ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΥΠΟΥΡΓΕΙΟ ΕΣΩΤΕΡΙΚΩΝ, ΑΠΟΚΕΝΤΡΩΣΗΣ & ΗΛΕΚΤΡΟΝΙΚΗΣ ΔΙΑΚΥΒΕΡΝΗΣΗΣ

Εθνικό Κέντρο Δημόσιας Διοίκησης & Αυτοδιοίκησης

ΥΠΟΕΡΓΟ: «ΔΗΜΙΟΥΡΓΙΑ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΩΝ ΓΙΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΔΥΝΑΜΙΚΟΥ ΤΗΣ ΔΗΜΟΣΙΑΣ ΔΙΟΙΚΗΣΗΣ ΒΑΣΕΙ ΣΧΕΔΙΩΝ ΕΚΠΑΙΔΕΥΣΗΣ»

ΤΙΤΛΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ:

«Εξ αποστάσεως εκπαίδευση στη γλώσσα PYTHON»

ΕΚΠΑΙΔΕΥΤΙΚΟ ΥΛΙΚΟ

Κωδικός εκπαιδευτικού υλικού:

Κωδικός Πιστοποίησης προγράμματος:

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

Ε.Π. ΔΙΟΙΚΗΤΙΚΗ ΜΕΤΑΡΡΥΘΜΙΣΗ 2007-2013

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

ΥΠΟΕΡΓΟ: «ΔΗΜΙΟΥΡΓΙΑ ΕΚΠΑΙΔΕΥΤΙΚΟΥ ΥΛΙΚΟΥ ΠΡΟΓΡΑΜΜΑΤΩΝ ΓΙΑ ΤΗΝ ΑΝΑΠΤΥΞΗ ΤΟΥ ΑΝΘΡΩΠΙΝΟΥ ΔΥΝΑΜΙΚΟΥ ΤΗΣ ΔΗΜΟΣΙΑΣ ΔΙΟΙΚΗΣΗΣ ΒΑΣΕΙ ΣΧΕΔΙΩΝ ΕΚΠΑΙΔΕΥΣΗΣ»

ΤΙΤΛΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ:

«Εξ αποστάσεως εκπαίδευση στη γλώσσα PYTHON»

ΟΜΑΔΑ ΕΡΓΑΣΙΑΣ

Μέλη Ομάδας

Συντονιστής
Δρ. Αναστάσιος Σαλής
Προϊστάμενος Τμήματος Εφαρμογών Πληροφορικής & Τεχνικής Υποστήριξης Ε.Κ.Δ.Δ.Α.

Συντάκτες
Δρ. Νικόλαος Ζάχαρης, Επίκουρος Καθηγητής ΤΕΙ Πειραιά
Δρ. Γεώργιος Μαυρομμάτης, Υπεύθυνος Σπουδών & Έρευνας Ε.Κ.Δ.Δ.Α.

Αξιολογητές
Δρ. Ηλίας Μαραγκός, Αναπληρωτής Διευθυντής ΙΝ.ΕΠ.
Δρ. Μερκούριος Μαργαριτόπουλος, Προϊστάμενος Π.ΙΝ.ΕΠ.

Εθνικό Κέντρο Δημόσιας Διοίκησης & Αυτοδιοίκησης
Συγγραφική ομάδα: Νικόλαος Ζάχαρης & Γεώργιος Μαυρομμάτης

ΠΕΡΙΕΧΟΜΕΝΑ

1 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΚΑΙ ΔΟΜΕΣ ΕΛΕΓΧΟΥ ΤΗΣ Python

Σκοπός

Σκοπός του κεφαλαίου είναι η εισαγωγή στη γλώσσα προγραμματισμού Python. Στο κεφάλαιο αυτό θα βρείτε σύντομα στοιχεία για την ιστορία της γλώσσας καθώς και οδηγίες για την εγκατάστασή της στον υπολογιστή σας. Ακόμα, το κεφάλαιο περιέχει όλες τις πληροφορίες σχετικά με τα κύρια χαρακτηριστικά της γλώσσας: αναγνωριστές και δεσμευμένες λέξεις, είσοδος και έξοδος δεδομένων από την κονσόλα, δομές ελέγχου ροής και επανάληψης, καθώς και αλφαριθμητικά.

Προσδοκώμενα αποτελέσματα

Όταν θα έχετε μελετήσει το κεφάλαιο αυτό θα μπορείτε να:

- γνωρίζετε τα κύρια ιστορικά στοιχεία και χαρακτηριστικά της Python,
- αναγνωρίζετε τις εκδόσεις της Python και τα χαρακτηριστικά τους,
- εγκαθιστάτε την Python,
- χρησιμοποιείτε το περιβάλλον ανάπτυξης IDLE,
- γνωρίζετε τις κύριες διαδικασίες συγγραφής και εκτέλεσης κώδικα,
- περιγράφετε τα κύρια χαρακτηριστικά ενός προγράμματος Python,
- διακρίνετε και χρησιμοποιείτε τις δομές ελέγχου και επανάληψης,
- χρησιμοποιείτε τα αλφαριθμητικά της γλώσσας.

Έννοιες-Κλειδιά

- Ιστορία Python
- Εκδόσεις Python
- Εγκατάσταση Python
- Τύποι δεδομένων
- Εντολές ελέγχου ροής
- Εντολές επανάληψης
- Αλφαριθμητικά

Εισαγωγικές Παρατηρήσεις

Η Python είναι μια σύγχρονη γλώσσα προγραμματισμού που ανήκει στην κατηγορία των γλωσσών σεναρίου. Η εκμάθησή της είναι εύκολη και όπως θα δείτε μετά τη μελέτη του παρόντος κεφαλαίου, θα μπορείτε να γράφετε άμεσα και να εκτελείτε

προγράμματα στη γλώσσα αυτή, αρκεί να έχετε βασικές γνώσεις στη χρήση και στον προγραμματισμό υπολογιστών. Στο παρόν κεφάλαιο θα αποκτήσετε τη γνώση και τις ικανότητες που χρειάζονται για να δημιουργείτε τόσο απλά προγράμματα όσο και συνθετότερες εφαρμογές.

1.1 Ιστορία και εκδόσεις

Η δημιουργία της γλώσσας προγραμματισμού Python ξεκίνησε στα τέλη της δεκαετίας του ‘80 από τον Ολλανδό Guido von Rossum. Η πρώτη εμφάνιση της γλώσσας έγινε το 1991 και η έκδοση 1.0 κυκλοφόρησε το 1994. Η Python, που πήρε το όνομά της από την ομάδα των βρετανών κωμικών ηθοποιών Monty Python, είναι λογισμικό ανοικτού κώδικα που σήμερα υποστηρίζεται από μια μεγάλη ομάδα εθελοντών και είναι ελεύθερα διαθέσιμη από το Ίδρυμα Python Software Foundation.

Η κύρια έκδοση της γλώσσας είναι γραμμένη σε γλώσσα C. Αυτή τη στιγμή υπάρχουν δύο κύριες ροές εκδόσεων της Python. Η σειρά των εκδόσεων 2.x αποτελεί την παλαιότερη εκδοχή της, με τελευταία στη σειρά την έκδοση με κωδικό αριθμό 2.7.6 που κυκλοφόρησε το Νοέμβριο του 2013. Η σειρά 2.x εξακολουθεί να υποστηρίζεται για λόγους συμβατότητας και υποστήριξης του λογισμικού που έχει γραφτεί ήδη, αλλά σύμφωνα με το Ίδρυμα Python Software δεν πρόκειται να συνεχίσει η εξέλιξή της. Η νέα σειρά 3.x αποτελεί το παρόν και το μέλλον της γλώσσας. Η τελευταία έκδοση της σειράς είναι η 3.4.0 που κυκλοφόρησε τον Μάρτιο του 2014. Για να καταλάβετε τη δυναμική της γλώσσας καθώς και την υποστήριξη που διαθέτει, σημειώστε ότι στο μόνο στο διάστημα Ιανουάριος-Μάρτιος 2014 έχουν ήδη κυκλοφορήσει τρεις εκδόσεις της: οι 3.3.4, 3.3.5 και η τελευταία 3.4.0.

Μια πολύ συνηθισμένη πρακτική στις εκδόσεις λογισμικού, είναι η διατήρηση της συμβατότητας με τις προηγούμενες εκδόσεις, κανόνας που έχει εγκαταλειφθεί στην περίπτωση της Python. Οι εκδόσεις 3.x δεν είναι συμβατές με τις εκδόσεις 2.x. Η απόφαση αυτή λήφθηκε από τον δημιουργό της γλώσσας προκειμένου να γίνουν αλλαγές στον μεταγλωττιστή που θα του εξασφαλίσουν καλύτερη λειτουργικότητα και βελτιώσεις που διαφορετικά δεν θα μπορούσαν να γίνουν. Οι διαφορές ανάμεσα στις δύο σειρές εκδόσεων είναι αρκετές και άλλες είναι προφανείς, για παράδειγμα στη σύνταξη των εντολών, ενώ άλλες (ίσως και οι σημαντικότερες) δεν φαίνονται άμεσα, για παράδειγμα, οι εκδόσεις 3.x αποθηκεύουν όλα τα αλφαριθμητικά της γλώσσας σε μορφή Unicode.

Η γλώσσα, πέραν της κύριας έκδοσής της, διατίθεται και σε άλλες υλοποιήσεις που είναι γραμμένες σε διάφορες γλώσσες προγραμματισμού και συνεργάζονται εγγενώς με αυτές, χωρίς να αποκλείεται να διαλειτουργούν και με άλλες γλώσσες μέσα από τις βιβλιοθήκες που διαθέτουν:

- Η Jython είναι μια Java - based έκδοση της Python και μπορεί να χρησιμοποιηθεί για να συνεργαστεί με κώδικα Java.
- Η Iron Python είναι γραμμένη σε C# και συνεργάζεται με το περιβάλλον .net της Microsoft.
- Το PyPy είναι ένα πρότζεκτ που υποστηρίχθηκε από την Ευρωπαϊκή Ένωση αλλά και την Google. Ο πυρήνας του είναι γραμμένος σε ένα υποσύνολο της ίδιας της Python.

Σε ό,τι μας αφορά και με δεδομένο ότι τώρα ξεκινάμε να μάθουμε τη γλώσσα, η επιλογή είναι μάλλον προφανής: θα χρησιμοποιήσουμε την τελευταία έκδοση της σειράς 3. Στην επόμενη ενότητα θα δούμε τη διαδικασία επιλογής της κατάλληλης έκδοσης και εγκατάστασης της γλώσσας, στον υπολογιστή μας. Νωρίτερα, ας δούμε τα κυριότερα χαρακτηριστικά της, που μπορούν να συνοψιστούν στα ακόλουθα (Swaroop, 2005):

Ανοικτός κώδικας. Η γλώσσα διατίθεται δωρεάν τόσο για χρήση όσο και σε επίπεδο κώδικα. Όποιος ενδιαφέρεται μπορεί να βρει τον πηγαίο κώδικα όλων των εκδόσεων από τη διεύθυνση http://python.org/ftp/python.

Απλή. Ο κώδικας Python είναι πολύ κοντά στην αγγλική γλώσσα γεγονός που εξασφαλίζει υψηλή αναγνωσιμότητα και ευκολία στην εκμάθηση.

Υψηλού επιπέδου. Η Python ανήκει στις λεγόμενες γλώσσες συγγραφής σεναρίων (scripting languages). Αυτό σημαίνει ότι διαθέτει ισχυρές εντολές με τις οποίες ο προγραμματιστής μπορεί να υλοποιεί τον κώδικά του χωρίς να ασχολείται με θέματα όπως είναι η διαχείριση μνήμης, η υπερχείλιση αριθμών, κ.λπ. Με τον τρόπο αυτό, ο προγραμματιστής μπορεί να εστιάζει στο πρόβλημα που έχει να επιλύσει και όχι στις ιδιαιτερότητες της γλώσσας που χρησιμοποιεί.

Τρέχει μέσω Διερμηνευτή. Οι παραδοσιακές γλώσσες προγραμματισμού διαθέτουν μεταγλωττιστή (Compiler) ο οποίος μετατρέπει τον πηγαίο κώδικα σε δυαδικό (γλώσσα μηχανής). Η διαδικασία αυτή της μεταγλώττισης πρέπει να προηγηθεί της εκτέλεσης του προγράμματος. Ο κώδικας Python εκτελείται μέσω διερμηνευτή (Interpreter). Σε αυτό το μοντέλο, ο πηγαίος κώδικας μετατρέπεται σε έναν ενδιάμεσο

κώδικα (ψηφιοκώδικας, bytecode) που στη συνέχεια εκτελείται από την εικονική μηχανή της Python (Python Virtual Machine). Το γεγονός αυτό απαλλάσσει τον προγραμματιστή από θέματα που αφορούν στη μεταγλώττιση, στην ύπαρξη των κατάλληλων βιβλιοθηκών κ.λπ., ενώ της προσδίδει φορητότητα.

Φορητή. Η σχεδίαση που χρησιμοποιεί διερμηνευτή και εικονική μηχανή, την καθιστά ικανή να εκτελείται σε διαφορετικές πλατφόρμες (Windows, Linux, Mac, κ.ά.) χωρίς να απαιτείται αλλαγή στον κώδικα.

Αντικειμενοστρεφής αλλά και Διαδικαστική. Ο προγραμματιστής μπορεί να γράψει είτε κώδικα βασισμένο τις αρχές του διαδικαστικού προγραμματισμού ή κώδικα πλήρως αντικειμενοστρεφή. Ο διαδικαστικός (Procedural Programming), όπως και ο συγγενής του δομημένος προγραμματισμός, υποστηρίζεται από τις παραδοσιακές γλώσσες. Τα δεδομένα και οι διαδικασίες, που επιδρούν σε αυτά, είναι διαχωρισμένα και η επεξεργασία γίνεται μέσα από συναρτήσεις που επιδρούν πάνω στα δεδομένα. Ο αντικειμενοστρεφής προγραμματισμός (Object Oriented Programming) στηρίζει τη λειτουργία του στα αντικείμενα που συνδυάζουν τα δεδομένα με τις λειτουργίες τους σε μια ολότητα, η οποία υλοποιείται με τη μορφή κλάσεων.

Συνεργάσιμη με άλλες γλώσσες. Είναι δυνατόν να γραφτεί κώδικας σε άλλες γλώσσες και να χρησιμοποιηθεί σε μια εφαρμογή που είναι γραμμένη σε Python, αλλά και αντίστροφα.

Όλα τα παραπάνω, έχουν καταστήσει τη γλώσσα Python μια από τις δημοφιλέστερες γλώσσες προγραμματισμού. Στην Εικόνα 1.1 μπορείτε να δείτε τη διαχρονική κατάταξη της δημοφιλίας της Python, όπως αυτή αναφέρεται στο δείκτη TIOBE (TIOBE Programming Community Index, www.tiobe.com).

Programming Language	2014	2009	2004	1999	1994	1989
C	1	2	2	1	1	1
Java	2	1	1	12	-	-
Objective-C	3	38	48	-	-	-
C++	4	3	3	2	2	3
C#	5	8	8	29	-	-
PHP	6	5	6	-	-	-
(Visual) Basic	7	4	5	3	3	7
Python	8	6	11	31	22	-
JavaScript	9	9	9	20	-	-
Perl	10	7	4	5	17	23
Lisp	14	19	15	14	6	2

Εικόνα 1.1: Διαχρονική κατάταξη δημοφιλίας γλωσσών προγραμματισμού (www.tiobe.com)

Δραστηριότητα 1.1: Εξοικείωση με τα γενικά χαρακτηριστικά της Python

1. Αναζητήστε στην ελληνική Wikipedia (Βικιπαίδεια) και διαβάστε το άρθρο με τίτλο «Python». Διαβάστε τα δείγματα κώδικα που θα βρείτε εκεί και προσπαθήστε να κατανοήσετε τη λειτουργία τους.
2. Αναζητήστε στο Youtube την ανάρτηση με θέμα «Guido van Rossum on the History of Python» και δείτε την (το βίντεο είναι μεγάλης χρονικής διάρκειας, αλλά αξίζει να δείτε, τουλάχιστον επιλεκτικά, μερικά σημεία του).

1.2 Εγκατάσταση και το περιβάλλον IDLE

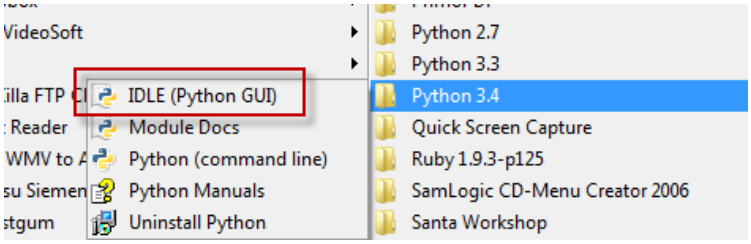
Ο επίσημος δικτυακός τόπος της γλώσσας βρίσκεται στη διεύθυνση www.python.org. Από εκεί θα κατεβάσετε την έκδοση που ταιριάζει με την πλατφόρμα (υλικό και λειτουργικό σύστημα) που χρησιμοποιείτε.

Η Python συνοδεύει τις περισσότερες εγκαταστάσεις Linux και δεν χρειάζεται να κάνετε κάτι περισσότερο. Αν χρησιμοποιείτε κάποιο λειτουργικό σύστημα της οικογένειας Linux, ανοίξτε ένα παράθυρο εντολών και πληκτρολογήστε την εντολή "python" στη γραμμή εντολών. Θα δείτε ότι η Python αποκρίνεται και σας περνά στο περιβάλλον της. Στην απίθανη περίπτωση που δεν είναι εγκατεστημένη, χρησιμοποιήστε το εργαλείο διαχείρισης λογισμικού που διαθέτει το λειτουργικό σας, για να την εγκαταστήσετε.

Για τα Windows υπάρχουν εκτελέσιμα προγράμματα που θα κατεβάσετε και θα τρέξετε στον υπολογιστή σας (Εικόνα 1.2). Η διαδικασία είναι ίδια με οποιαδήποτε άλλη εγκατάσταση λογισμικού στα Windows. Δεν χρειάζεται να κάνετε κάποιες ιδιαίτερες επιλογές, μπορείτε να αφήσετε την εγκατάσταση να ολοκληρωθεί με τις όποιες εξ ορισμού επιλογές σας δίνει. Η γλώσσα, στην τυπική περίπτωση εγκαθίσταται σε έναν κατάλογο της μορφής C:\pythonXX όπου XX ο αριθμός έκδοσης. Έτσι, μπορείτε να έχετε εγκαταστάσεις περισσότερων από μια εκδόσεων, οπότε δεν χρειάζεται αλλαγή.

Εικόνα 1.2: Εγκατάσταση Python - επιλογή έκδοσης

Για να γράψουμε κώδικα Python, μπορούμε να χρησιμοποιήσουμε ένα οποιοδήποτε απλό κειμενογράφο (π.χ. Notepad++). Ωστόσο, υπάρχουν πολύ πιο εξελιγμένα εργαλεία, τα ολοκληρωμένα περιβάλλοντα ανάπτυξης λογισμικού (IDE) που παρέχουν υπηρεσίες συγγραφής, εκτέλεσης και εκσφαλμάτωσης κώδικα. Η εγκατάσταση της Python περιλαμβάνει και ένα τέτοιο εργαλείο (Εικόνα 1.3), που έχει το όνομα IDLE (Integrated DeveLopment Environment). Σημειώστε ότι το λογισμικό αυτό είναι γραμμένο σε γλώσσα Python.

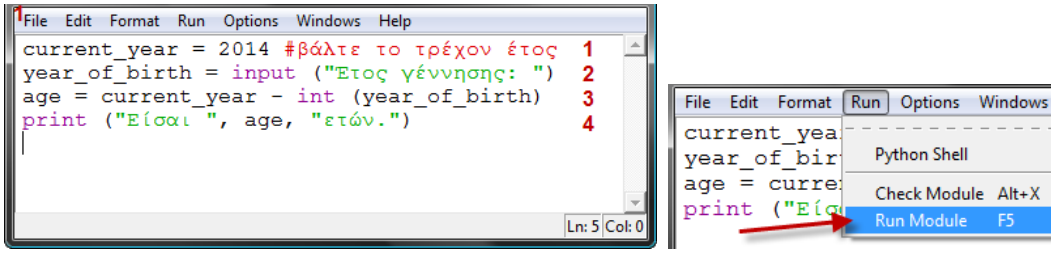


Εικόνα 1.3: Το IDLE

Το περιβάλλον IDLE είναι λιτό και σε πρώτη επαφή σας φέρνει σε ένα χώρο όπου μπορείτε να γράψετε το πρώτο σας πρόγραμμα. Πληκτρολογήστε τον κώδικα που βλέπετε στο Παράδειγμα 1.1, τηρώντας τη μορφή που βλέπετε (εκτός, βέβαια, από την αρίθμηση που είναι για αναφορά στις γραμμές. Για να πάτε στην επόμενη γραμμή, πατάτε το πλήκτρο enter.

Παράδειγμα 1.1: Το πρώτο μας πρόγραμμα

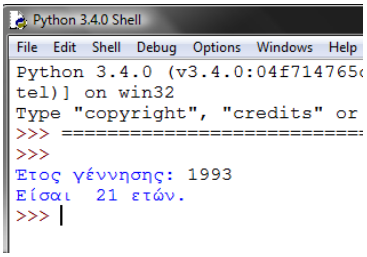
Το πρόγραμμα που ακολουθεί, ζητά από το χρήστη το έτος γέννησής του και υπολογίζει την ηλικία του, με αφαίρεση του έτους γέννησης από την τρέχουσα χρονιά.



Στη γραμμή 1 δηλώνουμε μια μεταβλητή με όνομα current_year και της δίνουμε τιμή 2014. Μπορείτε να δώσετε όποια τιμή θέλετε, ανάλογα με το τρέχον έτος. Το σύμβολο # προσδιορίζει έναρξη σχολίων. Ό,τι γράφεται μετά αγνοείται από τον διερμηνευτή και έχει χρησιμότητα μόνο για τον προγραμματιστή που διαβάζει τον κώδικα. Στη γραμμή 2 χρησιμοποιούμε τη συνάρτηση input που διαθέτει η Python για να εμφανίσουμε στην οθόνη το σχετικό μήνυμα και να αποθηκεύσουμε την απάντηση του χρήστη στη μεταβλητή year_of_birth. Η εντολή input διαβάζει σε μορφή αλφαριθμητικού και επομένως για να κάνουμε την αφαίρεση στη γραμμή 3, πρώτα

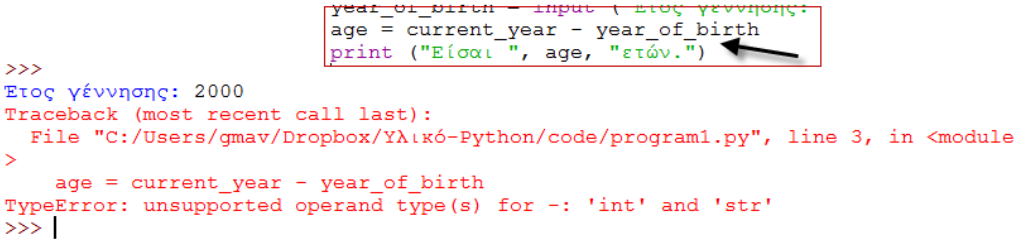
μετατρέπουμε το year_of_birth σε ακέραιο (int). Τέλος, στη γραμμή 4 χρησιμοποιούμε την εντολή print για να εμφανίσουμε στην οθόνη το αποτέλεσμα, μαζί με ένα κατατοπιστικό μήνυμα.

Για να εκτελέσετε το πρόγραμμα, επιλέγετε Run/Run Module ή πατάτε το πλήκτρο F5. Θα σας ζητήσει πρώτα να το αποθηκεύσετε. Δημιουργήστε ένα φάκελο, όπου σας εξυπηρετεί και αποθηκεύστε το αρχείο. Ένα πρόγραμμα Python συνήθως έχει επέκταση .py, επομένως μπορείτε να το ονομάσετε, π.χ. program1.py. Η εκτέλεση γίνεται στο python shell, που ανοίγει αυτόματα (Εικόνα 1.4).



Εικόνα 1.4: Εκτέλεση κώδικα στο Python shell

Αν ο κώδικας περιέχει λάθη, τότε το Python shell θα σας εμφανίσει μηνύματα, παρόμοια με αυτό που βλέπετε στην Εικόνα 1.5, όπου έχουμε αφαιρέσει τη μετατροπή τύπου από αλφαριθμητικό σε ακέραιο. Διαβάστε προσεκτικά το μήνυμα, πηγαίνετε στη γραμμή που σας αναφέρει, διορθώστε και πατήστε πάλι το F5 για να εκτελεστεί ο κώδικας. Παρατηρήστε ότι οι γραμμές 1 και 2 εκτελούνται κανονικά και το μήνυμα λάθους εμφανίζεται τη στιγμή που πρόκειται να εκτελεστεί η γραμμή 3 που έχει το πρόβλημα. Έτσι δουλεύει ένας διερμηνευτής (interpreter).



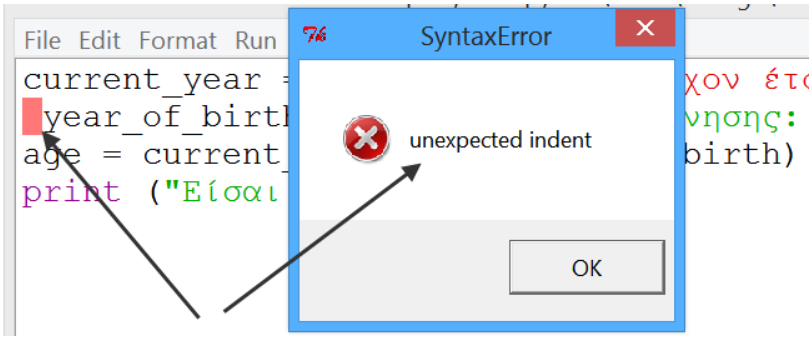
Εικόνα 1.5: Μηνύματα λάθους από την Python

1.3 Βασικά συστατικά της γλώσσας και I/O από την κονσόλα

Θα πάμε τώρα να δούμε από πιο κοντά τον κώδικα που γράψαμε στο Παράδειγμα 1.1, ώστε να έρθουμε σε επαφή με τα βασικά χαρακτηριστικά της γλώσσας Python.

Ένα σημαντικό σημείο που θα πρέπει να τονίσουμε, είναι ο τρόπος με τον οποίο θέλει η Python να γράφετε τις ομάδες των εντολών στον κώδικά σας. Στο Παράδειγμα 1.1 οι εντολές είναι γραμμένες μια σε κάθε γραμμή. Η γραφή μιας εντολής ανά γραμμή, δεν είναι υποχρεωτική. Μπορείτε να γράψετε περισσότερες εντολές στη

γραμμή, χωρίζοντάς τις με το ελληνικό ερωτηματικό (semicolon). Ακόμα, δεν έχουμε κενά ή tabs στην αρχή κάθε γραμμής. Αν αφήσετε ένα κενό στην αρχή μιας γραμμής, θα δείτε ότι η Python διαμαρτύρεται και σημειώνει λάθος (Εικόνα 1.6).



Εικόνα 1.6: Λάθος στις εσοχές του κώδικα

Γλώσσες όπως η C και η Java χρησιμοποιούν τα ζευγάρια αγκίστρων για να διαχωρίζουν τις ομάδες εντολών (blocks). Οι εσοχές μπαίνουν από τον προγραμματιστή απλά και μόνο για καλύτερη εμφάνιση του κώδικα, χωρίς να τις χρειάζεται ο μεταγλωττιστής, ο οποίος ελέγχει τα άγκιστρα. Η Python, αντί για άγκιστρα χρησιμοποιεί τις εσοχές (indents). Για παράδειγμα, αν θέλουμε να χρησιμοποιήσουμε μια εντολή if (θα την δούμε στην ενότητα 1.4), τότε οι εντολές που περιλαμβάνονται μέσα στους δύο κλάδους της θα πρέπει να έχουν την ίδια εσοχή και να βρίσκονται πιο μέσα από ό,τι οι γραμμές που περιέχουν το if και το else (Εικόνα 1.7). Το συνιστώμενο μήκος της εσοχής για τον κώδικα Python είναι τέσσερεις χαρακτήρες (μήκος που είναι ορισμένο το tab στον IDLE) και καλό είναι να το ακολουθούμε, παρόλο που δεν είναι υποχρεωτικό να έχει αυτή την τιμή.

```c
int main()
{
    int x =1;
    if (x==3)
    {
        printf("x is 3\n");
        printf("second command in if\n");
    }
    else
    {
        printf("x is not 3\n");
        printf("second command in else\n");
    }
    printf("outside if..else");
}
```

```python
x = 1
if x == 3:
  print("x is 3")
  print("δεύτερη εντολή στο if")
else:
  print("x is not 3")
  print("δεύτερη εντολή στο else")
print("Έξω από την if..else")
```

Εικόνα 1.7: Σύγκριση κώδικα C με Python

1.3.1 Μεταβλητές και ονόματα αναγνωριστών

Όπως συμβαίνει σε όλες τις γλώσσες, έτσι και η Python χρησιμοποιεί μεταβλητές για να αποθηκεύσει δεδομένα στη μνήμη. Η αναφορά στις μεταβλητές γίνεται μέσω του

ονόματός τους. Τα ονόματα των μεταβλητών (αναγνωριστές) πρέπει να υπακούν στους ακόλουθους κανόνες:

- Σχηματίζονται με συνδυασμούς γραμμάτων a..z, A..Z, αριθμών 0..9 καθώς και της κάτω παύλας _
- Δεν πρέπει να ξεκινάνε από αριθμό
- Υπάρχει διαφοροποίηση πεζών και κεφαλαίων
- Δεν πρέπει να είναι δεσμευμένη λέξη

Οι δεσμευμένες λέξεις έχουν ειδική σημασία γα τη γλώσσα και δεν μπορούν να χρησιμοποιηθούν από τον προγραμματιστή ως όνομα μεταβλητής. Στην Εικόνα 1.8 βλέπουμε τις δεσμευμένες λέξεις της γλώσσας Python (Python documentation, 2014).

```
False      class      finally    is         return
None       continue   for        lambda     try
True       def        from       nonlocal   while
and        del        global     not        with
as         elif       if         or         yield
assert     else       import     pass
break      except     in         raise
```

Εικόνα 1.8: Οι δεσμευμένες λέξεις της Python

Στο Παράδειγμα 1.1 χρησιμοποιήσαμε τις μεταβλητές current_year, year_of_birth, age. Προσέξτε ότι η Python διαφοροποιεί τα πεζά από τα κεφαλαία. Έτσι, οι αναγνωριστές NAME, Name, name αναφέρονται σε τρεις διαφορετικές μεταβλητές (θέσεις στη μνήμη). Στις μεταβλητές μπορούμε να αναθέτουμε σταθερές τιμές (literals), π.χ. ακέραιο, όπως κάναμε στην ανάθεση του τρέχοντος έτους: current_year = 2014, αλλά και αλφαριθμητικό, π.χ. myName = “George”. Στην πρώτη περίπτωση η current_year θα είναι τύπου αριθμός, ενώ στη δεύτερη η myName τύπου αλφαριθμητικό. Επειδή ο τύπος καθορίζεται δυναμικά κατά την εκτέλεση, μπορούμε αν χρειάζεται, να αναθέτουμε στο ίδιο όνομα μεταβλητής διαφορετικού τύπου τιμές μέσα στον κώδικα (Εικόνα 1.9). Ωστόσο αυτό, γενικά, δεν είναι καλή πρακτική.

```
myVar = 2014          >>>
print(myVar)          2014
myVar = "George"      George
print(myVar)          >>>
```

Εικόνα 1.9: Ανάθεση τιμών σε μεταβλητές

Άσκηση 1.1: Ποιες από τα παρακάτω ονόματα μεταβλητών είναι έγκυρα σύμφωνα με τους κανόνες ονομάτων της Python;		
MyName 2ndClass Serial#	FALSE False A	One Variable __grade a

1.3.2 Τύποι δεδομένων

Η Python χρησιμοποιεί δυναμική δήλωση τύπων και επομένως δεν απαιτεί να δηλώσουμε νωρίτερα τις μεταβλητές και τον τύπο τους, ο οποίος καθορίζεται κατά την εκτέλεση του κώδικα. Οι σημαντικότεροι βασικοί τύποι δεδομένων της γλώσσας είναι:

Λογικού τύπου (bool). Παίρνουν τιμές True και False. Αν τις χρησιμοποιήσουμε σε αριθμητικές παραστάσεις, η True αποτιμάται σε 1 και η False σε 0.

Ακέραιοι (int). Ο μόνος περιορισμός στο μέγεθός τους πηγάζει από τη μνήμη του υπολογιστή.

Δεκαδικοί κινητής υποδιαστολής (float). Το εύρος των αριθμών που μπορεί να αποθηκεύσει μια τέτοια μεταβλητή εξαρτάται από τη γλώσσα στην οποία έχει γραφτεί η Python (π.χ. C, Java, C#). Είναι δεδομένη η αδυναμία ενός υπολογιστή να παραστήσει με απόλυτη ακρίβεια δεκαδικούς αριθμούς και παρόμοια προβλήματα έχει και η Python, αλλά, στη συνηθισμένη περίπτωση, οι αριθμοί float επαρκούν, αρκεί να έχουμε κατά νου ότι οι συγκρίσεις μεταξύ τους μπορεί να μην είναι ακριβείς, λόγω σφαλμάτων προσέγγισης.

Στον Πίνακα 1.1 μπορείτε να δείτε τις κυριότερες πράξεις ανάμεσα σε αριθμητικούς τύπους μεταβλητών. Όταν οι τύποι αυτοί συμμετέχουν σε αριθμητικές παραστάσεις, τότε το αποτέλεσμα υπολογίζεται με βάση την προτεραιότητα των πράξεων, που επίσης βλέπουμε στον Πίνακα 1.1.

Πίνακας 1.1: Προτεραιότητα κυριότερων αριθμητικών τελεστών

Προτεραιότητα	Σύμβολο	Επεξήγηση
1	**	Ύψωση σε δύναμη
2	/	Διαίρεση
	*	Πολλαπλασιασμός
	//	Ακέραιο μέρος διαίρεσης
	%	Υπόλοιπο ακέραιας διαίρεσης
3	+	Πρόσθεση
	-	Αφαίρεση

Η αποτίμηση μιας παράστασης γίνεται από αριστερά προς τα δεξιά. Μεταξύ δύο συνεχόμενων τελεστών πρώτα υπολογίζεται αυτός που έχει τη μεγαλύτερη προτεραιότητα. Οι παρενθέσεις αλλάζουν την προτεραιότητα και υποχρεώνουν την αποτίμηση να ξεκινήσει από το εσωτερικό τους. Για παράδειγμα, η παράσταση 3+6/2 δίνει αποτέλεσμα 6.0, αφού η διαίρεση προηγείται της πρόσθεσης, ενώ η παράσταση

(3+6)/2 ισούται με 4.5 αφού, λόγω της παρένθεσης, θα προηγηθεί η πρόσθεση. Αν έχουμε αμφιβολία για μια παράσταση, συνιστάται να χρησιμοποιούμε παρενθέσεις, ακόμα και αν δεν χρειάζονται πραγματικά.

Η Python μας δίνει διάφορες δυνατότητες ανάθεσης τιμών σε αριθμητικές μεταβλητές, όπως μπορείτε να δείτε στο Παράδειγμα 1.2, που ακολουθεί.

Παράδειγμα 1.2: Ανάθεση αριθμητικών τιμών σε μεταβλητές

Ανάθεση	Επεξήγηση
x = 12	Θέτει x=12
x, y = 3, 5	Θέτει x=3 και y=5
x = y = 16.3	Θέτει x=16.3 και y=16.3
x = y = 2, 6	Θέτει x=(2,6) και y=(2,6)
x = y = 3,2,6	Θέτει x = (3,2,6) και y = (3,2,6)
Κώδικας	Αποτέλεσμα εκτέλεσης

`x=12`
`print(x)`
`x,y=3,5`
`print(x,y)`
`x=y=16.3`
`print(x,y)`
`x=y=2,6`
`print(x,y)`
`x=y=3,2,6`
`print(x,y)`
	`12`
	`3 5`
	`16.3 16.3`
	`(2, 6) (2, 6)`
	`(3, 2, 6) (3, 2, 6)`

Άσκηση 1.2: Γράψτε ένα πρόγραμμα που θα διαβάζει δύο αριθμούς από το πληκτρολόγιο και των οποίων θα υπολογίζει το άθροισμα, το γινόμενο και τη διαφορά τους. Το αποτέλεσμα θα εμφανίζεται στην οθόνη.

Άσκηση 1.3: Γράψτε ένα πρόγραμμα που θα διαβάσει τρεις αριθμούς από το πληκτρολόγιο, θα υπολογίζει το μέσο όρο τους και θα εμφανίζει το αποτέλεσμα, με κατάλληλο μήνυμα.

Αλφαριθμητικά (str). Είναι ακολουθίες από χαρακτήρες. Για να παραστήσουμε μια αλφαριθμητική σταθερά χρησιμοποιούμε είτε μονά είτε διπλά εισαγωγικά. Αν το αλφαριθμητικό περιέχει μονά εισαγωγικά ως τμήμα του, τότε το βάζουμε σε διπλά, ενώ αν περιέχει διπλά, το βάζουμε σε μονά.

Ακόμα, μπορούμε να περιλάβουμε σε αλφαριθμητικό τους ειδικούς χαρακτήρες (όπως οι \n και \t που είναι αλλαγή γραμμής και tab αντίστοιχα) βάζοντας νωρίτερα τον χαρακτήρα διαφυγής που είναι η ανάποδη διαίρεση (backslash). Στην περίπτωση που θέλουμε να έχουμε αλφαριθμητικό που θα εκτείνεται σε περισσότερες της μιας γραμμές, χρησιμοποιούμε τριπλά μονά ή τριπλά διπλά εισαγωγικά. Στην Εικόνα 1.10

μπορείτε να δείτε μερικά απλά παραδείγματα χρήσης των αλφαριθμητικών, ενώ στην ενότητα 1.4 θα δούμε αναλυτικότερα τις κύριες λειτουργίες τους.

```
print("Χρησιμοποιούμε διπλά εισαγωγικά")
print('ή μονά εισαγωγικά')
print("Αν περιέχει 'μονά', το βάζω σε διπλά")
print('και αντίστροφα "διπλά" μέσα σε μονά')
print("Μπορώ και με backslash, π.χ. \' \" ")
print("""τριπλά εισαγωγικά
   για παραπάνω από μια γραμμή\n
   το \\n είναι νέα γραμμή""")
```

```
Χρησιμοποιούμε διπλά εισαγωγικά
ή μονά εισαγωγικά
Αν περιέχει 'μονά', το βάζω σε διπλά
και αντίστροφα "διπλά" μέσα σε μονά
Μπορώ και με backslash, π.χ. ' "
τριπλά εισαγωγικά
   για παραπάνω από μια γραμμή

   το \n είναι νέα γραμμή
```

Εικόνα 1.10: Παραδείγματα εμφάνισης αλφαριθμητικών σταθερών

Όπως έχουμε ήδη δει (π.χ. Παράδειγμα 1.1), η έξοδος δεδομένων στην κονσόλα γίνεται με τη συνάρτηση print(). Παραδείγματα της εντολής print() είδαμε και στην Εικόνα 1.10. Η print() εμφανίζει ό,τι περιέχει μέσα στην παρένθεση (ορίσματα) και στη συνέχεια μετακινεί τον κέρσορα στην επόμενη γραμμή. Μπορούμε να βάζουμε περισσότερα του ενός ορίσματα τα οποία μπορεί να είναι και αριθμοί. Αν τα ορίσματα χωρίζονται με κόμμα, τότε ανάμεσά τους η print εμφανίζει ένα κενό, ενώ αν αντί για κόμμα χωρίζονται με + τότε τα ορίσματα εμφανίζονται χωρίς καμία παρεμβολή άλλων χαρακτήρων. Η εκτέλεση της εντολής χωρίς όρισμα στην παρένθεση, απλά αφήνει μια κενή γραμμή.

Η συνάρτηση input() χρησιμοποιείται για είσοδο δεδομένων από την κονσόλα, ενώ μπορεί να περιέχει αλφαριθμητικό όρισμα μέσα στην παρένθεση, οπότε το εμφανίζει και στη συνέχεια περιμένει για να διαβάσει την είσοδο του χρήστη.

Παράδειγμα 1.3: Χρήσεις της εντολής print()

```
name = "Peter"
print("Hi",name,"how are you?")
print("Hi"+name+"how are you?")
```

```
Hi Peter how are you?
HiPeterhow are you?
>>>
```

Άσκηση 1.4: Γράψτε ένα πρόγραμμα που θα περιέχει δύο ακριβώς εντολές print, που θα εμφανίζουν αντίστοιχα στην οθόνη τα παρακάτω μηνύματα:

1η: Το σύμβολο \n είναι ο ειδικός χαρακτήρας αλλαγής γραμμής και
το σύμβολο \t είναι ο ειδικός χαρακτήρας για το tab.

2η: Άνοιξη,
Καλοκαίρι,
Φθινόπωρο,
Χειμώνας.

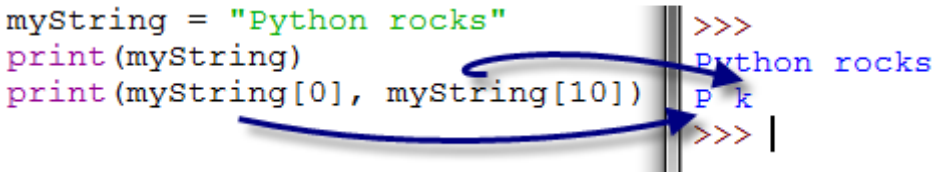
1.4 Αλφαριθμητικά

Μπορείτε να βλέπετε τα αλφαριθμητικά της Python σαν πίνακες χαρακτήρων, όπου το πρώτο στοιχείο (ο πρώτος χαρακτήρας) έχει δείκτη μηδέν. Έτσι, μπορείτε να

αναφέρεστε σε ένα συγκεκριμένο χαρακτήρα του αλφαριθμητικού χρησιμοποιώντας τις αγκύλες και μέσα τον αντίστοιχο ακέραιο δείκτη.

Παράδειγμα 1.4: αλφαριθμητικά και δείκτες χαρακτήρων

Έστω το αλφαριθμητικό myString = "Python rocks". Το μήκος του είναι δώδεκα χαρακτήρες (μετράμε και τα κενά διαστήματα). Δείτε στην εικόνα που ακολουθεί πώς μπορούμε να αναφερθούμε σε όποιον χαρακτήρα επιθυμούμε, μέσω του δείκτη.



Αφού το αλφαριθμητικό έχει μήκος δώδεκα, ο δείκτης έχει έγκυρη τιμή ανάμεσα στο 0 και στο 11, όπως μπορείτε να δείτε στην Εικόνα 1.11. Η προσπάθεια να αναφερθούμε σε δείκτη εκτός των ορίων, θα έχει σαν αποτέλεσμα μήνυμα λάθους: IndexError: string index out of range

myString	P	y	t	h	o	n		r	o	c	k	s
Δείκτης	0	1	2	3	4	5	6	7	8	9	10	11

Εικόνα 1.11: Αλφαριθμητικά και δείκτες χαρακτήρων

Τα αλφαριθμητικά της Python είναι αμετάβλητα (immutable). Αυτό σημαίνει ότι από τη στιγμή που θα δημιουργήσουμε ένα string, αυτό δεν μπορεί να τροποποιηθεί. Αν δοκιμάσετε στον κώδικα του Παραδείγματος 1.4 να αλλάξετε την τιμή ενός χαρακτήρα, π.χ. myString[10] = "W", αυτό θα έχει σαν αποτέλεσμα μήνυμα λάθους. Θα δούμε λίγο παρακάτω μια λύση στο «πρόβλημα» αυτό, αφού μελετήσουμε τις λειτουργίες των αλφαριθμητικών, που μας παρέχονται μέσω των δεικτών.

Η χρήση δεικτών στα αλφαριθμητικά της Python δεν περιορίζεται στη λειτουργία που είδαμε στο Παράδειγμα 1.4. Μπορούμε να κάνουμε πολύ περισσότερα πράγματα, που θα τα δούμε αμέσως τώρα.

<string> [start : stop]	Το μέρος του <string> από το δείκτη start μέχρι και το δείκτη stop-1
<string> [start :]	Το μέρος του <string> από το δείκτη start μέχρι και το τέλος του αλφαριθμητικού
<string> [: stop]	Το μέρος του <string> από την αρχή μέχρι και το δείκτη stop-1
<string> [:]	Όλο το <string>

Παράδειγμα 1.5: λειτουργίες δεικτών σε αλφαριθμητικά

```
myString = "Python rocks"          >>>
print(myString[3:6])               hon
print(myString[3: ])               hon rocks
print(myString[ :4])               Pyth
print(myString[ : ])               Python rocks
                                   >>>
```

Οι δείκτες μπορεί να είναι και αρνητικοί. Στην περίπτωση αυτή, έχουμε τις ακόλουθες λειτουργίες:

<string> [-x]	Ο x-στος χαρακτήρας, μετρώντας από δεξιά (x>0)
<string> [-x :]	Οι x τελευταίοι χαρακτήρες
<string> [:-x]	Όλοι εκτός από τους τελευταίους x χαρακτήρες

Παράδειγμα 1.5: λειτουργίες αρνητικών δεικτών σε αλφαριθμητικά

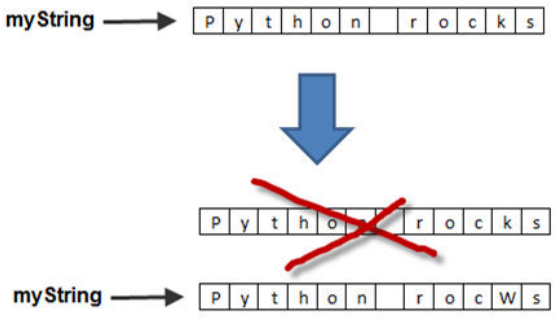
```
myString = "Python rocks"          >>>
print(myString[-2])                k
print(myString[-3: ])              cks
print(myString[ :-4])              Python r
                                   >>>
```

Όπως είπαμε, τα αλφαριθμητικά της Python είναι αμετάβλητα. Πώς, επομένως, μπορούμε να αλλάξουμε ένα αλφαριθμητικό; Μια λύση είναι να δημιουργήσουμε ένα νέο, τροποποιώντας τους χαρακτήρες που επιθυμούμε.

Παράδειγμα 1.6: τροποποίηση αλφαριθμητικού

```
myString = "Python rocks"                                  ...
print(myString)                                            Python rocks
myString = myString[0:10]+ "W" + myString[11:]             Python rocWs
print(myString)                                            >>>
```

Στο Παράδειγμα 1.6 ο τελεστής + κάνει τη συνένωση. Κόψαμε τους δέκα πρώτους χαρακτήρες του αρχικού αλφαριθμητικού που θέλουμε να κρατήσουμε (δείκτες από μηδέν έως και εννέα), αντικαταστήσαμε τον 11^{ο} χαρακτήρα (δείκτης 10) και στο τέλος προσθέσαμε τους υπόλοιπους (τον 12^{ο}), που έχει δείκτη 11.



Εικόνα 1.12: Αλφαριθμητικά και διαχείριση μνήμης

Η διαδικασία που υλοποιήσαμε στο Παράδειγμα 1.6 μπορεί να παρασταθεί όπως βλέπετε στην Εικόνα 1.12. Μπορείτε να καταλάβετε καλύτερα τη διαδικασία, αν σκεφτείτε ότι η μεταβλητή myString δείχνει σε ένα χώρο στη μνήμη, στον οποίο βρίσκεται το αλφαριθμητικό. Η γραμμή 3 του κώδικα δημιουργεί ένα νέο αλφαριθμητικό στη μνήμη, βάζοντας το δείκτη myString να δείξει σε αυτό.

Άσκηση 1.5: Γράψτε ένα πρόγραμμα που θα διαβάζει το όνομα και το επώνυμό σας και θα δημιουργεί ένα αλφαριθμητικό με τα αρχικά σας, χωρισμένα με τελείες. Για παράδειγμα αν το όνομα είναι Πέτρος και το επώνυμο Νικολάου, θα εμφανίζει Π.Ν.

1.4.1 Βασικές συναρτήσεις αλφαριθμητικών

Στην ενότητα αυτή, θα δούμε μερικές σημαντικές συναρτήσεις (μεθόδους) που παρέχει η Python για την διαχείριση αλφαριθμητικών.

Α. **len (<string>)** Επιστρέφει το μήκος του αλφαριθμητικού ορίσματος. Σημειώστε ότι η len μπορεί να χρησιμοποιηθεί και σε άλλους τύπους δεδομένων εκτός του string (Πλειάδες, Λίστες και Λεξικά που θα τα δούμε όλα στο Κεφάλαιο 2).

Παράδειγμα 1.7: η συνάρτηση len()

```
myString = "Python rocks"
print("The length of string \""+myString+"\" is",len(myString))
```

```
>>>
The length of string "Python rocks" is 12
>>>
```

Β. <**string1>.find(<string2>)** Ψάχνει στο <string1> για να βρει το <string2> και επιστρέφει τη θέση (δείκτη) της πρώτης ταύτισης. Αν δεν βρει ταύτιση, επιστρέφει -1.

Παράδειγμα 1.8: η μέθοδος find()

```
demoString = 'η γλώσσα κόκκαλα δεν έχει και κόκκαλα τσακίζει'
index = demoString.find('κόκκαλα')
if index == -1:
    print("Search word not found")
else:
    print("Search word found starting at",index)
```

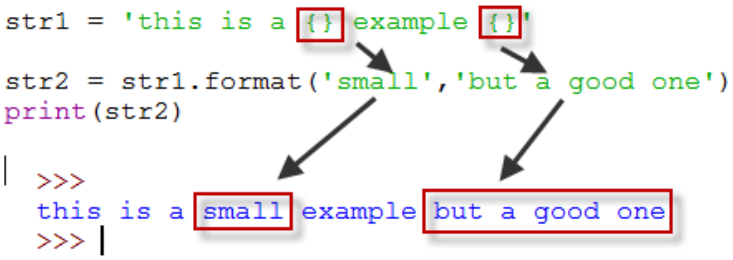
```
>>>
Search word found starting at 9
>>>
```

Την εντολή if θα την δούμε στην επόμενη ενότητα, αλλά πιστεύουμε δεν δυσκολευτήκατε να καταλάβετε τη λογική του Παραδείγματος 1.8.

Γ. <**string1>.find(<string2>, <number>)** Ψάχνει στο <string1> ξεκινώντας από τη θέση <number> για να βρει το <string2>. Επιστρέφει το δείκτη με τη θέση της πρώτης ταύτισης ή -1 αν δεν βρει ταύτιση.

Δ. **str.format()** Μορφοποιεί αριθμούς και αλφαριθμητικά. Η κύρια εφαρμογή της γίνεται στις περιπτώσεις εξόδου, για παράδειγμα σε συνδυασμό με την εντολή print(), αλλά αυτό δεν είναι απαραίτητο, όπως μπορείτε να δείτε στο Παράδειγμα 1.9, όπου δημιουργούμε ένα αλφαριθμητικό με δυναμικό τρόπο.

Παράδειγμα 1.9: δημιουργία αλφαριθμητικού με την format()



Όπως μπορείτε να δείτε, ο κώδικας δημιουργεί το αλφαριθμητικό str2 χρησιμοποιώντας ως πρότυπο το str1 με θέσεις οι οποίες παίρνουν τιμές με δυναμικό τρόπο. Οι θέσεις παριστάνονται με ζεύγη αγκυλών και στο όρισμα της format θα πρέπει να περιλαμβάνονται ισάριθμα αντικείμενα που θα τις αντικαταστήσουν.

Μέσα στα άγκιστρα μπορούμε να βάλουμε ακέραιους αριθμούς που είναι δείκτες για το όρισμα της format. Στην περίπτωση αυτή, το άγκιστρο αντικαθίσταται με το αντίστοιχο όρισμα που βρίσκεται στη θέση που δείχνει ο δείκτης. Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε ονόματα της επιλογής μας για κάθε άγκιστρο-θέση αντικατάστασης. Το Παράδειγμα 1.10 θα σας δώσει να καταλάβετε τις λειτουργίες αυτές.

Παράδειγμα 1.10: format() με δείκτες και ονόματα θέσης

```
print("Η {0} και η {1} είναι Ήπειροι".format("Ευρώπη", "Ασία"))
print("Η {1} και η {0} είναι Ήπειροι".format("Ευρώπη", "Ασία"))
print("Η {cont1} και η {cont2} είναι Ήπειροι".format(cont1="Ασία",cont2="Ευρώπη"))
```

```
Η Ευρώπη και η Ασία είναι Ήπειροι
Η Ασία και η Ευρώπη είναι Ήπειροι
Η Ασία και η Ευρώπη είναι Ήπειροι
>>>
```

Η format() έχει και άλλες, πολλές δυνατότητες και η κάλυψή της ξεφεύγει από το πλαίσιο του παρόντος συγγράμματος. Για παράδειγμα, μπορεί να χρησιμοποιηθεί για μορφοποίηση αριθμών με τους προσδιοριστές f και d για δεκαδικούς και ακεραίους, αντίστοιχα.

Παράδειγμα 1.11: η format για μορφοποίηση αριθμών

Στη γραμμή 2 αναφερόμαστε δύο φορές στο ίδιο όρισμα, όπως φαίνεται από το δείκτη μηδέν που υπάρχει και στα δύο άγκιστρα. Στο δεύτερο άγκιστρο, ζητάμε να

μορφοποιηθεί ο δεκαδικός num με προσέγγιση 4 δεκαδικά ψηφία. Στη γραμμή 3 και πάλι τα δεκαδικά είναι τέσσερα, αλλά ζητάμε ο αριθμός να γραφτεί σε συνολικά δέκα θέσεις, ενώ στη γραμμή 6 μορφοποιούμε την εμφάνιση ακεραίου, βάζοντάς τον να πιάσει χώρο 10 χαρακτήρων.

```
1 num=3.12391
2 print("{0} rounded to 4 decimal digits:{0:.4f}".format(num))
3 num=3.12399
4 print("{0} rounded to 4 decimal digits:{0:10.4f}".format(num))
5 num=12
6 print("{0} printed with format specifier   :{0:10d}".format(num))
```

```
3.12391 rounded to 4 decimal digits:3.1239
3.12399 rounded to 4 decimal digits:    3.1240
12 printed with format specifier   :         12
```

1.5 Δομές ελέγχου

Στις δομές ελέγχου περιλαμβάνονται η if που είναι η εντολή λήψης αποφάσεων και ελέγχου ροής του κώδικα και οι εντολές for και while που είναι οι εντολές επανάληψης της Python.

1.5.1 Η εντολή if

Στον Πίνακα 1.2 βλέπουμε τη γενική σύνταξη της εντολής if, που είναι η ακόλουθη:

Πίνακας 1.2: Γενική σύνταξη εντολής if και παράδειγμα

```
if <συνθήκη1>:
        <εντολή1>
        <εντολή2>
        ……
elif <συνθήκη2>:
        <εντολή3>
        <εντολή4>
        ……
else:
        <εντολή5>
        <εντολή6>
        ……
<εντολές εκτός του if>
```

```
if noOfPages == 0:
        print(“Document is empty”)
        flag = False
elif noOfPages > 0:
        print(“Document is not empty”)
        flag == True
else:
        print(“Wrong number of pages”)
print(“This is outside if”)
```

Η δήλωση if παρουσιάζεται μόνο μια φορά, ενώ μπορούμε να έχουμε όσα elif χρειάζονται (μέχρι και κανένα). Το else είναι προαιρετικό και μπορεί να είναι μέχρι ένα. Μετά τη συνθήκη του if και μετά το else ακολουθεί άνω-κάτω τελεία. Παρατηρήστε τις εσοχές που καθορίζουν τις ομάδες εντολών. Όσοι έχετε γνώσεις άλλων γλωσσών προγραμματισμού, όπως C, Java, C++ κ.λπ. θα παρατηρήσατε ότι δεν χρησιμοποιούνται άγκιστρα, αλλά η λειτουργικότητά τους αντικαθίσταται από τις εσοχές.

Στην Εικόνα 1.13 βλέπετε ένα παράδειγμα της χρήσης της if. Πέραν της σύνταξης της εντολής, μπορείτε να δείτε και μερικά καινούργια πράγματα: Στη δεύτερη γραμμή διαβάζουμε τη μεταβλητή sys.platform και την αποθηκεύουμε στη μεταβλητή os. Η μεταβλητή αυτή περιλαμβάνεται στη βιβλιοθήκη sys, την οποία κάνουμε εισαγωγή στο πρόγραμμά μας με την εντολή import, που βλέπετε στη γραμμή 1. Η μέθοδος str.startswith() επιστρέφει True αν το str αρχίζει από το αλφαριθμητικό που της βάζουμε ως όρισμα, διαφορετικά επιστρέφει False. Η τελευταία εντολή print βρίσκεται εκτός του if, πράγμα που φαίνεται από την εσοχή της.

```
import sys
os = sys.platform
if os.startswith ("win"):
    print("Το λειτουργικό σας σύστημα είναι Windows")
else:
    print("Το λειτουργικό σας σύστημα δεν είναι Windows")
print(os)
```

Εικόνα 1.13: Παράδειγμα if - εισαγωγή βιβλιοθήκης

Στις εντολές if αλλά και όπου αλλού υπάρχουν λογικές συνθήκες, μπορούμε να χρησιμοποιήσουμε τους λογικούς τελεστές. Στον Πίνακα 1.3 μπορείτε να δείτε τους κυριότερους λογικούς τελεστές.

Πίνακας 1.3: Λογικοί τελεστές

<	Μικρότερο από
>	Μεγαλύτερο από
<=	Μικρότερο ή ίσο
>=	Μεγαλύτερο ή ίσο
==	Έλεγχος για ισότητα
!=	Διάφορο
not	Λογική άρνηση (όχι)
and	Λογική σύζευξη (και)
or	Λογική διάζευξη (ή)

Στο παράδειγμα που ακολουθεί χρησιμοποιούμε σύνθετη λογική συνθήκη και τη μέθοδο format για να μορφοποιήσουμε την έξοδο.

Παράδειγμα 1.12: Εντολή if και λογικοί τελεστές

Μια τηλεφωνική εταιρεία υπολογίζει το λογαριασμό ενός συνδρομητή ως εξής: για τα πρώτα 200 λεπτά ομιλίας η χρέωση είναι 0,022 ευρώ ανά λεπτό. Για τα επόμενα 200 λεπτά (από 201 μέχρι 400) η χρέωση είναι 0,015 ευρώ ανά λεπτό και για τα πέραν των πρώτων 400 λεπτών η χρέωση είναι 0,012 ευρώ ανά λεπτό. Το πρόγραμμα που ακολουθεί, διαβάζει το χρόνο ομιλίας ενός συνδρομητή και υπολογίζει τη χρέωση.

```
minutes = int (input("Δώστε το χρόνο ομιλίας: "))
if minutes <= 200:
    bill = minutes * 0.022
elif minutes > 200 and minutes <= 400:
    bill = 200 * 0.022 + (minutes - 200)* 0.015
elif minutes > 400:
    bill = 200 * 0.022 + 200 * 0.015 + (minutes - 400)* 0.012
print("Ο λογαριασμός είναι {:6.2f} ευρώ".format(bill))
```

1.5.2 Η εντολή for

Μια ιδιαίτερα χρήσιμη συνάρτηση που παρέχει η Python είναι η range(). Την αναφέρουμε εδώ διότι βρίσκει μεγάλη εφαρμογή στην εντολή επανάληψης for. Οι σημαντικότερες δυνατότητες της συνάρτησης range() είναι οι ακόλουθες:

A. **range (n)** Ο n είναι ακέραιος αριθμός. Η συνάρτηση επιστρέφει τη λίστα (μια ακολουθία) ακεραίων από το 0 μέχρι και το n-1.

B. **range (m, n)** Οι m, n ακέραιοι. Επιστρέφει τη λίστα των ακεραίων από το m μέχρι και το n-1.

A. **range (m, n, k)** Οι m, n, k είναι ακέραιοι. Επιστρέφει τη λίστα των αριθμών ανάμεσα σε m και n-1 με βήμα k.

Έχοντας δει τη συνάρτηση range(), πάμε τώρα να δούμε τη γενική σύνταξη της εντολής for:

Πίνακας 1.4: Γενική σύνταξη εντολής for και παράδειγμα

for <μεταβλητή> in <ακολουθία>
  <εντολή1>
  <εντολή2>
  ……..
<εντολές εκτός του for>
	print("Προπαίδεια του 9")
	for i in range(10):
	  print("{0} x 9 = {1}".format(i,i*9))
	print("Τέλος")

Η εντολή for έχει πολύ περισσότερες δυνατότητες από την απλή απαρίθμηση που είδατε στο παραπάνω παράδειγμα. Θα τις εκτιμήσετε όταν, στο Κεφάλαιο 2 μάθετε τις Λίστες και τις άλλες δομές δεδομένων της Python. Στο παράδειγμα που ακολουθεί, ο κώδικας διαβάζει μια πρόταση από την κονσόλα, διαπερνά έναν-έναν τους χαρακτήρες της και εμφανίζει σε μια γραμμή μόνο τους κεφαλαίους. Παρατηρήστε πώς εμποδίζουμε την print να αλλάξει κενή γραμμή.

Παράδειγμα 1.13: Εντολή for και διάτρεξη αλφαριθμητικού

```
sentence = input("Enter a phrase:")
for letter in sentence:
    if letter.isupper(): print(letter,end="")

Enter a phrase:I am learning Python. I Like it!
IPIL
```

1.5.3 Η εντολή while

Η γενική σύνταξη της εντολής while φαίνεται στον Πίνακα 1.5.

Πίνακας 1.5: Γενική σύνταξη εντολής while και παράδειγμα

while <συνθήκη>:	num = int (input("Δώστε αριθμούς, 0 για τέλος: "))
<εντολή1>	sum = 0;
<εντολή2>	while num != 0:
……	sum +=num
<εντολές εκτός του while>	num = int(input())
	print("Το σύνολο είναι {:.3f}".format(sum))

Στο παράδειγμα, διαβάζουμε αριθμούς μέχρις ότου να δοθεί το μηδέν που σηματοδοτεί το τέλος της εισόδου. Το πρόγραμμα αθροίζει τους αριθμούς και στο τέλος εμφανίζει το άθροισμα.

Η επανάληψη σταματά όταν η συνθήκη αποτιμηθεί σε False. Είναι σημαντικό να φροντίζουμε ώστε κάποια από τις εντολές μέσα στο βρόγχο της επανάληψης να τροποποιεί τη συνθήκη. Μια άλλη προσέγγιση είναι να χρησιμοποιήσουμε την εντολή break, όπως μπορείτε να δείτε στο Παράδειγμα 1.14.

Παράδειγμα 1.14: Ατέρμων βρόγχος και εντολή break

```
num = int (input("Δώστε αριθμούς, 0 για τέλος: "))
sum = 0;
while True:
    if num == 0: break;
    sum +=num
    num = int(input())
print("Το σύνολο είναι {:.3f}".format(sum))
```

```
Δώστε αριθμούς, 0 για τέλος: 2
3
4
5
0
Το σύνολο είναι 14.000
```

1.5.4 Άλλες δυνατότητες των δομών ελέγχου

Ο εντολές επανάληψης μπορούν να συνδυαστούν με τις εντολές break, continue και pass.

break. Η εντολή break σταματά την εκτέλεση μιας επαναληπτικής διαδικασίας, όπως είδαμε στο Παράδειγμα 1.14.

continue. Πηγαίνει στο επόμενο βήμα ενός βρόγχου.

pass. Είναι η κενή εντολή, που δεν κάνει τίποτε. Μπορούμε να την χρησιμοποιήσουμε σε ένα if αν σε έναν από τους δύο κλάδους του δεν θέλουμε να γίνει καμία επεξεργασία.

Οι βρόγχοι επανάληψης for και while μπορούν να συνταχθούν με else. Ο κώδικας που περιλαμβάνεται στο else εκτελείται, για μεν το for όταν τελειώσουν οι επαναλήψεις, για δε το while όταν η συνθήκη στο γίνει ψευδής. Προϋπόθεση και στις δύο περιπτώσεις είναι να μην έχει διακοπεί η εκτέλεση με την εντολή break.

Παράδειγμα 1.15: Εντολή pass

```
# εμφανίζει όλους τους χαρακτήρες εκτός των αριθμών
str=input ("enter a string:")
for ch in str:
    if ch.isdigit(): pass
    else: print (ch,end="")
```

Στο Παράδειγμα 1.16 που ακολουθεί, υπολογίζουμε τις τετραγωνικές ρίζες μέχρι 10 αριθμών, εφόσον είναι θετικοί. Παρατηρήστε ότι κάνουμε import τη βιβλιοθήκη math, για να μπορούμε να χρησιμοποιήσουμε τη μέθοδο sqrt() που υπολογίζει την τετραγωνική ρίζα.

Παράδειγμα 1.16: Εντολή continue

```
#υπολογίζει την τετραγωνική ρίζα 10 αριθμών
#αν ο αριθμός ειναι αρνητικός, διαβάζει τον επόμενο
import math
for i in range(1,11):
    x = float(input("enter a number:"))
    if x < 0: continue; #ή και pass
    else:print("Square root is ",math.sqrt(x))
```

Στο Παράδειγμα 1.17 το else θα εκτελεστεί μόνο αν δεν πληκτρολογηθεί αρνητικός αριθμός. Αν δοθεί αρνητικός, ο βρόγχος τερματίζει με break και επομένως το else δεν εκτελείται.

Παράδειγμα 1.17: for .. else

```
import math
for i in range(1,11):
    x = float(input("enter a number:"))
    if x < 0: break;
    else:print("Square root is ",math.sqrt(x))
else: print("for ended normally, no negative numbers found")
```

Άσκηση 1.6: Γράψτε ένα πρόγραμμα που θα διαβάζει από το πληκτρολόγιο έναν ακέραιο αριθμό και θα εξετάζει αν διαιρείται ακριβώς με το άθροισμα των ψηφίων του (Υπόδειξη: μετατρέψτε τον αριθμό σε αλφαριθμητικό και εργαστείτε με κάθε ένα ψηφίο του).

Άσκηση 1.7: Γράψτε ένα πρόγραμμα που θα διαβάζει τρεις αριθμούς και θα εμφανίζει το μεγαλύτερο.

Άσκηση 1.8: Γράψτε ένα πρόγραμμα που θα διαβάζει αριθμούς και θα υπολογίζει το μέσο όρο τους. Το πλήθος των αριθμών θα το δίνει ο χρήστης στην αρχή.

Άσκηση 1.9: Γράψτε ένα πρόγραμμα που θα διαβάζει μια πρόταση (ένα αλφαριθμητικό) και έναν χαρακτήρα. Το πρόγραμμα θα μετρά πόσες φορές εμφανίζεται ο χαρακτήρας μέσα στο αλφαριθμητικό.

Άσκηση 1.10: Ένας δεσμός (link) στη γλώσσα HTML έχει τη μορφή
 κείμενο δεσμού
Γράψτε ένα πρόγραμμα που θα διαβάζει έναν τέτοιο δεσμό και θα εξαγάγει το URL.
Για παράδειγμα, αν η είσοδος είναι
ΕΚΔΔΑ
Το πρόγραμμα θα εμφανίζει: http://www.ekdd.gr

ΓΛΩΣΣΑΡΙ

Διερμηνευτής: ένα πρόγραμμα που παίρνει σαν είσοδο μια-μια εντολή ενός προγράμματος την ελέγχει, την μετατρέπει σε γλώσσα μηχανής και την εκτελεί.

Μεταγλωττιστής: ένα πρόγραμμα που δέχεται τον πηγαίο κώδικα ως είσοδο, τον ελέγχει και τον μετατρέπει σε εκτελέσιμη γλώσσα μηχανής.

Ψηφιοκώδικας (bytecode): ο ενδιάμεσος κώδικας που παράγουν πολλές σύγχρονες γλώσσες προγραμματισμού. Ο κώδικας απευθύνεται σε μια αφηρημένη υπλογιστική πλατφόρμα και εκτελείται στον τοπικό υπολογιστή μέσα από μια εικονική μηχανή.

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

Python documentation (2014). http://docs.python.org/3/. Προσπελάστηκε την 14-1-2014.

Swaroop, C.H. (2005). A Byte of Python. Ανακτήθηκε την 10-1-2014 από www.ibiblio.org

ΟΔΗΓΟΣ ΠΕΡΑΙΤΕΡΩ ΜΕΛΕΤΗΣ

- Στη διεύθυνση http://www.python-course.eu/python3_course.php περιέχεται ένα πλήρες εκπαιδευτικό υλικό αναφορικά με τη γλώσσα Python 3. Μπορείτε να διαβάσετε τα θέματα «Data Types and Variables» σχετικό με τους τύπους δεδομένων της γλώσσας και «Formatted output with string modulo and the format method» με περιεχόμενο τη μορφοποίηση με τη μέθοδο format.
- Στη διεύθυνση http://ebeab.com/2012/10/10/python-string-format/ διαβάστε το άρθρο «Python String Format» με πληροφορίες και παραδείγματα μορφοποίησης.
- Στην διεύθυνση https://swaroopch.com/book/python/ θα βρείτε ένα πλήρες online βιβλίο για την Python. Διαβάστε τις ενότητες «Installation» με πληροφορίες για την εγκατάσταση της γλώσσας σε διάφορες πλατφόρμες, «Basics» με τα βασικά χαρακτηριστικά της γλώσσας, «Operators and Expressions» και «Control Flow» με στοιχεία για τελεστές/παραστάσεις και εντολές ελέγχου ροής αντίστοιχα.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

2 ΠΡΟΗΓΜΕΝΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΗΣ ΓΛΩΣΣΑΣ

Σκοπός

Σκοπός του κεφαλαίου είναι η επίδειξη των βασικών δομών δεδομένων που παρέχει η Python για την αναπαράσταση των πληροφοριών σε μια εφαρμογή. Επιπλέον θα δούμε το τρόπο κατασκευής των συναρτήσων με σκοπό να επαναχρήσιμοποιήσουμε ένα σύνολο εντολών σε πολλά σημεία του προγράμματος μας. Επίσης θα παρουσιάσουμε τις βασικές έννοιες του αντικειμενοστρεφούς προγραμματισμού και θα γίνει επίδειξη μέσω εφαρμογής τόσο η κατασκευή μιας κλάσης όσο και εν συνεχεία η δημιουργία αντικειμένων της. Ακόμη θα παρουσιαστούν πιο προχωρημένες τεχνικές για τη καλύτερη διαχείριση της εφαρμογής, όπως είναι η υπερφόρτωση συναρτήσεων και τελεστών καθώς και η κληρονομικότητα των κλάσεων.

Στο τέλος του κεφαλαίου θα γίνει ανάπτυξη ενός αρθρώματος που περιέχει συναρτήσεις και θα επιδειχθεί πως γίνεται η εισαγωγή και η χρήση τους σε ένα σενάριο. Τέλος θα γίνει επίδειξη της λειτουργίας του αρθρώματος tkInter μέσω του οποίου μπορούμε να κατασκευάσουμε γραφικές διεπαφές.

Προσδοκώμενα αποτελέσματα

Όταν θα έχετε μελετήσει το κεφάλαιο αυτό θα μπορείτε να:

- γνωρίζετε τι είναι η λίστα και τις βασικές λειτουργίες διαχείρισης των στοιχείων της,
- γνωρίζετε τι είναι το λεξικό και τις βασικές λειτουργίες διαχείρισης των στοιχείων του,
- γνωρίζετε τι είναι η πλειάδα και τις διαφορές που έχει από τη λίστα,
- γνωρίζετε τι είναι το σύνολο και τις βασικές λειτουργίες διαχείρισης των στοιχείων του, καθώς και τις πράξεις της ένωσης, τομής και διαφοράς μεταξύ δύο συνόλων,
- κατασκευάζετε συναρτήσεις για την επίλυση βασικών προβλημάτων,
- αναγνωρίζετε την χρησιμότητα των προεπιλεγμένων τιμών των ορισμάτων μιας συνάρτησης,
- κατανοείτε τη βασική ορολογία τους αντικειμενοστραφούς προγραμματισμού,

- κατασκευάζετε κλάσεις για τη περιγραφή απλών οντοτήτων ενός προβλήματος,
- κατασκευάζετε και να εισαγάγετε λειτουργικότητα από ένα άρθρωμα,
- κατανοείτε τη χρησιμότητα των αρθρωμάτων στη κατασκευή σύνθετων εφαρμογών

Έννοιες-Κλειδιά

- Δομές δεδομένων
- Λίστα
- Πλειάδα
- Λεξικό
- Σύνολο
- Συνάρτηση
- Κλάση
- Αντικείμενο
- Κληρονομικότητα
- Άρθρωμα

Εισαγωγικές Παρατηρήσεις

Η Python διαθέτει πλήθος από τύπους δεδομένων μέσω των οποίων μπορούμε να υποστηρίξουμε την απόθηκευση δεδομένων ώστε εν συνεχεία να μπορούμε να επιτελέσουμε σε αυτό το σύνολο των τιμών, εργασίες όπως : την εύρεση μιας τιμής, το διαμερισμό του συνόλου, τη συνένωση διαφορετικών συνόλων, τον έλεγχο αν μια τιμή είναι μέλος του συνόλου κ.λπ. Η λίστα αποτελεί είναι μια από τις πιο ευέλικτες δομές δεδομένων που διαθέτει η Python και την υποστηρίζει με πλήθος τελεστών.

2.1 Δομές Δεδομένων - Λίστες

Η λίστα είναι μια διατεταγμένη συλλογή από αντικείμενα που περιλαμβάνονται σε αγκύλες [ ]. Τα περιεχόμενα μιας λίστας μπορεί να είναι διαφορετικών τύπων. Για παράδειγμα η Lista1 περιλαμβάνει και αλφαριθμητικά και ακεραίους και πραγματικούς αριθμούς.

```
Lista1 = ['eggs', 'spam', 'onions', 2, 3, 2.4]
print (Lista1)
```

Τα περιεχόμενα της λίστας είναι

```
['eggs', 'spam', 'onions', 2, 3, 2.4]
```

Μια λίστα μπορεί να περιέχει σαν τιμή μια άλλη λίστα :

```
Lista2= [Lista1, 'more eggs', 1200]
print (Lista2)
```

Τα περιεχόμενα της δεύτερης λίστας είναι

```
[['eggs', 'spam', 'onions', 2, 3, 2.4], 'more eggs', 1200]
```

Η Lista2 περιέχει σαν υπό-λίστα τη Lista1.

Η συνάρτηση split ενός αλφαριθμητικού δημιουργεί μια λίστα αποτελούμενη από τις λέξεις του αλφαριθμητικού.

```
str = "good morning my dear friend"
t = str.split ()
print (t)
```

Τα περιεχόμενα της λίστας t είναι :

```
['good', 'morning', 'my', 'dear', 'friend']
```

Το προκαθορισμένο διαχωριστικό για το χωρισμό των επιμέρους τμημάτων του αλφαριθμητικού είναι το κενό, αλλά στη split μπορούμε να ορίσουμε και μια ακολουθία χαρακτήρων σαν διαχωριστικό.

```
str = "goodXXmorningXXmyXXdearXXfriend"
t = str.split ("XX")
print (t)
```

Τα περιεχόμενα της λίστας t είναι :

```
['good', 'morning', 'my', 'dear', 'friend']
```

Οι λίστες εμπεριέχουν τα στοιχεία τους σε διάταξη, δηλαδή έχει σημασία η σειρά αναγραφής των στοιχείων τους.

```
lista=["Peter","Georgia", "George" ]
listal=["Peter","Georgia", "George"]
lista2=["Georgia", "George","Peter"]

if lista==listal:
   print("equal")
else:
   print("unequal")

if lista==lista2:
  print("equal")
else:
  print("unequal")
```

Η εκτέλεση του σεναρίου δημιουργεί τα παρακάτω αποτελέσματα :

```
equal
unequal
```

Οι λίστες lista και lista1έχουν το ίδιο πλήθος και την ίδια διάταξη τα στοιχεία τους, οπότε εκτυπώνεται το μήνυμα equal, ενώ οι λίστες lista και lista2 έχουν το ίδιο πλήθος αλλά τα στοιχεία τους δεν έχουν την ίδια διάταξη, οπότε εκτυπώνεται το μήνυμα unequal.

2.1.1 Τελεστές πράξεων σε Λίστες

Ο τελεστής + ανάμεσα σε δύο λίστες δημιουργεί μια νέα λίστα, όπου τα περιεχόμενα της, είναι όλες οι επιμέρους τιμές των δύο λιστών. Επίσης μπορούμε να ορίσουμε λίστες που περιέχουν άλλες λίστες καθώς επίσης να πολλαπλασιάσουμε λίστα με αριθμό ώστε να πολλαπλασιάσουμε τις διατάξεις εμφάνισης των τιμών της λίστας.

```
1  Lista1=["George","Maria"]
2  Lista2=["Nick","Lina"]
3  all=Lista1+Lista2
4  print(all)
5  all=[Lista1,Lista2]
6  print(all)
7  Lista1+=['1']
8  print (Lista1)
9  Lista2+='1'
10 print (Lista2)
11 Lista2=Lista2*2
12 print(Lista2)
```

Εικόνα 2.1: Πράξεις ανάμεσα σε λίστες

Η εκτέλεση του ανωτέρω σεναρίου δημιουργεί τα παρακάτω αποτελέσματα :

```
['George', 'Maria', 'Nick', 'Lina']
[['George', 'Maria'], ['Nick', 'Lina']]
['George', 'Maria', '1']
['Nick', 'Lina', '1']
['Nick', 'Lina', '1', 'Nick', 'Lina', '1']
```

2.1.2 Η συνάρτηση range

Ο μετρητή σε μια επανάληψη for μπορεί να παίρνει τιμές από μια καθορισμένη λίστα ή μπορούμε να χρησιμοποιήσουμε τη συνάρτηση range για να δημιουργήσουμε τη λίστα με τις τιμές.

Πίνακας 2.1: Παράδειγμα συνάρτησης range

Το σενάριο showrange.py	Το αποτέλεσμα

`for i in [0,1,2]:`
`  print(i)`
`print()`
`for i in range(3):`
`  print(i)`
	0
	1
	2
	
	0
	1
	2

Η κλήση της συνάρτησης range μπορεί να γίνει με ένα όρισμα, όπως στο ανωτέρω σενάριο, ώστε να δημιουργήσει μια λίστα που αποτελείται από την ακολουθία των αριθμών από 0 μέχρι και το όρισμα -1, με βήμα 1. Μπορεί να πάρει και δύο παραμέτρους range(m, n) ώστε να δημιουργήσει μια ακολουθία από m μέχρι και το n - 1, με βήμα 1. Τέλος μπορεί να γίνει κλήση με τρεις παραμέτρους range(m, n, κ) ώστε να δημιουργήσει μια λίστα από την ακολουθία των αριθμών από m μέχρι και το n - 1, με βήμα κ.

2.1.3 Διάσχιση των στοιχείων μιας λίστας

Τα στοιχεία της λίστας είναι σε διάταξη και μπορούμε να τα διασχίσουμε με χρήση της δομής επανάληψης for :

α) με χρήση δείκτη στη διάταξη των στοιχείων

Το σενάριο elementsbyIndex.py	Το αποτέλεσμα

`lista = ['eggs', 3, 2.4]`
`for i in range(len(lista)):`
`  print(lista[i])`
	`eggs`
	`3`
	`2.4`

β) με απευθείας αναφορά στα στοιχεία της λίστας

Το σενάριο elementsbyRef.py	Το αποτέλεσμα

`lista = ['eggs', 3, 2.4]`
`for i in lista:`
`  print(i)`
	`eggs`
	`3`
	`2.4`

Η συνάρτηση len με παράμετρο μια λίστα επιστρέφει το πλήθος των στοιχείων της λίστας.

2.1.4 Λειτουργίες με χρήση δεικτών σε λίστες

Οι δείκτες που είδαμε στα αλφαριθμητικά, έχουν την ανάλογη λειτουργικότητα και στην περίπτωση των Λιστών.

Πίνακας 2.2: Χρήση δεικτών σε λίστες

Το σενάριο ListIndexes.py	Το αποτέλεσμα της εκτέλεσης

`1 group=['George','Maria','Sofia']`
`2 print(group[0:2])`
`3 print(group[:-1])`
`4 print(group[-2])`
`5 print(group[-1:])`
`6 print(group[2])`
	`['George', 'Maria']`
	`['George', 'Maria']`
	`Maria`
	`['Sofia']`
	`Sofia`

Στη γραμμή 2 εμφανίζονται τα δύο πρώτα στοιχεία της λίστας, στις θέσεις 0 και 1. Στη 3 γραμμή εμφανίζονται όλα τα στοιχεία εκτός από το τελευταίο της λίστας. Στη 4 γραμμή εμφανίζει το προτελευταίο στοιχείο της λίστας ενώ στη επόμενη όλα τα στοιχεία από το προτελευταίο μέχρι το τέλος. Στη τελευταία γραμμή εμφανίζει το στοιχείο στη δεύτερη θέση, που είναι το τελευταίο στη λίστα. Εκτός από τις γραμμές 4 και 6 όπου το αποτέλεσμα είναι αλφαριθμητικό, σε όλες τις άλλες περιπτώσεις το αποτέλεσμα είναι λίστα.

Η αναφορά στα στοιχεία της λίστας μπορεί να γίνεται και με αρνητικούς αριθμούς οπότε ισχύουν τα ακόλουθα :

Lista[-n] =Lista [len[Lista]-n]

Η ανωτέρω εντολή επιστρέφει το στοιχείο της λίστας το οποίος βρίσκεται σε θέση n από το τέλος.

Lista[m:n] τα m, n μπορεί να είναι και αρνητικοί:

Η ανωτέρω εντολή επιστρέφει μια λίστα, όπου.

- Ο πρώτος αριθμός είναι η εναρκτήρια θέση του πρώτου στοιχείου,
- Ο δεύτερος αριθμός είναι η θέση του στοιχείου που δεν θα συμπεριληφθεί στη λίστα,
- Διαβάζουμε τη λίστα από αριστερά προς τα δεξιά,
- Τα ενδιάμεσα στοιχεία είναι η επιστρεφόμενη τιμή.

Παράδειγμα 2.1: Δείκτες σε λίστα

Έστω ότι εκτελείται ο κώδικας:

Lista= [1, 2, "aa", "bb"]

print (Lista[1:-2]

Το αποτέλεσμα είναι μια λίστα που περιλαμβάνει όλα τα στοιχεία από τη θέση 1 μέχρι και πριν τη θέση -2 από το τέλος, δηλαδή είναι η λίστα [2].

2.1.5 Αναζήτηση σε Λίστα

Η μέθοδος index μιας λίστας, με όρισμα ένα στοιχείο της λίστας επιστρέφει τη θέση του στοιχείου στη λίστα. Αν δεν υπάρχει το στοιχείο στη λίστα τότε εμφανίζεται μήνυμα σφάλματος.

Πίνακας 2.3: Αναζήτηση σε λίστα

Το σενάριο ListIndex.py	Το αποτέλεσμα της εκτέλεσης
1 Lista=["best","test","one"] 2 goal=input("enter element:") 3 if goal in Lista: 4 print(Lista.index(goal)) 5 else: 6 print("not found")	>>> enter element:one 2 >>>

Το στοιχείο "one" βρίσκεται στη δεύτερη θέση του πίνακα. Για την αποφυγή σφαλμάτων πριν από τη χρήση της index κάνουμε έλεγχο με το τελεστή in αν το στοιχείο προς αναζήτηση, υπάρχει στη λίστα ή όχι.

2.1.6 Βασικές Μέθοδοι σε λίστες - append,extend,remove,pop,sort,reverse

Η μέθοδος append μιας λίστας με όρισμα μια τιμή, προσθέτει στο τέλος της λίστας τη τιμή σαν νέο στοιχείο της. Η μέθοδος extend μιας λίστας με όρισμα μια λίστα, συνενώνει τις δύο λίστες.

Πίνακας 2.4: Συνένωση λιστών

Το σενάριο appendextend.py	Το αποτέλεσμα της εκτέλεσης
1 Listal=["one","two"] 2 Listal.append("three") 3 print(Listal) 4 Lista2=["four","five"] 5 Listal.extend(Lista2) 6 print(Listal)	>>> ['one', 'two', 'three'] ['one', 'two', 'three', 'four', 'five'] >>>

Στη γραμμή 2 προσθέτουμε στο τέλος της λίστας τη τιμή "three" ενώ στη γραμμή 5 συνενώνουμε στη πρώτη λίστα τα στοιχεία της δεύτερης.

Η μέθοδος remove αφαιρεί τη τιμή του ορίσματος από τη λίστα. Αν δεν υπάρχει το στοιχείο στη λίστα τότε εμφανίζεται μήνυμα σφάλματος. Η μέθοδος pop με παράμετρο έναν ακέραιο αριθμό, αφαιρεί και επιστρέφει το στοιχείο στην θέση του ακεραίου αριθμού. Η pop χωρίς όρισμα αφαιρεί και επιστρέφει τη τελευταία τιμή της λίστας.

Πίνακας 2.5: Αφαίρεση στοιχείων από λίστα

Το σενάριο removepop.py	Το αποτέλεσμα της εκτέλεσης
1 Lista1=["one","two", "three", "four","five"] 2 Lista1.remove("three") 3 print(Lista1) 4 popvalue = Lista1.pop(1) 5 print(popvalue) 6 print(Lista1) 7 popvalue = Lista1.pop() 8 print(Lista1) 9	>>> ['one', 'two', 'four', 'five'] two ['one', 'four', 'five'] ['one', 'four'] >>>

Στη γραμμή 2 αφαιρούμε τη τιμή "three" από τη λίστα ενώ στη γραμμή 4 αφαιρούμε και αποδίδουμε στη μεταβλητή popvalue τη τιμή two που είναι στη θέση 1 της λίστας. Τέλος στη γραμμή 7, αφαιρούμε και αποδίδουμε στη μεταβλητή popvalue τη τιμή "five" που είναι στη τελευταία θέση της λίστας.

Πίνακας 2.6: Ταξινόμηση Λίστας

Το σενάριο sortreverse.py	Το αποτέλεσμα της εκτέλεσης
1 Cities=["cairo","athens", "rome", "paris"] 2 Cities.sort() 3 print(Cities) 4 Cities.reverse() 5 print(Cities)	>>> ['athens', 'cairo', 'paris', 'rome'] ['rome', 'paris', 'cairo', 'athens'] >>>

Η μέθοδος sort ταξινομεί τα στοιχεία της λίστας ενώ η reverse, αντιστρέφει τη διάταξη των στοιχείων της λίστας.

Άσκηση 2.1 Δημιουργείστε μια εφαρμογή για τη καταχώρηση των ονομάτων των υπαλλήλων μιας εταιρίας σε μια λίστα, η οποία εμφανίζει επαναληπτικά ένα μενού με τις εξής επιλογές: 1) Εμφάνιση 2) Προσθήκη 3) Διαγραφή 4)Έξοδος. Πατώντας 1 θα γίνεται εμφάνιση όλων των στοιχείων της λίστας, δηλαδή θα εμφανίζονται όλα τα ονόματα. Πατώντας το 2 θα γίνετε ερώτηση για το όνομα του υπαλλήλου (μόνο το μικρό όνομα) και εν συνεχεία θα προστίθεται στη λίστα. Πατώντας το 3 θα γίνεται ερώτηση για το όνομα, εν συνεχεία θα γίνεται έλεγχος αν υπάρχει και θα εμφανίζεται αντίστοιχο μήνυμα σε περίπτωση απουσίας του ονόματος από τη λίστα και μετά θα γίνεται η διαγραφή. Με το 4 θα σταματά η επανάληψη και θα τερματίζει η εφαρμογή.

2.2 Συναρτήσεις

Η δημιουργία των συναρτήσεων διευκολύνει το προγραμματισμό των εφαρμογών, επιτρέποντας την επαναχρησιμοποίηση μιας λειτουργίας, η οποία μπορεί να αποτελείται από μια ή μερικές εκατοντάδες γραμμές κώδικα, απλά χρησιμοποιώντας το όνομα της με τις κατάλληλες παραμέτρους. Για τη δημιουργία μιας συνάρτησης, ξεκινάμε με τη λέξη def, ακολουθεί το όνομα της συνάρτησης (δικής μας επιλογής) και τέλος ζεύγος παρενθέσεων με ή χωρίς ορίσματα. Το σώμα της συνάρτησης καθορίζεται από την εσοχή (indent).

Πίνακας 2.7: Σύνταξη συναρτήσεων

Τρόπος σύνταξης των συναρτήσεων	
def <όνομα συνάρτησης> (<ορισματα -ή κενό>) *εντολή-1* *εντολή-2* *<κώδικας εκτός συνάρτησης>*	*def <όνομα συνάρτησης> (<ορισματα-ή κενό>)* *εντολή-1* *εντολή-2* *return <μεταβλητή>* *<κώδικας εκτός <συνάρτησης>*

Το όνομα μιας συνάρτησης δεν πρέπει να ίδιο με κάποια δεσμευμένη λέξη της Python και επίσης στις παρενθέσεις μετά το όνομα, μπορεί να υπάρχουν ή όχι

ορίσματα, τα οποία μπορεί να έχουν προεπιλεγμένες τιμές. Επίσης μια συνάρτηση μπορεί να επιστέφει ή όχι μια τιμή.

2.2.1 Παραδείγματα συναρτήσεων: Αναζήτηση σε λίστα

Η δημιουργία των συναρτήσεων παρομοιάζεται με τη κατασκευή μικρών ανεξάρτητων προγραμμάτων, των οποίων ο συνδυασμός τους βοηθά στη σύνταξη μεγαλύτερων προγραμμάτων.

Το σενάριο searchlist.py

```
1  def find(item, list):
2      if item in list:
3          return True
4      else:
5          return False
6
7  def printresult(found):
8      if found:
9          print("search item found")
10     elif not found:
11         print("search item not found")
12     else :
13         print("Error in value option")
14
15 a='test'
16 lista=['ena','dyo' , 'tria', 'test', 'pente']
17 result=find(a,lista)
18 printresult(result)
19 a='notexist'
20 lista=['ena','dyo', 'tria', 'test', 'pente']
21 printresult(find(a,lista))
```

Στο σενάριο searchlist.py κατασκευάζονται δυο συναρτήσεις για την αναζήτηση μια τιμής σε μια λίστα. Η συνάρτηση find, γραμμές 1 έως 5, δέχεται δύο ορίσματα, τη τιμή που αναζητούμε και τη λίστα με τις τιμές, και επιστρέφει μια λογική τιμή ανάλογα με

το αποτέλεσμα της έρευνας. Στις γραμμές 7 έως 13 κατασκευάζουμε τη συνάρτηση printresult, η οποία δέχεται μια λογική τιμή που αντιστοιχεί στο αποτέλεσμα της αναζήτησης μια τιμής σε μια λίστα και εκτυπώνει ένα μήνυμα. Η δεύτερη συνάρτηση δεν επιστρέφει τιμή. Στις γραμμές 15 έως 20 παρουσιάζονται παραδείγματα χρήσης των δύο συναρτήσεων, τη μια φορά με υπαρκτή τιμή και τη δεύτερη με ανύπαρκτη τιμή.

2.2.2 Προεπιλεγμένες τιμές σε παραμέτρους συναρτήσεων

Στις παραμέτρους των συναρτήσεων μπορούμε να δίνουμε προεπιλεγμένες τιμές. Αν κατά την κλήση της συνάρτησης δεν δώσουμε καθόλου τα συγκεκριμένα ορίσματα, τότε χρησιμοποιούνται οι προεπιλεγμένες τιμές. Σε περίπτωση περισσότερων τους ενός ορισμάτων, οι αναθέσεις ζευγαρώνονται από αριστερά προς τα δεξιά, εκτός και αν καθορίσουμε συγκεκριμένη παράμετρο με το όνομα της.

Το σενάριο calculateVat.py

```
1  def ypologismos_fpa(timi_monadas, pososto=23, posotita=1):
2      return posotita* timi_monadas * pososto / 100
3
4  fpa= ypologismos_fpa(1000)
5  print(fpa)
6  fpa = ypologismos_fpa(1000,8)
7  print(fpa)
8  fpa = ypologismos_fpa(1000,8,2)
9  print(fpa)
10 fpa = ypologismos_fpa(1000,posotita=2)
11 print(fpa)
```

Το αποτέλεσμα της εκτέλεσης του book.py είναι

```
>>>
230.0
80.0
160.0
460.0
>>>
```

Η συνάρτηση ypologismos_fpa δέχεται τρία ορίσματα, τη τιμή μονάδας (timi_monadas), το ποσοστό του φόρου (pososto) με προκαθορισμένη τιμή το 23, καθώς και τη ποσότητα (posotita) με προκαθορισμένη τιμή τη 1. Στη γραμμή 4,

γίνεται χρήση της συνάρτησης όπου αγνοούνται τα ορίσματα του ποσοστού του φόρου καθώς και της ποσότητας, οπότε θα χρησιμοποιηθούν οι προεπιλεγμένες τιμές 23 και 1 αντίστοιχα. Στη γραμμή 10, γίνεται χρήση της συνάρτησης όπου αγνοείται το ποσοστό φόρου αλλά είναι απαραίτητο για να αποδοθεί η τιμή 2 στη ποσότητα να δηλώνεται ρητά η απόδοση της τιμής στο συγκεκριμένο όρισμα.

Το σενάριο retainValues.py

```
1  def lista_omadas(name, team=[]):
2      team.append(name)
3      return team
4
5  omada=lista_omadas("Peter")
6  print(omada)
7  omada=lista_omadas("John")
8  print(omada)
```

Το αποτέλεσμα της εκτέλεσης του retainValues.py είναι

```
>>>
['Peter']
['Peter', 'John']
>>>
```

Οι αναθέσεις ορισμάτων σε παραμέτρους γίνονται μια μόνο φορά στην πρώτη κλήση. Οπότε αν το όρισμα είναι αντικείμενο που μπορεί να πολλαπλασιαστεί, π.χ. μια λίστα, η αρχική τιμή δίνεται μόνο την πρώτη φορά και η τιμή δεν χάνεται στις διαδοχικές κλήσεις.

Άσκηση 2.2 Να δημιουργήσετε μια συνάρτηση το όνομα distance η οποία δέχεται 4 αριθμούς που αντιστοιχούν στα σημεία (x1, y1) και (x2, y2) και υπολογίζει τη μεταξύ τους απόσταση που δίνεται από τη παρακάτω σχέση :

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Σε περίπτωση που δεν δίνεται το σημείο x1,y1 τότε να θεωρείται ότι αντιστοιχεί στο σημείο 0,0.

2.3 Δομές Δεδομένων - Λεξικά, Πλειάδες, Σύνολα

2.3.1 Λεξικά (Dictionaries ή Hashes)

Το λεξικό είναι ένα μη διατεταγμένο σύνολο ζευγαριών που το καθένα αποτελείται από Κλειδί και αντίστοιχη Τιμή, και περικλείεται από άγκιστρα. Η σύνταξη του είναι :

{<key-i>:<value-i>,..., <key-n>:<value-n>}

Σε ένα λεξικό, το κάθε κλειδί εμφανίζεται μια μόνο φορά και η πρόσβαση στην αντίστοιχη τιμή γίνεται με τη χρήση του κλειδιού μέσα σε [].

Το σενάριο showLexico.py	
1	player = {"name":"Nikos","age":20,"levels":[1,2,3] }
2	print(player)
3	print(player["name"])
4	print(player["age"])
5	print(player["levels"])
6	print(player["levels"][0])

Το αποτέλεσμα της εκτέλεσης του showLexico.py είναι

```
>>>
{'age': 20, 'name': 'Nikos', 'levels': [1, 2, 3]}
Nikos
20
[1, 2, 3]
1
>>>
```

Στην πρώτη γραμμή του σεναρίου δηλώνουμε το λεξικό player και στις επόμενες εκτελούμε διάφορες εντολές εμφάνισης δεδομένων του.

2.3.2 Τροποποίηση λεξικού

Σε ένα λεξικό μπορούμε να προσθέτουμε και νέα ζεύγη κλειδί - τιμή και κάθε φορά δημιουργούμε νέο κλειδί με την αντίστοιχη τιμή εκτός αν ήδη υπάρχει το κλειδί, οπότε τροποποιείται η τιμή του.

Το σενάριο modifyLexico.py	
1	`player = {"name":"Nikos","age":20,"levels":[1,2,3]}`
2	`print(player)`
3	`player["status"]="active"`

4	`print(player)`
5	`player["name"]="George"`
6	`print(player)`
7	`del player["name"]`
8	`print(player)`
9	`player.clear()`
10	`print(player)`

Το αποτέλεσμα της εκτέλεσης του modifyLexico.py είναι

```
>>>
{'name': 'Nikos', 'age': 20, 'levels': [1, 2, 3]}
{'status': 'active', 'name': 'Nikos', 'age': 20, 'levels': [1, 2, 3]}
{'status': 'active', 'name': 'George', 'age': 20, 'levels': [1, 2, 3]}
{'status': 'active', 'age': 20, 'levels': [1, 2, 3]}
{}
>>>
```

Στη γραμμή 3 κάνουμε αναφορά στο κλειδί "status" με τιμή "active" και επειδή δεν υπάρχει το κλειδί στο λεξικό, τότε δημιουργείται νέο ζεύγος κλειδί-τιμή ("status"-"active"). Στη γραμμή 5 κάνουμε αναφορά στο κλειδί "name" που υπάρχει στο λεξικό και έχει τιμή Nikos, και σε αυτή την περίπτωση τροποποιείται η τιμή Nikos σε George. Στη γραμμή 7 διαγράφουμε από το λεξικό το ζεύγος κλειδί-τιμή που έχει σαν κλειδί τη "name" ενώ στη γραμμή 9 διαγράφουμε όλα τα περιεχόμενα του λεξικού.

2.3.3 Αναζήτηση σε λεξικό

Ο τελεστής in μας επιτρέπει να ελέγξουμε την ύπαρξη ενός κλειδιού μέσα σε ένα λεξικό. Στο παρακάτω σενάριο δηλώνουμε ένα Λεξικό και στη συνέχεια ψάχνουμε σύμφωνα με το κλειδί που θα δώσει ο χρήστης.

Το σενάριο searchLexico.py	
1	`player = {"name" : "Nikos", "age" : 20, "levels" : [1, 2, 3] }`
2	`key = input("enter key to search for:")`
3	
4	`if key in player:`
5	`    print(player[key])`
6	`else:`
7	`    print("key not found")`

Το αποτέλεσμα της εκτέλεσης του searchLexico.py είναι

```
>>>
enter key to search for:name
Nikos
>>>
```

Στη γραμμή 4 γίνεται έλεγχος αν το κλειδί που έχει πληκτρολογήσει ο χρήστης στη γραμμή 2 υπάρχει στη λίστα, και αν ναι τότε εμφανίζει τη τιμή του διαφορετικά εμφανίζει το μήνυμα ότι το κλειδί δεν βρέθηκε.

Άσκηση 2.3 Δημιουργείστε μια εφαρμογή για τη καταχώρηση των ποσοτήτων των προϊόντων ενός super market σε ένα λεξικό. Το μενού επιλογών εμφανίζεται επαναληπτικά και ο χρήστης μπορεί να πατήσει : 1) Εμφάνιση 2) Τροποποίηση 3) Διαγραφή 4)Έξοδος. Πατώντας 1 θα γίνεται εμφάνιση όλων των στοιχείων του λεξικού, δηλαδή θα εμφανίζεται το όνομα του προϊόντος και η ποσότητα του. Πατώντας το 2 θα γίνετε ερώτηση για το όνομα του προϊόντος και εν συνεχεία θα γίνεται ανάγνωση της ποσότητας, η οποία μπορεί να είναι και αρνητική. Αν δεν υπάρχει το προϊόν τότε θα προστίθεται στο λεξικό, διαφορετικά θα τροποποιείται ανάλογα η ποσότητα του. Σε περίπτωση μη ύπαρξης και αρνητικής τιμής θα εμφανίζετε μήνυμα σφάλματος. Πατώντας το 3 θα γίνεται ερώτηση για το όνομα του προϊόντος, εν συνεχεία θα γίνεται έλεγχος αν υπάρχει και θα εμφανίζεται αντίστοιχο μήνυμα σε περίπτωση απουσίας του ονόματος από το λεξικό και μετά θα γίνεται η διαγραφή. Με το 4 θα σταματά η επανάληψη και θα τερματίζει η εφαρμογή.

2.3.4 Πλειάδα (Tuple)

Η Πλειάδα είναι μια Λίστα που δεν μπορεί να τροποποιηθεί, δηλαδή να αλλάξει δηλαδή τις τιμές της. Δημιουργείται κατά τον ίδιο τρόπο όπως και η Λίστα αλλά περικλείεται σε παρενθέσεις. Στη Πλειάδα τα στοιχεία είναι σε διάταξη και μπορούν να εφαρμοστούν οι μέθοδοι των λιστών που δεν τις αλλάζουν, π.χ. δείκτες, find(), κ.λπ., αλλά όχι αυτές που τις αλλάζουν, π.χ. insert, append, sort, pop, κ.λπ.

Το σενάριο showTuples.py	Το αποτέλεσμα της εκτέλεσης

```
1  tup1 = ("one","two","three")
2  tup2=("one","two","three")
3  if tup1 == tup2:
4      print("tuples are equal")
5  else:
6      print("tuples not equal")
7
8  print(tup1.index("two"))
9  #tup1.append("four")
```

```
>>>
tuples are equal
1
>>>
```

Η σύγκριση μεταξύ τους, γραμμή 3, καθώς και η αναζήτηση της θέσης μιας τιμής, γραμμή 8, δεν μεταβάλουν τις πλειάδες. Αν η γραμμή 9 δεν ήταν σχόλιο τότε θα δημιουργούσε σφάλμα αφού δεν μπορούμε να τροποποιήσουμε τη πλειάδα. Η επιλογή για την αναπαράσταση ενός πλήθους τιμών σε πλειάδα ή λίστα εξαρτάται αν χρειάζεται να τροποποιήσουμε τις τιμές ή τη διάταξη τους. Σε περίπτωση που δεν υπάρχει τέτοια ανάγκη τότε η πλειάδα υπερτερεί έναντι της λίστας αφού προσφέρει προστασία από "τυχαία" επέμβαση στα δεδομένα καθώς και σε ταχύτητα εκτέλεσης.

Το σενάριο List2Tuble.py

```
1  tup1 = ("one","two","three")
2  list1=("monday","friday")
3
4  tup2List = list(tup1)
5  print(tup2List)
6
7  list2tup = tuple(list1)
8  print(list2tup)
```

Το αποτέλεσμα της εκτέλεσης

```
>>>
['one', 'two', 'three']
('monday', 'friday')
>>>
```

Πάντως σε οποιαδήποτε περίπτωση είναι εύκολη υπόθεση η μετατροπή ανάμεσα στις δύο αναπαραστάσεις αφού, η συνάρτηση list δέχεται σαν όρισμα μια πλειάδα και επιστρέφει μια λίστα με τις τιμές της πλειάδας, και η συνάρτηση tuple δέχεται σαν όρισμα μια λίστα και επιστρέφει μια πλειάδα με τις τιμές της λίστας.

2.3.5 Σύνολο

Το σύνολο είναι ένα μη διατεταγμένο πλήθος τιμών, στο οποίο κάθε τιμή μπορεί να συμμετέχει μια φορά. Το σύνολο μπορεί να συμμετέχει σε πράξεις όπως, τομή, ένωση και διαφορά. Οι τιμές του συνόλου περικλείονται σε άγκιστρα.

Το σενάριο showset.py	Το αποτέλεσμα της εκτέλεσης

```
1  set1 = {"one", "two"}
2  print(set1)
3  set1.add("two")
4  set1.update({"three","four"})
5  print(set1)
6  set2=set()
7  set2.add("monday")
8  print(set2)
```

```
>>>
{'one', 'two'}
{'one', 'three', 'two', 'four'}
{'monday'}
>>>
```

Στη γραμμή 1 δημιουργούμε ένα σύνολο με δύο τιμές. Στη γραμμή 3 προσθέτουμε μια ακόμη τιμή, αλλά αγνοείται επειδή ήδη υπάρχει στο σύνολο. Στη γραμμή 4 προσθέτουμε περισσότερες τιμές. Στη γραμμή 6 δημιουργούμε ένα νέο κενό σύνολο. Αν σε περίπτωση γράφουμε set2 = {} τότε δεν θα δημιουργούσαμε σύνολο αλλά λεξικό. Για τη διαγραφή ενός στοιχείου από το σύνολο μπορούμε να χρησιμοποιήσουμε είτε τη discard είτε τη remove, όπου και οι δύο δέχονται σαν όρισμα το στοιχείο που θα αφαιρέσουν από το σύνολο.

Το σενάριο removefromset.py	Το αποτέλεσμα της εκτέλεσης

```
1  set1 = {"one", "two", "three"}
2  set1.discard("four");
3  set1.discard("two");
4  print(set1)
5
6  if("three" in set1):
7      set1.remove("three")
8  else:
9      print("Not found")
10
11 print(set1)
```

```
>>>
{'one', 'three'}
{'one'}
>>>
```

Η διαφορά είναι ότι σε περίπτωση που δεν υπάρχει το στοιχείο, η remove εμφανίζει μήνυμα λάθους κατά την εκτέλεση ενώ η discard αγνοεί το σφάλμα. Για αυτό το λόγο θα πρέπει να ελέγχουμε με το τελεστή in αν υπάρχει το στοιχείο που θέλουμε να αφαιρέσουμε μέσα στο σύνολο.

Το σενάριο popandclear.py	Το αποτέλεσμα της εκτέλεσης

```
1  set1 = {"one", "two", "three"}
2  if(len(set1) > 1):
3      x = set1.pop()
4  print("x = ", x)
5  print(set1)
6  set1.clear()
7  print(set1)
```

```
>>>
x =  three
{'one', 'two'}
set()
>>>
```

Η μέθοδος pop αφαιρεί και επιστρέφει ένα τυχαίο στοιχείο του συνόλου. Η χρήση της pop σε κενό σύνολο δημιουργεί σφάλμα κατά την εκτέλεση, για αυτό και πρέπει να ελέγχουμε το πλήθος των στοιχείων του συνόλου ότι είναι μεγαλύτερο από 1. Η μέθοδος clear διαγράφει όλα τα στοιχεία του συνόλου. Η χρήση της clear σε κενό σύνολο δεν δημιουργεί πρόβλημα.

Το σενάριο mathset.py	Το αποτέλεσμα της εκτέλεσης

```
1  set1 = {"one", "two"}
2  set2 = {"three", "two"}
3  set3 = set1.union(set2)
4  print(set3)
5
6  set4 = {"one", "two"}
7  set5 = {"three", "two"}
8  set6 = set4.intersection(set5)
9  print(set6)
10
11 set7 = {"one", "two", "four"}
12 set8 = {"three", "two"}
13 set9 = set7.difference(set8)
14 print(set9)
```

```
>>>
{'two', 'one', 'three'}
{'two'}
{'four', 'one'}
>>>
```

Ανάμεσα σε δύο σύνολα, οι βασικές εργασίες που μπορούμε να επιτελέσουμε είναι, ο υπολογισμός της ένωσης, της τομής και της διαφοράς τους. Στη γραμμή 3, του σεναρίου mathset.py υπολογίζουμε την ένωση των δύο συνόλων, του set1 και του set2. Στη γραμμή 8 υπολογίζουμε τη τομή των δύο συνόλων, του set4 και του set5, και στη γραμμή 13, τη διαφορά του set7 από το set8.

Άσκηση 2.4 Όλοι οι υπάλληλοι σε μια εταιρία έργων ασχολούνται σε διάφορα προγράμματα. Αυτή τη χρονική στιγμή "τρέχουν" δύο έργα. Επιλέξτε τις κατάλληλες δομές δεδομένων για να αναπαραστήσετε τα δύο έργα. Έστω στο πρώτο έργο συμμετέχουν οι "Kostas" "Thanasis" "Eleni" και στο δεύτερο οι "Eleni" "Dimitris" "Petros". Να εμφανίσετε στην οθόνη το σύνολο των υπαλλήλων που συμμετέχουν στα δύο έργα, το σύνολο των υπαλλήλων που απασχολούνται και στα δύο έργα, το σύνολο των ανθρώπων από το δεύτερο έργο που δεν απασχολείται στο πρώτο.

2.4 Εισαγωγή στην αντικειμενοστρέφεια

2.4.1 Αντικειμενοστραφής ορολογία και ορισμοί

Ο αντικειμενοστραφής προγραμματισμός δεν είναι κάτι καινούργιο στο χώρο των γλωσσών του προγραμματισμού αφού όλες υλοποιούν και υποστηρίζουν τις βασικές αρχές και λειτουργίες του. Η ιδέα είναι ότι πλέον δεν προγραμματίζω με τα δεδομένα αλλά με τα αντικείμενα του προβλήματος, τα οποία ενσωματώνουν μέσα τους και τα

απαιτούμενα δεδομένα, ονομάζονται ιδιότητες, για να αλληλεπιδράσουν με το πρόβλημα καθώς και τις λειτουργίες, ονομάζονται μέθοδοι, που μπορούν να εφαρμοστούν πάνω στα δεδομένα του προβλήματος.

Πριν δημιουργήσουμε τα αντικείμενα θα πρέπει να περιγράψουμε τη μορφή - δομή τους με την κατασκευή ενός προτύπου, που ονομάζεται κλάση. Η ορολογία και μια σύντομη επεξήγηση ακολουθεί παρακάτω :

Class: Η κλάση είναι ένα ορισμένο από τον χρήστη, πρωτότυπο (πρότυπο) ενός αντικειμένου - το οποίο μπορεί να είναι οτιδήποτε στα πλαίσια ενός προβλήματος, δηλαδή ένα φυσικό αντικείμενο, μια υπηρεσία κ.λπ. - και περιλαμβάνει το σύνολο των χαρακτηριστικών του. Η κλάση δεν καταλαμβάνει χώρο στη μνήμη του υπολογιστή αφού είναι ένα γενικό σχέδιο, μια περιγραφή.

Instance: Το Στιγμιότυπο, είναι ένα ξεχωριστό και συγκεκριμένο αντικείμενο (Object) της κλάσης. Για παράδειγμα η κλάση Άνθρωπος περιγράφει γενικά ένα άνθρωπο και τα χαρακτηριστικά του, δηλαδή ότι έχει όνομα, ημερομηνία γέννησης, ότι μπορεί να τρέχει, να εργάζεται κ.λπ. Αυτή η κλάση - η γενική περιγραφή ενός ανθρώπου - θα μας βοηθήσει στα πλαίσια ενός προβλήματος να δημιουργήσουμε υπαλλήλους, μαθητές κ.λπ. δηλαδή συγκεκριμένα στιγμιότυπα της κλάσης Άνθρωπος, τα οποία όλα θα είναι ξεχωριστά (ο Κώστας Παπαδόπουλος με ημερομηνία γέννησης 19/6/1956) αλλά όλα έχουν τις ίδιες ιδιότητες και λειτουργίες αφού προέρχονται από την ίδια κλάση, δηλαδή όλοι μπορούν να τρέχουν να εργάζονται κ.λπ. Το αντικείμενο καταλαμβάνει χώρο στη μνήμη του υπολογιστή, τόσο όσο περιγράφεται στη κλάση, αφού σε αυτό θα αποθηκευτούν οι τιμές του κάθε ξεχωριστού αντικειμένου.

Class Members: Τα μέλη, τα περιεχόμενα της κλάσης μπορεί να είναι :

- **Methods**: Μέθοδοι, οι συναρτήσεις της κλάσης
- **Data members**: Μεταβλητές που κρατάνε δεδομένα, οι οποίες μπορεί να είναι
    - class variables: είναι κοινές για όλα τα στιγμιότυπα της κλάσης, δηλώνονται μέσα στην κλάση αλλά έξω από τις μεθόδους της
    - instance variables: μεταβλητές που δηλώνονται μέσα σε μέθοδο και αναγνωρίζονται μόνον μέσα στο στιγμιότυπο.

Overloading: Η υπερφόρτωση αναφέρεται τόσο σε Συναρτήσεις (Μεθόδους) όσο και σε Τελεστές. Η ανάθεση περισσότερων από μιας συμπεριφορές σε μια Συνάρτηση ή έναν Τελεστή, ώστε να εξαρτάται από τον τύπο των παραμέτρων που δίνονται.

Inheritance: Κληρονομικότητα. Η μεταφορά ιδιοτήτων μιας κλάσης (προτύπου) σε μια άλλη που δημιουργείται από αυτήν.

2.4.2 Ορισμός και ανατομία μιας Κλάσης

Η δημιουργία των κλάσεων διευκολύνει και απλοποιεί τη ανάπτυξη των εφαρμογών. Ας υποθέσουμε ότι κατασκευάζουμε μια εφαρμογή, η οποία διαχειρίζεται βιβλία, τα οποία περιγράφονται από το τίτλο, το isbn και τη τιμή τους. Άρα με τις μέχρι τώρα γνώσεις, για να περιγράψουμε ένα βιβλίο θα μπορούσαμε να χρησιμοποιήσουμε δύο αλφαριθμητικά και ένα πραγματικό αριθμό. Το δεύτερο βιβλίο θα χρειαστεί άλλα δυο αλφαριθμητικά και ένα πραγματικό αριθμό καθώς και το τρίτο και ούτω καθεξής. Άρα για 3 βιβλία θα έχουμε 6 αλφαριθμητικά και τρεις πραγματικούς αριθμούς, τα οποία δεν συνδέονται μεταξύ τους.

Ο αντικειμενοστραφής προγραμματισμός προτείνει να διαχειριζόμαστε τα αντικείμενα του προβλήματος, στην περίπτωση μας τα βιβλία, όπως και στο φυσικό κόσμο, σαν ένα κοινό σύνολο από χαρακτηριστικά δεδομένα που το περιγράφουν, δηλαδή ο τίτλος, το isbn, η τιμή καθώς και τις ενέργειες που μπορούμε να κάνουμε σε αυτά, και να κατασκευάσουμε ένα γενικό σχέδιο-πρότυπο περιγραφής βιβλίων, τη κλάση Book.

Το σενάριο book.py

```
1  class Book:
2      bookNo=0
3
4      def __init__(self, title, isbn, cost=10):
5          self.booktitle=title
6          self.bookisbn=isbn
7          self.bookcost=cost
8          Book.bookNo +=1
9
10     def displayTotal(self):
11       print("Total Books = {0}".format(Book.bookNo))
12
13     def showBook(self):
14       print("Title: "+self.booktitle+" ISBN:"+self.bookisbn)
15
16     def __str__(self):
```

```
17          return "Book: {0} ISBN: {1} Cost:
{2}".format(self.booktitle,self.bookisbn, self.bookcost)
18
19      def __lt__(self, other):
20          return self.bookcost  < other.bookcost
21
22  book1=Book("Introduction to Python", "ABCllllllll")
23  book2=Book("Advanced Java", "ABC5555555", 20)
24  book1.displayTotal()
25  book1.showBook()
26  book2.showBook()
27  print(book1)
28  if(book1 < book2):
29      print("book2 is more expensive")
```

Το αποτέλεσμα της εκτέλεσης του book.py είναι

```
>>>
Total Books = 2
Title: Introduction to Python ISBN:ABCllllllll
Title: Advanced Java ISBN:ABC5555555
Book: Introduction to Python ISBN: ABCllllllll Cost: 10
book2 is more expensive
>>>
```

Το σενάριο Book.py αρχίζει με τη δεσμευμένη λέξη class που δηλώνει μια νέα κλάση και εν συνεχεία το όνομα της, Book. Στη γραμμή 2, η μεταβλητή bookNo είναι κοινή σε όλα τα αντικείμενα της κλάσης και χρησιμοποιείται για να γνωρίζουμε το συνολικό πλήθος των αντικειμένων που έχουμε δημιουργήσει κατά τη λειτουργία της εφαρμογής. Η αρχική τιμή της μεταβλητής είναι 0. Στις γραμμές 4 έως 8 ορίζουμε το κατασκευαστή της κλάσης, δηλαδή τη μέθοδο που θα εκτελεστεί κατά τη δημιουργία ενός νέου αντικειμένου. Στη γραμμή 22 όπως και στη 23 που δημιουργούμε καινούργια αντικείμενα της κλάσης, τα book1 και book2, θα γίνει κλήση του κατασκευαστή, ο οποίος έχει τέσσερα ορίσματα :

1. self, είναι η δεσμευμένη λέξη που αναφέρεται στη διεύθυνση του αντικειμένου που κάνει κλήση το κατασκευαστή, δηλαδή στη γραμμή 15 είναι η διεύθυνση του book1 και στη γραμμή 16 είναι του book2.

2. title, ο τίτλος του βιβλίου
3. isbn, το isbn του βιβλίου
4. cost, η τιμή του βιβλίου που έχει προκαθορισμένη τιμή το 10, δηλαδή αν παραλείψουμε αυτό το όρισμα κατά τη κλήση, τότε θα χρησιμοποιηθεί η συγκεκριμένη τιμή.

Στη γραμμή 5 και 6 του κατασκευαστή δίνουμε το τίτλο "Introduction to Python", το isbn "ABCllllllll", και τη τιμή 10, στο book1 όταν θα εκτελεστεί η γραμμή 22 ενώ κατά την εκτέλεση της γραμμής 23 θα αποδώσουμε το τίτλο "Advanced Java", το isbn "ABC5555555", τη τιμή 20 στο book2. Το self θα ταυτιστεί αυτόματα με τη διεύθυνση του book1 και του book2 και δεν χρειάζεται να το περνάμε εμείς σαν παράμετρο, αλλά είναι μια ενέργεια που γίνεται από τη Python.

Στη γραμμή 8 αυξάνουμε τη τιμή της bookNo κατά 1 και επειδή η συγκεκριμένη μεταβλητή είναι κοινή σε όλα τα αντικείμενα για αυτό το λόγο προηγείται του ονόματος της, το όνομα της κλάσης, δηλαδή είναι Book.bookNo. Στις γραμμές 10 και 11 δημιουργούμε τη μέθοδο displayTotal η οποία εκτυπώνει το πλήθος των βιβλίων ενώ στις γραμμές 13 και 14 δημιουργούμε τη μέθοδο showBook η οποία εκτυπώνει το τίτλο και το isbn του βιβλίου.

Στις γραμμές 16 και 17 υπερφορτώνουμε τη μέθοδο str που αναπαριστά το αντικείμενο σαν αλφαριθμητικό και στις γραμμές 19 και 29 υπερφορτώνουμε το τελεστή < ώστε να μπορούμε να συγκρίνουμε δύο βιβλία ως προς τη τιμή τους. Ο συγκεκριμένος τελεστής λειτουργεί χωρίς πρόβλημα όταν πρόκειται για αριθμούς ενώ δεν μπορεί να χρησιμοποιηθεί για βιβλία εκτός αν τον υπερφορτώσουμε και αφού κάνουμε τις απαραίτητες συγκρίσεις να επιστρέψουμε μια λογική τιμή.

Κατά την εκτέλεση του προγράμματος, στις γραμμές 22 και 23 δημιουργούμε τα δύο βιβλία, εν συνεχεία εμφανίζουμε το συνολικό πλήθος των βιβλίων, και στις γραμμές 25 και 26 καλούμε τη μέθοδο showbook ώστε να εμφανίσουμε το τίτλο και το isbn του κάθε βιβλίου. Στη γραμμή 27 εκτυπώνεται στην οθόνη, η αλφαριθμητική αναπαράσταση του book1, δηλαδή η print κάνει κλήση της υπερφορτωμένης μεθόδου str. Στις γραμμές 28 και 29 κάνουμε χρήση του υπερφορτωμένου τελεστή μικρότερο ώστε να συγκρίνουμε τις τιμές δύο βιβλίων.

2.4.3 Κληρονομικότητα

Η κληρονομικότητα είναι πολύ χρήσιμο εργαλείο στη κατασκευή μεγάλων εφαρμογών αφού δεν χρειάζεται να υλοποιήσουμε εξαρχής όλη τη λειτουργικότητα σε μια κλάση από τη στιγμή που την έχουμε υλοποιήσει σε μια άλλη κλάση. Για παράδειγμα στη κλάση Book, το σενάριο book.py, έχουμε ενσωματώσει τα δεδομένα και τις μεθόδους για τη διαχείριση ενός βιβλίου.

Εάν εν συνεχεία θέλουμε να κατασκευάσουμε τη κλάση παιδικό βιβλίο δεν χρειάζεται να υλοποιήσουμε όλη την λειτουργικότητα εξαρχής αφού κατά μεγάλο ποσοστό είναι παρόμοια με αυτή του βιβλίου. Χρειάζεται στο σώμα της νέας κλάσης να προσθέσουμε μόνο την επιπλέον πληροφορία που είναι η προτεινόμενη ηλικία για την ανάγνωση του συγκεκριμένου παιδικού βιβλίου.

Το σενάριο BookForKids.py

```
1  # επικόλληση των γραμμών 1 ως 20 του σεναρίου book.py
2
3  class BookForKids(Book) :
4     def __init__(self, title, isbn, cost, proposedage):
5         #super().__init__(title, isbn, cost)
6         Book.__init__(self, title, isbn, cost)
7         self.kidsage=proposedage
8
9     def showBook(self):
10        #super().showBook()
11        Book.showBook(self)
12        print("Age: {0}".format(self.kidsage))
13
14 kidBook=BookForKids("The snow queen", "1234789", 20, 10)
15 kidBook.showBook()
16 kidBook.displayTotal()
```

Το αποτέλεσμα της εκτέλεσης του BookForKids.py είναι

```
>>>
Title: The snow queen ISBN:1234789
Age: 10
Total Books = 1
>>>
```

Στη γραμμή 3, μετά το όνομα της κλάσης, μέσα σε παρενθέσεις δηλώνουμε τη βασική κλάση. Στις γραμμές 4 έως 7 ορίζουμε το κατασκευαστή της κλάσης BookForKids και στις γραμμές 9 έως 12 υπερφορτώνουμε τη μέθοδο showBook ώστε να εκτυπώσει και τα επιπλέον στοιχεία της κλάσης. Και στο κατασκευαστή και στη μέθοδο showBook κάνουμε κλήση των εκδόσεων τους, στη βασική κλάση, γραμμή 6 και 11. Η κλήση τους μπορεί να γίνει με δύο τρόπους, είτε κάνοντας αναφορά στη βασική κλάση σαν super() είτε χρησιμοποιώντας το όνομα της βασικής κλάσης και εν συνεχεία το όνομα της μεθόδου, με πρώτη παράμετρο τη διεύθυνση του αντικειμένου που έκανε την αντίστοιχη κλήση, δηλαδή το self.

Άσκηση 2.5: Να δημιουργήσετε την κλάση Rectangle η οποία περιγράφει ένα ορθογώνιο παραλληλόγραμμο. Ο κατασκευαστής θα δέχεται δύο τιμές, πλάτος και ύψος. Να δημιουργήσετε δύο μεθόδους : α) την εμβαδόν - η οποία υπολογίζει και επιστρέφει το εμβαδόν σαν πλάτος επί ύψος β) τη περίμετρο - η οποία υπολογίζει και επιστρέφει τη περίμετρο σαν 2 * πλάτος και ύψος. Υπερφορτώστε το τελεστή μεγαλύτερο > ο οποίος συγκρίνει δύο παραλληλόγραμμα ως προς τη περίμετρο. Δημιουργείστε δυο παραλληλόγραμμα και συγκρίνετε τα μεταξύ τους.

2.5 Βιβλιοθήκες Python

2.5.1 Τι είναι μια βιβλιοθήκη

Η βιβλιοθήκη ή άρθρωμα (module) είναι μια συλλογή από συναρτήσεις ή/και κλάσεις που σχετίζονται με κάποια συγκεκριμένη λειτουργικότητα. Για να μπορέσουμε να χρησιμοποιήσουμε μια βιβλιοθήκη θα πρέπει να δώσουμε στον μεταφραστή την εντολή να το εισαγάγει μέσα στον κώδικά μας και αυτό γίνεται με την εντολή import.Έχουμε ήδη χρησιμοποιήσει το module math σε προηγούμενες ενότητες.

Υπάρχουν modules που έρχονται μαζί με τη γλώσσα και συνιστούν τις εξ ορισμού βιβλιοθήκες της (π.χ. math, sys), ενώ μπορούμε να δημιουργήσουμε και τις δικές μας. Η Python διατηρεί ένα μονοπάτι με τους καταλόγους όπου ψάχνει για να βρει modules. Μπορούμε να δούμε τι περιέχει μέσω της μεταβλητής path του module sys:

Το σενάριο showPath.py	**Το αποτέλεσμα**

`import sys`
`print(sys.path)`
	`['C:\\Windows\\SYSTEM32\\python33.zip',`
	`'C:\\Python33\\DLLs',`
	`'C:\\Python33\\lib', 'C:\\Python33',`
	`'C:\\Python33\\lib\\site-packages']`

Προσοχή, το αποτέλεσμα της εκτέλεσης μπορεί να διαφέρει ανάλογα με το μονοπάτι της εγκατάστασης της Python.

2.5.2 Δημιουργία Module

Η δημιουργία ενός η περισσοτέρων modules είναι χρήσιμη λειτουργία στην ανάπτυξη μιας εφαρμογής αφού μπορούμε να ενσωματώσουμε σε αυτά ομοειδείς λειτουργίες και δηλώσεις, τις οποίες εν συνεχεία μπορούμε να εισάγουμε σε άλλα αρχεία της εφαρμογής. Κατά αυτό το τρόπο επαναχρησιμοποιούμε το κώδικα και έχουμε καλύτερη διαχείριση και συντήρηση των λειτουργιών του.

Εν συνεχεία θα δημιουργήσουμε δύο αρχεία σε Python, το inep.py το οποίο θα αποτελεί το module και το testmodule.py το οποίο θα κάνει εισαγωγή το inep και θα χρησιμοποιεί τις λειτουργίες του. Μέσα στο inep.py πληκτρολογούμε τις εντολές παρακάτω εικόνας.

```
1   # factorial
2
3   def factorial(n):
4       a, b = 1, 1
5       while (b <= n):
6               a = a * b
7               print(a)
8               b = b +1
9       return a
10
11  #isPrime
12
13  def isprime(n):
14      for i in range(2, n-1):
15          if((n % i) == 0):
16              return False
17      return True
18
```

Εικόνα 2.2: Ο κώδικας του inep.py

Η συνάρτηση factorial υπολογίζει το παραγοντικό ενός αριθμού n δηλαδή το γινόμενο όλων των αριθμών από το 1 μέχρι τον αριθμό n. Ενώ η isprime υπολογίζει αν ένας αριθμός n είναι πρώτος ή όχι. Πρώτοι λέγονται οι αριθμοί που διαιρούνται ακριβώς, χωρίς να αφήνουν υπόλοιπο, μόνο με τη μονάδα και τον εαυτό τους. Τις δύο ανωτέρω συναρτήσεις τις χρησιμοποιούμε αρκετά συχνά σε μαθηματικές εφαρμογές και στη περίπτωση που ήταν χρήσιμές σε αρκετά προγράμματα, τότε τις ενσωματώνουμε σε ένα module ώστε να μπορούμε να τις επαναχρησιμοποιήσουμε και σε άλλα σενάρια.

Ο κώδικας του testmodule.py | Το αποτέλεσμα της εκτέλεσης

```
1  import inep
2
3  print(inep.factorial(4))
4  print(inep.isprime(17))
```

```
>>>
24
True
>>>
```

Στη γραμμή 1 του testmodule.py, κάνουμε εισαγωγή το inep και στις γραμμές 3 και 4 καλούμε τις συναρτήσεις του. Εν συνεχεία τροποποιούμε τον ανωτέρω κώδικα ώστε να πάρει τη μορφή τη παρακάτω μορφή :

Ο κώδικας του testmodule.py	Το αποτέλεσμα της εκτέλεσης

`1 from inep import isprime`
`2`
`3 print(isprime(23))`
	`True`

Με την αλλαγή αυτή, η συνάρτηση isprime έχει ενσωματωθεί πλήρως στο testmodule.py ειναι δηλαδή σαν να μην υπήρχε το inep και είχαμε γράψει απευθείας τη συνάρτηση στο σώμα του testmodule. Φυσικά κάθε προσπάθεια να εκτελέσουμε τη συνάρτηση factorial θα εμφανίσει σφάλμα. Εναλλακτικά θα μπορούσαμε να κάνουμε εισαγωγή όλες τις συναρτήσεις με την παρακάτω εντολή :

Ο κώδικας του testmodule.py	Το αποτέλεσμα της εκτέλεσης

`from inep import isprime, factorial`
 	
`print(isprime(17))`
`print(factorial(8))`
	`True`
	
	`40320`

Στη περίπτωση που κάνουμε εισαγωγή όλες τις συναρτήσεις θα μπορούσαμε εναλλακτικά να χρησιμοποιήσουμε στη γραμμή 1 την εντολή : from inep import *

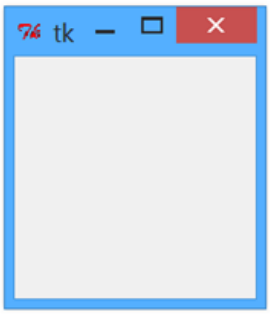
Το module tkinter είναι το παλαιότερο και πλέον γνωστό module για τη δημιουργία γραφικής διεπαφής (Graphical User Interface) στην Python, αν και υπάρχουν πολλές άλλες δυνατότητες και εναλλακτικές, όπως για παράδειγμα το module wxPython ή το Jython που φέρνει στην Python όλη τη λειτουργικότητα των κλάσεων της Java (περισσότερες πληροφορίες στη διεύθυνση ttps://wiki.python.org/moin/GuiProgramming).

Το σενάριο testTkinter.py	Το αποτέλεσμα της εκτέλεσης
`>>> import tkinter` `>>> tkinter._test()`	

Το tkinter είναι στηριγμένο πάνω στο πλαίσιο tcl/tk (tcl = tool command line, γλώσσα προγραμματισμού, tk η επέκταση της tcl, βιβλιοθήκη για χειρισμό γραφικής διεπαφής). Η ενσωμάτωση του module γίνεται στην python 3+ με την εντολή import tkinter ενώ στην 2+ με την εντολή import Tkinter.

Η βασική δομή κώδικα για τη χρήση του tkinter είναι η ακόλουθη:

Το σενάριο basicTkinter.py	Το αποτέλεσμα της εκτέλεσης

`import tkinter`
`topwindow = tkinter.Tk()` 1
`# Εδώ δημιουργούνται τα widgets,`
`# π.χ. κουμπιά, ετικέττες, κλπ`
`topwindow.mainloop()`
2
	

Τα βασικά σημεία στη λειτουργία του basicTkinter.py είναι :

1. Δημιουργία κεντρικού παράθυρου στο οποίο αναφερόμαστε μέσω της topwindow
2. Έναρξη παρακολούθησης των γεγονότων στο GUI και ενημέρωσής του παραθύρου, όποτε χρειαστεί, κατά τη διάρκεια της λειτουργίας του.

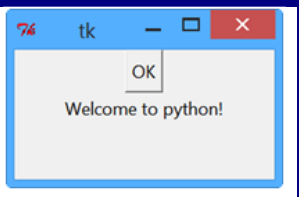
Με τον όρο widgets εννοούμε τα αντικείμενα που χρησιμοποιούμε για να δημιουργήσουμε ένα GUI: Button, Label, Text, κλπ. Στη διεύθυνση http://www.tutorialspoint.com/python/python_gui_programming.htm θα βρείτε μια λίστα με τα κυριότερα widgets καθώς και τη σύνταξή τους.

Ένα πιο ολοκληρωμένο παράδειγμα είναι το σενάριο windowWithEvent.py το οποίο αρχίζει με την εισαγωγή του module tkinter. Στις γραμμές 2 και 3 ορίζεται η συνάρτηση endf, η οποία πρόκειται να εκτελεστεί με το πάτημα του πλήκτρου που θα τοποθετήσουμε στο παράθυρο του σεναρίου. Η λειτουργία της endf είναι να κλείσει το παράθυρο, με τη κλήση της destroy.

Στη γραμμή 6 δημιουργούμε το κεντρικό παράθυρο και η προγραμματιστική αναφορά θα γίνεται μέσω της topwindow. Στην γραμμή 7 δημιουργούμε το κουμπί εντολών btn, με κείμενο το OK και με συνάρτηση λειτουργίας κατά το πάτημα του, την endf.

Το σενάριο windowWithEvent.py	Το αποτέλεσμα της εκτέλεσης

```
1  import tkinter
2
3  def endf():
4      topwindow.destroy()
5
6  topwindow = tkinter.Tk()
7  btn=tkinter.Button(topwindow,text="OK", command=endf)
8  btn.pack()
9
10 label = tkinter.Label(topwindow, text="Welcome to python!")
11 label.pack()
12
13 topwindow.mainloop()
```



Στην επόμενη γραμμή τοποθετούμε το κουμπί στο παράθυρο. Στις γραμμές 10 και 11 δημιουργούμε την ετικέτα label με κείμενο το Welcome to python και το τοποθετούμε στο κεντρικό παράθυρο. Τέλος στη γραμμή 13 αρχίζει η παρακολούθηση των γεγονότων στο κεντρικό παράθυρο.

Άσκηση 2.6 Να δημιουργήστε το module geometry το οποίο περιέχει τις συναρτήσεις **CircleArea** - η οποία δέχεται ένα όρισμα, την ακτίνα ενός κύκλου R και επιστρέφει το εμβαδόν ενός κύκλου 3.1415 * R * R, την **RectangleArea** - η οποία δέχεται δύο ορίσματα τις πλευρές A και B ενός ορθογωνίου παραλληλογράμμου και επιστρέφει το εμβαδόν του A * B, την **VolumeCylinder** - η οποία δέχεται δύο ορίσματα, την ακτίνα της βάσης R και το ύψος H ενός κυλίνδρου και επιστρέφει τον όγκο του 3.1415 * R * R * H. Να δημιουργήσετε και το αρχείο testgeometry.py στο οποίο θα κάνετε εισαγωγή μόνο τις συναρτήσεις RectangleArea και CircleArea και θα εκτυπώσετε το εμβαδόν ενός ορθογωνίου με τιμές 10 και 13 και το εμβαδόν ενός κύκλου με ακτίνα τη τιμή 7.5.

ΓΛΩΣΣΑΡΙ

Αντικείμενο: είναι ένα ξεχωριστό παράδειγμα-στιγμιότυπο της κλάσης του.

Άρθρωμα: είναι ένα αρχείο που περιέχει δηλώσεις και ορισμούς σε Python και ολοκληρη η λειτουργικότητα του ή μέρος αυτής μπορεί να γίνει εισαγωγή σε ένα νέο αρχείο.

Δομές δεδομένων: αναφέρονται σε διαφορετικούς τρόπους οργάνωσης και αποθήκευσης των δεδομένων μέσα σε έναν υπολογιστή, ώστε αυτά να μπορούν να χρησιμοποιηθούν με το πλέον αποδοτικό τρόπο.

Κλάση: είναι μία αυτοτελής και αφαιρετική αναπαράσταση των ιδιοτήτων και των μεθόδων κάποιας κατηγορίας αντικειμένων σε ένα περιβάλλον προγραμματισμού.

Κληρονομικότητα: είναι η μεταβίβαση των μεθόδων και των ιδιοτήτων μιας γονικής κλάσης σε μια νέα κλάση χωρίς να τη περιορίζει να προσθέσει δικές της, νέες μεθόδους και ιδιότητες.

Λεξικό: είναι μια δομή δεδομένων που αντιστοιχίζει μοναδικά κλειδιά για τις τιμές.

Λίστα: είναι διατεταγμένη ακολουθία στοιχείων η οποία μπορεί να τροποποιηθεί κατά τη διάρκεια της εφαρμογής.

Πλειάδα: είναι διατεταγμένη ακολουθία στοιχείων η οποία δεν μπορεί να τροποποιηθεί κατά τη διάρκεια της εφαρμογής.

Συνάρτηση: είναι ένας τύπος υποπρογράμματος (ένα σύνολο εντολών) που μπορεί να δέχεται ή όχι παραμέτρους και να επιλύει ένα μέρος του συνολικού προβλήματος.

Σύνολο: είναι μια δομή δεδομένων που περιέχει μοναδίαια στοιχεία χωρίς διάταξη.

ΟΔΗΓΟΣ ΠΕΡΑΙΤΕΡΩ ΜΕΛΕΤΗΣ

- Στη διεύθυνση http://www.greenteapress.com/thinkpython/html/index.html παρέχεται δωρεάν το βιβλίο Think in Python το οποίο καλύπτει όλα τα περιεχόμενα του κεφαλαίου.
- Στη διεύθυνση http://www.swaroopch.com/notes/python/ παρέχεται δωρεάν το βιβλίο A Byte of Python το οποίο καλύπτει όλα τα περιεχόμενα του κεφαλαίου.
- Στη διεύθυνση http://www.diveintopython.net/ παρέχεται δωρεάν το βιβλίο Dive into Python το οποίο καλύπτει όλα τα περιεχόμενα του κεφαλαίου.
- Στη διεύθυνση http://www.learnpython.org/en/Classes_and_Objects υπάρχει ένας δικτυακός τόπος με εκπαιδευτικά θέματα και αλληλεπιδραστικά παραδείγματα με τα περιεχόμενα του κεφαλαίου.
- Στη διεύθυνση http://www.tutorialspoint.com/python/ υπάρχει ένας δικτυακός τόπος με παραδείγματα σχετικά με τα περιεχόμενα του κεφαλαίου.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

3 ΑΠΟΘΗΚΕΥΣΗ & ΔΙΑΧΕΙΡΙΣΗ ΠΛΗΡΟΦΟΡΙΑΣ

Σκοπός

Σκοπός του κεφαλαίου είναι η παρουσίαση των δυνατοτήτων διαχείρισης δεδομένων που παρέχει η Python. Το κεφάλαιο περιέχει όλη την πληροφορία που χρειάζεται για να μπορείτε να δημιουργείτε και να διαχειρίζεστε αρχεία κειμένου καθώς και να χρησιμοποιείτε σχεσιακές βάσεις δεδομένων για αποθήκευση και ανάκτηση των δεδομένων των εφαρμογών σας. Ειδικότερα, παρουσιάζεται η υποστήριξη της Python για διαχείριση των βάσεων δεδομένων SQLite και mySQL.

Προσδοκώμενα αποτελέσματα

Όταν θα έχετε μελετήσει το κεφάλαιο αυτό θα μπορείτε να:

- χρησιμοποιείτε αρχεία κειμένου,
- αποθηκεύετε και ανακτάτε δεδομένα από ΒΔ SQLite,
- αποθηκεύετε και ανακτάτε δεδομένα από ΒΔ mySQL,
- αναπτύσσετε εφαρμογές που χρησιμοποιούν μόνιμα αποθηκευτικά μέσα

Έννοιες-Κλειδιά

- Αρχεία κειμένου
- Σχεσιακές Βάσεις Δεδομένων
- SQLite
- mySQL

Εισαγωγικές Παρατηρήσεις

Η αποθήκευση και διαχείριση δεδομένων αποτελεί απαραίτητη λειτουργία ενός πληροφοριακού συστήματος. Στα Κεφάλαια 1 και 2 ήρθαμε σε επαφή με το «γενικό» σύστημα εργαλείων της γλώσσας. Στο παρόν κεφάλαιο θα μάθουμε πώς μπορούμε να δημιουργούμε, διαβάζουμε και επεξεργαζόμαστε αρχεία δεδομένων. Επιπλέον θα δούμε με ποιο τρόπο η Python χρησιμοποιεί τις σχεσιακές βάσεις δεδομένων (Β.Δ.). Για το σκοπό αυτό έχουμε επιλέξει αφενός την SQLite που είναι μια απλή αλλά λειτουργική και αποτελεσματική υλοποίηση σχεσιακής Β.Δ. που βρίσκει πολλές εφαρμογές, ιδιαίτερα σε μέτριας ισχύος υπολογιστικά περιβάλλοντα, όπως είναι τα κινητά τηλέφωνα, και αφετέρου την mySQL που είναι μια ισχυρότατη κλασική σχεσιακή Β.Δ.. Και οι δύο Β.Δ. είναι ελεύθερο λογισμικό και μπορούμε να τις χρησιμοποιούμε δωρεάν.

3.1 Αρχεία Κειμένου

Ένα αρχείο κειμένου είναι μια ακολουθία από χαρακτήρες που βρίσκονται αποθηκευμένοι σε ένα σταθερό μέσο όπως ένας σκληρός δίσκος, ένα CDROM κ.λπ.

Η Python διαθέτει μια σειρά από μεθόδους με τις οποίες μπορούμε να ανοίγουμε, να διαβάζουμε και να τροποποιούμε τα αρχεία κειμένου. Πάμε αμέσως να τις δούμε παραθέτοντας και παραδείγματα εφαρμογής τους.

Δημιουργία αρχείου για εγγραφή: <fhandle> = open(<filename>, 'w')

Με την εντολή αυτή δημιουργείται ένα νέο, κενό αρχείο, στο οποίο μπορούμε να εισαγάγουμε δεδομένα. Το <filename> μπορεί να περιέχει τόσο το μέσον αποθήκευσης όσο και το μονοπάτι των φακέλων όπου θα γίνει η δημιουργία. Αν δεν προσδιοριστεί μονοπάτι, τότε το αρχείο δημιουργείται στον τρέχοντα κατάλογο. Σε περίπτωση που στο φάκελο υπάρχει ήδη ένα αρχείο με το όνομα που προσδιορίζουμε, τότε αυτό αδειάζει και τα δεδομένα που περιέχει χάνονται. Πρέπει επομένως να είμαστε προσεκτικοί στη χρήση αυτής της εντολής. Το <fhandle> είναι ένα αντικείμενο τύπου file, του οποίου μπορούμε να χρησιμοποιήσουμε τις μεθόδους write() και close() για να γράψουμε τα δεδομένα και να κλείσουμε το αρχείο, αντίστοιχα. Η μέθοδος close() μεταφέρει τυχόν δεδομένα που, ανάλογα με το πώς τα διαχειρίζεται το λειτουργικό σύστημα, μπορεί να βρίσκονται ακόμα σε προσωρινή αποθήκευση (buffer).

Παράδειγμα 3.1: Δημιουργία αρχείου κειμένου και εγγραφή δεδομένων

```
fname = input ("Δώστε όνομα νέου αρχείου κειμένου:")
fhandle = open (fname,"w")
text = input("""Πληκτρολογήστε τις γραμμές του αρχείο,
τέλος με enter σε κενή γραμμή \n""")
firstline = True
while text !='':
    if firstline:
        fhandle.write(text)
        firstline = False
    else:
        fhandle.write("\n"+text)
    text = input()
fhandle.close()
```

Ο χρήστης πληκτρολογεί το όνομα του αρχείου, και ο κώδικας το ανοίγει για εγγραφή (access mode = 'w'). Στη συνέχεια, μέσα στο βρόγχο while ο χρήστης πληκτρολογεί τα περιεχόμενα κάθε γραμμής και πατά enter. Η εντολή while ελέγχει το αλφαριθμητικό που δόθηκε και τερματίζει αν είναι κενό. Αν έχουν δοθεί δεδομένα, αυτά γράφονται στο αρχείο με τη μέθοδο write(). Στο τέλος η close() κλείνει το αρχείο

και απελευθερώνει το αντικείμενο fhandle. Η boolean μεταβλητή firstline χρησιμοποιείται για να ελέγξει αν είμαστε στην πρώτη γραμμή, οπότε απλά γράφεται το κείμενο. Στις επόμενες επαναλήψεις, πρώτα αλλάζουμε γραμμή και μετά γράφουμε. Ακολουθεί ένα δείγμα της εκτέλεσης και το αρχείο που δημιουργείται.

```
Δώστε όνομα νέου αρχείου κειμένου:names.txt
Πληκτρολογήστε τις γραμμές του αρχείο,
τέλος με enter σε κενή γραμμή
Παναγιώτης Παναγιώτου
Πέτρος Πέτρου
Νίκος Νίκου

>>>
```

names.txt

```
1 Παναγιώτης Παναγιώτου
2 Πέτρος Πέτρου
3 Νίκος Νίκου
```

Εικόνα 3.1: Δημιουργία αρχείου κειμένου - δείγμα εκτέλεσης

Άνοιγμα υπάρχοντος αρχείου για ανάγνωση: <fhandle> = open(<filename>, 'r')

Ανοίγουμε ένα αρχείο που έχει όνομα <filename> για ανάγνωση. Το αρχείο πρέπει να υπάρχει ήδη. Μπορούμε να παραλείψουμε το access mode 'r' διότι ένα αρχείο ανοίγει εξ ορισμού για ανάγνωση. Με τη μέθοδο read() διαβάζουμε τα περιεχόμενά του.

Παράδειγμα 3.2: Άνοιγμα αρχείου κειμένου και ανάγνωση δεδομένων

```
fname = 'names.txt'
fhandle = open (fname,"r")
data = fhandle.read()
fhandel.close()
print (data)
```

Όπως βλέπετε στο Παράδειγμα 3.2, η ανάγνωση δεδομένων από το αρχείο κειμένου, που δημιουργήσαμε στο Παράδειγμα 3.1, δεν μπορεί να είναι ευκολότερη! Τα δεδομένα που διαβάζονται αποθηκεύονται στη μεταβλητή data και στη συνέχεια μπορούμε να τα υποβάλουμε σε οποιαδήποτε επεξεργασία θέλουμε, στο παράδειγμα τα εμφανίζουμε στην οθόνη. Προσέξτε ότι, στο αλφαριθμητικό data περιλαμβάνονται όλοι οι χαρακτήρες που θα βρεθούν στο αρχείο, στο παράδειγμά μας οι δύο πρώτες γραμμές περιέχουν τον χαρακτήρα αλλαγής γραμμής \n. Αυτός είναι και ο λόγος που η εντολή print(data) τα εμφανίζει σε γραμμές, όπως διαβάστηκαν. Ωστόσο στη συνηθισμένη περίπτωση, αυτό θα πρέπει να το λαμβάνουμε υπόψη και αν θέλουμε να έχουμε τα «καθαρά» δεδομένα, θα πρέπει να αφαιρέσουμε από την είσοδο τυχόν πρόσθετους χαρακτήρες. Αυτό μπορεί να γίνει με τη μέθοδο str.strip(), που όταν χρησιμοποιηθεί χωρίς όρισμα, σβήνει τους λευκούς χαρακτήρες (π.χ. νέα γραμμή, κενά) που προηγούνται και έπονται του αλφαριθμητικού. Στο Παράδειγμα 3.3 διαβάζουμε κάθε μια γραμμή και την αποθηκεύουμε σε μια, αρχικά κενή, λίστα.

Παράδειγμα 3.3: Ανάγνωση δεδομένων σε Λίστα

```
fname = 'names.txt'
fhandle = open (fname,"r")
array = []
for line in fhandle:
    array.append(line.strip())
fhandle.close()
print(array)
```

```
['Παναγιώτης Παναγιώτου', 'Πέτρος Πέτρου', 'Νίκος Νίκου']
```

Άνοιγμα αρχείου για επέκταση: <fhandle> = open(<filename>, 'a')

Αν το αρχείο υπάρχει, τότε ανοίγεται χωρίς να πειραχθούν τα περιεχόμενά του. Οι επόμενες εντολές write() θα προσθέσουν τα δεδομένα στο τέλος του. Αν το αρχείο δεν υπάρχει, τότε δημιουργείται ένα νέο, κενό.

Παράδειγμα 3.4: Προσθήκη δεδομένων σε αρχείο

```
fname = 'names.txt'
fhandle = open (fname,"a")
fhandle.write("\nΒασίλειος Βασιλείου")
fhandle.close()
for line in open(fname,"r"):
    print(line,end="")
```

```
Παναγιώτης Παναγιώτου
Πέτρος Πέτρου
Νίκος Νίκου
Βασίλειος Βασιλείου
```

Παρατηρήστε ότι συντάξαμε την εντολή for, ενθυλακώνοντας το άνοιγμα του αρχείου. Θα μπορούσε να γίνει σε δύο ξεχωριστά βήματα, πρώτα να ανοίξουμε το αρχείο και μετά να διατρέξουμε τις γραμμές του. Ωστόσο η Python είναι μια γλώσσα μινιμαλιστική και τέτοιες συντομεύσεις χρησιμοποιούνται για να κάνουν τον κώδικα λακωνικό και περισσότερο ευανάγνωστο.

Το fhandle, που χρησιμοποιούμε για να αναφερθούμε στο αρχείο που ανοίγουμε, είναι τύπου file και όπως τα περισσότερα αντικείμενα μιας κλάσης, έχει ιδιότητες (attributes). Στον Πίνακα 3.1 που ακολουθεί, μπορείτε να δείτε μερικές από αυτές.

Πίνακας 3.1: Ιδιότητες αντικειμένου file

fhandle.closed	Είναι boolean και έχει τιμή True αν το αρχείο είναι κλειστό και False αν είναι ανοικτό
fhandle.mode	Επιστρέφει το access mode με το οποίο ανοίχτηκε το αρχείο
fhandle.name	Επιστρέφει το όνομα του αρχείου (που γνωρίζει το Λ.Σ.)

Στο Παράδειγμα 3.2 ανοίξαμε το αρχείο για να διαβάσουμε τα περιεχόμενά του, γνωρίζοντας ότι αυτό υπάρχει (αφού το είχαμε δημιουργήσει στο προηγούμενο

παράδειγμα). Αν όμως προσπαθήσουμε να ανοίξουμε ένα αρχείο που δεν υπάρχει (π.χ. διότι ο χρήστης έδωσε λάθος όνομα) θα έχουμε σφάλμα εκτέλεσης. Για να προλαμβάνουμε τέτοια προβλήματα μπορούμε να χρησιμοποιήσουμε την εντολή παγίδευσης λαθών (ή εξαιρέσεων, exception handling). Η απλούστερη μορφή της εντολής φαίνεται στον Πίνακα 3.2.

Πίνακας 3.2: Εντολή try .. except .. else

```
try:
  <επικίνδυνη-ες λειτουργία-ες>
except:
  <εντολές αν υπάρχει εξαίρεση>
else:
<εντολές αν δεν υπάρχει εξαίρεση>
```

```
try:
   fhandle = open("datafile.txt", "w")
   fhandle.write("data")
except:
   print ("Error in file open or write")
else:
   print("Data written")
   fhandle.close()
```

Παράδειγμα 3.5: Έλεγχος πρόσβασης σε αρχείο

```
exist = 'names.txt'
notexist = 'notexist.txt'
lista = ['one','two','three']
fhandle = open(exist,"w")
for item in lista:
    fhandle.write(item+"\n")
fhandle.close()
for filename in [exist,notexist]:
    try:
        fhandle = open(filename, "r")
    except:
        print ('Error in file {}'.format(filename))
    else:
        print ("File {} contains are:".format(fhandle.name))
        for lines in fhandle:
            print(lines.strip())
        fhandle.close()
```

```
...
File names.txt contains are:
one
two
three
Error in file notexist.txt
```

Στην αρχή του Παραδείγματος 3.5 δημιουργούμε ένα αρχείο και γράφουμε τα δεδομένα που περιέχει η λίστα. Κατόπιν, δοκιμάζουμε να ανοίξουμε τόσο το αρχείο που έχουμε δημιουργήσει, όσο και ένα που δεν υπάρχει. Δείτε ότι στην πρώτη περίπτωση εκτελείται ο κώδικας του else και εμφανίζονται τα περιεχόμενα του αρχείου, ενώ στη δεύτερη το αρχείο δεν υπάρχει και εκτελείται ο κώδικας του except.

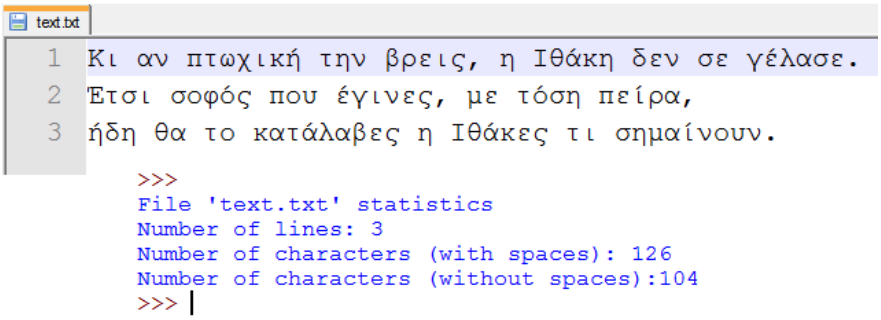
Η readline() είναι μια ακόμα μέθοδος με την οποία μπορούμε να διαβάζουμε μια-μια τις γραμμές ενός αρχείου κειμένου. Η γραμμή ενός τέτοιου αρχείου τερματίζει όταν η εντολή συναντήσει τον χαρακτήρα νέας γραμμής \n, ο οποίος συμπεριλαμβάνεται στο αλφαριθμητικό που η readline() επιστρέφει. Όταν φθάσουμε στο τέλος του αρχείου (EOF) η readline() επιστρέφει ένα κενό αλφαριθμητικό. Στο Παράδειγμα 3.6

ανοίγουμε ένα αρχείο κειμένου και μετράμε τις γραμμές και τους χαρακτήρες που αυτό περιέχει.

Παράδειγμα 3.6: Μέτρημα γραμμών και χαρακτήρων αρχείου κειμένου

```
filename = 'text.txt'
fh = open(filename,'r')
lines = 0
chars = 0
nospace=0
line = fh.readline().strip("\n")
while line != '':
    lines += 1
    for ch in line:
        chars += 1
        if ch !=' ': nospace += 1
    line = fh.readline().strip("\n")
print("File '{0}' statistics".format(fh.name))
print ("Number of lines: {0}".format(lines))
print ("Number of characters (with spaces): {}".format(chars))
print ("Number of characters (without spaces):{}".format(nospace))
fh.close()
```

Αφού ανοίξουμε το αρχείο, διαβάζουμε τις γραμμές του. Το τέλος αρχείου εντοπίζεται στη συνθήκη της εντολής while, όπου η readline() επιστρέφει κενό αλφαριθμητικό. Αφαιρούμε το χαρακτήρα αλλαγής γραμμής (strip) αλλά όχι τυχόν κενά, που θεωρούνται μέρος του κειμένου. Στη συνέχεια, μετράμε γραμμές, χαρακτήρες καθώς και τους χαρακτήρες πλην των κενών. Τέλος, εμφανίζουμε τα αποτελέσματα και κλείνουμε το αρχείο. Το περιεχόμενο του αρχείου text.txt που χρησιμοποιήσαμε, καθώς και το αποτέλεσμα της εκτέλεσης μπορείτε να τα δείτε στην Εικόνα 3.2.



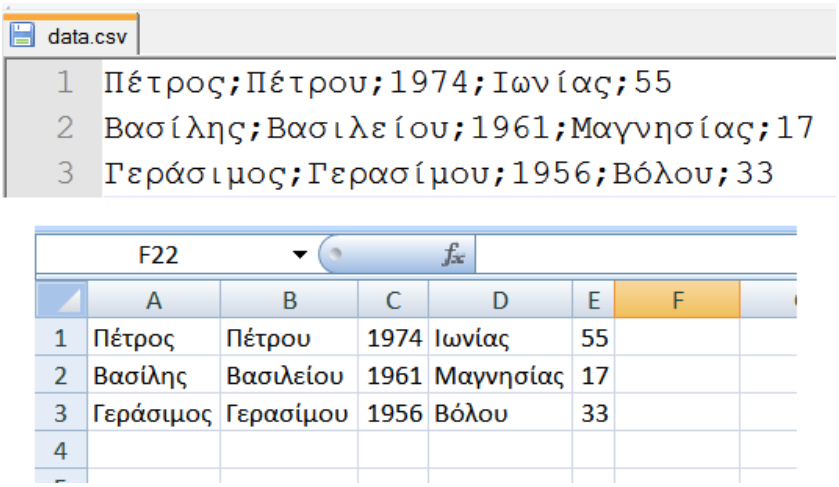
Εικόνα 3.2: Μέτρημα γραμμών & χαρακτήρων αρχείου κειμένου - εκτέλεση κώδικα

Άσκηση 3.1: Γράψτε ένα πρόγραμμα που θα διαβάζει το αρχείο text.txt του Παραδείγματος 3.6 και θα δημιουργεί ένα νέο αρχείο με τις λέξεις του κειμένου μια σε κάθε γραμμή. Οι λέξεις του πηγαίου αρχείου χωρίζονται με κενό. Μπορείτε να χρησιμοποιήσετε τη μέθοδο split() των αλφαριθμητικών. Βρείτε τις οδηγίες χρήσης της από το διαδίκτυο.

Άσκηση 3.2: Γράψτε ένα πρόγραμμα που θα διαβάζει ένα αρχείο κειμένου και θα δημιουργεί μια απεικόνιση της κάθε λέξης του με έναν αριθμό ο οποίος είναι το μήκος της λέξης. Τα αποτελέσματα θα αποθηκεύονται σε ένα νέο αρχείο κειμένου, όπου σε κάθε γραμμή του θα είναι η λέξη, ένα κόμμα και μετά ο αριθμός.

Δραστηριότητα 3.1: Comma Separated Values (αρχεία CSV)

Ένα αρχείο CSV είναι αρχείο κειμένου όπου όλες οι γραμμες του έχουν την ίδια δομή. Κάθε γραμμή περιέχει μια σειρά από τιμές (πεδία, fields) που διαχωρίζονται με κάποιον ειδικό χαρακτήρα (ή και περισσότερους) που ονομάζεται οριοθέτης (delimiter). Πολύ συχνά ο χαρακτήρας αυτός είναι το κόμμα, γι' αυτό και τα αρχεία ονομάζονται οριοθετημένα με κόμματα. Στην Ελλάδα, ωστόσο, το κόμμα χρησιμοποιείται για τους δεκαδικούς αριθμούς, για το λόγο αυτό ο διαχωριστικός χαρακτήρας μπορεί να είναι άλλος, π.χ. το ελληνικό ερωτηματικό. Οι τιμές μπορεί να περιλαμβάνουν στην αρχή και στο τέλος τους κενά ώστε όλες οι γραμμές να έχουν το ίδιο πλήθος χαρακτήρων. Αρχεία CSV δημιουργούνται από προγράμματα λογιστικών φύλλων (Excel, Libre Calc, κ.λπ.) αλλά και βάσεις δεδομένων. Στην Εικόνα 3.2 μπορείτε να δείτε το ίδιο αρχείο, data.csv που έχει ανοιχτεί με έναν κειμενογράφο και με το Excel.



Εικόνα 3.3: Αρχείο CSV

Δημιουργήστε με ένα λογιστικό φύλλο το παραπάνω αρχείο CSV. Γράψτε ένα πρόγραμμα Python που θα ανοίγει το αρχείο αυτό, θα διαβάζει τα περιεχόμενά του και θα δημιουργεί μια λίστα όπου κάθε στοιχείο της θα είναι επίσης λίστα με τα περιεχόμενα της γραμμής. Για παράδειγμα: [['Πέτρος','Πέτρου','1974','Ιωνίας','55'], [κ.ο.κ.],…].

3.2 Διαχείριση πληροφορίας σε Βάσεις Δεδομένων

3.2.1 Δεδομένα και Πληροφορία

Η συλλογή, αποθήκευση, ανάκτηση και επεξεργασία δεδομένων αποτελεί μια από τις σημαντικότερες λειτουργίες της Πληροφορικής. Τα δεδομένα υπάρχουν παντού,

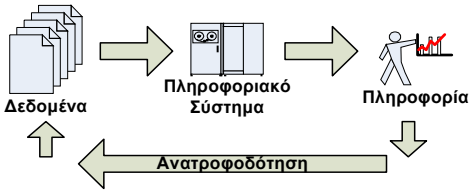
καταγράφονται και χρησιμοποιούνται σε κάθε έκφανση της ανθρώπινης δραστηριότητας, κυβερνητική, εμπορική, επιστημονική κ.ο.κ., προκειμένου να δημιουργήσουν το πολυτιμότερο προϊόν της σύγχρονης εποχής, την πληροφορία.

Τι είναι όμως Δεδομένο και τι Πληροφορία; Παρόλο που δεν υπάρχει ένας επίσημος, κοινά αποδεκτός ορισμός για τους όρους, οι ορισμοί που έχουν δοθεί από τον οργανισμό τυποποίησης των Η.Π.Α. ANSI είναι χαρακτηριστικοί:

- **Δεδομένα (data)** είναι ο ακατέργαστος αριθμός, το γεγονός, σύμβολο, γράμμα κλπ, στο οποίο μπορούμε να προσδώσουμε έννοια, ενώ
- **Πληροφορία (information)** είναι το νόημα που δίνουμε σε ένα σύνολο από δεδομένα, αφού τα υποβάλλουμε σε κάποιας μορφής επεξεργασία.

Εικόνα 3.4: από τα δεδομένα στην πληροφορία

Σε πολλές περιπτώσεις οι όροι χρησιμοποιούνται εναλλακτικά (όπως και ο όρος Γνώση, Knowledge) και συχνά συγχέονται και ταυτίζονται, πράγμα που είναι αναμενόμενο: το δεδομένο αποτελεί πληροφορία που όμως δεν έχει ενταχθεί ακόμα σε κάποιο ευρύτερο πλαίσιο, δεν έχει δηλαδή αποκτήσει κάποιο ιδιαίτερο νόημα (Εικόνα 3.4). Είναι προφανές ότι αυτό που για κάποιον αποτελεί πληροφορία μπορεί την επόμενη στιγμή να αποτελέσει δεδομένο για κάποια άλλη διαδικασία ή λειτουργία κ.ο.κ. (Εικόνα 3.5).



Εικόνα 3.5: Η ανατροφοδότηση πληροφορίας-δεδομένων

3.2.2 Οργάνωση Αρχείων

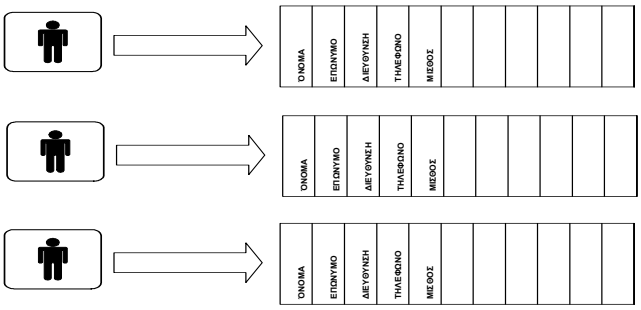
Προκειμένου να είμαστε σε θέση να χρησιμοποιούμε αποδοτικά τα δεδομένα πρέπει να τους δώσουμε δομή και οργάνωση. Μόνον έτσι μπορεί κάποιος να τα αναζητήσει και στη συνέχεια να τα ανακτήσει εύκολα.

Κατά την εποχή πριν από την έλευση των Συστημάτων Βάσεων Δεδομένων τα δεδομένα φυλάσσονταν σε αρχεία του Λειτουργικού Συστήματος που τα δημιουργούσαν τα προγράμματα εφαρμογών.

- Η λογική μονάδα οργάνωσης των αρχείων δεδομένων είναι συνήθως η *εγγραφή* (*record*). Κάθε εγγραφή αντιστοιχεί σε μία οντότητα (π.χ. υπάλληλος, ασφαλισμένος, υλικό αποθήκης, αθλητής κ.λπ.).
- Κάθε εγγραφή περιέχει μιά σειρά από πληροφορίες, τα *πεδία* (*fields*), που είναι οι ιδιότητες της εγγραφής, τα χαρακτηριστικά της δηλαδή που την περιγράφουν, σύμφωνα βέβαια με τις ειδικές απαιτήσεις της συγκεκριμμένης εφαρμογής.

Παράδειγμα 3.7: Μια εγγραφή για την οντότητα «Υπάλληλος»

Η οντότητα υπάλληλος παριστάνεται με εγγραφή που περιέχει ως πεδία το όνομα, επώνυμο, αριθμό μητρώου, διεύθυνση κατοικίας, μισθολογικό κλιμάκιο, προστατευόμενα μέλη κ.ο.κ.. Με τον τρόπο αυτό σε κάθε υπάλληλο αντιστοιχεί μία εγγραφή που περιέχει τα προσωπικά του δεδομένα (Εικόνα 3.6).



Εικόνα 3.6: Σε κάθε υπάλληλο αντιστοιχεί μια εγγραφή

Συνήθως, τα πληροφοριακά συστήματα απαρτίζονται από περισσότερα του ενός αρχεία δεδομένων, με τον προγραμματιστή να καθορίζει και να δημιουργεί τις συσχετίσεις μεταξύ τους. Στα δεδομένα αυτά έχει πρόσβαση το πρόγραμμα εφαρμογής, το οποίο αναζητά, ανασύρει, συνδυάζει, μεταβάλλει και γενικότερα τα επεξεργάζεται παράγοντας νέα δεδομένα και πληροφορία. Είναι γεγονός ότι και αυτά τα συστήματα πολλές φορές ονομάζονταν «βάσεις δεδομένων».

Ωστόσο, σε μιά τέτοια υλοποίηση όπου τα δεδομένα και οι εντολές επεξεργασίας τους είναι άρρηκτα συσχετισμένα παρουσιάζονται πολλά μειονεκτήματα, ανάμεσα στα οποία:

- τα δεδομένα είναι εξαρτημένα από τον προγραμματιστή ο οποίος καθορίζει τη δομή των αρχείων καθώς και τις διαδικασίες επεξεργασίας τους,

- τα αρχεία δεδομένων είναι «δεμένα» με τη γλώσσα προγραμματισμού που τα δημιούργησε, επειδή οι διάφορες γλώσσες προγραμματισμού δημιουργούν διαφορετικής τεχνολογίας αρχεία,
- το πρόγραμμα εφαρμογής που χρησιμοποιεί τα αρχεία δεδομένων περιέχει τον σχετικό κώδικα αναζήτησης και συνδυασμού τους, με αποτέλεσμα οι δυνατότητές του να είναι εκ των προτέρων καθορισμένες και σχετικά άκαμπτες,
- δεν είναι εύκολο να μεταβληθεί η δομή των αρχείων, ενώ η μεταβολή συνήθως απαιτεί μεγάλης έκτασης αλλαγές και στα προγράμματα,
- απαιτείται μεγάλη προσπάθεια από πλευράς του προγραμματιστή προκειμένου να εξασφαλίσει την ασφάλεια των δεδομένων και τα διαφορετικά επίπεδα στα δικαιώματα πρόσβασης των χρηστών σε αυτά,
- σε μεγάλες εφαρμογές όπου πολλοί χρήστες από κοινού επιδρούν και χειρίζονται ένα αρχείο είναι επίσης δύσκολο να ελεγχθεί η ακεραιότητα και εγκυρότητα των δεδομένων.

Η βασική διαφορά της υλοποίησης εφαρμογής με χρήση αρχείων δεδομένων από αυτή των συστημάτων βάσεων δεδομένων βρίσκεται στο ότι στις δεύτερες η ευθύνη αναζήτησης και ανάσυρσης των δεδομένων ανατίθεται πλέον σε ειδικά για το σκοπό αυτό σχεδιασμένο λογισμικό, το **Σύστημα Διαχείρισης της Βάσης Δεδομένων**.

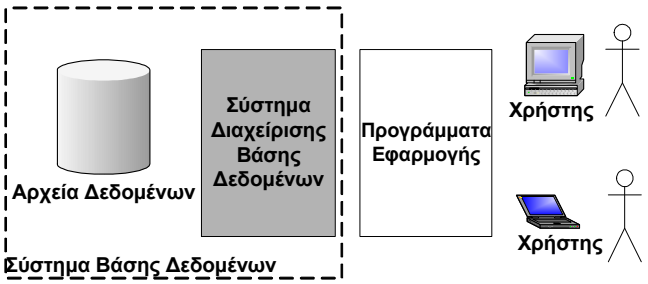
3.2.3 Βάσεις Δεδομένων

Μετά τη διαπίστωση των μειονεκτημάτων που σκιαγραφήσαμε παραπάνω, γίνεται φανερή η χρησιμότητα των Συστημάτων Βάσεων Δεδομένων (Σ.Β.Δ. ή και απλά Β.Δ., Database System). Μπορούμε να ορίσουμε μια Βάση Δεδομένων ως "*μια συλλογή από εγγραφές ή πληροφορίες, αποθηκευμένες ψηφιακά με κάποιο συστηματικό, δομημένο τρόπο έτσι ώστε να μπορεί ένα πρόγραμμα να την συμβουλεύεται για να παίρνει απαντήσεις σε ερωτήματα*".

Ειδικότερα ένα Σύστημα Β.Δ. αποτελείται από:

- τη Βάση Δεδομένων (database): ένα ή περισσότερα αρχεία-συλλογές δεδομένων που είναι κατάλληλα δομημένα, και
- το Σύστημα Διαχείρισης Βάσης Δεδομένων (database management system): μία συλλογή από προγράμματα που έργο τους είναι να επιτελούν όλες τις βασικές λειτουργίες πάνω στα αρχεία δεδομένων: δημιουργία, εισαγωγή, ενημέρωση, διαγραφή, αναζήτηση κλπ.

Μέσα από αυτή την οργάνωση, ο προγραμματιστής απαλλάσσεται από την υποχρέωση να γράφει τα προγράμματα της διαχείρισης και ανάκτησης των δεδομένων. Αυτή πλέον έχει ανατεθεί στο Σύστημα της Β.Δ. Ο προγραμματιστής μπορεί να ασχολείται με το μέρος μόνο των υπολογισμών και όποια άλλη ειδική απαίτηση έχει το πληροφοριακό σύστημα που υλοποιεί (Εικόνα 3.7). Έτσι, το Σύστημα Διαχείρισης Β.Δ. μπορεί να συνεργάζεται και να δέχεται ερωτήματα και εντολές από περισσότερες από μία εφαρμογές, τα οποία υλοποιεί επικοινωνώντας με τα αρχεία της Β.Δ..



Εικόνα 3.7: Σχέση Εφαρμογών με Βάσεις Δεδομένων

Από τα διάφορα μοντέλα Β.Δ. που έχουν προταθεί, το σχεσιακό έχει επικρατήσει στην αγορά λόγω της απλής, αλλά ισχυρής, δομής του που μπορεί να παρασταθεί μαθηματικά. Σήμερα στην αγορά κυκλοφορεί μεγάλος αριθμός από εφαρμογές βάσεων δεδομένων που στηρίζονται στο σχεσιακό μοντέλο (Relational Databases):

- SQL Server της Microsoft,
- Oracle της Oracle,
- DB2 και Informix της IBM,
- MySQL, λογισμικό ανοικτού κώδικα από την Oracle,
- PostgreSQL, λογισμικό ανοικτού κώδικα,
- Sybase, από την Sybase Inc, κ.λπ.

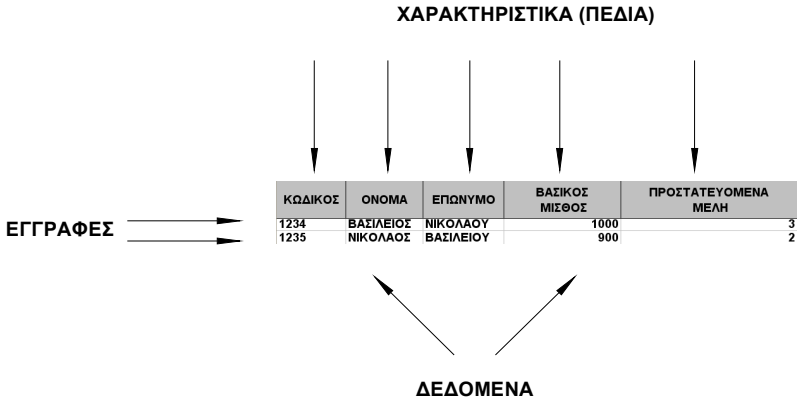
3.2.4 Πίνακες

Μία σχεσιακή Β.Δ. αποθηκεύει τα δεδομένα της σε **πίνακες**, ή αλλοιώς **σχέσεις** (Εικόνα 3.8). Κάθε πίνακας περιγράφει μία οντότητα με λογική αντίστοιχη των αρχείων, π.χ. ΥΠΑΛΛΗΛΟΣ, ΚΑΘΗΓΗΤΗΣ, ΧΩΡΑ, ΠΑΡΑΓΓΕΛΙΑ κλπ.

Ο πίνακας αποτελείται από γραμμές και στήλες.

- Κάθε στήλη λέγεται **χαρακτηριστικό** ή **ιδιότητα** ή **γνώρισμα** της εγγραφής. Όλα μαζί τα χαρακτηριστικά προσδιορίζουν την οντότητα.

- Κάθε γραμμή αντιστοιχεί σε μία εγγραφή, ένα στιγμιότυπο δηλαδή της οντότητας π.χ. σε έναν υπάλληλο, έναν καθηγητή, μιά χώρα, μία παραγγελία κ.ο.κ.



Εικόνα 3.8: Πίνακας Β.Δ.

Κάθε χαρακτηριστικό έχει έναν τύπο ανάλογα με το τι είδους δεδομένα θα αποθηκεύσει, π.χ. ακέραιος αριθμός, δεκαδικός, χαρακτήρας, αλφαριθμητικό, κλπ. Το σύνολο των επιτρεπτών τιμών του χαρακτηριστικού είναι το πεδίο ορισμού του.

Κλειδί ενός πίνακα ονομάζουμε ένα χαρακτηριστικό (στήλη) που μπορεί να διακρίνει (ξεχωρίσει) με μοναδικό τρόπο τη μία γραμμή (εγγραφή) από την άλλη.

Είναι δυνατόν το κλειδί να είναι ένας συνδυασμός από περισσότερα του ενός χαρακτηριστικά. Στην περίπτωση αυτή το κλειδί λέγεται **Σύνθετο**. Απαιτείται ιδιαίτερη προσοχή στην επιλογή του κλειδιού. Υποθέστε, για παράδειγμα, ότι η νομοθεσία επιτρέπει τη χορήγηση του ίδιου αριθμού κυκλοφορίας σε περισσότερα από ένα αυτοκίνητα, αφού αποσυρθεί το προηγούμενο. Άλλο παράδειγμα, η επιλογή ευαίσθητων δεδομένων (π.χ. ο αριθμός πιστωτικής κάρτας). Είναι πολύ πιθανό να θέλουμε να τον διατηρούμε κρυφό για λόγους ασφαλείας, πράγμα που καταργείται αν τον κάνουμε κλειδί.

Παράδειγμα 3.8: Καθορισμός κλειδιού σε έναν πίνακα Β.Δ.

Έστω ο πίνακας με τα στοιχεία αυτοκινήτων που κυκλοφορούν στην Ελλάδα. Υποψήφια κλειδιά μπορεί να είναι:

- ο αριθμός κυκλοφορίας του,
- ο αριθμός πλαισίου της μηχανής,
- κάθε άλλο χαρακτηριστικό που ικανοποιεί τις απαιτήσεις του κλειδιού,

- ένας ανεξάρτητος αριθμός μητρώου που θα ορίζεται από την εφαρμογή, μία συνηθισμένη πρακτική που μας απαλλάσσει από τα προβλήματα που περιγράψαμε.

Είναι δυνατόν στον ίδιο πίνακα να οριστούν περισσότερα από ένα κλειδιά. Στην περίπτωση αυτή ένα από αυτά ορίζεται να είναι το **Πρωτεύον** κλειδί, και όλα τα άλλα είναι **Δευτερεύοντα** κλειδιά.

Ένα κλειδί ενός πίνακα που βρίσκεται ως απλό χαρακτηριστικό σε έναν άλλο πίνακα, λέγεται **Ξένο** κλειδί. Τα Ξένα κλειδιά χρησιμοποιούνται για να συσχετίσουν (συνδέσουν) τους πίνακες (Σχήμα 6).

Εικόνα 3.9: Ξένο κλειδί

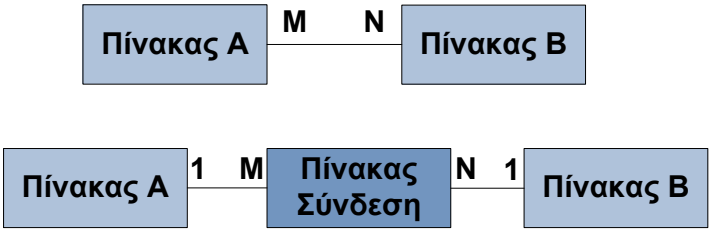
3.2.5 Σχέσεις και Κανονικοποίηση

Οι κατηγορίες συνδέσεων (συσχετίσεων) ανάμεσα σε πίνακες είναι τρείς:

ένα-προς-ένα (1:1). Κάθε στιγμιότυπο μιάς οντότητας Α μπορεί να συσχετιστεί με μόνο ένα στιγμιότυπο μιάς οντότητας Β και κάθε στιγμιότυπο από το Β μπορεί να σχετίζεται με μόνο ένα στιγμιότυπο από το Α. Συχνά όταν έχουμε συσχέτιση 1:1 προτιμάμε να χρησιμοποιούμε έναν πίνακα ενώνοντας τους δύο πίνακες σε έναν.

ένα-προς-πολλά (1:N). Κάθε ένα στιγμιότυπο μιάς οντότητας Α μπορεί να συσχετιστεί με οποιοδήποτε αριθμό από στιγμιότυπα της οντότητας Β, ενώ ένα στιγμιότυπο από την Β μπορεί να συσχετιστεί με ένα μόνο από την Α. Η συσχέτιση γίνεται με αντιγραφή του κλειδιού από την 1-πλευρά στη Ν-πλευρά ως ξένο κλειδί.

πολλά-προς-πολλά (M:N). Κάθε ένα στιγμιότυπο μιάς οντότητας Α μπορεί να συσχετιστεί με οποιοδήποτε αριθμό από στιγμιότυπα της οντότητας Β, και ένα στιγμιότυπο από την Β μπορεί να συσχετιστεί με οποιοδήποτε αριθμό από στιγμιότυπα της οντότητας Α. Η υλοποίηση μιάς συσχέτισης M:N (Σχήμα 7) γίνεται με δύο συσχετίσεις τύπου 1:N, χρησιμοποιώντας έναν τρίτο ενδιάμεσο **πίνακα σύνδεσης** (junction table).



Εικόνα 3.10: Σχέση M:N και πίνακας σύνδεσης

Προκειμένου να λειτουργήσει αποδοτικά μία βάση δεδομένων, απαιτείται να ακολουθούμε μία σειρά από κανόνες όταν σχεδιάζουμε τους πίνακες και τις συσχετίσεις τους. Κεντρικός πυρήνας των κανόνων είναι η ονομαζόμενη διαδικασία **κανονικοποίησης** (normalization) που υλοποιείται με μία σειρά από διαδοχικά βήματα (κανονικά επίπεδα). Η παρουσίαση των επιπέδων κανονικοποίησης Β.Δ. ξεφεύγει από τα πλαίσια του παρόντος συγγράμματος. Ωστόσο, μπορούμε να πούμε ότι τα επίπεδα έχουν σωρευτικό χαρακτήρα: η επόμενη μορφή απαιτεί καταρχήν να ισχύσει η προηγούμενη. Στον Πίνακα 3.3 μπορείτε να δείτε πότε μια Β.Δ. βρίσκεται σε $1^{η}$, $2^{η}$ και $3^{η}$ κανονική μορφή, που είναι τα τρία πρώτα κανονικά επίπεδα (1NF, 2NF, 3NF).

Πίνακας 3.3: 1NF, 2NF, 3NF

1η Κανονική Μορφή (1NF) όταν
1. Κάθε πίνακας έχει ένα πρωτεύον κλειδί. 2. Δεν επιτρέπεται να υπάρχουν πεδία που να είναι επαναλαμβανόμενα, να έχουν περισσότερες από μία ή σύνθετες τιμές. 3. Απαγορεύονται οι φωλιασμένες σχέσεις – «πίνακας μέσα σε πίνακα».
2η Κανονική Μορφή (2NF) όταν
1. Ο πίνακας είναι 1NF 2. Τα πεδία που δεν ανήκουν στο πρωτεύον κλειδί, είναι πλήρως εξαρτημένα από αυτό.
3η Κανονική Μορφή (3NF) όταν
1. Ο πίνακας είναι 2NF 2. Ένα πεδίο που δεν ανήκει στο πρωτεύον κλειδί πρέπει να εξαρτάται μόνο από αυτό. Δεν υπάρχει δηλαδή κάποιου τύπου μεταβατική εξάρτηση.

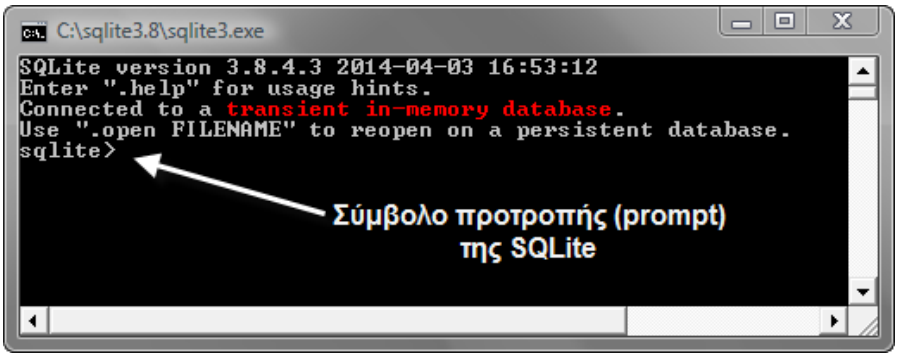
3.3 Σύνδεση και Διαχείριση βάσεων δεδομένων SQLite

3.3.1 Εισαγωγή στην SQLite

Η Βάση Δεδομένων SQLite είναι ένα σύστημα συμβατό με το ISO SQL. Η Β.Δ. ασφαλώς και δεν αποτελεί ανταγωνιστή των δυνατοτήτων άλλων Β.Δ. όπως οι Oracle και mySQL. Όμως, είναι μια ελαφριά υλοποίηση που χρησιμοποιείται ευρύτατα από περιηγητές ιστού (π.χ. firefox) και άλλες εφαρμογές που εκτελούνται σε περιβάλλοντα με μικρή υπολογιστική ισχύ, π.χ. φορητές συσκευές. Η SQLite δεν απαιτεί να τρέχει

ξεχωριστό σύστημα Β.Δ. (RDBMS Server) και εκτελείται όποτε χρειαστεί, επομένως δεν απαιτεί πρόσθετο χώρο στη μνήμη, ούτε ιδιαίτερη υπολογιστική ισχύ. Η έκδοσή της για το περιβάλλον Windows είναι ένα και μόνον εκτελέσιμο αρχείο (sqlite3.exe), δεν απαιτεί καμία ρύθμιση, ούτε ιδιαίτερη εγκατάσταση. Η SQLite αποθηκεύει ολόκληρη τη βάση με τα δεδομένα σε ένα μόνο αρχείο, πράγμα που καθιστά εύκολη τη μεταφορά της, ενώ παράλληλα είναι και πολύ γρήγορη για μικρό ή μεσαίο αριθμό εγγραφών.

Τα περισσότερα σύγχρονα συστήματα Linux έρχονται με ήδη εγκατεστημένη την SQLite. Για να την εγκαταστήσετε στα Windows, αλλά και όπου αλλού την χρειάζεστε, θα κατεβάσετε τα σχετικά αρχεία (precompiled binaries) από τον επίσημο δικτυακό της τόπο (https://sqlite.org/). Κατεβάστε το αρχείο και αποσυμπιέστε το σε έναν φάκελο της αρεσκείας σας, κατά προτίμηση χωρίς ελληνικούς χαρακτήρες στο όνομα της διαδρομής, π.χ. σε ένα φάκελο της μορφής C:\sqlite. Τρέξτε το εκτελέσιμο αρχείο που θα δημιουργηθεί και θα βρεθείτε στο κέλυφος (shell) της Β.Δ. (Εικόνα 3.11).



Εικόνα 3.11: Το περιβάλλον εντολών της SQLite

Στο σύμβολο προτροπής **sqlite>** μπορούμε να δίνουμε είτε εντολές με τις οποίες διαχειριζόμαστε το περιβάλλον της Β.Δ. ή εντολές με τις οποίες διαχειριζόμαστε τις βάσεις δεδομένων μας (κυρίως εντολές SQL). Δεν είναι δυνατόν σε ένα σύγγραμμα που αφορά στην Python να καλύψουμε όλες τις δυνατότητες του περιβάλλοντος SQLite αλλά ούτε και τη γλώσσα SQL. Για το λόγο αυτό θα αρκεστούμε στις σημαντικότερες σχετικές εντολές και στη σύνδεσή τους με την Python.

Οι εντολές περιβάλλοντος ξεκινάνε με τελεία. Με αυτές μπορούμε να ρυθμίσουμε θέματα που αφορούν στην εμφάνιση της πληροφορίας, να πάρουμε βοήθεια για τις εντολές, να δημιουργήσουμε ή να συνδεθούμε με Β.Δ., κ.ά. Μπορείτε να δείτε τις κυριότερες από αυτές στον Πίνακα 3.4, που ακολουθεί.

Πίνακας 3.4: Εντολές περιβάλλοντος SQLite

.help	Εμφανίζει συνοπτική βοήθεια
.exit ή .quit	Έξοδος από το περιβάλλον SQLite
.databases	Εμφανίζει τις συνδεδεμένες Β.Δ.
.tables	Εμφανίζει τους πίνακες της ανοικτής Β.Δ.
.headers on/off	Ρυθμίζει αν θα εμφανίζονται επικεφαλίδες στις στήλες των αποτελεσμάτων
.mode	Ρυθμίζει τη μορφή που θα εμφανίζονται τα αποτελέσματα. Μπορεί να είναι “csv”, “column”, “html”, “insert”, “line”, “list”, “tabs”, “tcl”. Η συνηθισμένη επιλογή είναι η “column”.
.open <filename>	Ανοίγει μια υπάρχουσα Β.Δ. με όνομα <filename> ή δημιουργεί μια νέα με αυτό το όνομα, αν δεν υπάρχει.
.schema <πίνακας>	Εμφανίζει τις εντολές SQL που δημιούργησαν τον πίνακα

Δημιουργία/Άνοιγμα βάσης δεδομένων.

Σε ένα παράθυρο εντολών πληκτρολογούμε *sqlite3 <όνομα ΒΔ>*. Περνάμε σε περιβάλλον sqlite και αν το όνομα που δώσαμε υπάρχει στον φάκελο, ανοίγει σύνδεση με τη βάση, διαφορετικά, αν δεν υπάρχει, τότε δημιουργείται μια νέα Β.Δ..

Ισοδύναμα, μέσα από το περιβάλλον της sqlite, μπορούμε να χρησιμοποιήσουμε την εντολή *.open <όνομα ΒΔ>* για να συνδεθούμε ή να δημιουργήσουμε μια νέα Β.Δ. (Εικόνα 3.12).

```
sqlite> .open my.db
sqlite> .databases
seq  name             file
---  ---------------  ----------------------
0    main             c:\sqlite308\my.db
sqlite>
```

Εικόνα 3.12: Δημιουργία Β.Δ.

Η διαχείριση της Β.Δ. γίνεται μέσα από εντολές SQL, που τις βασικότερες από αυτές θα τις δούμε παρακάτω. Σημειώστε ότι, στη σύνταξη των εντολών που περιγράφουμε, οι αγκύλες σημαίνουν προαιρετικό τμήμα, ενώ οι τελείες δυνατότητα επανάληψης. Παρατηρήστε ακόμα, ότι οι εντολές τερματίζονται με το ελληνικό ερωτηματικό. Στην περίπτωση που πατήσουμε enter πριν ολοκληρωθεί η εντολή, η sqlite μας περνά σε prompt αναμονής **...>** όπου μπορούμε να συνεχίσουμε τη σύνταξη της εντολής, σηματοδοτώντας το τέλος της με το ελληνικό ερωτηματικό. Στην ενότητα 3.3.2 θα δούμε πώς αυτές οι εντολές χρησιμοποιούνται στον κώδικα Python.

Δημιουργία πίνακα σε βάση δεδομένων.

CREATE TABLE <όνομα_πίνακα> (<πεδίο1> <τύπος1> [(μέγεθος)], [...]);

Στην Εικόνα 3.13 δημιουργούμε έναν πίνακα στη Β.Δ. my.db που ανοίξαμε νωρίτερα. Ο πίνακας έχει όνομα person και περιέχει τα πεδία name, surname, address που είναι κείμενο (text) και age που είναι ακέραιος (integer).

```
sqlite> create table person
   ...> (name text, surname text, address text, age integer);
sqlite> .schema
CREATE TABLE person
(name text, surname text, address text, age integer);
```

Εικόνα 3.13: Δημιουργία πίνακα στην SQLite

Η εντολή Create Table έχει αρκετές δυνατότητες και η σύνταξή της μπορεί να είναι αρκετά πιο σύνθετη από το παράδειγμα της Εικόνας 3.13. Για παράδειγμα, μπορείτε να καθορίσετε περιορισμούς στα πεδία, όπως να μην επιτρέπεται να είναι κενό, να έχει κάποια εξ ορισμού τιμή, κ.λπ. Στο δικτυακό τόπο της sqlite μπορείτε να βρείτε την πλήρη περιγραφή της σύνταξης αυτής της εντολής όσο και όλων των άλλων που, ενδεικτικά, θα δούμε στο παρόν σύγγραμμα.

Σε κάθε πίνακα που δημιουργείται στη Β.Δ., δίνεται από το σύστημα ένα πρωτεύον κλειδί που είναι ένας ακέραιος με πρόσημο και μήκος 64 bit. Το πεδίο αυτό εισάγεται σε κάθε πίνακα αυτόματα. Η SQLite το αναγνωρίζει με όνομα rowid, oid ή και _rowid_ . Μάλιστα, αν ορίσουμε εμείς ως κλειδί κάποιο πεδίο τύπου ακεραίου, τότε αυτό ταυτίζεται με το rowid.

Εισαγωγή τιμών σε πίνακα.

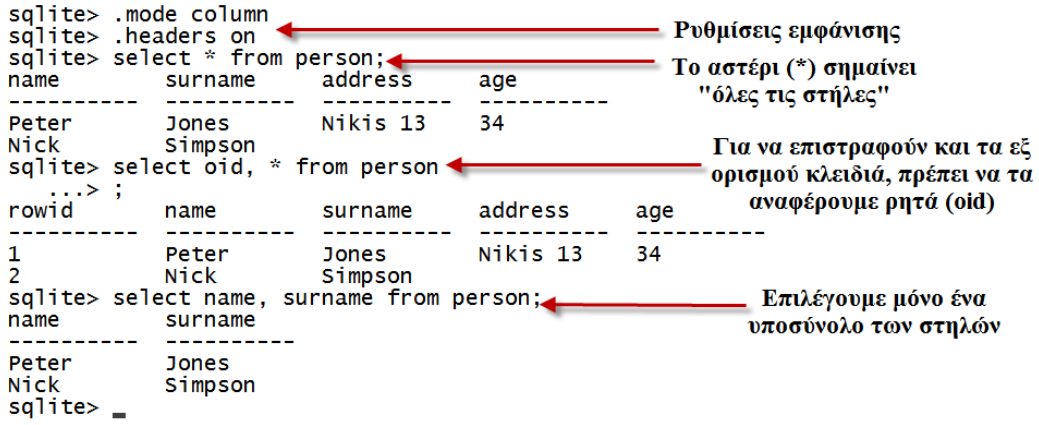
INSERT INTO όνομα_πίνακα (όνομα_πεδίου1 [, ...]) VALUES (νέα_τιμή1 [, ...]);

Εφόσον εισάγονται τιμές ισάριθμες με το πλήθος των στηλών του πίνακα, τότε δεν χρειάζεται να αναφέρουμε τα ονόματα των πεδίων, διότι η αντιστοίχηση γίνεται αυτόματα. Σε διαφορετική περίπτωση, θα πρέπει να αναφέρουμε τα ονόματα των στηλών και στη συνέχεια τις αντίστοιχες τιμές. Στην Εικόνα 3.14 μπορείτε να δείτε τρεις περιπτώσεις. Στην πρώτη εντολή INSERT INTO δίνουμε τιμές για όλα τα πεδία του πίνακα, στη δεύτερη μόνο για το όνομα και το επώνυμο (name, surname), ενώ στην τρίτη μπορείτε να δείτε το μήνυμα λάθους που εμφανίζεται όταν δεν καθορίσουμε σε ποιες στήλες του πίνακα αναφερόμαστε.

```
sqlite> insert into person values ("Peter","Jones", "Nikis 13",34);      1
sqlite> insert into person (name, surname) values ("Nick","Simpson");   2
sqlite> insert into person values ("AAA","BBB");                        3
Error: table person has 4 columns but 2 values were supplied
sqlite>
```

Εικόνα 3.14: Εισαγωγή δεδομένων σε πίνακα SQLite

Θα δούμε περισσότερα για την εντολή SELECT παρακάτω, ωστόσο, για την ώρα ας χρησιμοποιήσουμε την απλούστερη μορφή της για να δούμε τα δεδομένα που μόλις εισαγάγαμε στον πίνακα person. Το αποτέλεσμα το βλέπουμε στην Εικόνα 3.15. Νωρίτερα έχουμε ρυθμίσει την εμφάνιση των αποτελεσμάτων σε μορφή στηλών (.mode column) με εμφάνιση επικεφαλίδων (.headers on). Παρατηρήστε ότι παρόλο που με τον χαρακτήρα * μπορούμε να αναφερθούμε σε όλες τις στήλες του πίνακα, για να ανακτήσουμε και το κλειδί που καθορίζεται αυτόματα από την SQLite πρέπει να αναφερθούμε ειδικά σε αυτό (oid). Αν θέλουμε να ανακτήσουμε ορισμένες στήλες, δεν έχουμε παρά να αναφερθούμε σε αυτά με το όνομά τους.

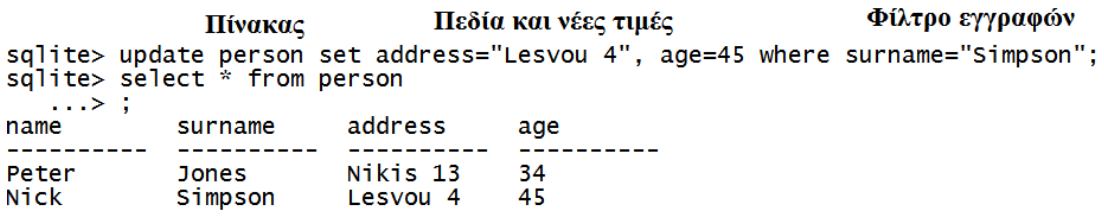


Εικόνα 3.15: Ανάκτηση εγγραφών με τη Select από ΒΔ SQLite

Ενημέρωση τιμών σε πίνακα.

UPDATE όν_πίνακα SET όν_πεδίου = νέα_τιμή [, ...] WHERE Έκφραση-φίλτρο;

Η εντολή έχει τρία τμήματα. Καταρχήν καθορίζουμε σε ποιόν πίνακα θα γίνει η λειτουργία της ενημέρωσης (όν_πίνακα). Στο τμήμα της εντολής μετά το SET, καθορίζουμε τα πεδία που θα ενημερωθούν και τις νέες τιμές που θα πάρουν, σε ζεύγη αναθέσεων του τύπου όν_πεδίου = τιμή, χωρισμένα με κόμμα. Η έκφραση WHERE μπορεί να προσδιορίσει μια μοναδική εγγραφή (π.χ. Where oid = 1) ή ένα υποσύνολο των εγγραφών (π.χ. where age >30). Στην περίπτωση που το where απουσιάζει, τότε οι αλλαγές γίνονται σε όλες τις γραμμές του πίνακα. Στην Εικόνα 3.16 βλέπετε ένα παραδειγμα ενημέρωσης μιας συγκεκριμένης εγγραφής, καθώς και το αποτέλεσμα της εντολής αυτής.

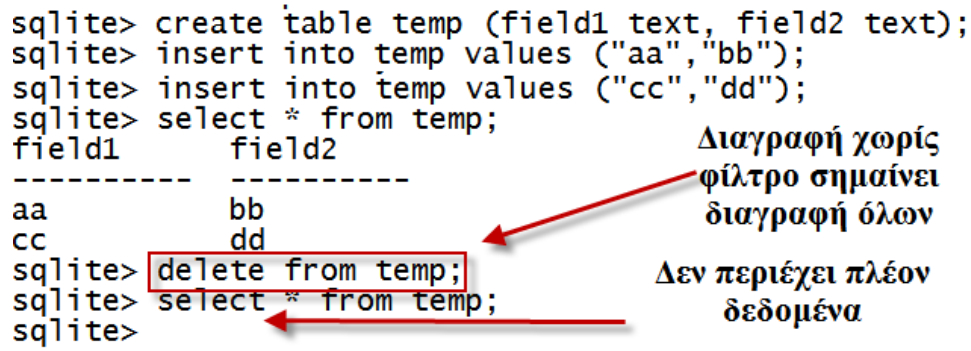


Εικόνα 3.16: Ενημέρωση πεδίων πίνακα SQLite

Διαγραφή γραμμών σε πίνακα.

DELETE FROM όνομα_πίνακα WHERE έκφραση-φίλτρο;

Η εντολή θα διαγράψει από τον πίνακα του οποίου το όνομα προσδιορίζουμε μετά τη δήλωση FROM, εκείνες τις εγγραφές (γραμμές) που ικανοποιούν τα κριτήρια που περιλαμβάνουμε στην έκφραση μετά το WHERE. Όπως και στην περίπτωση της ενημέρωσης, η έκφραση που ακολουθεί το WHERE μπορεί να μην επιλέγει καμία γραμμή, οπότε και δεν θα έχουμε καμία διαγραφή. Στην περίπτωση που δεν έχουμε καθορίσει έκφραση WHERE, διαγράφονται όλες οι γραμμές του πίνακα (αλλά όχι ο ίδιος ο πίνακας, που παραμένει κενός).



Εικόνα 3.17: Διαγραφή περιεχομένων πίνακα

Όπως μπορείτε να δείτε στην Εικόνα 3.17, δημιουργήσαμε στη ΒΔ έναν πίνακα με όνομα temp, εισαγάγαμε δύο εγγραφές και στη συνέχεια χρησιμοποιήσαμε την εντολή DELETE χωρίς να θέσουμε κριτήριο WHERE, γεγονός που είχε σαν αποτέλεσμα τη διαγραφή όλων των γραμμών. Αν θέλαμε να διαγράψουμε την πρώτη μόνο εγγραφή, θα μπορούσαμε, για παράδειγμα, να δώσουμε την εντολή: *delete from temp where field2 = "bb";*

Ανάκτηση δεδομένων από ΒΔ.

SELECT λίστα_επιστροφής FROM πηγή_δεδομένων WHERE Έκφραση-φίλτρο;

Η εντολή SELECT είναι ιδιαίτερα ισχυρό εργαλείο της γλώσσας SQL και μπορεί να πάρει πολύ σύνθετες μορφές. Η μορφή που περιγράψαμε παραπάνω είναι η γενική σύνταξη της πλέον τυπικής περίπτωσης SELECT. Η λίστα_επιστροφής είναι μια λίστα πεδίων από έναν πίνακα (ή και περισσότερους), η πηγή_δεδομένων είναι συνήθως ένας ή περισσότεροι πίνακες και η έκφραση_φίλτρο περιορίζει τις εγγραφές που θα επιλεγούν, σε αυτές που ικανοποιούν τη συνθήκη.

Ήδη στις Εικόνες 3.15 και 3.16 έχουμε δει απλές περιπτώσεις χρήσης της εντολής αυτής. Η παρουσίαση όλων των δυνατοτήτων της SELECT ξεφεύγει από τους σκοπούς του παρόντος και για το λόγο αυτό θα περιοριστούμε στα παραδείγματα αυτά

καθώς και στα παραδείγματα που βλέπετε στην Εικόνα 3.18. Μπορείτε να βρείτε πληθώρα πηγών για την εντολή αυτή (και γενικά για τη γλώσσα SQL) στο διαδίκτυο.

```
sqlite> select * from person;
name        surname     address     age
----------  ----------  ----------  ----------
Peter       Jones       Nikis 13    34
Nick        Simpson     Lesvou 4    45
sqlite> select name, surname from person order by age desc;
name        surname
----------  ----------
Nick        Simpson
Peter       Jones
sqlite> select * from person where age < 40;
name        surname     address     age
----------  ----------  ----------  ----------
Peter       Jones       Nikis 13    34
```

Ταξινόμηση

Εικόνα 3.18: Παραδείγματα SELECT

3.3.2 Python και SQLite

Η Python, από την έκδοση 2.5 και μετά, υποστηρίζει εγγενώς τη σύνδεση και διαχείριση ΒΔ SQLite μέσα από το module sqlite3 το οποίο θα πρέπει απλά να κάνουμε import στο πρόγραμμά μας.

Έλεγχος αριθμού έκδοσης του module.

Για να δούμε ποιά έκδοση του module έχουμε εγκατεστημένη καθώς και ποια έκδοση της SQLite αυτό υποστηρίζει, μπορούμε να περάσουμε στο κέλυφος της Python (επιλέγοντας Run/Python Shell μέσα από το IDLE).

```
Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2
tel)] on win32
Type "copyright", "credits" or "license()"
>>> import sqlite3
>>> sqlite3.version
'2.6.0'
>>> sqlite3.sqlite_version
'3.7.12'
>>>
>>>
```

Εισαγωγή του module

Αριθμός έκδοσης του module

Έκδοση της SQLite

Εικόνα 3.19: Έλεγχος έκδοσης του module sqlite3

Δημιουργία και σύνδεση με ΒΔ sqlite-Δημιουργία πίνακα.

Για να συνδεθούμε με μία ΒΔ sqlite μέσα από τον κώδικα Python, ακολουθούμε τη διαδικασία που μπορεί να συνοψιστεί στα ακόλουθα, γενικά, βήματα:

1. Δημιουργούμε ένα αντικείμενο Connection με τη μέθοδο connect() η οποία δέχεται ως παράμετρο το όνομα του αρχείου (ή την πλήρη διαδρομή, αν βρίσκεται σε άλλο φάκελο, εκτός του τρέχοντος) και επιστρέφει το αντικείμενο:

connection = sqlite3.connect ("πλήρης_διαδρομή_αρχείου")

Αν το όρισμα που δίνουμε είναι Β.Δ. που ήδη υπάρχει στο φάκελο, τότε αυτή ανοίγει, ενώ αν η Β.Δ. δεν υπάρχει, τότε δημιουργείται μια νέα, κενή.

2. Μπορούμε να έχουμε πρόσβαση στη Β.Δ. μέσω του αντικειμένου connection, αλλά για να είμαστε σύμφωνοι με τις προδιαγραφές των Β.Δ. (και επομένως σίγουροι ότι θα μπορούμε να εκτελέσουμε οποιαδήποτε λειτουργία θέλουμε), δημιουργούμε ένα αντικείμενο Cursor:

cursor = connection.cursor( )

3. Χρησιμοποιώντας την μέθοδο execute() του αντικειμένου cursor, μπορούμε να εκτελέσουμε οποιαδήποτε εντολή SQL θέλουμε, ως εξής:

cursor.execute ("Εγκυρη_εντολή_SQL")

Παράδειγμα 3.9: Κώδικας Python για δημιουργία Β.Δ. sqlite

```
import sqlite3
conn = sqlite3.connect("B://tmp//movies.db")
cursor = conn.cursor()
cursor.execute('''create table film
(filmname text not null, director text, year integer)''')
cursor.close()
print("database created")
```

Ο παραπάνω κώδικας δημιουργεί μια Β.Δ. με όνομα movies.db και έναν πίνακα με όνομα film και πεδία filmname, director, year (όνομα ταινίας, σκηνοθέτης, έτος κυκλοφορίας). Παρατηρήστε πώς εισάγουμε το πλήρες όνομα του φακέλου με τους χαρακτήρες διαφυγής (/). Επίσης, παρατηρήστε το αλφαριθμητικό όρισμα της μεθόδου execute που περικλείεται σε τριπλά εισαγωγικά, ώστε να μπορεί να επεκταθεί σε περισσότερες της μιας γραμμές. Προσέξτε ότι δεν ελέγχουμε προηγουμένως μήπως η βάση ήδη υπάρχει. Στην περίπτωση που η βάση υπάρχει ήδη, ο κώδικας δημιουργεί ένα σύνδεσμο με αυτήν. Η εντολή δημιουργίας του πίνακα, μπορεί να εκτελεστεί κανονικά αν ο πίνακας δεν περιέχεται ήδη στη βάση, αν όμως περιέχεται θα έχουμε εξαίρεση (runtime error). Είναι ευθύνη του προγραμματιστή να προβλέψει όλα αυτά τα πιθανά σενάρια, αλλά στο παράδειγμά μας παραλείπουμε τους ελέγχους για να επικεντρώσετε την προσοχή σας στις βασικές εντολές της δημιουργίας/ανοίγματος της Β.Δ.. Μπορείτε, αφού εκτελέσετε τον κώδικα, στη συνέχεια να περάσετε στο περιβάλλον της sqlite και να ελέγξετε αν η Β.Δ. και ο πίνακας δημιουργήθηκαν κανονικά (δείτε σχετικά στην Εικόνα 3.20).

```
sqlite> .open b:/tmp/movies.db
sqlite> .schema
CREATE TABLE film
(filmname text not null, director text, year integer);
sqlite>
```

Εικόνα 3.20: Αποτέλεσμα εκτέλεσης κώδικα Python για δημιουργία Β.Δ. και πίνακα

Αποθήκευση δεδομένων σε πίνακα ΒΔ sqlite.

Για να αποθηκεύσουμε (αλλά και για να ενημερώσουμε) δεδομένα σε πίνακα μιας Β.Δ., πρέπει καταρχήν να έχουμε συνδεθεί με τη βάση, η οποία θα περιέχει ήδη τον πίνακα. Στη συνέχεια, εκτελούμε την σχετική εντολή SQL όπως μπορείτε να δείτε στο Παράδειγμα 3.10. Η αποθήκευση των δεδομένων στη Β.Δ. γίνεται μόνον εφόσον εκτελέσουμε τη μέθοδο commit() της connection. Στην Εικόνα 3.21 μπορείτε να δείτε τόσο αυτόν όσο και έναν δεύτερο τρόπο σύνταξης της εντολής insert, όπου χρησιμοποιούμε μπαλαντέρ (αγγλικό ερωτηματικό, ?) σε αριθμό όδιο με τα πεδία που ενημερώνουμε, ενώ οι τιμές ακολουθούν σε μορφή πλειάδας (tuple).

```
import sqlite3
conn = sqlite3.connect("B://tmp//movies.db")
cursor = conn.cursor()
cursor.execute('''insert into film(filmname, director, year)
values ("World War Z"," Marc Forster",2013)''')
datatup = ('Arthur','Steve Gordon',1981)
cursor.execute('''insert into film(filmname, director, year)
values (?,?,?)''',datatup)
conn.commit()
cursor.close()
conn.close()
print("data saved")
```

Α' τρόπος Εγγραφή με execute SQL εντολή

Β' τρόπος, 1ο βήμα Δημιουργία πλειάδας με τα δεδομένα

Β' τρόπος, 2ο βήμα execute SQL εντολή, με μπαλαντέρ

Αποθήκευση

Εικόνα 3.21: Αποθήκευση δεδομένων σε πίνακα ΒΔ sqlite

Ανάκτηση δεδομένων από πίνακα ΒΔ sqlite.

Εκτελούμε την κατάλληλη εντολή select προκειμένου να φιλτράρουμε τα δεδομένα που θέλουμε. Στη συνέχεια μπορούμε να χρησιμοποιήσουμε είτε τη μέθοδο fetchone() για να ανακτήσουμε την επόμενη διαθέσιμη γραμμή ή τη fetchall() για να ανακτήσουμε όλες όσες ικανοποιούν τα κριτήρια που θέσαμε με το select. Σε περίπτωση που δεν υπάρχει γραμμή, επιστρέφεται η τιμή None.

Στο Παράδειγμα 3.10 που ακολουθεί, μπορείτε να δείτε τρεις περιπτώσεις χρήσης των δυο αυτών μεθόδων για ανάκτηση δεδομένων. Παρατηρήστε ότι σε όλες αρχικά εκτελούμε το ερώτημα (query) για να φιλτράρουμε τα δεδομένα/ Στην περίπτωση (1) ανακτούμε την πρώτη εγγραφή που περιέχεται στο ερώτημα, στην περίπτωση (2) τις ανακτούμε όλες και τις εμφανίζουμε με μια εντολή επανάληψης for, ενώ στην περίπτωση (3) η ανάκτηση γίνεται μια γραμμή κάθε φορά. Παρατηρήστε τη χρήση του None για τον έλεγχο τερματισμού του while.

Παράδειγμα 3.10: Ανάκτηση δεδομένων από Β.Δ. Sqlite

```
import sqlite3
conn = sqlite3.connect("B://tmp//movies.db")
cursor = conn.cursor()
cursor.execute('select director from film where year = 2013')
onerow = cursor.fetchone()                                          1
print(onerow)
print("================")
cursor.execute('select * from film')
rows=cursor.fetchall()
for record in rows:                                                 2
    print(record)
print("================")
cursor.execute('select rowid, director from film ')
row=cursor.fetchone()
while row is not None:                                              3
    print(row)
    row=cursor.fetchone()
print("================")
cursor.close()
conn.close()
```

Στην Εικόνα 3.22 μπορείτε να δείτε το αποτέλεσμα της εκτέλεσης του κώδικα.

```
(' Marc Forster',)
================
('World War Z', ' Marc Forster', 2013)
('Arthur', 'Steve Gordon', 1981)
================
(1, ' Marc Forster')
(2, 'Steve Gordon')
================
```

Εικόνα 3.22: Αποτέλεσμα εκτέλεσης κώδικα ανάκτησης δεδομένων από ΒΔ Sqlite

Διαγραφή δεδομένων από πίνακα ΒΔ sqlite.

Η διαγραφή υλοποιείται μέσα από την εκτέλεση της κατάλληλης SQL εντολής delete. Η ασφαλέστερη προσέγγιση είναι να γίνει η διαγραφή με βάση το κλειδί της γραμμής, χωρίς φυσικά αυτό να είναι απαραίτητο. Οι διαγραφές αποθηκεύονται μόνον όταν εκτελεστεί η μέθοδος commit() της connection. Στο Παράδειγμα 3.11 εμφανίζονται μια-μια οι εγγραφές του πίνακα και διαγράφεται αυτή που θα επιλέξει ο χρήστης. Παρατηρήστε ότι στη select που εκτελούμε για να ανασύρουμε όλες τις εγγραφές της Β.Δ. δηλώνουμε ρητά το rowid που είναι το αυτόματα οριζόμενο μοναδικό κλειδί της κάθε γραμμής. Όπως μπορείτε να δείτε, το χρησιμοποιούμε για να διαγράψουμε τη γραμμή που επιλέγει ο χρήστης και αν δεν το δηλώσουμε αυτό δεν επιστρέφεται αυτόματα. Το πεδίο rowcount της cursor.execute() επιστρέφει το πλήθος των γραμμών που επηρεάστηκαν (στην περίπτωσή μας, διαγράφτηκαν) από την εκτέλεση της εντολής SQL. Η rows είναι μια λίστα, που κάθε στοιχείο της είναι μια

πλειάδα με τα πεδία κάθε γραμμής. Στην περίπτωση της Β.Δ. movies.db και του πίνακα film η rows θα έχει περιεχόμενο:

[('World War Z', ' Marc Forster', 2013, 1), ('Arthur', 'Steve Gordon', 1981, 2)]

Ενώ κάθε πλειάδα record περιέχει τέσσερα στοιχεία, με δείκτες από το μηδέν μέχρι και το τρία, με record[3] να είναι το κλειδί.

Παράδειγμα 3.11: Διαγραφή γραμμής από πίνακα Β.Δ. sqlite

```
import sqlite3
conn = sqlite3.connect("B://tmp//movies.db")
cursor = conn.cursor()
cursor.execute('select *, rowid from film')
rows=cursor.fetchall()
for record in rows:
    print("Movie title:",record[0],"\nDirector:",record[1],"\nYear:",record[2])
    ans = input("Delete? (y/n):")
    if ans == 'y':
        s=cursor.execute("delete from film where film.rowid={}".format(record[3])).rowcount
        print(str(s),"record deleted!")
        break
conn.commit()
cursor.close()
conn.close()
```

Άσκηση 3.3: Γράψτε ένα πρόγραμμα που θα ανοίγει τη Β.Δ. movies.db, θα εμφανίζει μια-μια τις εγγραφές της και ο χρήστης θα επιλέγει ποια από αυτές θέλει να τροποποιήσει. Μόλις αυτή επιλεγεί από τον χρήστη, το πρόγραμμα θα εμφανίζει μια οθόνη όπου θα πληκτρολογεί τα νέα δεδομένα για τα πεδία της εγγραφής και στη συνέχεια θα ενημερώνεται η Β.Δ. (με την κατάλληλη SQL εντολή update).

3.4 Σύνδεση και Διαχείριση βάσεων δεδομένων mySQL

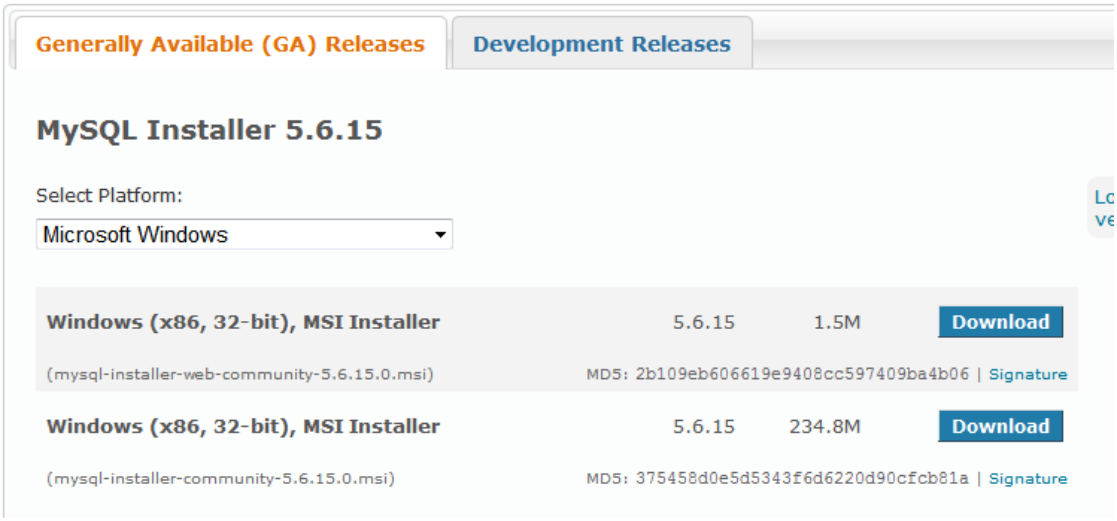
Η MySQL είναι το πιο διαδεδομένο Σύστημα Διαχείρισης Βάσεων Δεδομένων (Relational Data Base Manamgement System) ανοικτού κώδικα και υποστηρίζεται σχεδόν από όλα τα λειτουργικά συστήματα. Η σύνδεση του πελάτη με τη βάση και η διαχείριση της, γίνεται μέσω δικτύου. Για να δημιουργήσουμε μια βάση δεδομένων στη MySQL και να μπορούμε να τη διαχειριστούμε με οποιαδήποτε εφαρμογή ή γλώσσα προγραμματισμού, θα πρέπει πρώτα να την εγκαταστήσουμε - αν δεν υπάρχει ήδη. Εν συνεχεία, για να δημιουργήσουμε εφαρμογές με τη Python θα πρέπει, μετά την εγκατάσταση της MySQL, να εγκαταστήσουμε επιπλέον και το απαραίτητο πρόγραμμα που περιέχει τους οδηγούς για τη σύνδεση και τη διαχείριση της, με τη συγκεκριμένη γλώσσα προγραμματισμού.

3.4.1 Εγκατάσταση της εφαρμογής MySQL

Υπάρχουν δύο τρόποι για να εγκαταστήσουμε την εφαρμογή MySQL σε έναν υπολογιστή.

Α Τρόπος

Από τη σελίδα http://dev.mysql.com/downloads/installer κατεβάζουμε τη τελευταία έκδοση της Β.Δ., για το περιβάλλον των Windows.



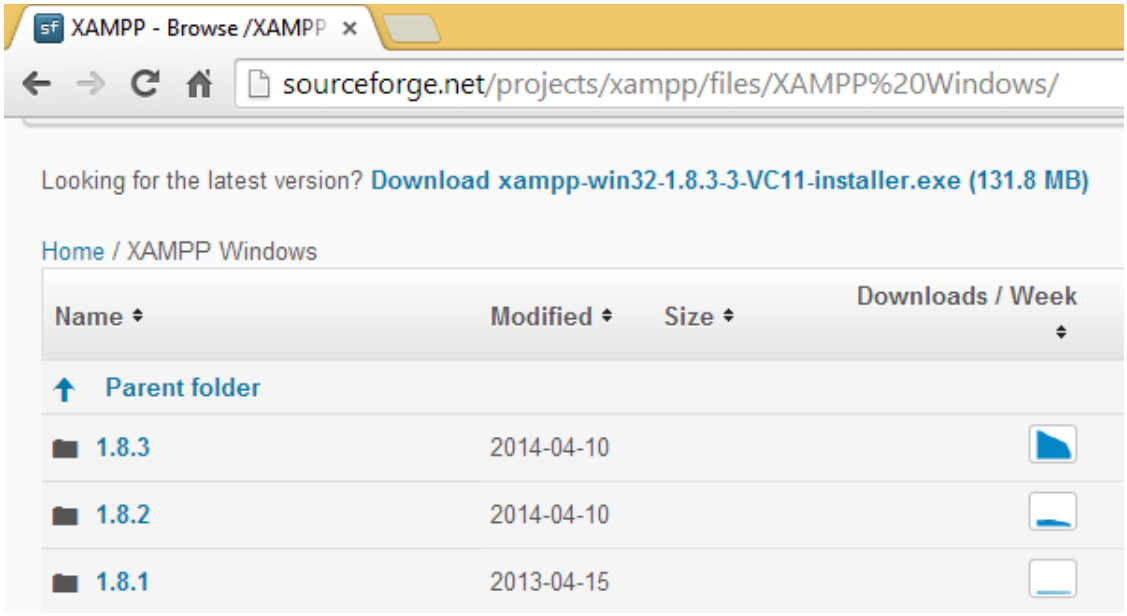
Εικόνα 3.23: Επιλογή του προγράμματος εγκατάστασης

Υπάρχουν δύο προγράμματα που διαφέρουν σημαντικά σε μέγεθος, δες Εικόνα 3.23. Το αρχείο με το μικρότερο μέγεθος (πρώτο στη σειρά) αφού το κατεβάσουμε και το τρέξουμε θα ανιχνεύσει την έκδοση του λειτουργικού μας και εν συνεχεία αυτόματα θα κατεβάσει τα απαραίτητα αρχεία, ενώ το δεύτερο αρχείο - αυτό με το μεγάλο μέγεθος - θα ξεκινήσει αμέσως την εγκατάσταση της Β.Δ. Η εγκατάσταση της Β.Δ., βήμα προς βήμα υπάρχει στο δικτυακό τόπο της MySQL στη διεύθυνση http://dev.mysql.com/doc/refman/5.6/en/mysql-installer-gui.html.

Β Τρόπος

Ο δεύτερος τρόπος είναι και ο προτεινόμενος, αφού δεν χρειάζεται να εγκαταστήσουμε τίποτα στον υπολογιστή μας. Θα πρέπει να κατεβάσουμε και να αποσυμπιέσουμε το πακέτο XAMPP (αν δεν υπάρχει ήδη) το οποίο περιέχει τη MySQL, σε μορφή άμεσης εκτέλεσης, χωρίς εγκατάσταση.

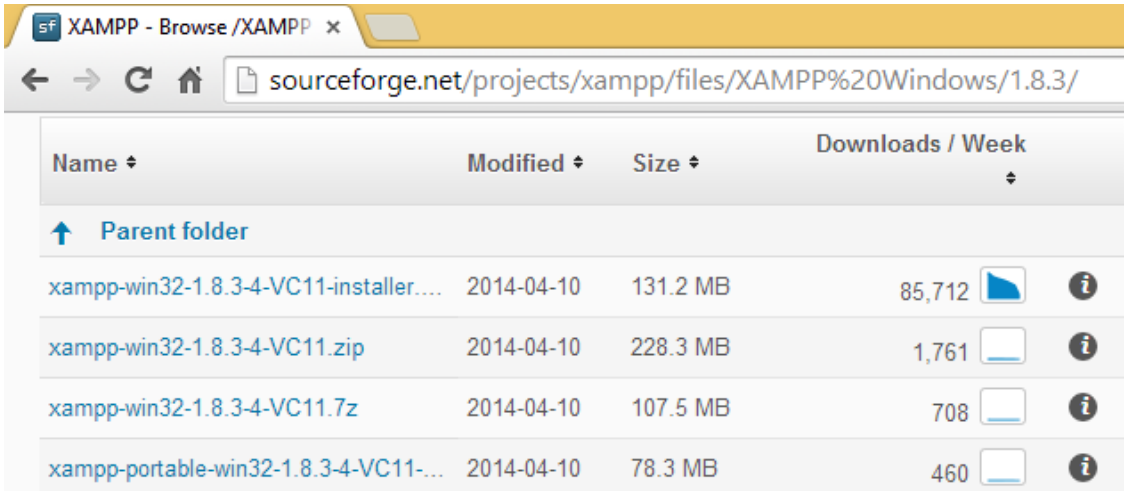
Από τη σελίδα http://sourceforge.net/projects/xampp/files/XAMPP%20Windows/ επιλέγουμε το φάκελο με τη τελευταία έκδοση του πακέτου XAMPP, δες Εικόνα 3.24, για το περιβάλλον των Windows.



Εικόνα 3.24: Επιλογή της έκδοσης του XAMPP

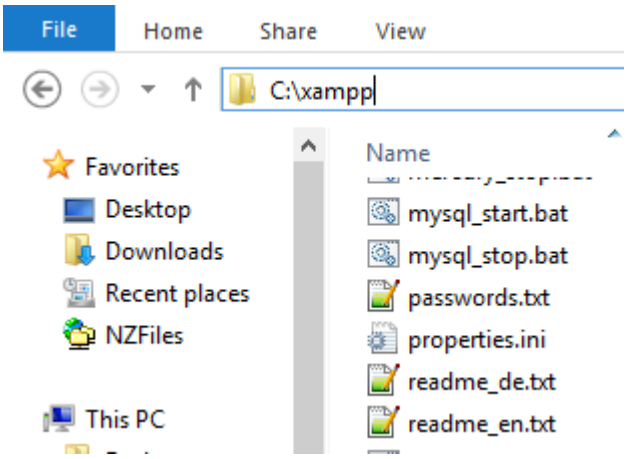
Εν συνεχεία, δες Εικόνα 3.25, επιλέγουμε το αρχείο που είναι σε συμπιεσμένη μορφή (π.χ. zip), και αφού το κατεβάσουμε στον υπολογιστή μας, το αποσυμπιέζουμε σε ένα φάκελο έστω c:\xampp. Η Εικόνα 3.26 εμφανίζει τα περιεχόμενα του φακέλου xampp.

Κάθε φορα που θέλουμε να εκκινήσουμε τη MySQL πατάμε το αρχείο mysql_start.bat και όταν θέλουμε να τερματίσουμε την εφαρμογή τότε πατάμε το mysql_stop.bat.

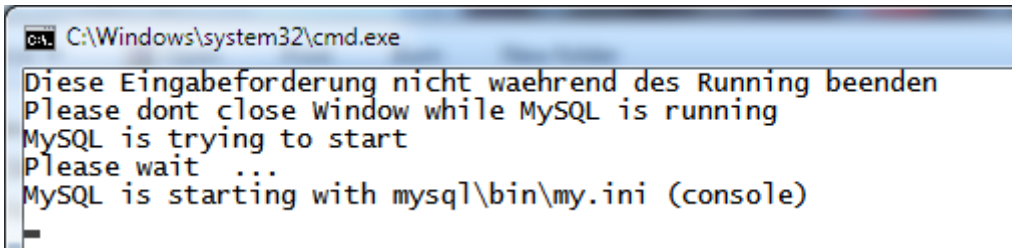


Εικόνα 3.25: Επιλογή της μορφής του αρχείου εγκατάστασης του XAMPP

Όταν τρέχει η MySQL εμφανίζεται ένα παράθυρο (δες Εικόνα 3.27), το οποίο θα πρέπει να μένει ανοικτό καθ' όλη τη διάρκεια εκτέλεσης της εφαρμογής.



Εικόνα 3.26: Τα αρχεία του φακέλου XAMPP

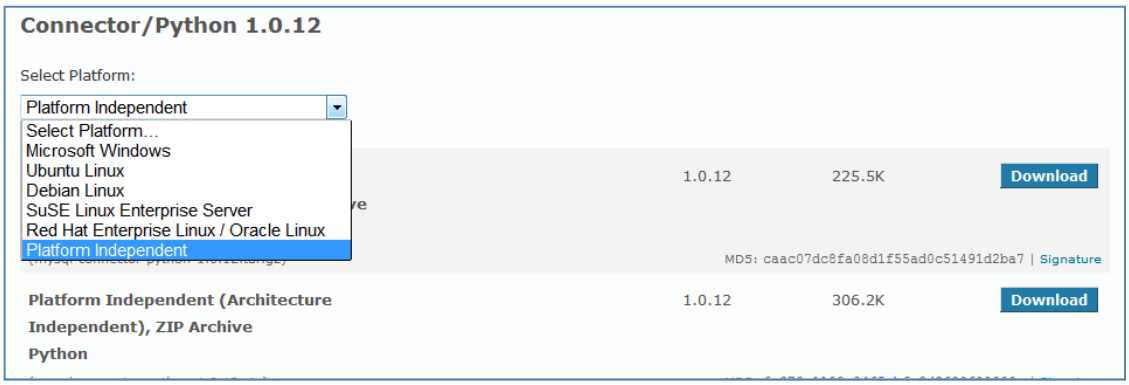


Εικόνα 3.27: Το παράθυρο της εφαρμογής MySQL

Υπάρχουν και άλλες διανομές που προσφέρουν το πακέτο με τα τρία προγράμματα Apache, PHP και MySQL, όπως για παράδειγμα WampServer, USBServer κ.λπ. οι οποίες μπορούν να χρησιμοποιηθούν εναλλακτικά του XAMPP.

3.4.2 Εγκατάσταση των οδηγών για τη σύνδεση της MySQL με τη Python

Από τη διεύθυνση http://dev.mysql.com/downloads/connector/python/ κατεβάζουμε το πρόγραμμα που εγκαθιστά τις απαραίτητες βιβλιοθήκες στην Python για τη σύνδεση της με τη Mysql.



Εικόνα 3.28 Επιλογή της μορφής του Python Connector

Κατά την επιλογή της πλατφόρμας, δες Εικόνα 3.28, να επιλέξετε τη μορφή Platform Independent. Κατεβάζουμε και αποσυμπιέζουμε το αρχείο σε έναν οποιονδήποτε φάκελο, έστω c:\connector. Εν συνεχεία, στο παράθυρο της γραμμής των εντολών (για να εμφανιστεί πατάμε το Κουμπί έναρξη και εκτελούμε την εντολή cmd) γράφουμε **cd c:\connector** και πατάμε ENTER. Σε αυτό το σημείο χρειάζεται να γνωρίζουμε το μονοπάτι της εγκατάστασης της Python στον υπολογιστή μας, έστω ότι είναι στο c:\python33.

Οπότε στη γραμμή των εντολών πληκτρολογούμε **c:\python33\python setup.py install** και πατάμε ENTER. Τώρα, είμαστε έτοιμοι να δημιουργήσουμε σενάρια σε Python που συνδέονται και αλληλεπιδρούν με τη MySQL. Σε αυτό το σημείο μπορούμε να κλείσουμε το παράθυρο της γραμμής εντολών και να διαγράψουμε - αν φυσικά το θέλουμε - το φάκελο c:\connector.

3.4.3 Δημιουργία Βάσης Δεδομένων

Για να δημιουργήσουμε μια νέα Βάση Δεδομένων στη MySQL θα πρέπει

να γράψουμε όλες τις εντολές σε ένα αρχείο κειμένου, και
να το φορτώσουμε στο περιβάλλον της MySQL

Βήμα 1

Σε ένα απλό επεξεργαστή κειμένου γράφουμε τις εντολές της παρακάτω εικόνας, και εν συνεχεία αποθηκεύουμε το αρχείο με το όνομα inep.sql στο φάκελο bin που βρίσκεται μέσα στο φάκελο της εγκατάστασης της MySQL. Αν για παράδειγμα χρησιμοποιείτε το πακέτο XAMPP τότε αποθηκεύστε το αρχείο στο φάκελο c:\xampp\mysql\bin

```
1  CREATE DATABASE IF NOT EXISTS inep DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci;
2  GRANT ALL PRIVILEGES ON inep.* TO 'inep'@'localhost' IDENTIFIED BY '78910';
3
4  USE inep;
5
6  CREATE TABLE IF NOT EXISTS USERS (
7   id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
8   uname VARCHAR( 50 ),
9   usurname  VARCHAR( 50 ),
10  age  INT,
11  city VARCHAR( 50 ) default 'Athens'
12 );
13
14 INSERT INTO USERS (uname, usurname, age) VALUES ("Nick", "Zacharis", 40);
15 INSERT INTO USERS (uname, usurname, age, city) VALUES ("George", "Loukas", 55, "LA");
```

Εικόνα 3.29 Τα περιεχόμενα του αρχείου inep.sql

Στη γραμμή 1 δημιουργούμε τη ΒΔ με όνομα inep και με κωδικοποίηση σε UTF-8 των δεδομένων της. Στη 2 γραμμή δημιουργούμε τον χρήστη με όνομα inep και κωδικό 78910 και του δίνουμε πλήρη πρόσβαση στη ΒΔ inep. Στην γραμμή 4 δηλώνουμε ότι οι επόμενες εντολές αν δεν έχουν ρητή αναφορά στο όνομα της ΒΔ, τότε θα αναφέρονται στην inep. Στις γραμμές 6 εως 12 δημιουργούμε το πίνακα USERS με πεδία :

έναν αύξοντα ακέραιο αριθμό σαν αναγνωριστικό,

το όνομα και το επώνυμο του χρήστη σαν αλφαριθμητικά,

την ηλικία του σαν ακέραιο αριθμό

και τέλος την πόλη που κατοικεί με προκαθορισμένη τιμή την Αθήνα (Athens).

Τέλος στις γραμμές 14 και 15 κάνουμε εισαγωγή δύο εγγραφών στον πίνακα USERS.

Βήμα 2

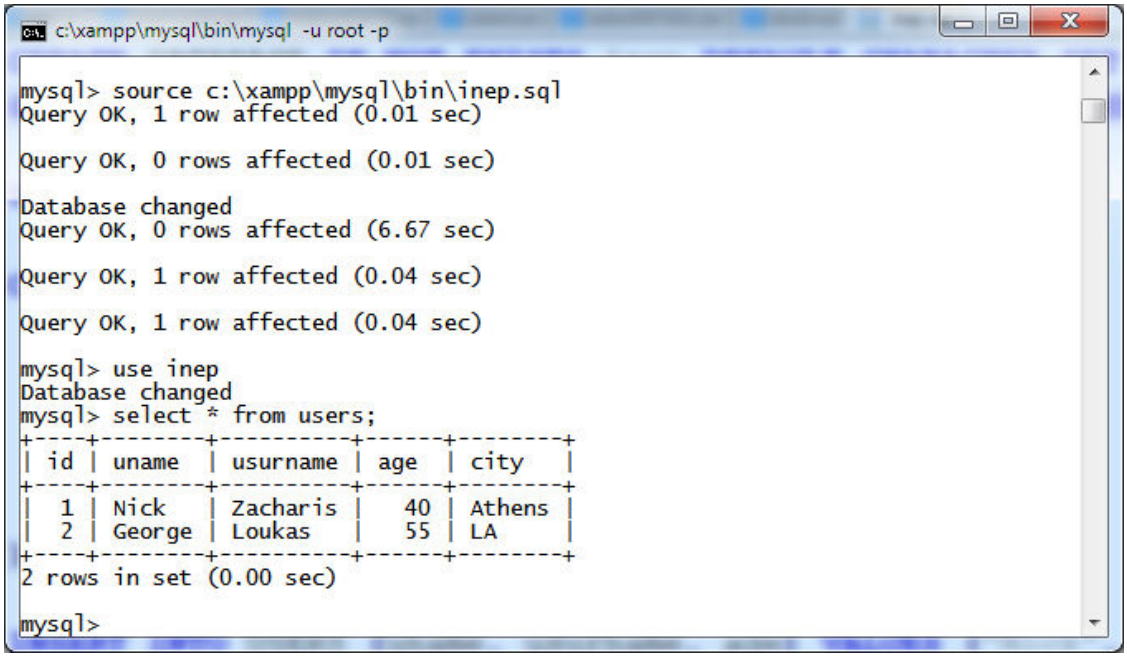
Από το παράθυρο της γραμμή εντολών (εκτέλεση του cmd.exe) μεταβαίνουμε στο φάκελο bin που βρίσκεται μέσα στο φάκελο της εγκατάστασης της MySQL και εκτελούμε την εντολή

> mysql - u root -p

Η ανωτέρω εντολή εκτελεί το πρόγραμμα πελάτη της MySQL μέσω του οποίου κάνουμε σύνδεση στον εξυπηρετητή της MySQL και μπορούμε να κάνουμε διαχείριση και επεξεργασία των δεδομένων.

Η παράμετρος -u root σημαίνει ότι θα κάνουμε είσοδο στον εξυπηρετητή σαν ο χρήστης root, ο οποίος είναι και ο προκαθορισμένος διαχειριστής, και η παράμετρος -

p είναι για να μας ζητήσει το πρόγραμμα το κωδικό πρόσβασης, ο οποίος είναι κενός, οπότε πατάμε Enter όταν εμφανιστεί το αντίστοιχο μήνυμα για να πληκτρολογήσουμε το κωδικό πρόσβασης. Στη περίπτωση που έχουμε αλλάξει το κωδικό πρόσβασης, όπως στη περίπτωση του Α τρόπου της εγκατάστασης της MySQL, τότε θα πρέπει να τον πληκτρολογήσουμε σε αυτό το σημείο.



Εικόνα 3.30 Το πρόγραμμα πελάτης της MySQL

Μετά την επιτυχή σύνδεση της εφαρμογής πελάτη αλλάζει το prompt σε mysql> και μπορούμε πλέον να δώσουμε εντολές προς τη MySQL. Η πρώτη εντολή είναι να φορτώσουμε το αρχειο inep.sql και για αυτο το λόγο πληκτρολογούμε :

mysql>source c:\xampp\mysql\bin\inep.sql

Εν συνεχεία με την εντολή

mysql>use inep

απευθύνουμε τις εντολές προς τη βάση δεδομένων με όνομα inep.

Τέλος, για να διαπιστώσουμε ότι όλα λειτουργούν κανονικά εκτελούμε την εντολή *mysql>select * from users;*

Για να κλείσουμε τη σύνδεση με τη βάση δεδομένω εκτελούμε την εντολή

mysql>quit

Από τη στιγμή που έχουμε δημιουργήσει τη βάση δεδομένων στη MySQL μπορούμε να δημιουργήσουμε σενάρια τα οποία μπορούν να επιλέξουν και να εμφανίσουν τις

τιμές των πεδίων των εγγραφών, να κάνουν εισαγωγή, τροποποίηση και διαγραφή εγγραφών.

3.4.4 Δημιουργία ερωτήματος επιλογής και εμφάνισης εγγραφών

Το σενάριο selectRecords.py περιέχει τις εντολές σε Python για τη σύνδεση στη βάση δεδομένων inep, καθώς και τη δημιουργία ερωτήματος για την επιλογή και εμφάνιση όλων των ονομάτων και επωνύμων του πίνακα users.

Το σενάριο selectRecords.py

```
1  import mysql.connector
2  conn = mysql.connector.connect(user='inep', database='inep', password='78910')
3  curs = conn.cursor()
4  query = "SELECT uname, usurname from users"
5  curs.execute(query)
6  for (uname, usurname,) in curs:
       print("{}{}".format(uname, usurname))
7  curs.close()
8  conn.close()
```

Η εκτέλεση του σεναρίου εμφανίζει :

```
>>>
Nick Zacharis
George Loukas
>>>
```

Στη γραμμή 1 κάνουμε αναφορά στον οδηγό σύνδεσης της Python με τη MySQL, και στη γραμμή 2 δημιουργούμε το αντικείμενο conn το οποίο περιέχει όλα τα στοιχεία της σύνδεσης με τη βάση δεδομένων inep και με τα δικαιώματα που έχει ο χρήστης inep. Εν συνεχεία στη γραμμή 3 δημιουργούμε το δρομέα για τη συγκεκριμένη σύνδεση και στη γραμμή 4 κατασκευάζουμε την ερώτηση που πρόκειται να απευθύνουμε στη βάση δεδομένων. Στις γραμμές 5 και 6 εκτελούμε το ερώτημα και εκτυπώνουμε επαναληπτικά τα αποτελέσματα στην οθόνη. Τέλος απελευθερώνουμε τους πόρους του συστήματος κλείνοντας το δείκτη και τη σύνδεση με τη βάση δεδομένων.

3.4.5 Εισαγωγή εγγραφών στη βάση δεδομένων

Το σενάριο insertRecords.py περιέχει τις εντολές σε Python για τη σύνδεση στη βάση δεδομένων inep, καθώς και τη δημιουργία ερωτήματος για την εισαγωγή μιας νέας εγγραφής στο πίνακα users.

Το σενάριο insertRecords.py

```
1   import mysql.connector
2   conn = mysql.connector.connect(user='inep', database='inep', password='78910')
3   curs = conn.cursor()
4   query = ("INSERT INTO users (uname,usurname,age,city) "
            "VALUES ('George','Mavrommatis', 28, 'Patra')"  )
5   curs.execute(query)
6   conn.commit()
7   curs.close()
8   conn.close()
```

Στη γραμμή 1 κάνουμε αναφορά στον οδηγό σύνδεσης της Python με τη MySQL, και στη γραμμή 2 συνδεόμαστε στη βάση δεδομένων inep και με τα δικαιώματα του χρήστη inep. Εν συνεχεία, στη γραμμή 3, δημιουργούμε το δρομέα για τη συγκεκριμένη σύνδεση και στη γραμμή 4 κατασκευάζουμε το ερώτημα για την εισαγωγή μιας νέας εγγραφής στο πίνακα users. Στην γραμμή 5 εκτελούμε το ερώτημα και στην επόμενη γραμμή αναμένουμε μέχρι να ολοκληρωθεί η εκτέλεση τους. Τέλος, στις γραμμές 7 και 8, απελευθερώνουμε τους πόρους του συστήματος κλείνοντας το δείκτη και τη σύνδεση με τη βάση δεδομένων.

3.4.6 Διαγραφή εγγραφών στη βάση δεδομένων

Για τη διαγραφή μιας ή περισσοτέρων εγγραφών στο πίνακα users, θα χρησιμοποιήσουμε το σενάριο insertRecords.py, αντίγραφο του οποίου θα μετονομάσουμε σε deleteRecords.py. Η μόνη αλλαγή που θα γίνει θα είναι στο ερώτημα δηλαδή στη γραμμή 4.

Το σενάριο deleteRecords.py

```
1..3  Είναι ίδιες με το  σενάριο insertRecords.py
4     query = ("Delete FROM users where uname = 'Nick'")
5..8  Είναι ίδιες με το  σενάριο insertRecords.py
```

Στη γραμμή 4 διαγράφουμε όλους τις εγγραφές από το πίνακα users που έχουν όνομα Nick.

3.4.7 Τροποποίηση εγγραφών στη βάση δεδομένων

Για τη τροποποίηση των τιμών των πεδίων μιας ή περισσοτέρων εγγραφών στο πίνακα users, θα χρησιμοποιήσουμε το σενάριο insertRecords.py, αντίγραφο του οποίου θα μετονομάσουμε σε updateRecords.py. Η μόνη αλλαγή που θα γίνει θα είναι στο ερώτημα δηλαδή στη γραμμή 4.

Το σενάριο updateRecords.py

1..3 Είναι ίδιες με το σενάριο insertRecords.py

```
4   query = ("update users set uname = 'Nick' where city='Patra'")
```

5..8 Είναι ίδιες με το σενάριο insertRecords.py

Στη γραμμή 5 τροποποιούμε/αλλάζουμε τα ονόματα όλων των εγγραφών που έχουν σαν πόλη τη Patra, σε Nick.

Άσκηση 3.4: Τροποποιήστε το αρχείο selectRecords.py ώστε να εμφανίζονται στα αποτελέσματα το επώνυμο, η πόλη και η ηλικία των χρηστών που είναι κάτω των 50 ετών.

Άσκηση 3.5: Τροποποιήστε το αρχείο insertRecords.py ώστε να κάνετε εισαγωγή ένα νέο χρήστη με στοιχεία John, Petrou, 45, Lamia.

Άσκηση 3.6: Τροποποιήστε το αρχείο updateRecords.py ώστε να αλλάξετε τις ηλικίες σε τιμή 34 όλων των εγγραφών που έχουν σαν πόλη τη Lamia ή τη Patra.

ΓΛΩΣΣΑΡΙ

MySQL Python Connector : ένα πρόγραμμα που περιέχει τους οδηγούς για τη σύνδεση της Python με τη MySQL.

MySQL: ένα πρόγραμμα διαχείρισης βάσεων δεδομένων.

Αρχείο κειμένου: μια ακολουθία από χαρακτήρες που βρίσκονται αποθηκευμένοι σε ένα σταθερό αποθηκευτικό μέσο.

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

Dubois, Hinz, Pedersen (2006). Ο Επίσημος Οδηγός MySQL 5, Εκδόσεις Γκιούρδας

ΟΔΗΓΟΣ ΠΕΡΑΙΤΕΡΩ ΜΕΛΕΤΗΣ

- Στη διεύθυνση http://zetcode.com/db/sqlite/ διαβάστε έναν μέτριας έκτασης οδηγό για τη ΒΔ SQLite, που μπορεί να χρησιμοποιηθεί και για αναζήτηση συγκεκριμένης λειτουργίας, όποτε μας χρειαστεί.

- Ο δικτυακός τόπος http://www.pythoncentral.io/ περιέχει μια μεγάλη ποικιλία από θέματα σχετικά με προγραμματισμό σε Python. Εκεί, θα βρείτε και ένα tutorial για την sqlite.
- Ο επίσημος οδηγός της MySQL είναι στη διεύθυνση https://dev.mysql.com/doc/refman/5.6/en/
- Στη διεύθυνση http://dev.mysql.com/doc/connector-python/en/index.html υπάρχει ένας περιεκτικός οδηγός για τη χρήση του οδηγού σύνδεσης με τη MySQL.
- Στη διεύθυνση http://el.wikibooks.org/wiki/Βασικές_γνώσεις_PHP_και_MySQL/Εισαγωγή _στην_MySQL υπάρχει μια σύντομη εισαγωγή στις λειτουργίες της MySQL.
- Στη διεύθυνση http://wiki.gentoo.org/wiki/MySQL/Startup_Guide υπάρχει ένα αναλυτικό εγχειρίδιο χρήσης.

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

4 ΔΙΑΔΙΚΤΥΑΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Σκοπός

Σκοπός του κεφαλαίου είναι η επίδειξη της λειτουργικότητας και η ανάπτυξη βασικών εφαρμογών για την επικοινωνία στο διαδίκτυο, με χρήση της γλώσσας προγραμματισμού Python. Στο κεφάλαιο αυτό θα βρείτε σύντομα στοιχεία για το διαδίκτυο, τα πρωτόκολλα επικοινωνίας καθώς και για το μοντέλο πελάτη εξυπηρετητή. Επίσης θα γίνει επίδειξη της σύνδεσης και επικοινωνίας ενός πελάτη με ένα εξυπηρετητή του παγκοσμίου ιστού για το κατέβασμα ενός αρχείου. Ακόμα, θα γίνει ανάπτυξη ενός σεναρίου για την αποστολή ενός ηλεκτρονικού μηνύματος στη δικτυακή υπηρεσία αλληλογραφίας GMAIL. Τέλος, θα μάθετε πώς μπορείτε να επεξεργάζεστε την πληροφορία που κινείται στο διαδίκτυο, σε δύο από τις πλέον διαδεδομένες μορφές τυποποίησης, τις xml και json.

Προσδοκώμενα αποτελέσματα

Όταν θα έχετε μελετήσει το κεφάλαιο αυτό θα μπορείτε να:

- γνωρίζετε τι είναι το διαδικτύο και τα πρωτοκόλλα επικοινωνίας
- τις βασικές υπηρεσίες και το τρόπο λειτουργίας τους
- κατανοήσετε τις διαφορές ενός εξυπηρετητή και ενός πελάτη
- κατεβάσετε μια ιστοσελίδα ή ένα δυαδικό αρχείο και να το αποθηκεύετε σε αρχείο στο τοπικό δίσκο
- στείλετε ένα μήνυμα ηλεκτρονικής αλληλογραφίας
- αναγνωρίζετε τα χαρακτηριστικά του πρότυπου xml,
- αναγνωρίζετε τα χαρακτηριστικά του πρότυπου json,
- αναπτύσσετε εφαρμογές που ανταλλάσσουν και επεξεργάζονται τυποποιημένα δεδομένα μέσω του διαδικτύου

Έννοιες-Κλειδιά

- Διαδίκτυο
- Πρωτόκολλο επικοινωνίας
- Μοντέλο πελάτη εξυπηρετητή
- HTTP
- SMTP
- πρότυπο xml
- πρότυπο json

Εισαγωγικές Παρατηρήσεις

Ο διαδικτυακός προγραμματισμός είναι σήμερα το κυριότερο πεδίο εφαρμογής κάθε σύγχρονης γλώσας προγραμματισμού. Η Python υποστηρίζει εγγενώς τις διαδικτυακές υπηρεσίες και έχει ένα σύνολο κλάσεων οργανωμένο σε βιβλιοθήκες ώστε να διευκολύνει το έργο του προγραμματιστή στην ανάπτυξη διαδικτυακών εφαρμογών. Στο παρόν κεφάλαιο θα αποκτήσετε τη γνώση και τις ικανότητες που χρειάζονται για να συνδεθείτε και να επικοινωνήσετε με εξυπηρετητές του διαδικτύου. Όπως θα δούμε, οι δομές δεομένων της Python που περιγράφτηκαν στο Κεφάλαιο 2, ταιριάζουν απόλυτα με τα σύγχρονα πρότυπα ανταλλαγής πληροφορίας στο διαδίκτυο, γεγονός που κάνει τη γλώσσα αυτή ιδανική για το περιβάλλον αυτό.

4.1 Επικοινωνία στο διαδίκτυο

Το διαδίκτυο αποτελείται από ένα μεγάλο σύνολο δικτύων υπολογιστών, στο οποίο οι διασυνδεμένοι υπολογιστές επικοινωνούν και ανταλλάσουν πληροφορίες μεταξύ τους. Για να επιτευχθεί αυτός ο σκοπός, χρησιμοποιούνται προκαθορισμένοι κανόνες ανταλλαγής δεδομένων που καθορίζονται από επιστημονικές επιτροπές και ονομάζονται, πρωτόκολλα επικοινωνίας.

Υπάρχουν κανόνες που καθορίζουν την "χαμηλού" επιπέδου επικοινωνία όπως για παράδειγμα την ονοματοδοσία των υπολογιστών που συμμετέχουν στο δίκτυο, το τρόπο τεμαχισμού της πληροφορίας σε πακέτα δεδομένων και την διοχέτευση τους στο δίκτυο από τον αποστολέα, τον τρόπο παραλαβής και σύνθεσης της πληροφορίας από τον παραλήπτη, τον τρόπο εντοπισμού των σφαλμάτων και επαναποστολής τυχόν απολεσθέντων πακέτων δεδομένων κ.λπ.

Από τη στιγμή που λύθηκαν τα προβλήματα σε χαμηλό επίπεδο επικοινωνίας δημιουργήθηκαν "υψηλού" επιπέδου κανόνες για εξειδικευμένες εργασίες, όπως, για παράδειγμα, την αποστολή ενός ηλεκτρονικού μηνύματος αλληλογραφίας (email), ή την αίτηση και παραλαβή μιας ιστοσελίδας.

Στο διαδίκτυο οι υπολογιστές χωρίζονται σε δυο μεγάλες κατηγορίες :

α) τους εξυπηρετητές: είναι οι υπολογιστές που προσφέρουν πληροφορίες και υπηρεσίες, και

β) τους πελάτες: είναι οι υπολογιστές που αιτούνται και παραλαμβάνουν υπηρεσίες και πληροφορίες από τους εξυπηρετητές.

Παράδειγμα 4.1: Εξυπηρετητές και διαδικτυακές διευθύνσεις και αναγνωριστικά

Ο υπολογιστής που προσφέρει πληροφορίες κάτω από το όνομα Wikipedia, ή ο υπολογιστής που προσφέρει την υπηρεσία Google Drive είναι ο εξυπηρετητής ενώ ο υπολογιστής που αιτείται μια ιστοσελίδα, η ανεβάζει ένα αρχείο, είναι ο πελάτης.

Ένας υπολογιστής που είναι συνδεδεμένος στο δίκτυο έχει ένα διεύθυνση η οποία μπορεί να είναι σε μορφή αριθμών π.χ. 60.10.20.30 ή σε πιο κατανοητή μορφή π.χ. www.google.gr. Στην ουσία οι υπολογιστές για την επικοινωνία μεταξύ τους χρησιμοποιούν τη πρώτη μορφή ενώ η δεύτερη μορφή είναι πιο κατανοητή και εύκολο-μνημόνευτη από εμάς τους ανθρώπους, αλλά αποτελεί στην ουσία ένα "ψευδώνυμο" της αριθμητικής μορφής.

Ένας υπολογιστής συνδεδεμένος στο δίκτυο, διαθέτει μια πληθώρα από πόρτες επικοινωνίας, οι οποίες μπορούν να χρησιμοποιηθούν από το λογισμικό που τρέχει στο συγκεκριμένο υπολογιστή για να προσφέρουν υπηρεσίες σε άλλους δικτυακούς υπολογιστές. Η λέξη εξυπηρετητής αναφέρεται ταυτόσημα και στον υπολογιστή που είναι συνδεδεμένος στο δίκτυο αλλά και στο λογισμικό του τρέχει στον υπολογιστή και προσφέρει υπηρεσίες σε μια συγκεκριμένη πόρτα επικοινωνίας. Σε έναν υπολογιστή μπορούν να τρέχουν ταυτόχρονα πολλά προγράμματα, όπου το καθένα καταλαμβάνει μια διαφορετική πόρτα επικοινωνίας και προσφέρει υπηρεσία μέσω αυτής.

Ένας πελάτης εκτός από τη διεύθυνση του υπολογιστή που θέλει να συνδεθεί, χρειάζεται να ξέρει και :

α) τη πόρτα επικοινωνίας, στην οποία προσφέρονται οι υπηρεσίες. Για παράδειγμα για να διαβάσουμε τη σελίδα της εταιρίας Google, θα πρέπει να συνδεθούμε στον υπολογιστή με το όνομα www.google.gr στη πόρτα 80.

β) το τρόπο, δηλαδή τις συγκεκριμένες εντολές που πρέπει να χρησιμοποιήσει για να επικοινωνήσει με τον εξυπηρετητή με σκοπό να ζητήσει και να παραλάβει τις πληροφορίες.

Αυτό εν συντομία δηλώνει η διεύθυνση που πληκτρολογούμε στο πρόγραμμα πλοήγησης:

http://www.google.gr:80

Το http είναι ο τρόπος επικοινωνίας (υψηλού επιπέδου πρωτόκολλο επικοινωνίας) που χρησιμοποιείται για την ανταλλαγή πληροφοριών με το μηχάνημα www στο δικτυακό τόπο google.gr το οποίο προσφέρει υπηρεσίες στη πόρτα επικοινωνίας 80. Αν στη καθημερινή μας ζωή με το διαδίκτυο δεν γράφουμε το :80 είναι επειδή το

πρωτόκολλο επικοινωνίας http ορίζει ότι η προκαθορισμένη πόρτα, αν δεν ορίζεται ρητά, θα είναι η 80.

Η Python διαθέτει πληθώρα modules για την υποστήριξη λειτουργιών σε επίπεδο εξυπηρετητή και πελάτη καθώς και για την επικοινωνία και ανταλλαγή πληροφοριών μέσω γνωστών υπηρεσιών του διαδικτύου. Στις επόμενες ενότητες θα δούμε παραδείγματα χρήσης του πρωτοκόλλου επικοινωνίας http (αίτηση και παραλαβή ιστοσελίδας) καθώς και του smtp (αποστολή ηλεκτρονικής αλληλογραφίας).

4.2 Σύνδεση με HTTP εξυπηρετητές διαδικτύου

Το πρωτόκολλο επικοινωνίας HTTP (HyperText Transfer Protocol) αποτελεί σήμερα έναν από τους πιο διαδεδομένους τρόπους επικοινωνίας και ανταλλαγής πληροφοριών ανάμεσα σε δύο υπολογιστές, στο διαδίκτυο. Η ηλεκτρονική αλληλογραφία είναι μακράν η πρώτη υπηρεσία, από πλευράς χρήσης, στο διαδίκτυο (killer application).

Ένας HTTP εξυπηρετητής, είναι ένας υπολογιστής που είναι συνδεδεμένος στο διαδίκτυο, τρέχει ένα λογισμικό το οποίο καταλαμβάνει μια πόρτα επικοινωνίας στον υπολογιστή και εν συνεχεία παρέχει υπηρεσίες του πρωτοκόλλου HTTP. Τα δύο πιο διαδεδομένα λογισμικά για αυτή την εργασία είναι :

α) IIS (Internet Information Server), εταιρία Microsoft

b) Apache, Οργανισμός The Apache Software Foundation

Στις ρυθμίσεις του HTTP εξυπηρετητή, εκτός από τη πόρτα επικοινωνίας, ορίζουμε και ποιο φάκελο από τον τοπικό υπολογιστή θα εξυπηρετεί, δηλαδή θα προσφέρει στο διαδίκτυο τα αρχεία του (προσοχή, μπορούμε να ορίσουμε πολλούς φακέλους αλλά ας κρατήσουμε απλό το παράδειγμα). Έστω ότι ο φάκελος που ορίσαμε είναι ο c:\test ο οποίος έχει σαν μέσα του 2 αρχεία το index.html και το money.pdf καθώς και ένα φάκελο το may ο οποίος με τη σειρά του έχει ένα αρχείο, το agores.xls

Επίσης ας υποθέσουμε ότι ο υπολογιστής που τρέχει το λογισμικό του εξυπηρετητή έχει όνομα www.bestcompany.gr και παρέχει υπηρεσίες στη πόρτα επικοινωνίας 7777

Τότε ο πελάτης (ο εξωτερικός υπολογιστής) που θέλει να λάβει το αρχείο money.pdf θα πρέπει να κάνει μια αίτηση μέσω του πρωτοκόλλου για το πόρο (Uniform Resource Locator) ως εξής :

http://www.bestcompany.gr:7777/money.pdf

ενώ αν ήθελε το αρχείο agores.xls

θα έκανε την αίτηση για το αρχείο

http://www.bestcompany.gr:7777/may/agores.xls

Στα ανωτέρω παραδείγματα, ο πελάτης αιτείται αρχικά ένα αρχείο τύπου pdf ενώ στη δεύτερη περίπτωση αιτείται ένα αρχείο τύπου xls (excel). Βέβαια, ο πιο διαδεδομένος τύπος μορφοποίησης ενός αρχείου είναι σαν ιστοσελίδα και συντάσσεται μέσω της γλώσσας HTML (HyperText Markup Language).

Η αίτηση του πελάτη καθώς και η απόκριση του εξυπηρετητή έχουν συγκεκριμένη μορφή στο πρωτόκολλο HTTP, με σκοπό να επιτρέπουν :

1. Την αίτηση και την παραλαβή ενός αρχείου (δες ανωτέρω παράδειγμα, π.χ. το pdf)
2. Την αποστολή δεδομένων από τη πλευρά του πελάτη με σκοπό να τύχουν επεξεργασίας από τον εξυπηρετητή (π.χ. η αποστολή ερώτησης στον εξυπηρετητή της google, ή το ανέβασμα (upload) ενός αρχείου)
3. Την ερώτηση από την πλευρά του πελάτη για πληροφορίες σχετικά με ένα αρχείο, με σκοπό να μάθει ποια είναι η τελευταία φορά που τροποποιήθηκε, το μέγεθος του, κ.λπ. ώστε να μην προβεί στο κατεβάσμα (download) του.

Η Python διαθέτει ένα μεγάλο πλήθος από αντικείμενα για να διευκολύνει τις προγραμματιστικές εργασίες μας στο διαδίκτυο.

4.2.1 Παράδειγμα εμφάνισης του κώδικα μιας ιστοσελίδας

Δημιουργείστε το αρχείο web.py και πληκτρολογήστε τις εντολές που εμφανίζονται παρακάτω :

```
1  import urllib.request
2  response = urllib.request.urlopen('http://www.wikipedia.org/')
3  html = response.read()
4  print(html)
```

Εικόνα 4.1: Ανάκτηση περιεχομένου από μια ιστοσελίδα

Στη γραμμή 1, κάνουμε εισαγωγή το module urllib.request το οποίο προσφέρει μια υψηλού επιπέδου προγραμματιστική διεπαφή για να διευκολύνει το έργο της αίτησης ενός πελάτη προς ένα εξυπηρετητή. Στη γραμμή 2, με τη συνάρτηση urlopen, ανοίγουμε μια σύνδεση με ένα δικτυακό τόπο και δημιουργούμε το αντικείμενο της αίτησης, μέσω του οποίου θα κατεβάσουμε το αρχείο.

```
<!DOCTYPE html>
<html lang="mul" dir="ltr">
<head>
<!-- Sysops: Please do not edit the main template directly; update /temp and synchronise. -->
<meta charset="utf-8">
<title>Wikipedia</title>
<!--[if lt IE 7]><meta http-equiv="imagetoolbar" content="no"><![endif]-->
<meta name="viewport" content="initial-scale=1.0, user-scalable=yes">
<link rel="apple-touch-icon" href="//bits.wikimedia.org/apple-touch/wikipedia.png">
```

Εικόνα 4.2: Μέρος του κώδικα σε HTML της ιστοσελίδας της wikipedia

Εν συνεχεία, χρησιμοποιούμε τη μέθοδο read της αίτησης για να διαβάσουμε το κώδικα της σελίδας και να τον αποθηκεύσουμε στη μεταβλητή html, δες γραμμή 3. Τέλος στη γραμμή 4, εκτυπώνουμε στην οθόνη, τα περιεχόμενα της html.

Το αποτέλεσμα της εκτέλεσης του web.py είναι να εμφανιστεί στην οθόνη όλος ο κώδικας σε HTML, της αρχικής ιστοσελίδας του δικτυακού τόπου wikipedia, μέρος της οποίας εμφανίζεται στη εικόνα 4.1.

Το "κατέβασμα" και η αποθήκευση ενός αρχείου εικόνας είναι εξίσου απλή διαδικασία. Το σενάριο saveimage.py παρουσιάζει τις εντολές για να αποθηκεύσουμε το λογότυπο από το δικτυακό τόπο της διαύγειας στο τοπικό δίσκο.

Το σενάριο saveimage.py

```
1  import urllib.request
2
3  url = 'http://sites.diavgeia.gov.gr/ekdd/images/logo.png'
4  response = urllib.request.urlopen(url)
5  fimg = open('c:\\temp\\logo.png', 'wb')
6  fimg.write(response.read())
7  fimg.close()
```

Το ίδιο μπορούμε να κάνουμε και σε μια ιστοσελίδα, όπως φαίνεται και στο παρακάτω σενάριο.

Το σενάριο savehtml.py

```
1  import urllib.request
2
3  url = 'http://www.ekdd.gr'
4  page = urllib.request.urlopen(url).read().decode('utf-8')
5  f = open("c:\\temp\\page.html", 'wb')
6  f.write(page.encode('utf-8'))
7  f.close()
```

Στη απόκριση του εξυπηρετητή, όταν πρόκειται για ιστοσελίδες θα πρέπει στη γραμμή 4 να αποκωδικοποιήσουμε τα bytes σε κείμενο και να τα ξανά-κωδικοποιήσουμε στη γραμμή 6 κατά την εγγραφή στο τοπικό αρχείο.

Άσκηση 4.1: Κατεβάστε την πρώτη σελίδα της ΔΙΑΥΓΕΙΑΣ (http://et.diavgeia.gov.gr/) και αποθηκεύστε την, σε αρχείο με το όνομα diaygeia.html

4.3 Επικοινωνία με εξυπηρετητή ηλεκτρονικής αλληλογραφίας

Το Simple Mail Transfer Protocol (SMTP) είναι το πρωτόκολλο επικοινωνίας για την αποστολή ενός ηλεκτρονικού μηνύματος, του γνωστού μας email. Η υπηρεσία υλοποιείται μέσω ενός εξυπηρετητή (SMTP Server) στον οποίο θα

συνδεθεί ο πελάτης (Mail client), και θα αποστείλει τις κατάλληλες εντολές, για την αποστολή ενός μηνύματος σε ένα παραλήπτη.

Περισσότερες πληροφορίες μπορείτε να διαβάσετε στη Wikipedia όπου, εκτός από μια σύντομη περιγραφή του τρόπου λειτουργίας του πρωτοκόλλου, περιλαμβάνονται σύνδεσμοι στους διεθνείς οργανισμούς που ασχολούνται με τις προδιαγραφές του ηλεκτρονικού ταχυδρομείου.

Δραστηριότητα 4.1: Αποστολή ενός ηλεκτρονικού μηνύματος

Για την επιτυχή εκτέλεση του παρακάτω σεναρίου θα χρειαστεί να δημιουργήσετε - αν δεν έχετε ήδη - ένα λογαριασμό ηλεκτρονικής αλληλογραφίας στο http://www.gmail.com. Αφού ολοκληρώσετε αυτό το βήμα, θα υποθέσουμε ότι ο λογαριασμός που δημιουργήσατε έχει τα εξής στοιχεία :

όνομα χρήστη	apostoleas
κωδικός χρήστη	1234

Άρα η διεύθυνση ηλεκτρονικής αλληλογραφίας του αποστολέα (δηλαδή η δική σας διεύθυνση), είναι apostoleas@gmail.com Εν συνεχεία θα υποθέσουμε ότι θέλουμε να στείλουμε, με τη βοήθεια της Python, ένα μήνυμα με τα εξής στοιχεία :

Διεύθυνση παραλήπτη	paraliptis@domain.gr
Θέμα μηνύματος	My first email
Κείμενο	Η python είναι σπουδαία γλώσσα.

Η διεύθυνση παραλήπτη είναι μη υπαρκτή και στο παρακάτω παράδειγμα θα πρέπει να αντικατασταθεί είτε με μια δεύτερη δική σας διεύθυνση είτε με την ηλεκτρονική διεύθυνση του αποστολέα (θα στείλετε μήνυμα στον εαυτό σας) με σκοπό να δοκιμάσετε τη λειτουργία του προγράμματος.

Δημιουργείστε το αρχείο smtp.py και πληκτρολογήστε τις εντολές που εμφανίζονται παρακάτω :

```
1  import smtplib
2  from email.mime.text import MIMEText
3
4  smtpserver = 'smtp.gmail.com'
5  smtpport   = 587
6  smtpuser = 'apostoleas@gmail.com'
7  smtppass = '1234'
8
9  recipients = ['paraliptis@domain.gr']
10
11 msg = MIMEText('Η python είναι σπουδαία γλώσσα.\n'.encode('utf-8'), _charset='utf-8')
12 msg['Subject'] = 'My first email'
13
14 try:
15    mailServer = smtplib.SMTP(smtpserver, smtpport)
16    mailServer.ehlo()
17    mailServer.starttls()
18    mailServer.login(smtpuser, smtppass)
19    mailServer.sendmail(smtpuser,recipients,msg.as_string())
20    mailServer.close()
21    print( "Successfully sent email")
22 except smtplib.SMTPException:
23    print ("Error: unable to send email")
```

Εικόνα 4.3: Αποστολή email με χρήση του smtplib

Στη γραμμή 1 κάνουμε εισαγωγή τη μονάδα smtplib που περιέχει όλες τις συναρτήσεις για την σύνδεση και την αποστολή ενός email σε ένα SMTP εξυπηρετητή. Στη γραμμή 2 κάνουμε εισαγωγή τη κλάση MIMEText με σκοπό να δημιουργήσουμε αντικείμενα που θα έχουν τη κατάλληλη διαμόρφωση για την αποστολή email και είναι σύμφωνα με τη προδιαγραφή Multipurpose Internet Mail Extensions (MIME). Στις γραμμές 4 και 5 δηλώνουμε τον εξυπηρετητή και την πόρτα επικοινωνίας στην οποία θα συνδεθούμε για να στείλουμε το email. Στις γραμμές 6 και 7 δηλώνουμε την ηλεκτρονική διεύθυνση του αποστολέα καθώς και το κωδικό του. Στη γραμμή 9 δηλώνουμε τον παραλήπτη του μηνύματος και στην επόμενη γραμμή δημιουργούμε το αντικείμενο msg το οποίο περιέχει το κείμενο του μηνύματος με τους χαρακτήρες του κωδικοποιημένους σε UTF-8. Στην γραμμή 12 ορίζουμε το θέμα του μηνύματος.

Οι γραμμές 15 έως 21 είναι μέσα σε ένα try except ώστε αν συμβεί σφάλμα τύπου SMTPException (π.χ. αργή συνδεση - timeout error κ.λπ.) κατά τη σύνδεση με τον εξυπηρετητή και την αποστολή του μηνύματος να το παγιδεύσουμε και να μην τερματίσει απότομα η λειτουργία του σεναρίου αλλά να εκτυπωθεί ένα μήνυμα στην οθόνη του υπολογιστή.

Στη γραμμή 15 δημιουργούμε το αντικείμενο mailServer, το οποίο αρχικοποιείται με τα στοιχεία - διεύθυνση και πόρτα επικοινωνίας - του SMTP εξυπηρετητή. Στη γραμμή 16 αποστέλλουμε ένα χαιρετισμό και στην επόμενη γραμμή ενεργοποιούμε την ασφαλή μεταφορά Transport Layer Security,η οποία υποστηρίζεται και απαιτείται τις περισσότερες φορές, από όλους τους σύγχρονους εξυπηρετητές. Στη γραμμή 18

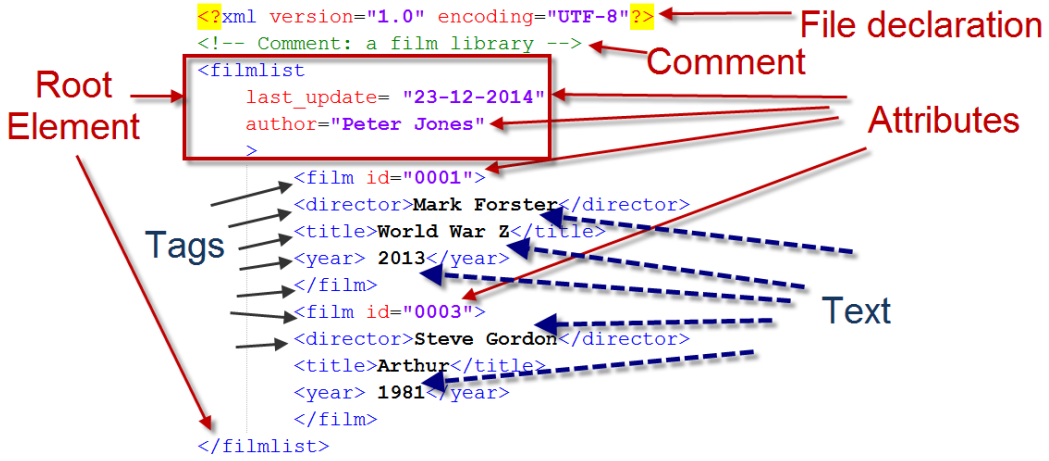
γίνεται η πιστοποίηση του χρήστη και αμέσως μετά γίνεται η αποστολή του μηνύματος. Στο τέλος κλείνει η σύνδεση. Ο ανωτέρω κώδικας λειτουργεί και σε οποιανδήποτε άλλο SMTP εξυπηρετητή, αρκεί να αντικαταστήσετε, το όνομα και το κωδικό του χρήστη, τη διεύθυνση και τη πόρτα επικοινωνίας του SMTP εξυπηρετητή.

Άσκηση 4.2: Δημιουργείστε ένα νέο λογαριασμό αλληλογραφίας στο δικτυακό τόπο http://mail.yahoo.com Δημιουργήστε το σενάριο mailyahoo.py ώστε να στείλετε ένα email προς οποιονδήποτε άλλον λογαριασμό στον οποίο έχετε πρόσβαση ή ακόμη και στον εαυτό σας, με θέμα "Συνάντηση Συλλόγου Πληροφορικής" και κείμενο "Παρακαλείστε να προσέλθετε στο σύλλογο τη Παρασκευή 12 Μαρτίου στις 8 το απόγευμα."

4.4 Επεξεργασία μορφοποιημένης πληροφορίας

4.4.1 Αρχεία XML

Η γλώσσα XML (eXtensible Markup Language) δημιουργήθηκε για την αποθήκευση και μεταφορά δεδομένων σε τυποποιημένη μορφή. Σε αντίθεση με την html που είναι γλώσσα με προκαθορισμένες ετικέτες (π.χ. head, body, κ.λπ.), η xml δεν περιέχει προκαθορισμένες ετικέτες (tags), αλλά αντίθετα, αυτές καθορίζονται από το χρήστη.



Εικόνα 4.4: Ανατομία ενός αρχείου xml (sample.xml)

Στην παραπάνω Εικόνα μπορείτε να δείτε τα βασικά συστατικά ενός xml αρχείου. Η πρώτη γραμμή είναι η δήλωση ότι πρόκειται για αρχείο xml, ενώ περιέχει και την κωδικοποίηση των χαρακτήρων, που στο παράδειγμα είναι η UTF-8. Η δεύτερη γραμμή είναι ένα σχόλιο που δεν λαμβάνεται υπόψη από το πρόγραμμα που θα επεξεργαστεί το αρχείο αυτό. Το στοιχείο (element) filmlist περιέχει δύο attributes, το last_update και το author. Το filmlist είναι το root element που περιλαμβάνει όλα τα υπόλοιπα στοιχεία. Παρατηρήστε ότι κλείνει στο τέλος του αρχείου, με την ετικέτα </filmlist>, ίδιο δηλαδή όνομα με ένα σύμβολο διαίρεσης μπροστά του. Το filmlist

περιέχει δύο παιδιά τύπου film, που μπορείτε να δείτε ότι περιλαμβάνεται ανάμεσα σε ετικέτες <film> και </film>. Κάθε τέτοιο στοιχείο περιέχει τις πληροφορίες μιας ταινίας, που αντιπροσωπεύονται από τα παιδιά του film, τα στοιχεία με ετικέτες director, title και year.

Η Python διαθέτει διάφορες βιβλιοθήκες με τις οποίες μπορεί κανείς να διαβάσει, δημιουργήσει και επεξεργαστεί αρχεία xml. Μια από αυτές είναι η lxml, την οποία χρησιμοποιούμε στο Παράδειγμα 4.2 που ακολουθεί για να διαβάσουμε το αρχείο sample.xml (Εικόνα). Στη συνέχεια κάνουμε εξαγωγή αρχικά τα στοιχεία film και μετά τα στοιχεία title, των οποίων το κείμενο εμφανίζουμε ένα-ένα.

Παράδειγμα 4.2: Εξαγωγή του Κειμένου (Text) για ένα στοιχείο αρχείου XML

```
from lxml import etree
doc = etree.parse("B:\\tmp\\sample.xml")
film_list= doc.findall('film')
for film in film_list:
    title = film.findtext('title')
    print(title)
```

```
>>>
World War Z
Arthur
>>>
```

Στην πρώτη γραμμή κάνουμε import το module etree από τη βιβλιοθήκη lxml και στη δεύτερη χρησιμοποιούμε τη μέθοδο parse() για να διαβάσουμε σε μορφή δένδρου το αρχείο xml. Η μέθοδος findall() ξεκινά πάντα από τη ρίζα και επιστρέφει μια λίστα στοιχείων που βρίσκει με βάση το όρισμα που της δίνουμε. Στην προκειμένη περίπτωση επιστρέφει μια λίστα με στοιχεία τις ταινίες. Η εντολή for περνά όλα τα στοιχεία (ταινίες) που περιέχει η λίστα και η findtext() επιστρέφει το κείμενο (text) από ένα στοιχείο με ετικέτα που δίνουμε ως παράμετρο. Τέλος, εμφανίζουμε τα κείμενα που ανακτούμε, δηλαδή τους τίτλους των ταινιών, όπως μπορείτε να δείτε στο δείγμα εκτέλεσης.

Παράδειγμα 4.3: Δημιουργία αρχείου xml με τη βιβλιοθήκη lxml

Ο κώδικας που ακολουθεί δημιουργεί ένα αρχείο xml στο δίσκο του οποίου η ρίζα emails περιέχει δύο παιδιά email και το καθένα από αυτά δύο στοιχεία-παιδιά με ετικέτες name και address.

```
1  from lxml import etree
2  root = etree.Element('emails')
3  xmldoc = etree.ElementTree(root)
4  a = etree.SubElement(root,'email',id='1')
5  a1 = etree.SubElement(a,'name')
6  a1.text = "George"
7  a2 = etree.SubElement(a,'address')
8  a2.text='george@george.gr'
9  a = etree.SubElement(root,'email',id='2')
10 a1 = etree.SubElement(a,'name')
11 a1.text = "Nick"
12 a2 = etree.SubElement(a,'address')
13 a2.text='nick@nick.gr'
14 xmldoc.write('b://tmp//newxmldoc.xml', encoding='UTF-8')
```

Στη γραμμή 1 κάνουμε εισαγωγή το module etree. Στη γραμμή 2 δημιουργούμε το στοιχείο-ρίζα, χρησιμοποιώντας τον κατασκευαστή Element. Στη γραμμή 3 δημιουργούμε τη δομή-δένδρου (element tree) με κορυφή τη ρίζα (root) που κατασκευάσαμε στη δεύτερη γραμμή. Στη γραμμή 4 ο SubElement() δημιουργεί ένα παιδί της ρίζας με ετικέτα email και attribute id = '1'. Στις γραμμές 5-6 και 7-8 δημιουργούμε δύο παιδιά του στοιχείου email, δίνοντας και τιμές στο κείμενο, χρησιμοποιώντας το πεδίο .text. Στις γραμμές 9-13 επαναλαμβάνουμε την ίδια διαδικασία για το δεύτερο στοιχείο email. Τέλος, στη γραμμή 14 χρησιμοποιούμε τη μέθοδο write του ElementTree για να αποθηκεύσουμε το xml αρχείο στο δίσκο. Στην εικόνα που ακολουθεί, βλέπετε το αρχείο που δημιουργήσαμε (η στοίχιση έγινε μετά τη δημιουργία, για να διαβάζεται ευκολότερα).

```
<emails
    username="George Nick"
    >
        <email id='1'>
        <name>George</name>
        <address>george@george.gr</address>
        </email
        <email id='2'>
        <name>Nick</name>
        <address>nick@nick.gr</address>
        </email
</emails>
```

Εικόνα 4.5: Ένα xml αρχείο που δημιουργήσαμε με κώδικα Python

4.4.2 Αρχεία json

Τα αρχεία json (JavaScript Object Notation) αποτελούν μια, εναλλακτική της xml, μέθοδο αποθήκευσης και μεταφοράς δεδομένων. Όπως θα δείτε, ένα αρχείο json είναι απόλυτα συμβατό με τις δομές δεδομένων της Python και ως εκ τούτου η χρήση τους είναι εύκολη διαδικασία.

Σε ένα αρχείο json, τα δεδομένα καταγράφονται σε ζευγάρια της μορφής

<όνομα (σε εισαγωγικά)> : <τιμή>

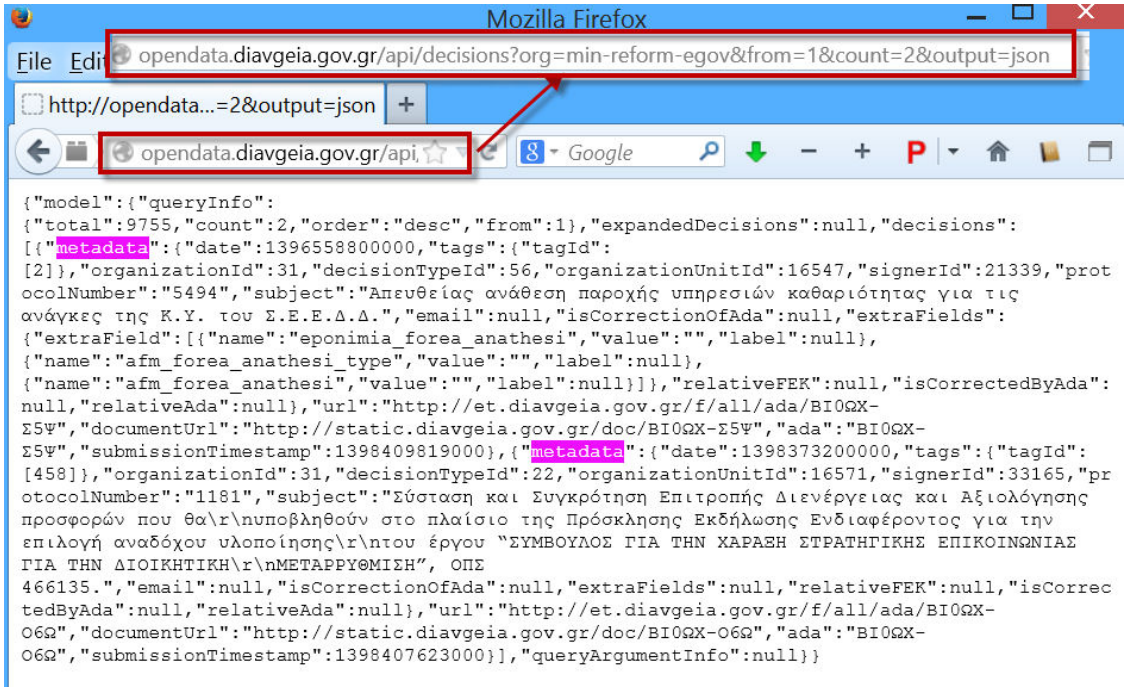
που όπως αντιλαμβάνεστε, μοιάζει με το λεξικό της Python. Τα ζεύγη χωρίζονται με κόμμα. Τα αντικείμενα json περιλαμβάνονται μέσα σε άγκιστρα, ενώ μπορούμε να έχουμε και πίνακες που περιλαμβάνονται σε αγκύλες. Είναι δυνατόν ένας πίνακας να περιέχει αντικείμενα και αντίστροφα. Τα δεδομένα μπορεί να είναι αλφαριθμητικό, αριθμός, σύνθετο αντικείμενο, πίνακας καθώς και οι τιμές true, false και null.

Στην Εικόνα 4.6 βλέπετε ένα παράδειγμα αρχείου σε μορφή json (πηγή http://www.jsoneditoronline.org/). Οι γραμμές 1 και 16 περιέχουν το ζεύγος των αγκίστρων που ορίζουν το αντικείμενο. Παρατηρήστε ότι το αντικείμενο, περιέχει πίνακα (γραμμές 2-6), διάφορες τιμές (γραμμές 7, 8, 9,15), άλλα αντικείμενα (γραμμές 10-14). Τα ονόματα των δεδομένων μπαίνουν σε εισαγωγικά και τα ονόματε είναι επιλογής του χρήστη (του προγραμματιστή), στο παράδειγμα έχουν επιλεγεί έτσι ώστε να περιγράφουν τον τύπο του κάθε δεδομένου, για λόγους επίδειξης.

```
1  {
2    "array": [
3      1,
4      2,
5      3
6    ],
7    "boolean": true,
8    "null": null,
9    "number": 123,
10   "object": {
11     "a": "b",
12     "c": "d",
13     "e": "f"
14   },
15   "string": "Hello World"
16 }
```

Εικόνα 4.6: Ένα αρχείο json

Στην Εικόνα 4.7 βλέπετε την κλήση του api του Προγράμματος «Διαύγεια» μέσα από έναν κοινό περιηγητή, όπου έχουμε ζητήσει τα δεδομένα να επιστραφούν σε μορφή json. Στην ένθετη εικόνα μπορείτε να δείτε την πλήρη σύνταξη του url-api που επιστρέφει τις δύο τελευταίες αναρτήσεις του Υπουργείου Διοικητικής Μεταρρύθμισης & Ηλεκτρονικής Διακυβέρνησης, στη Διαύγεια. Τα δύο σετ δεδομένων, ξεκινάνε με τη λέξη “metadata”, που μπορείτε να δείτε σημειωμένη πάνω στην Εικόνα 4.7




```
{"model":{"queryInfo":
{"total":9755,"count":2,"order":"desc","from":1},"expandedDecisions":null,"decisions":
[{"metadata":{"date":1396558800000,"tags":{"tagId":
[2]},"organizationId":31,"decisionTypeId":56,"organizationUnitId":16547,"signerId":21339,"prot
ocolNumber":"5494","subject":"Απευθείας ανάθεση παροχής υπηρεσιών καθαριότητας για τις
ανάγκες της Κ.Υ. του Σ.Ε.Ε.Δ.Δ.","email":null,"isCorrectionOfAda":null,"extraFields":
{"extraField":[{"name":"eponimia_forea_anathesi","value":"","label":null},
{"name":"afm_forea_anathesi_type","value":"","label":null},
{"name":"afm_forea_anathesi","value":"","label":null}]},"relativeFEK":null,"isCorrectedByAda":
null,"relativeAda":null},"url":"http://et.diavgeia.gov.gr/f/all/ada/BI0ΩX-
Σ5Ψ","documentUrl":"http://static.diavgeia.gov.gr/doc/BI0ΩX-Σ5Ψ","ada":"BI0ΩX-
Σ5Ψ","submissionTimestamp":1398409819000},{"metadata":{"date":1398373200000,"tags":{"tagId":
[458]},"organizationId":31,"decisionTypeId":22,"organizationUnitId":16571,"signerId":33165,"pr
otocolNumber":"1181","subject":"Σύσταση και Συγκρότηση Επιτροπής Διενέργειας και Αξιολόγησης
προσφορών που θα\r\nυποβληθούν στο πλαίσιο της Πρόσκλησης Εκδήλωσης Ενδιαφέροντος για την
επιλογή αναδόχου υλοποίησης\r\nτου έργου "ΣΥΜΒΟΥΛΟΣ ΓΙΑ ΤΗΝ ΧΑΡΑΞΗ ΣΤΡΑΤΗΓΙΚΗΣ ΕΠΙΚΟΙΝΩΝΙΑΣ
ΓΙΑ ΤΗΝ ΔΙΟΙΚΗΤΙΚΗ\r\nΜΕΤΑΡΡΥΘΜΙΣΗ", ΟΠΣ
466135.","email":null,"isCorrectionOfAda":null,"extraFields":null,"relativeFEK":null,"isCorrec
tedByAda":null,"relativeAda":null},"url":"http://et.diavgeia.gov.gr/f/all/ada/BI0ΩX-
O6Ω","documentUrl":"http://static.diavgeia.gov.gr/doc/BI0ΩX-O6Ω","ada":"BI0ΩX-
O6Ω","submissionTimestamp":1398407623000}],"queryArgumentInfo":null}}
```


Εικόνα 4.7: Ανάκτηση δεδομένων σε μορφή json από τη Διαύγεια

Στον επίσημο δικτυακό τόπο του json, θα βρείτε πληροφορίες και στοιχεία για το πρότυπο. Ακόμα, θα βρείτε αναφορές σε βιβλιοθήκες για επεξεργασία αρχείων json, σε διάφορες γλώσσες, ανάμεσά τους και για την Python. Στο Παράδειγμα 4.4 που ακολουθεί, χρησιμοποιούμε τη βιβλιοθήκη json για να επεξεργαστούμε δεδομένα σε μορφή json.

Παράδειγμα 4.4: Επεξεργασία δεδομένων με χρήση της βιβλιοθήκης json

Χρησιμοποιούμε το api της Διαύγειας για να ανακτήσουμε τις 2 τελευταίες αναρτήσεις του Υπουργείου Παιδείας. Στη συνέχεια, διατρέχουμε τον πίνακα των αποτελεσμάτων και εμφανίζουμε το θέμα της κάθε απόφασης. Παρατηρήστε ότι χρησιμοποιούμε μια εντολή try .. except για να παγιδεύσουμε τα λάθη ανάγνωσης από τον εξυπηρετητή. Αν δείτε μήνυμα λάθος, ελέγξτε αν έχετε πληκτρολογήσει σωστά τον κώδικα και ξαναπροσπαθήστε.

```
import urllib.request
import json

try:
    url = 'http://opendata.diavgeia.gov.gr/api/decisions?org=minedu&from=1&count=2&output=json'
    req = urllib.request.Request(url, headers={'accept' : '*/*' })
    response = urllib.request.urlopen(req,timeout=90000)
    the_page = response.read() #διαβάζει bytes
    doc=the_page.decode("utf-8") #μετατροπή σε αλφαριθμητικό
    doc=json.loads(doc) #μετατροπή σε λεξικό (dictionary)
    #print(doc) # για εμφάνιση όλου του json string, αφαιρούμε το σχόλιο
    for apofasi in range (len(doc['model']['decisions'])):
        print(doc['model']['decisions'][apofasi]['metadata']['subject'])
except urllib.error.HTTPError or urlib.error.URLError:
    print("Δεν ήταν δυνατή η σύνδεση με τον διακομιστή. Δοκιμάστε ξανά!")
```

Άσκηση 4.3: Χρησιμοποιήστε το api της Διαύγειας και κατεβάστε ένα μικρό σύνολο δεδομένων (ρυθμίζοντας τις παραμέτρους from και count) από το Υπουργείο από το οποίο προέρχεστε (θα βρείτε πληροφορίες για το api στη διεύθυνση http://opendata.diavgeia.gov.gr/). Κατεβάστε τα δεδομένα τόσο σε μορφή json, όσο και σε μορφή xml και αποθηκεύστε τα σε δύο αντίστοιχα αρχεία (η όλη διαδικασία μπορεί να γίνει είτε με αντιγραφή-επικόλληση από τον browser ή, καλύτερα, με κώδικα Python). Γράψτε δύο προγράμματα ένα για μορφή δεδομένων που κατεβάσατε, τα οποία θα διαβάζουν το αρχείο και θα εμφανίζουν την πληροφορία (ή όποιο μέρος της επιλέξετε) σε μια κατάλληλα σχεδιασμένη οθόνη. Δημιουργήστε διαδοχικές εκδόσεις του κώδικα, κάνοντάς τον περισσότερο παραμετρικό σε κάθε επόμενη έκδοση (π.χ. να επιλέγει ο χρήστης το Υπουργείο, το πλήθος των αποφάσεων, τα πεδία που θα εμφανίζονται, κ.λπ.)

Το πρότυπο json χρησιμοποιείται ευρέως για διακίνηση και ανταλλαγή δεδομένων στο διαδίκτυο, γεγονός που μπορεί να ενισχύσει τη διαλειτουργικότητα ανάμεσα στις διαδικτυακές εφαρμογές. Ως παραδείγματα, πέραν της «Διαύγεια» μπορούμε να αναφέρουμε τα facebook, twitter, youtube, κ.λπ.

ΓΛΩΣΣΑΡΙ

HTML: Οι εντολές που μορφοποιούν το περιεχόμενο μιας ιστοσελίδας.

HTTP: πρωτόκολλο επικοινωνίας για την αίτηση και λήψη δεδομένων ανάμεσα σε ένα πρόγραμμα πλοήγησης και σε ιστότοπο.

json: JavaScript Object Notation. Πρότυπο αρχείων με συγκεκριμένες και παράλληλα ευέλικτες προδιαγραφές σύνταξης που χρησιμοποιούνται για τη διακίνηση δεδομένων στο διαδίκτυο.

SMTP: πρωτόκολλο επικοινωνίας για την αποστολή emails.

TLS: πρωτόκολλο ασφαλούς επικοινωνίας μεταξύ ενός πελάτη και εξυπηρετητή

XML: eXtensible Markup Language. Πρότυπο σύνταξης αρχείων με συγκεκριμμένες και ευέλικτες προδιαγραφές σύνταξης για διακίνηση δεδομένων στο διαδίκτυο.

ΒΙΒΛΙΟΓΡΑΦΙΚΕΣ ΑΝΑΦΟΡΕΣ

Ν.Αβούρης, Κ.Σγάρμπας, Σ. Καξίρας, Μ.Κουκιάς, Β. Παλιουράς (2012). Εισαγωγή στους υπολογιστές με την γλώσσα Python, Εκδόσεις Πανεπιστημίου Πατρών

ΟΔΗΓΟΣ ΠΕΡΑΙΤΕΡΩ ΜΕΛΕΤΗΣ

- Η διεύθυνση http://el.wikipedia.org/wiki/Διαδίκτυο περιέχει πληροφορίες για την τεχνολογία και την ιστορία του διαδικτύου, το τρόπο πρόσβασης στις διαθέσιμες

υπηρεσίες και πληροφορίες, καθώς και τους κινδύνους και τα μέτρα προστασίας που πρέπει να λαμβάνονται υπόψη κατά τη πλοήγηση στο διαδίκτυο.

- Οι διευθύνσεις http://el.wikipedia.org/wiki/TCP/IP και http://el.wikipedia.org/wiki /Internet_Protocol περιέχουν πληροφορίες για τα πρωτόκολλα ελέγχου και μεταφοράς δεδομένων στο διαδίκτυο καθώς και για τη διευθυνσιοδότηση των κόμβων και την δρομολόγηση των πακέτων από έναν υπολογιστή προς έναν τελικό προορισμό, ανάμεσα στα διάφορα δίκτυα
- Στις διευθύνσεις http://el.wikipedia.org/wiki/Εξυπηρετητής και http://el.wikipedia .org/wiki/Μοντέλο_πελάτη-διακομιστή αναλύονται οι όροι πελάτης και εξυπηρετητής. Επίσης στη διεύθυνση http://el.wikipedia.org/wiki/Κατάλογος_θυρών_TCP_και_UDP περιγράφονται οι επίσημες πόρτες επικοινωνίας που χρησιμοποιούνται από τους εξυπηρετητές για να προσφέρουν υπηρεσίες στο διαδίκτυο.
- Στη διεύθυνση http://el.wikipedia.org/wiki/Domain_Name_System περιγράφεται το ιεραρχικό σύστημα ονοματοδοσίας για δίκτυα υπολογιστών.
- Το πρωτόκολλο επικοινωνίας για την αποστολή ηλεκτρονικής αλληλογραφίας περιγράφεται στην διεύθυνση http://el.wikipedia.org/wiki/SMTP
- Το πρωτόκολλο επικοινωνίας για την επικοινωνία ανάμεσα σε ένα πρόγραμμα πλοήγησης και ενός εξυπηρετητή στο παγκόσμιο ιστό περιγράφεται στη διεύθυνση http://el.wikipedia.org/wiki/Πρωτόκολλο_Μεταφοράς_Υπερκειμένου
- Το πρωτόκολλο της δομής ενός ηλεκτρονικού μηνύματος ώστε να περιλαμβάνει το μήνυμα με τη κωδικοποίηση των χαρακτήρων του καθώς και άλλα αρχεία, όλα μαζί σε ένα ενιαίο μήνυμα περιγράφεται στη διεύθυνση http://el.wikipedia.org/wiki/MIME

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ