ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ, ΕΡΕΥΝΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

Αράπογλου Α., Βραχνός Ε., Λέκκα Δ., Κανίδης Ε.,
Μακρυγιάννης Π., Μπελεσιώτης Β., Τζήμας Δ., Παπαδάκης Σπ.

Προγραμματισμός Υπολογιστών

Τετράδιο Εργασιών

Γ΄ ΕΠΑ.Λ.

ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΚΔΟΣΕΩΝ
«ΔΙΟΦΑΝΤΟΣ»

Προγραμματισμός Υπολογιστών

Γ΄ ΕΠΑ.Λ.

Τετράδιο Εργασιών

Γ' και Δ' τάξη ημερησίων και εσπερινών ΕΠΑ.Λ.
Τομέας Πληροφορικής

ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

ΠΡΟΕΔΡΟΣ:	**Κουζέλης Γεράσιμος,** Καθηγητής ΕΚΠΑ Γραφείο έρευνας, σχεδιασμού και εφαρμογών Β'
ΠΡΟΪΣΤΑΜΕΝΟΣ:	**Μάραντος Παύλος**
ΕΠΙΣΤΗΜΟΝΙΚΑ ΥΠΕΥΘΥΝΟΣ:	**Δρ. Τσαπέλας Θεοδόσιος,** Σύμβουλος Β' Πληροφορικής
ΣΥΓΓΡΑΦΙΚΗ ΟΜΑΔΑ	**Αράπογλου Αριστείδης,** Εκπαιδευτικός Πληροφορικής
	Βραχνός Ευριπίδης, Εκπαιδευτικός Πληροφορικής
	Κανίδης Ευάγγελος, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής
	Λέκκα Δήμητρα, Εκπαιδευτικός Πληροφορικής
	Μακρυγιάννης Παναγιώτης, Εκπαιδευτικός Πληροφορικής
	Μπελεσιώτης Βασίλειος, Σχολικός Σύμβουλος ΠΕ19- Πληροφορικής
	Παπαδάκης Σπυρίδων, Σχολικός Σύμβουλος ΠΕ19- Πληροφορικής
	Τζήμας Δημήτριος, Εκπαιδευτικός Πληροφορικής
ΕΠΙΜΕΛΕΙΑ - ΣΥΝΤΟΝΙΣΜΟΣ ΟΜΑΔΑΣ	**Κανίδης Ευάγγελος,** Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής
	Μπελεσιώτης Βασίλειος, Σχολικός Σύμβουλος ΠΕ19- Πληροφορικής
ΕΠΙΤΡΟΠΗ ΚΡΙΣΗΣ	**Βογιατζής Ιωάννης,** Επίκουρος Καθηγητής, Α.Τ.Ε.Ι. Αθηνών
	Εφόπουλος Βασίλειος, Σχολικός Σύμβουλος ΠΕ19- Πληροφορικής
	Κωτσάκης Σταύρος, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής
ΠΡΟΕΚΤΥΠΩΤΙΚΕΣ ΕΡΓΑΣΙΕΣ	**ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΕΚΔΟΣΕΩΝ «ΔΙΟΦΑΝΤΟΣ»**

ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ, ΕΡΕΥΝΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΙΝΣΤΙΤΟΥΤΟ ΕΚΠΑΙΔΕΥΤΙΚΗΣ ΠΟΛΙΤΙΚΗΣ

Αράπογλου Α., Βραχνός Ε., Λέκκα Δ., Κανίδης Ε.,
Μακρυγιάννης Π., Μπελεσιώτης Β., Τζήμας Δ., Παπαδάκης Σπ.

Προγραμματισμός Υπολογιστών

Τετράδιο Εργασιών

Γ' και Δ' τάξη ημερησίων και εσπερινών ΕΠΑ.Λ.
Τομέας Πληροφορικής

ΙΝΣΤΙΤΟΥΤΟ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΕΚΔΟΣΕΩΝ
«ΔΙΟΦΑΝΤΟΣ»

Εισαγωγικό σημείωμα

Το παρόν διδακτικό υλικό είναι άρρηκτα συνδεδεμένο με το Α.Π.Σ. του μαθήματος Προγραμματισμός Υπολογιστών (**ΦΕΚ 2010 τΒ'/16-09-2015**, Σελ. **23781-23791**) και με το αντίστοιχο Βιβλίο Μαθητή.

Το υλικό υποστηρίζει τους διδακτικούς στόχους της διδακτέας - εξεταστέας ύλης, όπως αυτό ορίζεται από την **Υ.Α. Φ6/160716/Δ4** - στο **ΦΕΚ 3143/30-9-2016/ τ.β'**- Καθορισμός διδακτέας - εξεταστέας ύλης των Πανελλαδικώς εξεταζόμενων μαθημάτων της Γ' τάξης Ημερήσιων και της Δ' τάξης Εσπερινών ΕΠΑ.Λ. για το σχολικό έτος **2016-2017**.

Έχει ως στόχο, μέσα από την υλοποίηση ποικίλων δραστηριοτήτων, πολλές από τις οποίες είναι λυμένες, να βοηθήσει στον εμπλουτισμό των γνώσεων και δεξιοτήτων για την επίλυση υπολογιστικών προβλημάτων και την ανάπτυξη κατάλληλων προγραμμάτων στη γλώσσα προγραμματισμού **Python**. Η προτεινόμενη ποικιλία δραστηριοτήτων δε διαφοροποιείται, ούτε αυξάνει την ύλη, όπως αυτή αναλύεται στο Βιβλίο Μαθητή. "Έχοντας υπόψη τη διαφορετικότητα των μαθητών ως προς το γνωστικό τους υπόβαθρο και ως προς τον τρόπο μάθησης, καταβάλλεται προσπάθεια, να εμπλέξει τους μαθητές στη μαθησιακή διαδικασία μέσα από την ενεργό συμμετοχή τους.

Αξίζει να σημειωθεί ότι για το γνωστικό αντικείμενο του προγραμματισμού, σύμφωνα με τις σύγχρονες θεωρίες για τη μάθηση, ο πειραματισμός των μαθητών μέσα στο εργαστηριακό περιβάλλον με τους Υπολογιστές, είναι άρρηκτα συνδεδεμένος τόσο για την ανάπτυξη θεωρητικών γνώσεων όσο και για την ανάπτυξη δεξιοτήτων. Υπό την έννοια αυτή προτείνεται, το εργαστηριακό και θεωρητικό μέρος να διδάσκονται ενιαία και μέσα στο σχολικό εργαστήριο Πληροφορικής.

Στις ασκήσεις που ακολουθούν, όλοι οι αλγόριθμοι διατυπώνονται στη γλώσσα **Python**, έκδοση **2.7.x**, ώστε να μπορούν να εκτελεστούν άμεσα στο προγραμματιστικό περιβάλλον. Όπου υπάρχει διαγραμματική αναπαράσταση, θεωρούμε ότι αυτή σχετίζεται μόνον με την υποβοήθηση της μάθησης και όχι με την εξεταστέα ύλη ή την κάθετη διδασκαλία διαγραμμάτων.

Σημαντικές Παρατηρήσεις

- Η έκδοση της **Python** που χρησιμοποιείται για την επίλυση και διερεύνηση αλγοριθμικών προβλημάτων είναι σε όλο το διδακτικό υλικό, η **Python 2** και πιο συγκεκριμένα η **2.7.10** αλλά μπορεί να χρησιμοποιηθεί οποιαδήποτε μεταγενέστερη της **2.7.10**, με την ανάλογη προσοχή σε διαφοροποιήσεις της.
- Αν θέλετε να γράψετε ελληνικούς χαρακτήρες μέσα σε ένα πρόγραμμα, είτε ως σχόλια είτε ως αλφαριθμητικά πρέπει στην πρώτη γραμμή του αρχείου **Python** να προσθέσετε την παρακάτω γραμμή κώδικα: **# _*_ coding: utf-8 -*-**
- Το περιβάλλον προγραμματισμού που προτείνεται για τη δημιουργία και τη διερεύνηση προγραμμάτων, είναι το **Python IDLE** της έκδοσης **Python 2.7.x.** Αν κριθεί σκόπιμο, μπορείτε να χρησιμοποιήσετε όμως και άλλα προγραμματιστικά περιβάλλοντα της **Python**, όπως το **PyScripter**, εκτός αν η εκφώνηση μιας άσκησης το περιορίζει.
- Η πρόοδος της ύλης γίνεται σε σπειροειδή μορφή και όχι κατ' ανάγκη σειριακά, μια και ένα μέρος της ύλης είναι γνωστό από τη Β' Τάξη και εδώ γίνεται εμβάθυνση σε αυτό.
- Στο βιβλίο χρησιμοποιείται και ο τελεστής // που επιστρέφει το πηλίκο της ακέραιας (ευκλείδειας) διαίρεσης δύο αριθμών. Ο τελεστής % επιστρέφει το υπόλοιπο της ακέραιας διαίρεσης.

Για λόγους απλοποίησης και μη διάσπασης της προσοχής του μαθητή χρησιμοποιείται το δεύτερο πληθυντικό πρόσωπο, καλύπτοντας και τα δύο γένη.

Περιεχόμενα

Μέρος Ι

ΚΕΦΑΛΑΙΟ 1 Από το πρόβλημα στην ανάπτυξη αλγορίθμου

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό, αρχικά θα προσεγγίσουμε θέματα που αφορούν στην αναγνώριση της δομής ενός προβλήματος και της πολυπλοκότητάς του. Στη συνέχεια, θα γνωρίσουμε τη διαδικασία της αφαίρεσης και την απλοποίηση ενός προβλήματος με την ανάλυσή του σε απλούστερα υποπροβλήματα. Τέλος θα γνωρίσουμε τον τρόπο που περιγράφουμε αλγοριθμικά τη λύση του εκφρασμένη με ψευδοκώδικα ή διάγραμμα ροής.

Λέξεις κλειδιά

Πρόβλημα, επίλυση προβλήματος, ανάλυση προβλήματος, αφαίρεση, πολυπλοκότητα, αλγόριθμος, αναπαράσταση αλγορίθμου.

Διδακτικές Ενότητες

Για να αναπτύξουμε προγράμματα που μας βοηθούν στην επίλυση των προβλημάτων, είναι αναγκαία η κατανόηση τριών βασικών εννοιών: της έννοιας του προβλήματος, της έννοιας του αλγορίθμου **(algorithm)** για τη περιγραφή της λύσης του και της δομής δεδομένων **(data structure)**, για να μπορούν να χρησιμοποιηθούν τα δεδομένα από τον αλγόριθμο. Η σχεδίαση αλγορίθμων αποτελεί σημαντικό μέρος της διαδικασίας επίλυσης ενός προβλήματος. Συνοδεύεται με την ανάλυση του προβλήματος σε απλούστερα, τη σύνθεση των λύσεων των επιμέρους προβλημάτων και κυρίως με την περιγραφή και μορφοποίηση της λύσης του. Η διαδικασία αυτή γίνεται, ώστε η λύση να μπορεί να αναπαρασταθεί σε μορφή κατανοητή από τον υπολογιστή.

Περιεχόμενα κεφαλαίου

Σημείωση

Το κεφάλαιο αυτό είναι εκτός διδακτέας και εξεταστέας ύλης.

ΚΕΦΑΛΑΙΟ 2 Ανάπτυξη προγράμματος

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό αναπτύσσονται, σε ένα πρώτο επίπεδο, θέματα σχετικά με τον κύκλο ανάπτυξης ενός προγράμματος και τις μεθοδολογίες που χρησιμοποιούνται. Αναλύονται θέματα ειδών προγραμματισμού και αναφέρονται αρχικές έννοιες για τον αντικειμενοστρεφή προγραμματισμό, καθώς και τεχνικές σχεδίασης και περιγραφής συστημάτων.

Λέξεις κλειδιά

Κύκλος ανάπτυξης προγράμματος, μοντέλα ανάπτυξης και σχεδίασης λογισμικού, προγραμματιστικά υποδείγματα, αντικειμενοστρεφής / δομημένος προγραμματισμός.

Διδακτικές Ενότητες

2. Ανάπτυξη προγράμματος
- **2.1** Κύκλος ανάπτυξης προγράμματος/λογισμικού
  - **2.1.1** Μοντέλο του καταρράκτη
  - **2.1.2** Μοντέλο σπείρας
  - **2.1.3** Η λογική συγγραφής προγράμματος ανάλογα με το είδος προγραμματισμού
  - **2.1.4** Προστακτικός προγραμματισμός
  - **2.1.5** Δηλωτικός προγραμματισμός
  - **2.1.6** Λοιπά πρότυπα και τεχνικές προγραμματισμού
  - **2.1.7** Ενδεικτικά περιβάλλοντα και γλώσσες προγραμματισμού
- **2.2** Εισαγωγή στις βασικές έννοιες του αντικειμενοστρεφούς προγραμματισμού
  - **2.2.1** Σύγκριση Διαδικαστικού και Αντικειμενοστρεφούς προγραμματισμού
  - **2.2.2** Αντικειμενοστρεφής σχεδίαση

Σημείωση

Το κεφάλαιο αυτό είναι εκτός διδακτέας και εξεταστέας ύλης.

ΚΕΦΑΛΑΙΟ 3 Βασικά στοιχεία γλώσσας προγραμματισμού

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό αναπτύσσονται τα βασικά χαρακτηριστικά του ολοκληρωμένου περιβάλλοντος ανάπτυξης της γλώσσας προγραμματισμού **Python**. Επίσης αναπτύσσονται οι βασικοί τύποι μεταβλητών, οι πρώτες βασικές εντολές (δηλώσεις) και η σύνταξη εκφράσεων με αριθμητικές και λογικές πράξεις.

Λέξεις κλειδιά

Προγραμματιστικό περιβάλλον ανάπτυξης, λογικές και αριθμητικές εκφράσεις, τελεστές, δομές και τύποι δεδομένων, μεταβλητή, εντολές, συναρτήσεις, διερμηνευτής.

Διδακτικές Ενότητες

3. Γνωριμία με το ολοκληρωμένο περιβάλλον ανάπτυξης της γλώσσας προγραμματισμού Python
   - 3.1 Μεταβλητές και τύποι δεδομένων
     - 3.1.1 Τύποι δεδομένων
   - 3.2 Αριθμητικές και λογικές πράξεις και εκφράσεις
   - 3.3 Βασικές (ενσωματωμένες) συναρτήσεις
   - 3.4 Δομή προγράμματος και καλές πρακτικές
   - 3.5 Τύποι και δομές δεδομένων στις γλώσσες προγραμματισμού

Δραστηριότητες - Ασκήσεις

Δραστηριότητα 1. Γνωριμία με το προγραμματιστικό περιβάλλον. Ένα απλό πρόγραμμα χαιρετισμού σε **Python**.

Ας δοκιμάσουμε να δημιουργήσουμε ένα μικρό πρόγραμμα σε Python, με στόχο να αποθηκεύσουμε προσωρινά ένα μήνυμα χαιρετισμού σε μια μεταβλητή και στη συνέχεια να εμφανιστεί στην οθόνη του υπολογιστή το μήνυμα: «ΚΑΛΗΜΕΡΑ ΣΕ ΟΛΗ ΤΗΝ ΕΛΛΑΔΑ».

Για το πρόγραμμα αυτό θα χρησιμοποιήσουμε μια μεταβλητή με το όνομα **greeting**, που θα αναφέρεται στο μήνυμα χαιρετισμού "ΚΑΛΗΜΕΡΑ ΣΕ ΟΛΗ ΤΗΝ ΕΛΛΑΔΑ" και την εντολή εμφάνισης **print**, μια εντολή *εξόδου* με τη δυνατότητα να εμφανιστεί στην οθόνη ένα μήνυμα, το αποτέλεσμα μιας πράξης ή το περιεχόμενο μιας μεταβλητής.

*Εάν συντάξουμε και εκτελέσουμε το πρόγραμμα στην **Python 2.7**:*

```
# Παράδειγμα 1 - Ex01_first_prog.py
# ----------------------------
greeting = 'ΚΑΛΗΜΕΡΑ ΣΕ ΟΛΗ ΤΗΝ ΕΛΛΑΔΑ'
print greeting
```

Θα δούμε ως αποτέλεσμα:

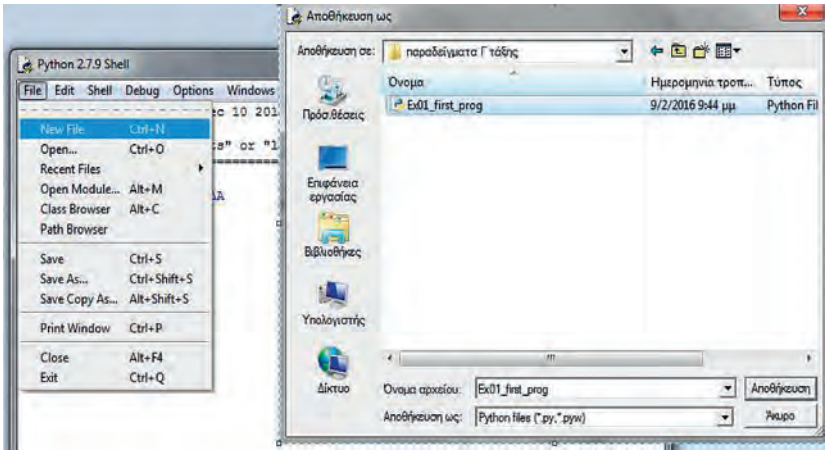
```
>>>
ΚΑΛΗΜΕΡΑ ΣΕ ΟΛΗ ΤΗΝ ΕΛΛΑΔΑ
```

Βοήθεια - παρατηρήσεις

Για τη συγγραφή προγραμμάτων, θα χρησιμοποιούμε τον *επεξεργαστή κώδικα* **(editor)** που μας προσφέρει το προγραμματιστικό περιβάλλον **IDLE**, με την ακόλουθη διαδικασία:

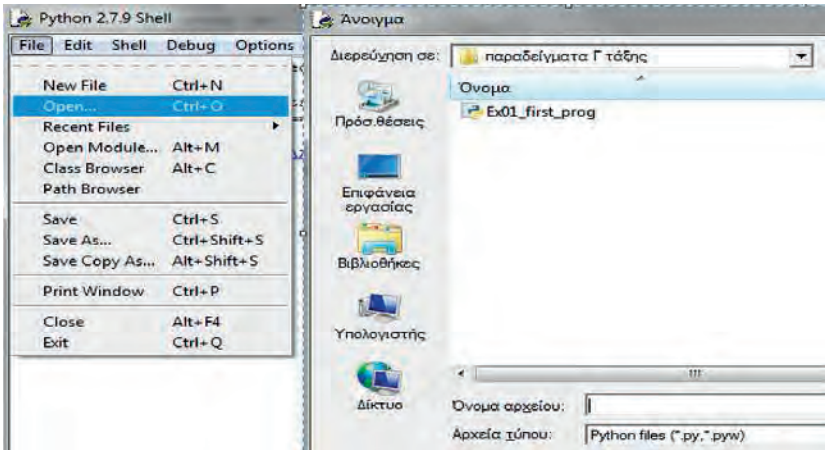
- άνοιγμα νέου αρχείου **(New File)**, από το μενού επιλογών **File**
- σύνταξη του κώδικα του προγράμματος
- αποθήκευση του προγράμματος **(Save**, όνομα αρχείου με κατάληξη **.py)**
- εκτέλεση του προγράμματος, από την επιλογή **Run→Run Module** (ή πατώντας το πλήκτρο F5), εμφάνιση των αποτελεσμάτων στο **Python Shell 2.7.x**

Για την αποθήκευση της εργασίας μας στην **Python**, πρέπει να χρησιμοποιήσουμε από το οριζόντιο μενού, στο πάνω μέρος του παραθύρου, το **File→Save As** (Αρχείο, Αποθήκευση ως). Στη συνέχεια στο αναδυόμενο παράθυρο που ανοίγει, επιλέγουμε το φάκελο που θα αποθηκεύσουμε το πρόγραμμά μας (Αποθήκευση σε:) και δίνουμε ένα χαρακτηριστικό όνομα αρχείου, όπως για παράδειγμα: **Ex01_first_prog** (εικόνα **3.1**). Στο αρχείο προστίθεται η κατάληξη .py. (Αποθήκευση ως: **Python files (*.py, *.pyw)**, που είναι η χαρακτηριστική κατάληξη για τα προγράμματα σε **Python**. Είναι χρήσιμο να δημιουργήσουμε από την αρχή ένα φάκελο, όπου εκεί θα αποθηκεύουμε όλα τα προγράμματά μας και τις ασκήσεις μας. Στο φάκελο μπορούμε για παράδειγμα να δώσουμε το όνομα «Παραδείγματα Γ τάξης».



Εικ. 3.1 *Αποθήκευση ενός προγράμματος* ***Python***

Αν θέλουμε αργότερα να επεξεργαστούμε ή/και να εκτελέσουμε το πρόγραμμά μας, τότε από το μενού **File**, επιλέγουμε άνοιγμα **(Open)**, βρίσκουμε το όνομα του αρχείου και το επιλέγουμε (Εικόνα 3.2).



Εικ. 3.2 *Επιλογή για το άνοιγμα ενός αρχείου προγράμματος* ***.py***

Δραστηριότητα 2. Εξάσκηση στο περιβάλλον **IDLE/Python**

Να ανοίξετε το περιβάλλον του **IDLE-Shell (>>>)** και να κάνετε τις παρακάτω ενέργειες:

- Πληκτρολογήστε **help()** και πατήστε **ENTER**. Τι κάνει αυτή η εντολή;
- Πληκτρολογήστε: **keywords**. Ποιο είναι το αποτέλεσμα της εντολής αυτής;
- Πληκτρολογήστε **print**. Διαβάστε προσεκτικά τη σύνταξη και λειτουργία της εντολής **(statement) print** και βγείτε από το περιβάλλον της βοήθειας πατώντας **ENTER**.
- Χρησιμοποιείστε την εντολή **print** για να εμφανίσετε στην οθόνη το μήνυμα: *Καλημέρα φίλοι μου, μόλις γνώρισα το περιβάλλον του διερμηνευτή* ***IDLE***. (Να μη ξεχάσετε να βάλετε εισαγωγικά, όπου χρειάζονται).
- Έχει διαφορά αν χρησιμοποιήσετε μονά ή διπλά εισαγωγικά; Μπορείτε να ξεκινήσετε με μονά εισαγωγικά και να κλείσετε με διπλά;
- Παρατηρείστε την οθόνη του **IDLE**. Γιατί κάποιες εντολές και λέξεις είναι με διαφορετικό χρώμα;
- Να γράψετε μια απλή γραμμή σχολίων της αρεσκείας σας (προσοχή να ξεκινάει με #). Τα σχόλια είναι πολύ χρήσιμα για να τεκμηριώσετε τον κώδικα σας.

Δραστηριότητα 3. Τύποι δεδομένων (ερωτήσεις αντιστοίχισης)

Σε ποιο τύπο δεδομένων στη γλώσσα προγραμματισμού **Python** αντιστοιχούν οι τιμές της αριστερής στήλης του παρακάτω πίνακα. Να συνδέσετε κατάλληλα τις τιμές της αριστερής στήλης με το σωστό τύπο δεδομένων της δεξιάς στήλης. Να σημειωθεί ότι περισσότερες από μία επιλογές της στήλης Α αντιστοιχούν σε κάποια από τις επιλογές της στήλης Β.

Στήλη Α (Τιμή)	**Στήλη Β** (Τύπος δεδομένων)
1. -27 2. 35.7 3. 'False' 4. True 5. "432.12" 6. 'μεταβλητή' 7. 12 / 2 8. 20 % 3	Α. **int** (ακέραια) Β. **float** (κινητής υποδιαστολής) Γ. **string** (συμβολοσειρά) Δ. **bool** (λογική)

Δραστηριότητα 4 (Σωστό-Λάθος). Δίνοντας όνομα σε μία μεταβλητή

Ποιο από τα παρακάτω υποψήφια ονόματα της αριστερής στήλης του πίνακα δεν είναι αποδεκτό ως όνομα μεταβλητής. Σημειώστε με **Λ (Λάθος)** αυτά που πιστεύετε ότι δεν είναι αποδεκτά ονόματα και με **Σ (Σωστό)** εκείνα που πιστεύετε ότι είναι αποδεκτά.

	Σ/Λ
x!b	
Metavliti3	
Metavliti+3	
Kila 2	
mikos_1	
245	
1onoma	
Print	

Δοκιμάστε στο περιβάλλον της **Python** να δώσετε τα παραπάνω ως ονόματα μεταβλητών, ελέγχοντας τις απαντήσεις σας. Στη συνέχεια δικαιολογήστε τις απαντήσεις σας απαριθμώντας τους βασικούς κανόνες που πρέπει να ακολουθούμε για να δώσουμε ένα όνομα σε μια μεταβλητή.

.....

.....

.....

Δραστηριότητα 5. Εμβάθυνση στις μεταβλητές και τύπους δεδομένων.

Να καταγράψετε τι πιστεύετε ότι θα εμφανιστεί στην οθόνη μετά την εκτέλεση των παρακάτω τμημάτων προγραμμάτων (ξεκινήστε από την αριστερή στήλη). Επαληθεύστε τις απαντήσεις σας εκτελώντας τα προγράμματα μέσα από το περιβάλλον της γλώσσας **Python**.

Α.
```
>>> x = 35
>>> y = 10
>>> x = x / y
>>> print x
..... Τι παρατηρείτε;
```

Β.
```
>>> x = 45
>>> y = 10
>>> divmod(x,y)
.....
>>> x / y
.....
>>> x % y
.....
>>> divmod(y,x)
.....
```

Γ.
```
>>> a = 0xB
>>> print a
.....
```

Δ.
```
>>> x,y,z = 1, 4, "today"
>>> print z, x
.....
```

Ε.
```
>>> x = 234
>>> y = 456.7
>>> x,y = y,x
>>> print x
.....
>>> print y
..... Τι παρατηρείτε;
```

ΣΤ.
```
>>> x = 2
>>> x = 2 **3 + 2/3
>>> print x
.....
>>>x = 2 **3 + 2/ float(3)
>>>print x
.....
```

Ζ.
```
>>>x = 2
>>>x-= 1
>>>x = x -1
>>>print x
.....
```

Βοήθεια - απαντήσεις:

Α.3 **Β.** (4, 5), 4, 5, (0, 10), **Γ.**11, **Δ.** today 1
Ε. 456.7, 234, **ΣΤ.** 8, 8.66666666667, **Ζ.**0

Δραστηριότητα 6. Πράξεις

Να συμπληρώσετε κατάλληλα τον παρακάτω πίνακα υπολογίζοντας το αποτέλεσμα, με την εφαρμογή της προτεραιότητας των πράξεων.

Στην αριστερή στήλη του πίνακα παρουσιάζεται μια πράξη που πρέπει να εκτελεστεί στον υπολογιστή χρησιμοποιώντας το προγραμματιστικό περιβάλλον της γλώσσας **Python 2.7**.

Αρχικά να συμπληρώσετε τη μεσαία στήλη με τα αποτελέσματα που πιστεύετε ότι θα εμφανιστούν στην οθόνη του υπολογιστή μετά την εκτέλεση της πράξης. Στη συνέχεια, να επαληθεύσετε τα στοιχεία που συμπληρώσατε, πληκτρολογώντας και εκτελώντας κάθε πράξη ξεχωριστά μέσα στο περιβάλλον της γλώσσας **Python**.

Πράξεις	Αναμενόμενο αποτέλεσμα	Αποτέλεσμα στην οθόνη
15 + 2/2		
5 * (3 + 2) / 10		
15 * 2 / 3		
15 * 2 / 4		
15*2.0/4		
2**3*3**2		
8/4%2		
11%3-2*2		
2*(5%3)+4/(1+3)		

Δραστηριότητα 7. Βρες το αποτέλεσμα. Διερεύνηση εκφράσεων
Δίνονται οι παρακάτω εκφράσεις σε **Python**

Α) (x+y)/(x**3+y**2+1)*z

Αν **x=2**, **y=3** και **z=1** ποιο αποτέλεσμα θα εμφανιστεί στην οθόνη του υπολογιστή;

Επαληθεύστε την απάντησή σας πληκτρολογώντας διαδοχικά στο προγραμματιστικό περιβάλλον της **Python** τις εκφράσεις αντικαθιστώντας τις μεταβλητές με τις αντίστοιχες τιμές τους. Δικαιολογήστε το αποτέλεσμα που εμφανίστηκε στην οθόνη.

Β) a+b*(a**c+c/2)**2

Αν **a=1** και **b=2** και **c=4**, ποιο αποτέλεσμα από τα παρακάτω θα εμφανιστεί στην οθόνη του υπολογιστή:

α) 27 β) 19 γ) 64

Γ) (x*y+x+2)**2+3**2

Αν **x=2**, **y=3** ποιο αποτέλεσμα από τα παρακάτω θα εμφανιστεί στην οθόνη του υπολογιστή:

α) 109 β) 81 γ) 36

Δραστηριότητα 8. Τύποι δεδομένων

Να δώσετε διαδοχικά τις παρακάτω εντολές στο διερμηνευτή της **Python** και να εξηγήσετε τα αποτελέσματα.

```
>>> a = 2
>>> type(a)
>>> b = 10
>>> p = a*b
>>> print a + b
>>> print " a + b "
>>> type( "python")
>>> type( 3.14 )
>>> type( a == 2 )
```

Δραστηριότητα 9. Πράξεις - Τύποι δεδομένων

Να δώσετε τις παρακάτω εντολές στο διερμηνευτή της **Python 2.7** και να εξηγήσετε τη λειτουργία του τελεστή της διαίρεσης.

```
>>> 3/2          >>> type( 1 / 2 )
>>> 3.0 / 2      >>> type( 1.0 / 2)
>>> 1/2          >>> int( 1.0 / 2.0)
>>> 1.0 / 2      >>> float( 1 ) / 2
```

Δραστηριότητα 10. Λογικοί τελεστές (Θέμα για συζήτηση).

Συζητήστε στην τάξη για τη χρήση των λογικών πράξεων **not, or** και **and**, καθώς και για τους τελεστές σύγκρισης στην καθημερινή ζωή και στον προγραμματισμό υπολογιστικών συσκευών. Δώστε παραδείγματα συνδέοντας λογικές εκφράσεις για τον έλεγχο καθημερινών λειτουργιών, όπως για παράδειγμα: **ΑΝ** (φανάρι πράσινο για τον πεζό) **ΚΑΙ** (δεν έρχεται κινούμενο όχημα) **ΤΟΤΕ** διασχίζω τη διάβαση.

Δραστηριότητα 11. Λογικοί τελεστές και πίνακες αληθείας

Συμπληρώστε τις παρακάτω σχέσεις με το Ψευδής/**False/0** ή Αληθής/**True/1**.

1 != 0	True
1 != 1	
0 != 1	
0 != 0	False

1 == 0	
1 == 1	True
0 == 1	
0 == 0	

Βοήθεια

!= όχι ίσο / Διάφορο	== ίσο / ταυτίζεται

Δραστηριότητα 12. Λογικές πράξεις

Να συμπληρώσετε τον παρακάτω πίνακα αληθείας

P	Q	P and Q	P or Q	not P	not P and not Q
True	True				
True	False				False (Η λύση δίνεται για βοήθεια)
False	True				
False	False				

Δραστηριότητα 13. Λογικές εκφράσεις

Ποια η τιμή αληθείας για τις παρακάτω προτάσεις (Ψευδής/**False/0** - Αληθής/**True/1**); Συμπληρώστε αρχικά τον πίνακα χειρόγραφα. Στη συνέχεια δοκιμάστε τις προτάσεις στο περιβάλλον της **Python** και συγκρίνετε τις απαντήσεις σας με τα αποτελέσματα που θα εμφανιστούν.

Εκφράσεις	Χειρόγραφος υπολογισμός	Αποτέλεσμα περιβάλλοντος Python
1 == 1 and 0 != 1		
"test" == 'test'		
1 == 1 or 2 != 1 or 5 == 5		
False and 1 != 0		Η λύση εδώ είναι **False**
not (4 == 4 and 1 != 0)		
not (5==5 or (1!=0 and 6 != 7))		Η λύση εδώ είναι **False**

Δραστηριότητα 14. Τελεστές (Ερωτήσεις αντιστοίχισης)

Κάντε τις κατάλληλες συνδέσεις. Γράψτε με τη σειρά κάτω από τον πίνακα τους αριθμούς της στήλης Α και δίπλα τους το αντίστοιχο γράμμα της στήλης Β, ώστε να σχηματίζεται η σωστή απάντηση.

Στήλη Α	Στήλη Β
1. *	α. Σχεσιακός τελεστής
2. False	β. Λογικός τελεστής
3. >	γ. Αριθμητικός τελεστής
4. and	δ. Αλφαριθμητική τιμή
5. length	ε. Λογική τιμή
6. "πλάτος"	στ. Όνομα Μεταβλητής

.....

.....

Δραστηριότητα 15. Η εντολή **print (statement)**

Να δώσετε τις παρακάτω εντολές στο διερμηνευτή της **Python** και να εξηγήσετε τα αποτελέσματα. Τι συμπεράσματα βγάζετε για τη λειτουργία της print;

```
>>>  print  1,000, 000
>>>  print  (1, 2, 3)
>>>  print  1, ; print 2, ; print 3;
>>>  print  1, 2, 3
>>>  print  1;2;3
```

Δραστηριότητα 16. Περισσότερα για την εντολή **print**

Μπορείτε να προβλέψετε τα αποτελέσματα των παρακάτω εντολών πριν τις εκτελέσετε στο διερμηνευτή της **Python**; Επαληθεύστε τις απαντήσεις σας μέσα από το περιβάλλον της **Python**.

```
>>> print “Python ” , 2
>>> print “Python ” + 2
>>> print “Python ” + str(2)
>>> print “Monty” + “Python”
>>> print 3 * “Python”
>>> print 3 * “Python” * 2
```

Δραστηριότητα 17. Μεταβλητές

Να γράψετε τις αντίστοιχες εντολές σε **Python** που επιτελούν τις παρακάτω λειτουργίες:

- Να οριστεί η μεταβλητή με όνομα **name**, ώστε να έχει ως τιμή το όνομά σας.
- Μηδενισμός της μεταβλητής **number**.
- Αύξηση κατά 1 της μεταβλητής **number**.
- Αύξηση της μεταβλητής **number** κατά **50%**.
- Διπλασιασμός της μεταβλητής **number**.
- Να οριστεί η μεταβλητή **value=456.7** και στη συνέχεια να οριστεί η μεταβλητή **value_square**, που να περιέχει το τετράγωνο της **value**.
- Να οριστεί η μεταβλητή με όνομα **logic** που να περιέχει την τιμή **False**.

Δραστηριότητα 18. Αντιμετάθεση τιμών μεταξύ δύο μεταβλητών

Να εξηγήσετε τη λειτουργία των παρακάτω τμημάτων κώδικα σε γλώσσα **Python**, αρχικά του τμήματος Α' (αριστερή στήλη) και στη συνέχεια του τμήματος Β' (δεξιά στήλη). Τα δύο τμήματα κώδικα επιτελούν την ίδια λειτουργία;

A

```
a = input( “a = ” )
b = input( “b = ” )
temp = a
a = b
b = temp
print a, b
```

B

```
a = input( “a = ” )
b = input( “b = ” )
a,b=b,a
print a, b
```

Δραστηριότητα 19.

Να γράψετε πρόγραμμα σε γλώσσα **Python**, που να διαβάζει το μήκος της ακτίνας ενός κύκλου και να τυπώνει τη διάμετρο, το μήκος και το εμβαδόν αυτού του κύκλου. **Βοήθεια:** Η διάμετρος του κύκλου δίνεται από τον τύπο **d=2*r**, η περίμετρος από τον τύπο **p=2*π*r** (όπου **π≈3,14**) και το εμβαδόν από τον τύπο $E=\pi*r^2$.

Ενδεικτική Λύση

```
import math # βιβλιοθήκη με μαθηματικές συναρτήσεις
r = float(input ("Δώστε την ακτίνα του κύκλου : "))
d = 2*r
print "Η διάμετρος του κύκλου με ακτίνα ",r, " είναι: ",d
p=2*math.pi*r
print "Η περίμετρος του κύκλου με ακτίνα " ,r, " είναι: ", p
E=math.pi* r**2
print "Το εμβαδόν του κύκλου με ακτίνα ",r, " είναι: ",E
```

Δραστηριότητα 20. Προσθέστε τα κατάλληλα σχόλια

Να συμπληρώσετε ως σχόλια (τεκμηρίωση) την επεξήγηση της ανάθεσης για κάθε πρόταση:

```
a = b = c = 0      # .....
a, b = 1, 2.5      # .....
a, b, c = (1,2,3)  # .....
a, b = b, a        # .....
```

Δοκιμάστε τις απαντήσεις σας στο περιβάλλον **IDLE** της **Python** χρησιμοποιώντας και την εντολή **print**, ώστε να εμφανιστούν τα αποτελέσματα για κάθε ανάθεση τιμών.

ΚΕΦΑΛΑΙΟ 4 Αλγοριθμικές δομές

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό γίνεται εμβάθυνση σε βασικές έννοιες, αλγοριθμικές δομές και τεχνικές προγραμματισμού, καθώς και στην αξιοποίησή τους για την επίλυση προβλημάτων με προγραμματισμό σε γλώσσα προγραμματισμού **Python**.

Λέξεις κλειδιά

Αλγοριθμικές δομές, δομή ακολουθίας, δομή επιλογής, δομή επανάληψης, συναρτήσεις.

Διδακτικές Ενότητες

4. Αλγοριθμικές δομές
- **4.1** Αλγοριθμικές δομές - Ροές εκτέλεσης προγράμματος
  - **4.1.1** Δομή ακολουθίας
  - **4.1.2** Δομή επιλογής **if (ΑΝ)**
  - **4.1.3** Δομή επανάληψης **(for** και **while)**
- **4.2** Συναρτήσεις
  - **4.2.1** Δημιουργώντας δικές μας συναρτήσεις
  - **4.2.2** Παράμετροι συναρτήσεων

Δραστηριότητα 1. Αλγοριθμικές δομές - Δομή ακολουθίας - Ροές εκτέλεσης προγράμματος.

Μελετήστε το παρακάτω τμήμα προγράμματος και προσπαθήστε να βρείτε το αποτέλεσμα που θα εμφανιστεί στην οθόνη μετά την εκτέλεσή του. Εκτελέστε το πρόγραμμα στο διερμηνευτή της **Python** και απαντήστε στα ερωτήματα:

```
# Διερεύνηση x += 1 / x = x + 1
x = 0
print 'x= ', x
x += 1
x = x + 1
x = x - 1
x -= 1
print 'x= ', x
```

- Τι θα εμφανιστεί από την εκτέλεση του προγράμματος;

  .....

- Τι αποτέλεσμα έχουν οι εντολές **x+=1** και **x-=1**;

  .....

- Με ποιες εντολές είναι ισοδύναμες οι δύο παραπάνω εντολές;

  .....

Δραστηριότητα 2. Δομή ακολουθίας (Υπολογισμός μέσου όρου τριών αριθμών)

Να γραφεί πρόγραμμα σε **Python** που να διαβάζει τις τιμές τριών αριθμών, να υπολογίζει το μέσο όρο τους και να τον εμφανίζει στην οθόνη.

Βοήθεια - ενδεικτική λύση. Να συγκρίνετε τη δική σας λύση με αυτήν που ακολουθεί:

Πρόγραμμα σε Python (έκδοση 2.7.x)

```
# Πρόγραμμα ΜΕΣΟΣ ΟΡΟΣ ΤΡΙΩΝ ΑΡΙΘΜΩΝ
A = input('Δώσε τον πρώτο αριθμό Α: ' )
B = input('Δώσε τον δεύτερο αριθμό Β: ' )
C = input('Δώσε τον τρίτο αριθμό Γ: ')
MESOS_OROS = (A+B+C)/3.0
print 'Ο μέσος όρος των αριθμών: ',A,B,C', είναι:',
MESOS_OROS
```

Διερεύνηση

Τι διαφορά μπορεί να υπάρξει στο αποτέλεσμα, αν γράψουμε την έκφραση: **MESOS_OROS = (A+B+C) / 3** αντί της **MESOS_OROS = (A+B+C)/3.0** (με τα A,B,C να έχουν ακέραιες τιμές); (Αφορά την έκδοση **Python 2.7.x**)

Δραστηριότητα 3. Γεννήτρια αριθμών

Να εκτελέσετε τρεις (3) τουλάχιστον φορές στο διερμηνευτή της **Python** τις παρακάτω εντολές και στη συνέχεια απαντήστε στα ερωτήματα:

- Τι τύπου είναι οι τιμές που εμφανίζονται (ακέραιες, κινητής υποδιαστολής);

  .....

- Ποιο το διάστημα των τιμών της μεταβλητής **number**;

  .....

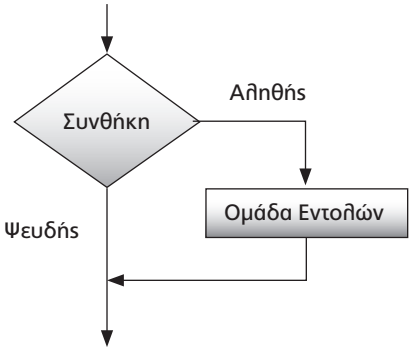
```
import random # αρχικά δηλώνουμε ότι θα χρησιμοποιήσουμε τη random
number = random.randint(1, 6) # επιστρέφει έναν τυχαίο αριθμό στο [1, 6]
print number
```

Βοήθεια

Η **random.randint(start, end)** επιστρέφει, με τυχαίο τρόπο, έναν ακέραιο αριθμό μέσα από το αριθμητικό διάστημα **[start, end]**.

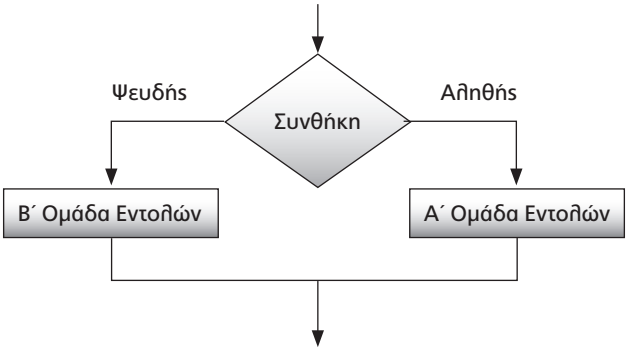
Θεωρητική ανασκόπηση. Δομή επιλογής
Απλή δομή επιλογής

```
if <συνθήκη ελέγχου> :
# εντολές που θα εκτελεσθούν αν ισχύει (αληθής-True) η συνθήκη ελέγχου
```



Σύνθετη δομή επιλογής if .... else (ΑΝ_ΑΛΛΙΩΣ)

```
if <συνθήκη ελέγχου> :
#εντολές που θα εκτελεσθούν, αν η συνθήκη ελέγχου είναι αληθής
else:
#εντολές που θα εκτελεσθούν αν η συνθήκη ελέγχου είναι ψευδής
```



Δραστηριότητα 4. Δομή επιλογής **if-else** (Λογικές εκφράσεις- συνθήκες)

Συμπληρώστε στη 2η στήλη του πίνακα, ποια θα είναι τα αποτελέσματα μετά την εκτέλεση του αντίστοιχου κώδικα **Python** της 1ης στήλης. Να λάβετε υπόψη, ότι το αποτέλεσμα εξαρτάται από την τιμή αληθείας των λογικών εκφράσεων της συνθήκης ελέγχου μέσα στη δομή επιλογής **if**.

Στη συνέχεια εκτελέστε κάθε τμήμα του κώδικα στο περιβάλλον της **Python** συγκρίνοντας τα αποτελέσματα που εμφανίζονται στην οθόνη με τα δικά σας.

if ...else	Αναμενόμενο Αποτέλεσμα (Χειρόγραφα)	Αποτέλεσμα περιβάλλοντος **Python**
1 == 1 and 0 != 1		

# " " και ' ' 	
if "test" == 'test':
  print "True"
else:
  print "False-0"
# Ελληνικά και Λατινικά 	
if "ΚΑΛΗΜΕΡΑ" == "ΚΑΛΗΜΕΡΑ":
  print "True"
else:
  print "False-0"		

if (1 == 1 or (2 != 1 or 5 == 5)):
  print "True"
else:
  print "False-0"		

if (2!=2 and 1!= 0):
  print "True"
else:
  print "False-0"
		False-0

# true και True 	
if (true):
  print "True"
else:
  print "False-0"
		Error (λόγω του true)

# true και True 	
if True:
  print "True"
else:
  print "False-0"		

Δραστηριότητα 5. Δομή επιλογής **if ... else ...** (Άρτιος ή περιττός)

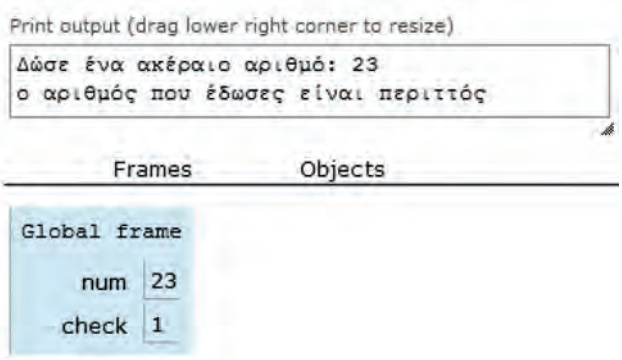
Να γραφεί πρόγραμμα σε **Python** που να διαβάζει έναν ακέραιο αριθμό και να ελέγχει αν είναι άρτιος ή περιττός. Στη συνέχεια να εμφανίζει στην οθόνη αντίστοιχο μήνυμα.

Οδηγία: α) Δημιουργείστε το πρόγραμμα, πρώτα σε χαρτί και στη συνέχεια καταχωρήστε και εκτελέστε το στο περιβάλλον της **Python**, β) μελετήστε τις τυχόν διαφορές με την αρχική σας δημιουργία.

Βοήθεια - ενδεικτική λύση

Έναν ακέραιος αριθμός x είναι άρτιος αν είναι πολλαπλάσιος του **2.** Δηλαδή αν το υπόλοιπο της διαίρεσης του **x** δια του **2** είναι **0**. Διαφορετικά είναι περιττός.

```
#πρόγραμμα άρτιος ή περιττός
num=int(input('Δώσε ένα ακέραιο αριθμό: '))
check = num % 2
if check == 0:
    print "Ο αριθμός είναι άρτιος"
else:
    print "Ο αριθμός που έδωσες είναι περιττός"
```



Εικ. 4.1: *Αποτέλεσμα εκτέλεσης κώδικα στο* ***PythonTutor****.*

Θεωρητική ανασκόπηση

Πολλαπλή Επιλογή με χρήση της εντολής **elif**

```
if <συνθήκη>:
    <εντολές>
elif <συνθήκη2>:
    <εντολές_2>
else:
    <εντολές_3>
```

Σημείωση: Οι πολλαπλές επιλογές μπορούν να υλοποιηθούν και με εμφωλευμένες δομές Αν-αλλιώς, δηλαδή μέσα σε μία δομή Αν-αλλιώς να περικλείονται και άλλες δομές Αν-αλλιώς.

Δραστηριότητα 6. Δομή επιλογής **if ... elif ... else ...** (Συμπλήρωσε κατάλληλα τις στήλες του πίνακα)

Μελετήστε τον κώδικα **Python** (με εντολές **if**) στην αριστερή στήλη του πίνακα που ακολουθεί. Στη συνέχεια στη δεξιά στήλη του ίδιου πίνακα γράψτε αντίστοιχο πρόγραμμα σε **Python** χρησιμοποιώντας τη δομή επιλογής **if elif else**. Το πρόγραμμα θα πρέπει να επιστρέφει τα ίδια αποτελέσματα στην οθόνη με αυτά του προγράμματος της αριστερής στήλης, για όλους τους δυνατούς συνδυασμούς τιμών των μεταβλητών **ar1** και **ar2**. Στη συνέχεια δοκιμάστε με διάφορες χαρακτηριστικές τιμές των **ar1** και **ar2** τα δύο προγράμματα στο περιβάλλον της **Python**.

Κώδικας με χρήση της απλής **if**	Κώδικας με χρήση της **if - elif - else**

ar1 = input('Δώσε έναν αριθμό A : ')
ar2 = input('Δώσε δεύτερο αριθμό B: ')
if ar1 < ar2:
  print "A < B"
if ar1 > ar2:
  print "A > B"
if ar1 == ar2:
  print " A = B "	

Βοήθεια: Παράδειγμα προγράμματος με πολλαπλή επιλογή Δείτε ένα άλλο παράδειγμα προγράμματος με πολλαπλή επιλογή.

```
x = input ("Δώσε αριθμό 0,1,2: ")
if (x == 0):
    print "έδωσες 0"
elif (x == 1):
    print "'έδωσες 1"
else:
    print "δεν έδωσες 0 ή 1"
```

Θεωρητική παρατήρηση: Στην περίπτωση της πολλαπλής επιλογής έχουμε λιγότερες συγκρίσεις σε σχέση με την απλή επιλογή. Στην απλή επιλογή σε κάθε εντολή **if** γίνεται νέος έλεγχος για το αν ισχύει η συνθήκη ελέγχου και ας έχει ικανοποιηθεί η συνθήκη ελέγχου προηγούμενης εντολής **if**. Για παράδειγμα αν η μεταβλητή **ar1** έχει την τιμή **1** και η **ar2** την τιμή **3**, θα ικανοποιηθεί η 1η συνθήκη και θα τυπωθεί A<B, αλλά το πρόγραμμα δε θα σταματήσει. Θα συνεχίσει και θα ελέγξει και τις υπόλοιπες συνθήκες ελέγχου στις εντολές **if** που ακολουθούν. Στην πολλαπλή επιλογή όμως, όταν βρεθεί η συνθήκη ελέγχου που είναι αληθής δε γίνονται οι άλλοι έλεγχοι για τις υπόλοιπες συνθήκες και ουσιαστικά σταματούν.

Θεωρητική ανασκόπηση

Εμφωλευμένη Επιλογή

Πρόκειται για εντολές if...else, που περικλείονται σε άλλες εντολές if...else, όπως στο παράδειγμα:

```
i=input("δώσε τον αριθμό i: ")
if i>0:
    if i<10:
        print i, "'θετικός αριθμός μικρότερος του 10"
    else:
        print i, "'θετικός αριθμός μεγαλύτερος ή ίσος του 10"
else:
    if i==0:
        print i, "μηδέν"
    else:
        print i, "αρνητικός"
```

Δραστηριότητα 7. Εμφωλευμένη επιλογή (Κλιμακωτή χρέωση)

Μια εταιρεία κινητής τηλεφωνίας ακολουθεί ανά μήνα την πολιτική τιμών που φαίνεται στον παρακάτω πίνακα:

Πάγιο=20€	
Χρόνος τηλεφωνημάτων (δευτερόλεπτα)	Χρονοχρέωση (€/ δευτερόλεπτο)
1-500 501-800 801 και άνω	0.02 0.009 0.007

Να γραφεί πρόγραμμα σε **Python** που:

- Να διαβάζει τη χρονική διάρκεια των τηλεφωνημάτων ενός συνδρομητή σε διάστημα ενός μήνα.
- Να υπολογίζει τη μηνιαία χρέωση του συνδρομητή.
- Να εμφανίζει σχετικό μήνυμα «η συνολική χρέωση του μήνα είναι: » και τη μηναία χρέωση του συνδρομητή.

Σημείωση: Η χρονοχρέωση θεωρείται κλιμακωτή. Δηλαδή τα πρώτα 500 δευτερόλεπτα χρεώνονται με 0.02(€/ δευτερόλεπτο), τα επόμενα 300 δευτερόλεπτα (από 501 -& 800) με 0.009 (€/ δευτερόλεπτο) και τα πέραν των 800 με 0.007 (€/ δευτερόλεπτο).

Βοήθεια - Ενδεικτική λύση

Πριν ξεκινήσουμε την επίλυση, είναι χρήσιμο για την κατανόηση της άσκησης, να διαλέγουμε διάφορα κατάλληλα χαρακτηριστικά δεδομένα μέσα από τα αριθμητικά διαστήματα που ορίζονται στην άσκηση. Με τα δεδομένα αυτά δοκιμάζουμε στο χαρτί τι αποτέλεσμα θα πρέπει να προκύψει. Για παράδειγμα, πόση είναι η τελική χρέωση αν μιλήσουμε 900 δευτερόλεπτα;

Μετά τις δοκιμές δημιουργούμε τους επιθυμητούς μαθηματικούς τύπους, τις κατάλληλες **if-else** για να φτιάξουμε τα διαστήματα τιμών που ισχύει ο μαθηματικός τύπος και τις υπόλοιπες εντολές που χρειάζονται για είσοδο των δεδομένων από το πληκτρολόγιο και έξοδο των αποτελεσμάτων στην οθόνη.

Παραδείγματα

Αν κάποιος έχει μιλήσει **400** δευτερόλεπτα η χρέωση θα είναι:

400*0.02+Πάγιο

Αν κάποιος έχει μιλήσει **720** δευτερόλεπτα η χρέωση θα είναι:

Τα αρχικά **500** δευτερόλεπτα επί **0.02€**, συν τα υπόλοιπα **220** δευτερόλεπτα επί **0.009€** συν το πάγιο.

500*0.02+(720-500)*0.009+Πάγιο

Αν κάποιος έχει μιλήσει **900** δευτερόλεπτα η χρέωση θα είναι:

Τα αρχικά **500** δευτερόλεπτα επί **0.02€**, συν τα **300** δευτερόλεπτα επί **0.009**, συν τα υπόλοιπα **100** δευτερόλεπτα επί **0.007€** συν το πάγιο.

500*0.02+300*0.009+(900-800)*0.007+Πάγιο.

Βοήθεια: Μια ενδεικτική λύση

```
#χρέωση τηλεφωνημάτων
time=input("Δώσε τη συνολική χρονική διάρκεια των τηλεφωνημάτων του μήνα σε δευτερόλεπτα")
pagio=20
if time<=500:
    amount=time*0.02+pagio
else:
    if time<=800:
        amount=500*0.02+(time-500)*0.009+pagio
    else:
        amount=500*0.02+300*0.009+(time-800)*0.007+pagio
print "Η συνολική χρέωση του μήνα είναι: ", amount
```

Θεωρητική ανασκόπηση. Δομή Επανάληψης For

Δομή επανάληψης (for)

```
for onoma_metavlitis in range (αρχή, μέχρι, βήμα):
    Εντολή_1
    Εντολή_2
    .....
    Εντολή_v
```

Λυμένο Παράδειγμα. Δομή Επανάληψης **for** (Άθροισμα πέντε ακεραίων αριθμών) Μελετήστε το παρακάτω πρόγραμμα, τα σχόλια δίπλα στον κώδικα και το αποτέλεσμά του. Το πρόγραμμα διαβάζει πέντε τυχαίους ακεραίους αριθμούς και υπολογίζει το άθροισμά τους. Στη συνέχεια καταχωρήστε το στο περιβάλλον της Python και εκτελέστε το δοκιμάζοντας διάφορα δεδομένα.

Πρόγραμμα σε **Python**

```
#Πρόγραμμα Άθροισμα 5 ακεραίων αριθμών
sum =0 #αρχικοποίηση της τιμής της μεταβλητής αθροίσματος
for i in range(1,6):
    x = int(input("Δώσε έναν αριθμό: " )) #διαβάζει έναν ακέραιο
    sum = x + sum #προσθέτει στο προηγούμενο άθροισμα το
                  νέο ακέραιο αριθμό
print 'Το αποτέλεσμα είναι ', sum #Το αποτέλεσμα που θα εμφανιστεί
                                  θα είναι το άθροισμα
                                  sum=((((0+x)+x)+x)+x)+x
```

Δραστηριότητα 8. Δομή Επανάληψης με **for** (συμπλήρωσης κώδικα). Υπολογισμός γινόμενου 10 διαδοχικών ακεραίων αριθμών.

Να συμπληρώσετε κατάλληλα τα κενά, ώστε το παρακάτω πρόγραμμα σε **Python** να υπολογίζει το γινόμενο των ακεραίων αριθμών **(1,2,3,4,5,6,7,8,9,10).** Δηλαδή το γινόμενο **Multi=1*2*3*4*5*6*7*8*9*10**, ή ισοδύναμα το γινόμενο **Multi=(((((((1 *2)*3)*4)*5)*6)*7)*8)*9)*10**.

```
# Πρόγραμμα γινόμενο ακεραίων αριθμών
Multi = ........
for i in range(.... , ....):
    Multi = Multi*.....
print ' Το αποτέλεσμα είναι ', Multi
```

Δραστηριότητα 9. Δομή Επανάληψης με **for** (Βρείτε τι θα εμφανίσει το πρόγραμμα).

Δίνεται το παρακάτω πρόγραμμα σε **Python**.

```
s = 0
for i in range(0, 10, 2):
    s = s + 1
    print i, s
print'--------'
s = 0
for i in range(10, 0, -2):
    s += 1
    print i, s
print s
```

1) Μελετήστε τον κώδικα ανά γραμμή εντολών και σημειώστε χειρόγραφα τα αποτελέσματα που περιμένετε να εμφανιστούν στην οθόνη του υπολογιστή.
2) Καταχωρήστε τον κώδικα στο περιβάλλον της Python και δοκιμάστε τον, συγκρίνοντας τα αποτελέσματα της οθόνης με αυτά του σημειώσατε.

Δραστηριότητα 10. Η συνάρτηση **range** και η δομή επανάληψης **for**.

Να εκτελέσετε τις παρακάτω συναρτήσεις (μία, μία ανά στήλη) στο διερμηνευτή και να σχολιάσετε τα αποτελέσματα. Ποια είναι η λειτουργία της ενσωματωμένης στην **Python** συνάρτησης **range()**;

```
>>> range( 5 )
>>> range( 1, 5 )
>>> range (1,5,1)
>>> range( 0, 5 )
>>> range( 1, 10, 2 )
>>> range( 10, 1, -2 )
>>> range(-10, -20, -10)
>>> range( 0 )
>>> range( 1 )
>>> type( range( 1 ) )
>>> range(4) == [0,1,2,3]
```

Να εκτελέσετε τα παρακάτω προγράμματα Α,Β,Γ στην Python και να σχολιάσετε τα αποτελέσματα.

Α	**Β**	**Γ**

sum = 0
 	
for i in [1,2,3,4,5,6]:
  print i
  sum = sum + i
 	
print "sum = ", sum
	sum = 0
	
	for i in range(1, 7) :
	  print i
	  sum = sum + i
	
	print "sum = ", sum
		sum = 0
		i = 1
		while i < 7 :
		  print i
		  sum = sum + i
		  i = i + 1
		print "sum = ", sum

Βοήθεια-Σημείωση

1) Η **range** (2) επιστρέφει τη λίστα [0,1]. Η **range** (1,7) επιστρέφει τη λίστα [1,2,3,4,5,6]. Για αυτό το λόγο η έκφραση **for i in range** (1,7) είναι ισοδύναμη με την έκφραση **for i in** [1,2,3,4,5,6].
2) Όταν το βήμα των επαναλήψεων είναι 1, στη **for i in range** (1,7) δεν είναι απαραίτητο να το αναφέρουμε και έτσι γράφουμε **range** (1,7) αντί **range** (1,7,1). Στη δομή επανάληψης **while** (όσο ισχύει συνθήκη **i<7** επανάλαβε) είναι όμως απαραίτητο για να γίνεται κάθε φορά έλεγχος αν ισχύει η συνθήκη **i<7** και να συνεχίσει η επανάληψη (αν ισχύει η συνθήκη) ή να σταματήσει (αν δεν ισχύει). Για αυτό **i=i+1**.
3) Στη δομή επανάληψης **while** (όσο ισχύει συνθήκη **i<7** επανάλαβε) πρέπει για να εκτελεστεί η επανάληψη, στην αρχή να δώσουμε την αρχική τιμή της μεταβλητής **i=1**, ώστε να γίνει ο πρώτος έλεγχος της συνθήκης **(1<7)** και να ξεκινήσει η επανάληψη.

Δραστηριότητα 11. (Η δομή επανάληψης for με εμφωλευμένες επαναλήψεις). Βρες τι κάνει το πρόγραμμα.

Μελετήστε και σημειώστε στο χαρτί, τι κάνει το παρακάτω πρόγραμμα, εάν δεχθούμε ότι με **h, m, s** συμβολίζουμε τις ώρες, λεπτά και δευτερόλεπτα αντίστοιχα. Στη συνέχεια καταχωρήστε το στο περιβάλλον της **Python**, εκτελέστε το και συγκρίνετε τα αποτελέσματα με αυτά που σημειώσατε. Παρατηρήστε ότι μέσα σε μία επανάληψη **for**, μπορεί να περικλείεται και άλλη (εμφωλευμένη) επανάληψη **for**.

```
for h in range (0,2):
    for m in range (0,3):
        for s in range (0,4):
            print h,m,s
```

Δραστηριότητα 12. Δομή Επανάληψης **for**. (Πολλαπλάσια του 9)

Να γραφεί πρόγραμμα που να ελέγχει όλους τους τριψήφιους ακεραίους αριθμούς και να εμφανίζει όσους είναι πολλαπλάσια του 9.

Βοήθεια: Όλοι οι τριψήφιοι ξεκινούν από το 100 και φτάνουν μέχρι το 999. Ένας αριθμός είναι πολλαπλάσιο του 9 αν διαιρείται ακριβώς με το 9, δηλαδή αν το υπόλοιπο της διαίρεσης του αριθμού n δια του 9 είναι 0.

```
# πρόγραμμα τριψήφιοι πολλαπλάσιοι του 9
for n in range(100,1000):
    if n % 9 ==0:
        print"ο αριθμός", n, "είναι πολλαπλάσιο του 9"
```

Τα τριψήφια πολλαπλάσια του 9 μπορούν επίσης να παραχθούν αν σκεφτούμε ότι αρκεί να ξεκινήσουμε από το 108 και να προσθέτουμε κάθε φορά 9 για να πάρουμε το επόμενο πολλαπλάσιο του 9.

```
# πρόγραμμα τριψήφιοι πολλαπλάσιοι του 9
for n in range(108,1000, 9):
        print"ο αριθμός", n, "είναι πολλαπλάσιο του 9"
```

Δραστηριότητα 13. Δομή Επανάληψης με **for**.

Μελετήστε το παρακάτω πρόγραμμα **Python**

```
arxh, telos, bhma = input('Δώσε τρεις τιμές, αρχή, τέλος, βήμα: ')
for i in range(arxh, telos, bhma):
    print (i)
```

Κατά την εκτέλεση του προγράμματος, ποιες τιμές πρέπει να εισάγουμε από το πληκτρολόγιο στις τρεις μεταβλητές, ώστε η εκτέλεση της εντολής επανάληψης να εμφανίσει διαδοχικά:

1) Όλους τους ακέραιους από το 1 μέχρι και το 100.
2) Τους άρτιους αριθμούς από το 0 έως το 100.
3) Τους περιττούς αριθμούς από το 0 έως το 100.

Δραστηριότητα 14 Δομές: **for** και **if**. (Υπολογισμός μέσου όρου, μέγιστου και ελάχιστου).

Ένα τμήμα της Γ' τάξης έχει 25 μαθητές. Να αναπτυχθεί πρόγραμμα σε **Python** που:

1) Να διαβάζει τους βαθμούς όλων των μαθητών στο μάθημα "Προγραμματισμός Υπολογιστών" και να υπολογίζει το μέσο όρο της βαθμολογίας του τμήματος για το μάθημα αυτό και να τον εμφανίζει στην οθόνη.
2) Να βρίσκει την υψηλότερη και τη χαμηλότερη βαθμολογία και να τις εμφανίζει στην οθόνη με κατάλληλο μήνυμα.

Οδηγία: Για την επίλυση να μη χρησιμοποιηθούν οι *ενσωματωμένες* μαθηματικές *συναρτήσεις* **max()** και **min()** της **Python**.

Δημιουργείστε τον κώδικα σε χαρτί και στη συνέχεια καταχωρήστε τον στο περιβάλλον της **Python**.

Βοήθεια - Λύση

Για την **εύρεση του μέσου όρου**, θα πρέπει να υπολογίσουμε αρχικά το άθροισμα όλων των βαθμών των μαθητών του τμήματος και στη συνέχεια να διαιρέσουμε το άθροισμα αυτό δια το πλήθος των μαθητών της τάξης.

Για να **υπολογίσουμε το άθροισμα** η διαδικασία είναι παρόμοια με αυτή του λυμένου παραδείγματος, άθροισμα πέντε ακεραίων αριθμών. Στη περίπτωσή μας θέλουμε το άθροισμα 25 αριθμών κινητής υποδιαστολής. Αρχικοποιούμε το άθροισμα=0 και στη συνέχεια, αφού ξέρουμε το πλήθος των βαθμών που θέλουμε να διαβάσουμε (ίσο με 25), χρησιμοποιούμε μια επανάληψη **for i in range** (1,26). Σε κάθε βήμα της επανάληψης διαβάζουμε το βαθμό του 1ου μαθητή στο 1ο βήμα, του 2ου μαθητή κ.ο.κ. Κάθε φορά στη προηγούμενη τιμή του αθροίσματος προσθέτουμε το νέο βαθμό, **athroisma=athroisma+bathmos**. Μετά το τέλος της επανάληψης, αφού αθροίστηκαν όλοι οι βαθμοί και έχουμε το τελικό άθροισμα, το διαιρούμε με το πλήθος των

βαθμών των μαθητών του τμήματος (25.0, ως αριθμός κινητής υποδιαστολής) για να υπολογίσουμε το μέσο όρο της βαθμολογίας όλων των μαθητών.

Για την **εύρεση του ελαχίστου** και αντίστοιχα του **μεγίστου** μπορούμε να εργαστούμε με δύο τρόπους

1ος Τρόπος

Όταν γνωρίζουμε ποια είναι η μέγιστη δυνατή τιμή που μπορεί να πάρει η μεταβλητή βαθμός (=20), τότε πριν την επανάληψη θέτουμε ως αρχική τιμή της μεταβλητής **min_bathmos**, που κρατάει την εκάστοτε ελάχιστη τιμή, την τιμή αυτή (στο παράδειγμα = 20). Στη συνέχεια, μέσα σε μια δομή επανάληψης, συγκρίνουμε σε κάθε βήμα το βαθμό κάθε μαθητή (δηλ στο 1ο βήμα το βαθμό του 1ου μαθητή, στο 2ο βήμα το βαθμό του 2ου μαθητή κ.ο.κ.) με την εκάστοτε ελάχιστη τιμή της **min_bathmos**. Αν ο βαθμός του μαθητή είναι μικρότερος από την τρέχουσα ελάχιστη τιμή που έχει η μεταβλητή **min_bathmos**, τότε θέτουμε στη μεταβλητή **min_bathmos** την τιμή του βαθμού του μαθητή (που περιέχεται στη μεταβλητή **bathmos**). Η επανάληψη με τις συγκρίσεις γίνεται και για τους 25 μαθητές, ξεκινώντας από τον πρώτο μέχρι το τελευταίο (σύμφωνα με την for i in range(1,26))

Ανάλογα βρίσκουμε την μέγιστη τιμή. Όταν γνωρίζουμε ποια είναι η ελάχιστη δυνατή τιμή που μπορεί να πάρει η μεταβλητή βαθμός (=0), τότε πριν την επανάληψη θέτουμε ως αρχική τιμή στη μεταβλητή **max_bathmos**, που κρατάει την εκάστοτε μέγιστη τιμή, την τιμή αυτή (στο παράδειγμα = 0). Στη συνέχεια και μέσα σε μια δομή επανάληψης, συγκρίνουμε σε κάθε βήμα το βαθμό κάθε μαθητή (δηλ στο 1ο βήμα το βαθμό του 1ου μαθητή, στο 2ο βήμα το βαθμό του 2ου μαθητή κ.ο.κ.) με την εκάστοτε μέγιστη τιμή της **max_bathmos**. Αν ο βαθμός του μαθητή είναι μεγαλύτερος από τη τρέχουσα μέγιστη τιμή που έχει η μεταβλητή **max_bathmos**, τότε θέτουμε στη μεταβλητή **max_bathmos** την τιμή του βαθμού του μαθητή (που περιέχεται στη μεταβλητή **bathmos**). Η επανάληψη με τις συγκρίσεις γίνεται και για τους 25 μαθητές, ξεκινώντας από τον πρώτο μέχρι τον τελευταίο (σύμφωνα με την for i in range(1,26))

2ος τρόπος

Στην περίπτωση που δε γνωρίζουμε την ελάχιστη δυνατή τιμή που είναι αποδεκτή. Αρχικά διαβάζουμε το βαθμό του πρώτου στη σειρά μαθητή και θέτουμε ως ελάχιστη και ως μέγιστη τιμή το βαθμό αυτό (**max_bathmos = bathmos** και **min_bathmos = bathmos**). Στη συνέχεια και μέσα σε μια δομή επανάληψης, συγκρίνουμε σε κάθε βήμα το βαθμό κάθε μαθητή με την εκάστοτε ελάχιστη τιμή της **min_bathmos**, ξεκινώντας από το δεύτερο στη σειρά μαθητή (δηλ στο 1ο βήμα το βαθμό του 2ου μαθητή, στο

2ο βήμα το βαθμό του 3ου μαθητή κ.ο.κ.). Αν ο βαθμός του μαθητή είναι μικρότερος από τη τρέχουσα ελάχιστη τιμή που έχει η μεταβλητή **min_bathmos**, τότε θέτουμε στη μεταβλητή **min_bathmos** την τιμή του βαθμού του μαθητή (που περιέχεται στη μεταβλητή **bathmos**). Η επανάληψη με τις συγκρίσεις γίνεται και για τους υπόλοιπους 24 μαθητές, ξεκινώντας από το δεύτερο μέχρι τον 25ο μαθητή (for i in range(2,26))

Παρόμοια ενεργούμε για την εύρεση μέγιστης τιμής. Μέσα σε μια δομή επανάληψης συγκρίνουμε σε κάθε βήμα το βαθμό κάθε μαθητή με την εκάστοτε μέγιστη τιμή της **max_bathmos**, ξεκινώντας από το δεύτερο στη σειρά μαθητή (δηλ στο 1ο βήμα το βαθμό του 2ου μαθητή, στο 2ο βήμα το βαθμό του 3ου μαθητή κ.ο.κ.). Αν ο βαθμός του μαθητή είναι μεγαλύτερος από την τρέχουσα μέγιστη τιμή που έχει η μεταβλητή **max_bathmos**, τότε θέτουμε στη μεταβλητή **max_bathmos** την τιμή του βαθμού του μαθητή (που περιέχεται στη μεταβλητή **bathmos**). Η επανάληψη με τις συγκρίσεις γίνεται και για τους υπόλοιπους 24 μαθητές, ξεκινώντας από το δεύτερο μέχρι τον 25ο μαθητή (for i in range(2,26))

Βοήθεια - ενδεικτική λύση

```
#Πρόγραμμα Βαθμολογία
athroisma=0
max_bathmos=0
min_bathmos=20
for i in range(1,26):
        print i,'os μαθητής'
        bathmos=input('Δώσε τη βαθμολογία του μαθητή: ')
        athroisma=athroisma+bathmos
        if bathmos>max_bathmos:
            max_bathmos=bathmos
        if bathmos<min_bathmos:
            min_bathmos=bathmos
Mesos_Oros=athroisma/25.0
print 'Ο μέσος όρος της βαθμολογίας του τμήματος
είναι',Mesos_Oros
Print 'Η υψηλότερη βαθμολογία είναι:', max_bathmos
Print 'Η χαμηλότερη βαθμολογία είναι:', min_bathmos
```

2η υλοποίηση

```
#Πρόγραμμα Βαθμολογία
bathmos=input('Δώσε τη βαθμολογία του 1ου μαθητή: ')
athroisma=bathmos
max_bathmos=bathmos
min_bathmos=bathmos
for i in range(2,26):
    print i,'os μαθητής'
    bathmos=input('Δώσε τη βαθμολογία του μαθητή: ')
    athroisma=athroisma+bathmos
    if bathmos>max_bathmos:
        max_bathmos=bathmos
    if bathmos<min_bathmos:
        min_bathmos=bathmos
Mesos_Oros=athroisma/25.0
print 'Ο μέσος όρος της βαθμολογίας του τμήματος είναι',
Mesos_Oros
print 'Η υψηλότερη βαθμολογία είναι:', max_bathmos
print 'Η χαμηλότερη βαθμολογία είναι:', min_bathmos
```

Θεωρητική παρατήρηση για την εύρεση ελάχιστης και μέγιστης τιμής.

Στη γλώσσα **Python** μπορούμε να επιλύσουμε το παραπάνω πρόβλημα αξιοποιώντας τη δομή δεδομένων της *λίστας*. Όπως θα δούμε και στο 8ο κεφάλαιο η χρήση της λίστας είναι ιδιαίτερα χρήσιμη. Όλες οι τιμές που εισάγονται σε μια λίστα, όπως οι βαθμοί των μαθητών του παραδείγματος, αποθηκεύονται προσωρινά στη λίστα και μπορούν να επαναχρησιμοποιηθούν στη συνέχεια κατά την εκτέλεση του προγράμματος και για πρόσθετους υπολογισμούς. Αντίθετα στην παραπάνω προτεινόμενη λύση οι προηγούμενοι βαθμοί χάνονται και δε μπορούμε να τους επαναχρησιμοποιήσουμε στη συνέχεια.

Σημείωση

Όσον αναφορά στη χρήση των έτοιμων ενσωματωμένων συναρτήσεων **min()** και **max()**, που μπορούν να χρησιμοποιηθούν για την άμεση εύρεση ελάχιστης και μέγιστης τιμής σε μία λίστα, πρέπει να τονιστεί ότι αποτελεί πρώτα στόχο του μαθήματος η κατανόηση της αλγοριθμικής λύσης του προβλήματος εύρεσης ελάχιστης και μέγιστης τιμής σε σχέση με την απευθείας εφαρμογή των έτοιμων συναρτήσεων που περιέχονται στις πλούσιες βιβλιοθήκες της **Python**. Σε κάθε περίπτωση, αυτό πρέπει να διευκρινίζεται στην εκφώνηση της άσκησης.

Θεωρητική ανασκόπηση. Δομή Επανάληψης while

Δομή Επανάληψης με while ή (όσο <συνθήκη> επανάλαβε)
Αρχική τιμή μεταβλητής
while **<συνθήκη>:**
Εντολή_1
Εντολή_2
....
Εντολή_κ

Σημείωση 1η: Θα πρέπει μέσα στο μπλοκ εντολών να υπάρχει κατάλληλη εντολή, ώστε να εξασφαλίζεται ότι κάποια στιγμή η συνθήκη θα γίνει ψευδής και θα διακοπεί ο βρόχος. Διαφορετικά ο βρόχος δε θα τερματίζει.

Σημείωση 2η: Πριν το βρόχο **while** θα πρέπει να δώσουμε μία αρχική τιμή στη μεταβλητή που ελέγχει τη συνθήκη του βρόχου, ώστε ανάλογα να εκτελεστεί ή όχι ο βρόχος.

Σημείωση 3η: Η δομή επανάληψης **while** χρησιμοποιείται κυρίως όταν δε γνωρίζουμε από την αρχή τον ακριβή αριθμό των επαναλήψεων που θα εκτελεστούν. Για το λόγο αυτό πρέπει να οριστεί μια συνθήκη, που όταν πάψει να ισχύει, να διακοπούν οι επαναλήψεις.

Σημείωση 4η: Η δομή επανάληψης **while** χρησιμοποιείται όταν θέλουμε να ελέγξουμε αν μία τιμή που εισάγεται από το πληκτρολόγιο είναι σύμφωνη με αυτό που θέλουμε. Για παράδειγμα αν η θερμοκρασία μιας πόλης στη νότια Ελλάδα είναι μεταξύ των -20 και των 50 βαθμών Κελσίου.

Λυμένο Παράδειγμα (Δομή επανάληψης **while**)

Να γραφεί, με χρήση του βρόχου **while**, πρόγραμμα σε γλώσσα Python, που να υπολογίζει το άθροισμα των αριθμών από το 1 ως το 100.

Πρόγραμμα σε **Python**

```
# Πρόγραμμα Αθροίζω
sum = 0                 # αρχική τιμή στο άθροισμα
i = 1                   #αρχική τιμή στη μεταβλητή ελέγχου
while i <= 100 :        # έλεγχος της επανάληψης
    sum= sum+ i
    i= i + 1            # αύξηση του μετρητή
print sum
```

Δραστηριότητα 15. Δομή επανάληψης με **while** (γινόμενο 200 ακεραίων αριθμών)

Κάντε τις απαραίτητες αλλαγές στο παραπάνω πρόγραμμα του λυμένου παραδείγματος, ώστε:

1) Να υπολογίζει το γινόμενο των αριθμών από το 1 ως το 200.
2) Να υπολογίζει το γινόμενο 10 ακεραίων αριθμών που θα τους διαβάζει από το πληκτρολόγιο.

Δραστηριότητα 16. Δομή επανάληψης με While. (Διερεύνηση προγράμματος). Δίνεται το παρακάτω τμήμα προγράμματος σε Python με αριθμημένες τις εντολές ανά γραμμή:

1.	x = 20
2.	s = 0
3.	while x < 100:
4.	x = x + 10
5.	s = s + x
6.	print x, s

Να απαντήσετε στα παρακάτω ερωτήματα:

1) Πόσες φορές θα εκτελεστεί η εντολή στη γραμμή 4;
2) Ποιες είναι όλες οι τιμές που θα πάρει η μεταβλητή x κατά την εκτέλεση του προγράμματος (μαζί με την αρχική);
3) Τι θα εμφανιστεί στην οθόνη στο τέλος του προγράμματος;

Δραστηριότητα 17. (Δομές επανάληψης **while / for**). Σωστό- Λάθος

Χαρακτηρίστε ως Σωστές ή Λανθασμένες τις παρακάτω προτάσεις, σημειώνοντας την ένδειξη Σ, αν η πρόταση είναι σωστή, ή Λ αν αυτή είναι λανθασμένη.

Προτάσεις	Σ/Λ
1) Η δομή **while** (Όσο) τερματίζει όταν η συνθήκη γίνει αληθής.	
2) Μια δομή επανάληψης **for** (Για) μπορεί να εκτελείται απεριόριστα.	
3) Η δομή **for** χρησιμοποιείται όταν ο αριθμός των επαναλήψεων δεν είναι προκαθορισμένος.	
4) Η δομή **while** χρησιμοποιείται όταν ο αριθμός επαναλήψεων είναι προκαθορισμένος.	
5) Η εντολές που περιλαμβάνονται μέσα στη δομή **while** θα εκτελεστούν τουλάχιστον μία φορά.	

Δραστηριότητα 18. Δομές επανάληψης **while/for**. Από τη δομή επανάληψης for στη δομή επανάληψης **while** και αντίστροφα.

Α. Γράψτε στη δεξιά στήλη του πίνακα και στα κενά κελιά, τμήμα προγράμματος σε **Python** ισοδύναμο με αυτό της αριστερής στήλης χρησιμοποιώντας τη δομή επανάληψης **while**, όπως στο πρώτο παράδειγμα της 1ης γραμμής.

for i in range (1,10):
  print i*i
	i=1
	while i<10:
	  print i*i
	  i=i+1

for i in range (10,51,2): print i*i	
for i in range (100,51,-2): print i*i	

Β. Γράψτε στην αριστερή στήλη τμήμα προγράμματος σε **Pyhon** ισοδύναμο με αυτό της δεξιάς στήλης χρησιμοποιώντας τη δομή επανάληψης **for**.

	z=2
	while z<10:
	  print z
	  z=z+4

	x = 1
	while x < = 10:
	  x = x + 2
	  print x

Δραστηριότητα 19. Δομές επανάληψης **While / For**. Δίνεται το παρακάτω τμήμα προγράμματος σε **Python**

```
x=60
while x>0:
    for i in range (2,7,2):
        x=x-10
    print x
```

1) Πόσες φορές θα εκτελεστεί η εντολή **x=x-10**;
2) Τι θα εμφανιστεί διαδοχικά στην οθόνη μετά την εκτέλεση του προγράμματος;

Δραστηριότητα 20.
Ο παρακάτω κώδικας σε **Python** σχεδιάζει μισό χριστουγεννιάτικο δέντρο:

```
lines = 1
maxlines = 12
while lines <= maxlines:
        print(lines*'*')
        lines +=1
```

```
*
**
***
****
*****
******
*******
********
*********
```

Αλλάξτε κατάλληλα τον κώδικα έτσι ώστε να σχεδιάζει και το άλλο μισό.

Θεωρητική ανασκόπηση. Δημιουργία - κλήση συνάρτησης

Συναρτήσεις
Ανάκληση γνώσεων από το Βιβλίο Μαθητή

Οι **συναρτήσεις** είναι επαναχρησιμοποιήσιμα μέρη προγραμμάτων. Μας επιτρέπουν να δίνουμε ένα όνομα σε ένα σύνολο εντολών και να το εκτελούμε καλώντας το όνομα αυτό, από οπουδήποτε στο πρόγραμμα και όσες φορές θέλουμε, διαδικασία που ονομάζεται κλήση **(calling)** της συνάρτησης. Δημιουργώντας δικές μας συναρτήσεις:

Για να ορίσουμε μια **δική μας συνάρτηση** χρησιμοποιούμε τη χαρακτηριστική δεσμευμένη λέξη def, ακολουθεί **ένα όνομα** που ταυτοποιεί την εκάστοτε συνάρτηση και **ένα ζευγάρι παρενθέσεων** που μπορούν να περικλείουν μια λίστα με ονόματα παραμέτρων, ενώ η γραμμή τελειώνει με άνω και κάτω τελεία **(:)**.

Δραστηριότητα 21. Δημιουργία και κλήση συναρτήσεων (Διερεύνηση κώδικα - Τι θα εμφανιστεί στην οθόνη)

Μελετήστε και σημειώστε χειρόγραφα το αποτέλεσμα κάθε προγράμματος. Στη συνέχεια καταχωρήστε τον κάθε κώδικα στο περιβάλλον της **Python** και συγκρίνετε το αποτέλεσμα με αυτό που σημειώσατε.

```
# Πρόγραμμα 1
print '\n'*1000 #clear screen
def print_1(t2):
    t1 = t2+10
    print t1
t1 = 5
print_1(t1)
```

```
# Πρόγραμμα 2
def print_1(t2):
    print t2
    t2 = t2+10
    print t2
t1 = 5
print_1(t1)
print t1
```

Δραστηριότητα 22. Δημιουργία συναρτήσεων (το πρώτο μου υποπρόγραμμα)

Να γράψετε το παρακάτω πρόγραμμα στο συντάκτη του **IDLE** της **Python** και αφού το αποθηκεύσετε σε ένα αρχείο να το εκτελέσετε.

```
sum = 0
for i in range (1,101):
    sum = sum + i
print sum
```

Στη συνέχεια επιλέξτε τον παραπάνω κώδικα και με Ctrl +] ή επιλέγοντας από το μενού του **IDLE Format → Indent region**, "σπρώξτε" τον κώδικα μια εσοχή μέσα.

Ονομάστε αυτό το τμήμα κώδικα με το όνομα **mysum** προσθέτοντας στην αρχή τη γραμμή def mysum( ):

```
def mysum( ) :
    sum = 0
    for i in range (1,101):
        sum = sum + i
    print sum
```

Μόλις ορίσατε το νέο σας υποπρόγραμμα. Με πλήκτρο **F5** το υποπρόγραμμα θα φορτωθεί στο διερμηνευτή και το **IDLE** θα σας μεταφέρει στο περιβάλλον του διερμηνευτή. Εκεί αν δώσετε την εντολή **mysum()** θα εκτελεστεί ο παραπάνω κώδικας.

Ουσιαστικά έχετε ονομάσει ένα τμήμα κώδικα με το όνομα **mysum** και αντί να ξαναγράφετε όλο τον κώδικα, απλά γράφετε το όνομα που του έχετε δώσει. Αυτό ήταν και το πρώτο σας υποπρόγραμμα.

Δραστηριότητα 23. Δημιουργία και κλήση συναρτήσεων (Μέγιστο τριών αριθμών)

Να γραφεί συνάρτηση σε **Python** που να υπολογίζει το μέγιστο μεταξύ τριών (3) αριθμών.

Υπόδειξη: Να δημιουργηθεί αρχικά μια συνάρτηση που να υπολογίζει το μεγαλύτερο μεταξύ δύο αριθμών με το όνομα **max_of_two**. Στη συνέχεια, να δημιουργηθεί μια νέα συνάρτηση που να βρίσκει το μεγαλύτερο μεταξύ τριών αριθμών με το όνομα **max_of_three**, χρησιμοποιώντας κατάλληλα τη συνάρτηση **max_of_two**.

Βοήθεια - Ενδεικτική λύση

```
#συνάρτηση εύρεση μέγιστου τριών αριθμών
def max_of_two( x, y ):
    if x > y:
        return x
    return y
def max_of_three ( x, y, z ):
    return max_of_two( x, max_of_two( y, z ) )
print(max_of_three(34, 8, -11)) #παράδειγμα για έλεγχο
```

Δραστηριότητα 24. Δημιουργία, κλήση και χαρακτηριστικά συναρτήσεων (ερωτήσεις Σωστού-Λάθους)

Συμπληρώστε τη δεύτερη στήλη του πίνακα με Σωστό (Σ) ή Λάθος (Λ).

Περιγραφή	Σ/Λ
Ορίζουμε τη συνάρτηση **Metatropi** ως εξής: **definition metatropi**	
Ορίζουμε επιτυχώς μια συνάρτηση: **1) def ektyposi (x1, x2)** **2) def ektyposi (x1 + x2)** **3) def ektyposi (x1; x2)**	1 ☐ 2 ☐ 3 ☐
Ορίζουμε επιτυχώς μια συνάρτηση: **1) def ektyposi (x1; x2):** **2) def ektyposi x1, x2** **3) def ektyposi (x1, x1):**	1 ☐ 2 ☐ 3 ☐
Η κλήση μιας συνάρτησης γίνεται με: **1) call** όνομα_συναρτησης() **2)** όνομα_συναρτησης () **3) run** όνομα_συναρτησης () **4)** Όλα τα παραπάνω	1 ☐ 2 ☐ 3 ☐ 4 ☐

Γενικές Δραστηριότητες - Ασκήσεις

Στο τετράδιό σας και στη συνέχεια στο προγραμματιστικό περιβάλλον της **Python-IDLE**, να επιλύσετε τα παρακάτω προβλήματα-δραστηριότητες.

Δραστηριότητα 25.

Να γράψετε πρόγραμμα σε **Python** που θα:

1) Διαβάζει ένα ποσό, σε ευρώ.
2) Να μετατρέπει το ποσό που διάβασε σε Λίρες Αγγλίας (δεχόμαστε ότι 1€ = 0,85 Λίρες).
3) Να εμφανίζει το ποσό σε ευρώ και Λίρες Αγγλίας.

Επέκταση: Αφού διαβάσει το ποσό, να διαβάζει και την ισοτιμία του ενός ευρώ έναντι της Λίρας Αγγλίας και στη συνέχεια με βάση την ισοτιμία αυτή να κάνει τη μετατροπή.

Δραστηριότητα 26.

Να γράψετε πρόγραμμα σε **Python** που να διαβάζει την ηλικία ενός προσώπου. Στη συνέχεια, αν είναι κάτω των 18 ετών να εμφανίζει τη λέξη "ΑΝΗΛΙΚΟΣ", αν είναι 18 και άνω να εμφανίζει "ΕΝΗΛΙΚΟΣ" και τέλος, αν είναι άνω των 70 ετών να εμφανίζει τη λέξη "ΗΛΙΚΙΩΜΕΝΟΣ".

Δραστηριότητα 27.

Η μισθοδοσία υπαλλήλου προβλέπει επίδομα τέκνων, με βάση τον παρακάτω πίνακα:

Αριθμός παιδιών	Ποσό επιδόματος μισθοδοσίας
0 έως και 2	0 ευρώ
3 (τρίτεκνη)	100 ευρώ συνολικά
άνω των 3 (πολύτεκνη)	Το αρχικό επίδομα των τριών παιδιών (100€) συν 20 € για κάθε παραπάνω παιδί άνω των τριών.

Να γράψετε πρόγραμμα σε γλώσσα **Python** το οποίο για μία οικογένεια:

1) Να διαβάζει τον αριθμό των παιδιών ενός (μόνο) υπαλλήλου.
2) Να εμφανίζει το μήνυμα τρίτεκνη ή πολύτεκνη οικογένεια στην αντίστοιχη περίπτωση.
3) Να υπολογίζει και να εμφανίζει το ποσό του επιδόματος που αναλογεί στον υπάλληλο.

Δραστηριότητα 28.
Ένας σκληρός δίσκος έχει χωρητικότητα **500 MB** για αποθήκευση αρχείων. Ο κάτοχός του τον γεμίζει με αρχεία. Θεωρώντας ότι το αποθηκευτικό μέσο είναι αρχικά άδειο, να γράψετε πρόγραμμα σε **Python** που θα διαβάζει το μέγεθος κάθε αρχείου σε **MB**, μέχρι το συνολικό μέγεθος να ξεπεράσει τη χωρητικότητά αυτή. Στη συνέχεια θα εμφανίζει το συνολικό πλήθος των αρχείων που έχουν αποθηκευθεί στο δίσκο.

Δραστηριότητα 29.
Ένα ασανσέρ έχει μέγιστο όριο ασφάλειας τα **500** κιλά. Να γράψετε πρόγραμμα σε **Python** που θα διαβάζει το βάρος και τη σειρά με την οποία κάθε άτομο εισέρχεται στο ασανσέρ (π.χ. 45 1, 89 2). Το πρόγραμμα θα τερματίζει όταν το ασανσέρ γεμίσει (σε σχέση με το μέγιστο επιτρεπτό όριο ασφαλείας). Στη συνέχεια θα εμφανίζει τη σειρά του τελευταίου ατόμου, που κατάφερε να μπει στο ασανσέρ.

Δραστηριότητα 30.
Σε ένα πλοίο υπάρχουν εισιτήρια Α' Θέσης (κωδικός 0) προς **50€** και Β' θέσης (κωδικός 1) προς **20€**, το ένα. Ο μέγιστος επιτρεπόμενος αριθμός επιβατών είναι **400** άτομα και θεωρούμε ότι τελικά το πλοίο γέμισε για το συγκεκριμένο προορισμό που εξετάζουμε.

Να γράψετε πρόγραμμα σε **Python**, το οποίο:

1) Να διαβάζει την κατηγορία εισιτηρίου (κωδικός **0** ή **1**) για κάθε επιβάτη.
2) Να εμφανίζει το πλήθος των επιβατών της Α' θέσης.
3) Να εμφανίζει το συνολικό ποσό που πλήρωσαν όλοι οι επιβάτες.

Δραστηριότητα 31.
Να συμπληρώσετε τα κενά στο παρακάτω πρόγραμμα ώστε:

1) Να εμφανίζει όλους τους αριθμούς από **1** μέχρι **100**.
2) Να εμφανίζει όλους τους αριθμούς από **1** μέχρι **100**, αλλά με αντίστροφη σειρά.
3) Να εμφανίζει όλους τους άρτιους αριθμούς από **20** μέχρι **80**.

```
for number in range( ____, _____, ____ ) :
    print number
```

Δραστηριότητα 32.

Να γραφεί πρόγραμμα σε **Python** το οποίο:

1) Να διαβάζει επαναληπτικά ακέραιους αριθμούς μέχρις ότου δοθεί ο αριθμός 0.
2) Να εμφανίζει στο τέλος, το πλήθος των θετικών αριθμών.
3) Να υπολογίζει και να εμφανίζει στο τέλος, το άθροισμα όλων των αριθμών που διαβάστηκαν (δόθηκαν από το πληκτρολόγιο).

Δραστηριότητα 33.

Να γράψετε πρόγραμμα σε **Python** με το οποίο: Να καταχωρούνται επαναληπτικά, ο αριθμός κυκλοφορίας οχήματος και ποσό κλήσης από παρκάρισμα ή άλλη αιτία, με την καταχώρηση να επαναλαμβάνεται μέχρι να δοθεί αριθμός κυκλοφορίας 99.

Στο τέλος να εμφανίζει:

1. Το πλήθος των οχημάτων που καταχωρήθηκαν.
2. Το συνολικό ποσό κλήσεων που θα εισπραχθεί.
3. Τον αριθμό κυκλοφορίας του τελευταίου από τα οχήματα που έλαβαν το μέγιστο ποσό κλήσης, καθώς και το ποσό της κλήσης αυτής.

Μέρος ΙΙ

ΚΕΦΑΛΑΙΟ 5 Κλασικοί Αλγόριθμοι II

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό αναπτύσσονται ορισμένοι από τους βασικούς αλγορίθμους της Πληροφορικής Επιστήμης. Η έννοια του αλγορίθμου αποτελεί τον ακρογωνιαίο λίθο της επιστήμης της Πληροφορικής, μια και ένα από τα βασικότερα προβλήματα που εξετάζει, είναι ο σχεδιασμός αλγορίθμων για την επίλυση προβλημάτων. Αυτό όμως δε σημαίνει ότι, αν επινοήσουμε έναν οποιονδήποτε αλγόριθμο που να λύνει το πρόβλημα που θέλουμε, αυτό είναι αρκετό. Το βασικό, αλλά και αιώνιο ερώτημα της Πληροφορικής, είναι "Μπορούμε καλύτερα;" Μπορούμε να σχεδιάσουμε έναν αλγόριθμο ο οποίος να κάνει καλύτερη διαχείριση των υπολογιστικών πόρων του συστήματος (μνήμης και επεξεργαστή); Σήμερα το βασικό πρόβλημα δεν είναι τόσο η χρήση της κύριας μνήμης, αλλά ο χρόνος εκτέλεσης του αλγορίθμου. Ο στόχος μας λοιπόν, δεν είναι να λύσουμε απλώς ένα πρόβλημα, αλλά να το λύσουμε με το βέλτιστο τρόπο. Πολλές φορές δεν υπάρχει ένας αλγόριθμος που να είναι βέλτιστος για όλες τις περιπτώσεις δεδομένων, αλλά εξαρτάται από την οργάνωση των δεδομένων. Για παράδειγμα η *σειριακή αναζήτηση* σαρώνει όλα τα στοιχεία μιας ταξινομημένης λίστας χωρίς να εκμεταλλεύεται τη διάταξη των στοιχείων. Ο σχεδιασμός ενός νέου αλγορίθμου που θα εκμεταλλεύεται την ιδιαίτερη δομή των δεδομένων, για να βρίσκει το ζητούμενο στοιχείο πιο γρήγορα, αποτελεί ένα από τα αντικείμενα αυτής της ενότητας.

Επειδή υπάρχουν διάφορες κατηγορίες δεδομένων, οι επιστήμονες της Πληροφορικής σχεδιάζουν διάφορους αλγορίθμους ανάλογα με την ιδιαίτερη οργάνωση των δεδομένων. Γι' αυτόν το λόγο δεν έχουμε μόνο έναν αλγόριθμο ταξινόμησης, αλλά αρκετούς, κάποιους από τους οποίους θα μελετήσουμε σε αυτήν την ενότητα.

Διδακτικοί στόχοι

Στόχοι

Μετά τη μελέτη του κεφαλαίου, θα μπορούμε να:

- αναλύουμε και να εφαρμόζουμε κατάλληλα κλασικούς αλγορίθμους για προβλήματα ταξινόμησης και αναζήτησης

- υλοποιούμε σε μια γλώσσα προγραμματισμού κλασικούς αλγορίθμους ταξινόμησης και αναζήτησης
- συγκρίνουμε και να επιλέγουμε τον κατάλληλο αλγόριθμο ανάλογα με το είδος του προβλήματος.

Λέξεις κλειδιά

Αλγόριθμοι, ταξινόμηση, αναζήτηση.

Διδακτικές Ενότητες

5. Κλασικοί Αλγόριθμοι ΙΙ
   5.1 Δυαδική αναζήτηση
   5.2 Ταξινόμηση Ευθείας ανταλλαγής
   5.3 Ταξινόμηση με Εισαγωγή
   5.4 Δραστηριότητες - Άλυτες
   Ερωτήσεις - Ασκήσεις

Σύνοψη

Μια μέθοδος για την αναζήτηση ενός στοιχείου σε μια λίστα, είναι η σειριακή αναζήτηση. Στην περίπτωση που η λίστα είναι ταξινομημένη, χρησιμοποιείται η δυαδική αναζήτηση η οποία εκμεταλλεύεται τη διάταξη των στοιχείων, για να βρει το ζητούμενο στοιχείο πιο γρήγορα. Για να χρησιμοποιηθεί η δυαδική αναζήτηση σε μη ταξινομημένα δεδομένα, πρέπει προφανώς πρώτα αυτά να ταξινομηθούν.

Σε αυτό το κεφάλαιο παρουσιάζονται δυο αλγόριθμοι ταξινόμησης. Ο αλγόριθμος της ευθείας ανταλλαγής, που βασίζεται στη σύγκριση γειτονικών ζευγών της λίστας και ο αλγόριθμος ταξινόμησης, που με εισαγωγή ταξινομεί μια λίστα βαθμιαία, εισάγοντας κάθε φορά το επόμενο στοιχείο, έτσι ώστε η λίστα να παραμένει ταξινομημένη.

Σημείωση: Το κεφάλαιο αυτό είναι μερικά εντός διδακτέας και εξεταστέας ύλης.

Δραστηριότητες - Ασκήσεις

Δραστηριότητα 1. Μελέτη περίπτωσης ενδεικτικής λύσης

Να γράψετε αλγόριθμο σε **Python** ο οποίος να διαβάζει για κάθε μαθητή το όνομα και το βαθμό του και να τα καταχωρεί σε δύο λίστες **student** και **grade**. Η εισαγωγή των δεδομένων σταματάει, όταν δοθεί αντί για όνομα η τιμή 0. Στη συνέχεια:

1) Να εμφανίζει τα ονόματα και τους βαθμούς των μαθητών σε αλφαβητική σειρά, κάνοντας κατάλληλη ταξινόμηση με βάση τη λίστα **student**.
2) Να διαβάζει το όνομα ενός μαθητή και να εμφανίζει τον αντίστοιχο βαθμό του, αφού προηγουμένως έχει βρει τη θέση του στη λίστα, αξιοποιώντας το γεγονός ότι τα ονόματα είναι σε αλφαβητική σειρά.
3) Να ταξινομεί τους μαθητές με βάση τους βαθμούς τους και να εμφανίζει τα ονόματα των τριών πρώτων μαθητών. Να χρησιμοποιήσετε τον αλγόριθμο ταξινόμησης που ήδη έχετε υλοποιήσει ως συνάρτηση.

Ενδεικτική Λύση

```
# Ταξινόμηση των λιστών A, B σε αύξουσα σειρά
# με βάση τη λίστα A
def bubbleSort( A, B ):
    N = len( A )
    for i in range( N ):
        for j in range(N-1, i, -1):
            if A[ j ] < A[ j-1 ] :
                A[ j ], A[ j-1 ] = A[ j-1 ], A[ j ]
                    B[ j ], B[ j-1 ] = B[ j-1 ], B[ j ]

# Δυαδική Αναζήτηση του στοιχείου key στη λίστα A.
# Η συνάρτηση επιστρέφει τη θέση του στοιχείου, αλλιώς
# αν δεν το βρει επιστρέφει τον αριθμό -1 .
```

```
def binarySearch( A, key ):
    last = len( A ) - 1
    first = 0
    pos = -1
    while first <= last and pos == -1 :
        mid = (last + first) // 2
        if A[ mid ]==key :
            pos = mid
        elif A[ mid ] < key :
            first = mid + 1
        else:
        last = mid - 1
    return pos

# Ακολουθεί το κύριο πρόγραμμα όπου χρησιμοποιούνται
# οι παραπάνω συναρτήσεις
#αρχικοποίηση
    index = 0
    student = [ ]
    grade = [ ]
#εισαγωγή ονόματος μαθητή
    print " Για να τερματιστεί η εισαγωγή δώσε 0 ως όνομα"
    name = raw_input("student[" + str(index) + "] = ")
    while name != ”0” :
        valGrade = input("grade[" + str(index) + "] = ")
        student.append(name)
        grade.append(valGrade)
        index += 1
        name = raw_input("student[" + str(index) + "] = ")
```

```
    bubbleSort( student, grade ) #κλήση συνάρτησης ταξινόμησης
#εμφάνιση ονομάτων μαθητών και βαθμών
   for index in range( len( student ) ):
      print student[index], grade[index]

   name = raw_input(" name = ")
   pos = binarySearch( student, name )
   #κλήση της συνάρτησης δυαδική αναζήτηση
   if pos == -1 :
      print " Το όνομα που έδωσες δεν υπάρχει στη λίστα "
   else :
      print grade[ pos ]

# Τώρα γίνεται ταξινόμηση ως προς τους βαθμούς,
# αλλά επειδή είναι σε αύξουσα
# οι τρεις μεγαλύτεροι είναι οι τρεις τελευταίοι
   bubbleSort( grade, student )
   print student[-1], student[-2], student[-3]
```

Παρατηρήσεις - Επεξηγήσεις

- Για την αναζήτηση θα χρησιμοποιήσουμε τον αλγόριθμο της δυαδικής αναζήτησης, αφού μας ζητείται να αξιοποιήσουμε την αλφαβητική σειρά των ονομάτων των μαθητών. Υπενθυμίζεται ότι αν τα ονόματα δεν ήταν σε αλφαβητική σειρά δεν θα μπορούσε να χρησιμοποιηθεί η δυαδική αναζήτηση, αλλά θα είμασταν υποχρεωμένοι να χρησιμοποιήσουμε τη σειριακή αναζήτηση που μάθαμε στην Β' τάξη του Λυκείου.
- Η συνάρτηση **str** χρησιμοποιείται για να μετατρέψει έναν αριθμό σε συμβολοσειρά ώστε να έχει νόημα η συνένωση συμβολοσειρών με τον τελεστή **'+'**
- Η συνάρτηση **len** επιστρέφει το πλήθος των στοιχείων μιας λίστας, όπως είχαμε δει στην Β' Λυκείου
- Μπορούμε να χρησιμοποιούμε τον τελεστή **//** της **ακέραιας (ευκλείδειας) διαίρεσης**.

- Προσθέτουμε κάθε όνομα μαθητή που διαβάζουμε στο τέλος της λίστας **student** με τη μέθοδο **append**. Το ίδιο κάνουμε για να προσθέσουμε τους αντίστοιχους βαθμούς κάθε μαθητή στη λίστα **grade**.
- Οι δυο λίστες είναι "παράλληλες", δηλαδή ο μαθητής με όνομα **student[2]** έχει βαθμό **grade[2]**, οπότε αν χρειαστεί να ταξινομήσουμε με βάση τους βαθμούς θα πρέπει να αντιμεταθέσουμε και τα ονόματα για να μην χαλάσει η αντιστοίχιση μεταξύ τους.

Δραστηριότητα 2. Ταξινόμηση ευθείας ανταλλαγής

Δίνεται η παρακάτω λίστα Α με 7 αριθμούς. Να εκτελέσετε τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής για την ταξινόμηση των αριθμών σε αύξουσα σειρά, συμπληρώνοντας παράλληλα τα κενά στον παρακάτω πίνακα, έτσι ώστε να φαίνονται τα στοιχεία της λίστας αμέσως μετά από κάθε πέρασμα του αλγορίθμου.

	A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
	55	34	5	3	2	1	1
1ο πέρασμα							
2ο πέρασμα							
3ο πέρασμα							
4ο πέρασμα							
5ο πέρασμα							
6ο πέρασμα							

Δραστηριότητα 3. Διερεύνηση του αλγορίθμου ταξινόμησης

Να εκτελέσετε την παρακάτω συνάρτηση συμπληρώνοντας τα κενά μέχρι να καταλήξετε στις κατάλληλες τιμές, ώστε να γίνεται η ταξινόμηση σωστά για τη λίστα A = [0, 1, 2, 3, 4, 5]

```
def bubbleSort( A ):
    N = len( A )
    for i in range( ____, ____ ):
        print " i = " , i
        print " j = " ,      # στην ίδια γραμμή
        for j in range(____,____, -1):
            print " j = " , j, j–1, ", " , # στην ίδια γραμμή
            if A[ j ] < A[ j-1 ] :
                A[ j ], A[ j-1 ] = A[ j-1 ], A[ j ]
        print
```

Δραστηριότητα 4.
Να γράψετε ένα πρόγραμμα σε **Python** το οποίο θα διαβάζει ονόματα μέχρι να δοθεί ως όνομα το '0' και θα τα εμφανίζει σε αλφαβητική σειρά.

Επέκταση
Στη συνέχεια το πρόγραμμά σας θα διαβάζει ένα όνομα και θα εμφανίζει κατάλληλο μήνυμα, αν το όνομα αυτό εμφανίζεται στη λίστα με τα ονόματα ή όχι.

Δραστηριότητα 5.
Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει βαθμούς μαθητών μέχρι να δοθεί αρνητικός και στη συνέχεια θα εμφανίζει τους τρεις μεγαλύτερους βαθμούς που διάβασε.

Δραστηριότητα 6. Αλγόριθμος Δυαδικής αναζήτησης

Δίνεται παρακάτω η λίστα Α με 14 αριθμούς. Να εκτελέσετε τον αλγόριθμο της Δυαδικής αναζήτησης για τον αριθμό 100 και να γράψετε στο τετράδιό σας τους αριθμούς που θα συγκριθούν με το 100 κατά την εκτέλεση του αλγορίθμου.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
A	1	1	2	3	5	8	13	21	34	55	89	94	96	99

Πόσες συγκρίσεις χρειάστηκαν μέχρι να διαπιστώσει ο αλγόριθμος ότι το 100 δεν υπάρχει στον πίνακα;

Δραστηριότητα 7. Αλγόριθμος Δυαδικής αναζήτησης

Να συμπληρώσετε τα κενά στο παρακάτω πρόγραμμα σε **Python**, έτσι ώστε να εκτελεί τον αλγόριθμο της Δυαδικής αναζήτησης ενός στοιχείου **key** σε μια λίστα **array**.

```
def binarySearch( array, key ) :
first = _______________
last = _______________
found = _______________
while _______________  and ______ found :
mid = ____________________________
if array[ mid ] == key :
    _______________
elif _______________ :
    first = _______________
else :
    last = _______________
return found
```

Δραστηριότητα 8.
Να γράψετε ένα πρόγραμμα το οποίο να διαβάζει τους βαθμούς και τα επίθετα των μαθητών της τάξης σας και στη συνέχεια να τα εμφανίζει σε φθίνουσα σειρά ως προς τους βαθμούς, δηλαδή όσοι έχουν καλύτερους βαθμούς να είναι πρώτοι.

Δραστηριότητα 9. Αλγόριθμος ευθείας ανταλλαγής
Να συμπληρώσετε τα κενά στο παρακάτω πρόγραμμα, έτσι ώστε να ταξινομεί τα στοιχεία της λίστας **array** σε φθίνουσα σειρά, σύμφωνα με τον αλγόριθμο της ευθείας ανταλλαγής.

```
for i in range( ____, ____, ____ ):
    for j in range( _____, _____, _____ ):
        if array[ _______ ] > array[ _______ ] :
            array [ j ] , array [ j+1 ] = array[ j+1 ] , array[ j ]
```

Να χρησιμοποιήσετε μια συνάρτηση αντίστοιχη με αυτή της δραστηριότητας 2 για περαιτέρω διερεύνηση.

Δραστηριότητα 10. Αλγόριθμος ευθείας ανταλλαγής
Πόσα περάσματα θα χρειαστούν για να ταξινομηθούν οι παρακάτω λίστες με τον αλγόριθμο ταξινόμησης ευθείας ανταλλαγής σε αύξουσα σειρά;

	0	1	2	3	4	5	6	7	8	9	10
α)	55	46	44	34	28	18	14	12	10	8	4
β)	4	6	8	18	30	50	56	68	70	3	1
γ)	28	28	28	28	28	28	28	30	30	30	30
δ)	28	28	28	6	6	6	6	6	6	6	6

ΚΕΦΑΛΑΙΟ 6 Διαχείριση Αρχείων

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό θα αναπτυχθούν οι τρόποι δημιουργίας και χειρισμού αρχείων. Επίσης, θα παρουσιαστούν οι βασικές λειτουργίες, όπως το άνοιγμα, κλείσιμο ενός αρχείου, το διάβασμα και γράψιμο μιας εγγραφής, μέσω της γλώσσας προγραμματισμού **Python**.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- δημιουργούμε αρχεία
- χειριζόμαστε απλά αρχεία, όπως για παράδειγμα αρχεία κειμένου, μέσω βασικών εντολών **(open, close, read, write)** που χρησιμοποιεί η γλώσσα προγραμματισμού **Python**
- εκτελούμε πρόσθετες λειτουργίες σε αρχεία, όπως η αναζήτηση.

Λέξεις κλειδιά

Αρχείο κειμένου, άνοιγμα αρχείου, ανάγνωση αρχείου, εγγραφή σε αρχείο, κλείσιμο αρχείου, διαδρομή.

Διδακτικές Ενότητες

6. Διαχείριση Αρχείων
- **6.1** Εισαγωγή - δημιουργία, άνοιγμα, κλείσιμο αρχείων
- **6.2** Ανάγνωση και εγγραφή σε αρχείο
- **6.3** Πρόσθετες λειτουργίες σε αρχεία

Ερωτήσεις - Ασκήσεις

Σύνοψη

Στο κεφάλαιο αυτό ασχοληθήκαμε με τη διαχείριση αρχείων δεδομένων, κυρίως από τη σκοπιά της γλώσσας προγραμματισμού **Python**. Επίσης, με θέματα ταυτοποίησής τους από το Λειτουργικό Σύστημα, όπως το όνομα, η θέση στο αποθηκευτικό μέσο και βασικές λειτουργίες σε αυτά.

Σημείωση: Το κεφάλαιο αυτό είναι μερικά εντός διδακτέας και εξεταστέας ύλης.

Δραστηριότητες - Ασκήσεις

Δραστηριότητα 1.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο:

Θα δημιουργεί μια λίστα με όλους τους άρτιους αριθμούς από το 2 μέχρι το 1000 και θα γράφει αυτούς τους αριθμούς σε ένα αρχείο έναν σε κάθε γραμμή.
Στη συνέχεια θα διαβάζει τους αριθμούς από το αρχείο και θα γράφει το άθροισμά τους στο τέλος του αρχείου χωρίς να διαγράψει τα περιεχόμενα του αρχείου.

Λύση

```
# δημιουργία μιας λίστας με ζυγούς αριθμούς
# και αποθήκευση των αριθμών στο αρχείο
even_list = range(2,1001,2)
myfile = open('even_list.out', 'w')
for number in even_list:
    myfile.write( str(number) + "\n")
myfile.close()

# άνοιγμα του αρχείου για ανάγνωση
myfile = open('even_list.out', 'r')
S = 0
for line in myfile:
    S = S + int(line)
myfile.close()

# άνοιγμα του αρχείου για προσθήκη στο τέλος
myfile = open('even_list.out', 'a')
myfile.write( str(S) + "\n" )
myfile.close()
```

Δραστηριότητα 2.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο θα διαβάζει ένα κείμενο από ένα αρχείο και θα εμφανίζει τις λέξεις κάθε πρότασης σε αντίστροφη σειρά, δηλαδή την πρώτη λέξη τελευταία και την τελευταία πρώτη.

Δεχόμαστε ότι κάθε πρόταση τελειώνει με τελεία '.' και οι λέξεις είναι χωρισμένες με ένα κενό. Δεν υπάρχουν άλλα σημεία στίξεως.

Δραστηριότητα 3.

Να γράψετε ένα πρόγραμμα το οποίο θα ενώνει δυο αρχεία κειμένου σε ένα τοποθετώντας τα περιεχόμενα του δεύτερου αρχείου μετά από αυτά του πρώτου.

Δραστηριότητα 4.

Να υλοποιήσετε μια συνάρτηση **copy(source, destination)** η οποία θα δημιουργεί ένα αντίγραφο του αρχείου με όνομα **source** στο αρχείο με όνομα **destination**.

ΚΕΦΑΛΑΙΟ 7 Προηγμένα στοιχεία γλώσσας προγραμματισμού

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό θα αναπτυχθούν προηγμένα χαρακτηριστικά της γλώσσας προγραμματισμού Python. Συγκεκριμένα θα αναπτυχθούν οι ιδιότητες των υποπρογραμμάτων, η εμβέλεια μεταβλητών, τα αρθρώματα και τα πακέτα.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- αναφέρουμε τα χαρακτηριστικά των υποπρογραμμάτων και τον τρόπο κλήσης τους, με την προσαρμογή τους στη γλώσσα προγραμματισμού
- εντοπίζουμε και απομονώνουμε κατάλληλα τμήματα κώδικα από έτοιμο πρόγραμμα για μετατροπή σε υποπρογράμματα
- σχεδιάζουμε τη λύση προγραμματιστικού προβλήματος με αρθρωτό τρόπο, όπου αυτό είναι σκόπιμο
- αναγνωρίζουμε την εμβέλεια μεταβλητών και παραμέτρων σε σύνθετα προγράμματα.

Λέξεις κλειδιά

Συνάρτηση, παράμετρος, εμβέλεια, άρθρωμα, τοπική μεταβλητή.

Διδακτικές Ενότητες

Σύνοψη

Όπως οι συναρτήσεις είναι επαναχρησιμοποιούμενα μέρη προγραμμάτων, τα αρθρώματα είναι επαναχρησιμοποιούμενα προγράμματα. Τα πακέτα είναι μια άλλη ιεραρχία, για να οργανώνουμε αρθρώματα. Η πρότυπη βιβλιοθήκη που συνοδεύει την **Python** είναι ένα παράδειγμα τέτοιων πακέτων και αρθρωμάτων.

Ένα άρθρωμα **(module)** είναι μια συλλογή σχετικών συναρτήσεων και αποθηκεύεται σε αρχείο με κατάληξη **.py**. Μπορούμε, να γράψουμε και τα δικά μας αρθρώματα, αποθηκεύοντας τις συναρτήσεις μας σε ένα αρχείο **.py**. Για να χρησιμοποιήσουμε ένα άρθρωμα σε ένα πρόγραμμα, θα πρέπει να το εισάγουμε με την εντολή **import**.

Εκτός από τις ενσωματωμένες βιβλιοθήκες (μονάδες) συναρτήσεων που περιλαμβάνονται στη γλώσσα **Python**, μπορούμε να βρούμε στους δικτυακούς τόπους υποστήριξης της γλώσσας και εξωτερικές μονάδες λογισμικού με πληθώρα επιπλέον συναρτήσεων για τη δημιουργία ισχυρών προγραμμάτων.

Ένα από τα καθοριστικά πλεονεκτήματα της γλώσσας **Python**, είναι ότι υποστηρίζεται από μια παγκόσμια κοινότητα προγραμματιστών που συνεισφέρουν με τη δημιουργία πολλών εξωτερικών βιβλιοθηκών. Τις βιβλιοθήκες αυτές μπορούμε να χρησιμοποιήσουμε, για να δημιουργήσουμε γρήγορα εντυπωσιακές και φιλικές προς το χρήστη εφαρμογές. Τις συλλογές αυτές μπορούμε να τις θεωρήσουμε ως πρόσθετες εργαλειοθήκες με έτοιμα εργαλεία, τα οποία μπορούμε να χρησιμοποιήσουμε μέσα στον κώδικά μας, αφού οι περισσότερες εργαλειοθήκες της **Python** ανήκουν στην κατηγορία του ελεύθερου λογισμικού.

Δραστηριότητες - Ασκήσεις

Δραστηριότητα 1.

Ανοίγουμε ένα νέο αρχείο στην **Python** στο οποίο θα προσθέσουμε τους ορισμούς των συναρτήσεων που θα δώσουμε παρακάτω.

Ορίζουμε τη συνάρτηση **python3** η οποία εμφανίζει τη λέξη **python 3** φορές. Επίσης ορίζουμε και τη συνάρτηση **python9** που εμφανίζει τη λέξη **python 9** φορές.

```
def python3():
    print "python"
    print "python"
    print "python"
>>> python3()
python
python
python
```

```
def python9():
    print "python"
    print "python"
    print "python"
    print "python"
    print"python"
    print"python"
    print "python"
    print "python"
    print "python"
```

Να ξαναγράψετε τη συνάρτηση **python9** χρησιμοποιώντας λιγότερες εντολές.

Να ορίσετε μια συνάρτηση η οποία να εμφανίζει 21 φορές τη λέξη **python** με αποκλειστική χρήση των δυο παραπάνω συναρτήσεων, χρησιμοποιώντας όσο το δυνατόν λιγότερες εντολές.

Δραστηριότητα 2.

Δίνονται οι παρακάτω συναρτήσεις σε **Python**:

```
def python3():
    for i in range(3):
        print "python"
def python100():
    for i in range(100):
        print "python"
```

```
def python12():
    for i in range(12):
        print "python"
```

Συμπληρώστε τον ορισμό της παρακάτω συνάρτησης, ώστε να αποτελεί γενίκευση των προηγούμενων και στη συνέχεια δώστε τις διπλανές κλήσεις στο διερμηνευτή:

```
def python( _____ ):
    for i in range( _____ ):
        print "python"
```

```
>>> python(3)
>>> python(9)
>>> python(21)
```

Σε τι διαφέρει η τελευταία συνάρτηση από όλες τις προηγούμενες; Ποια είναι η σχέση της με αυτές;

Δραστηριότητα 3.

Δώστε τις παρακάτω εντολές στο διερμηνευτή της **Python**. Τι παρατηρείτε; Τι πιστεύετε ότι κάνει η εντολή **return**;

```
>>> import math
>>> def root(number):
        return math.sqrt(number)
>>> root( 16 ) ; root( 2 )
```

```
>>> a = root( 16 )
>>> print a
>>> print root(a)
>>> print root(root(16))
```

Δραστηριότητα 4.

Να εντοπίσετε στο παρακάτω πρόγραμμα τα τμήματα κώδικα που πρέπει να γίνουν συναρτήσεις και να το ξαναγράψετε, έτσι ώστε να αποφευχθεί η επανάληψή τους και να μειωθεί ο όγκος του.

```
sum1 = 0
for i in range(100):
    sum1 = sum1 + i
print sum1
sum2 = 0
for j in range(100):
    sum2 = sum2 + j
print sum2 + sum1
sum3 = 0
for k in range(sum1):
    sum3 = sum3 + k
print sum3 + sum2 + sum1
```

Δραστηριότητα 5.

Να εκτελέσετε τα παρακάτω τμήματα κώδικα και εξηγήστε τα αποτελέσματα. Ήταν αυτά που αναμένατε; Ποια ερμηνεία δίνετε; Αν διακρίνετε κάποιο πρόβλημα τι μπορείτε να κάνετε για να το διορθώσετε;

```
>>> import math
>>> def donothing(value):
        local = value+4
        print local

>>> donothing( 496 )
>>> print(local)
```

```
>>> myglobal = 496
>>> def foo( value ):
        myglobal = value + 2
        print(myglobal)

>>> foo( 8128 )
>>> print(myglobal)
```

Δραστηριότητα 6.

Να ορίσετε μια συνάρτηση με όνομα count η οποία δέχεται δύο ορίσματα (**sequence** και **item**) και να επιστρέφει πόσες φορές εμφανίζεται το **item** στη λίστα **sequence**.

Βοήθεια
Η **count([1,2,1,1], 1)** πρέπει να επιστρέφει 3

```
def count(sequence, item):
    counter = 0
    for i in sequence:
        if i == item:
            counter += 1
    return counter
```

Δραστηριότητα 7.

Να κάνετε τα παρακάτω βήματα:

1) Να ορίσετε συνάρτηση με όνομα **purify**, η οποία δέχεται μια λίστα αριθμών, απομακρύνει όλους τους περιττούς από τη λίστα και επιστρέφει το αποτέλεσμα. Για παράδειγμα η **purify([1,2,3])** επιστρέφει 2.
2) Να αναζητήσετε πληροφορίες για τη λειτουργία των μεθόδων **pop** και **remove** των λιστών.

Βοήθεια

1ος τρόπος: Δημιουργούμε μια νέα λίστα στην οποία προσθέτουμε τους άρτιους αριθμούς

```
def purifybyConstruction( List ) :
    i = 0
    evenList = [ ]
    while i < len(List) :
        if List[ i ] % 2 == 0 :
            evenList.append(List[ i ])
        i = i + 1
    return evenList
```

2ος τρόπος: Αφαιρούμε από τη λίστα τους περιττούς. Γιατί όταν αφαιρούμε έναν αριθμό ο δείκτης i δεν αυξάνεται;

```
def purifybyDestruction( List ) :
    i = 0
    while i < len(List) :
        if List[ i ] % 2 == 1 :
            List.pop( i )
        else:
            i = i + 1
    return List
```

3ος τρόπος: Είναι σωστή η παρακάτω λύση; Εκτελέστε τη στο περιβάλλον **Python IDLE** για τη λίστα [1, 2, 3, 5, 7, 9]. Να εντοπίστε και να εξηγήσετε το πρόβλημα.

```
def purify( List ) :
    for item in List :
        if item % 2 == 1 :
            List.remove(item)
        print List
    return List
```

Δραστηριότητα 8.

Να ορίσετε μια συνάρτηση **product** η οποία δέχεται μια λίστα ακεραίων και επιστρέφει το γινόμενο όλων των στοιχείων της λίστας.

Βοήθεια

Για παράδειγμα, η **product([4, 5, 5])** επιστρέφει 100.
Μια ενδεικτική λύση είναι:

```
def product(the_list):
    p = 1
    for item in the_list:        # για κάθε στοιχείο της λίστας
        p *= item
    return p
```

ή

```
def product(the_list):
    p = 1
    for item in range( len (the_list ) ):
                                 # για κάθε στοιχείο της λίστας
        p *= item
    return p
```

Δραστηριότητα 9.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο θα διαβάζει μια λίστα αριθμών και θα υπολογίζει το μέσο όρο των θετικών αριθμών. Για την εισαγωγή των αριθμών και τον υπολογισμό του μέσου όρου να υλοποιήσετε κατάλληλες συναρτήσεις.

Δραστηριότητα 10.

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει δύο λίστες του ίδιου μεγέθους και θα δημιουργεί μια τρίτη. Σε αυτήν κάθε στοιχείο είναι ο μέσος όρος των αντίστοιχων στοιχείων των δύο λιστών, δηλαδή των στοιχείων που βρίσκονται στις ίδιες θέσεις. Για την εισαγωγή των λιστών και τον υπολογισμό της τρίτης λίστας να υλοποιήσετε κατάλληλες συναρτήσεις.

```
def readList( ):
    ..................................................
def avgList( A, B ):
   ..................................................
```

Βοήθεια

Παρακάτω δίνεται ένα παράδειγμα εκτέλεσης της **avgList** :

```
>>> A = [ 2, 4, 12, 15 ]
>>> B = [ 4, 8, 15, 18 ]
>>> avgList( A, B )
[ 3.0, 6.0, 13.5, 16.5 ]
```

ΚΕΦΑΛΑΙΟ 8 Δομές Δεδομένων ΙΙ

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Μια **δομή δεδομένων** μπορεί να οριστεί ως ένα σχήμα οργάνωσης σχετικών μεταξύ τους στοιχείων δεδομένων. Η επιλογή της κατάλληλης δομής δεδομένων παίζει σημαντικό ρόλο στην ανάπτυξη του αλγορίθμου, για την επίλυση ενός προβλήματος.

Στο κεφάλαιο αυτό θα αναπτυχθούν οι *ενσωματωμένες* δομές της **Python**, όπως τα αλφαριθμητικά, οι λίστες, οι πλειάδες και τα λεξικά. Σχετικά με τις λίστες και τα αλφαριθμητικά, που είχαν καλυφθεί και στην ύλη της Β' Λυκείου, θα γίνει μια πιο εκτενής παρουσίαση των βασικών λειτουργιών τους, μέσα από διάφορα παραδείγματα. Επίσης, με χρήση των ενσωματωμένων δομών της **Python** θα παρουσιαστούν υλοποιήσεις σύνθετων δομών δεδομένων, όπως είναι η στοίβα και η ουρά. Τέλος θα παρουσιαστούν θεωρητικά, οι δομές των γράφων και των δέντρων.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου, θα μπορούμε να:

- αναλύουμε τρόπους αναπαράστασης στη μνήμη βασικών στατικών και δυναμικών δομών
- αναφέρουμε τα ιδιαίτερα χαρακτηριστικά των δομών: πίνακα, λίστας, στοίβας και ουράς
- κατονομάζουμε τη χρησιμότητα των δομών: δέντρα και γράφοι
- επιλέγουμε και χρησιμοποιούμε κατάλληλες δομές στα προγράμματα που υλοποιούμε.

Λέξεις κλειδιά

Δομές δεδομένων, λίστα, στοίβα, ουρά, λεξικό, γράφος.

Διδακτικές Ενότητες

8. Δομές Δεδομένων II
- **8.1** Συμβολοσειρές **(strings)**
- **8.2** Λίστες
- **8.3** Στοίβα
- **8.4** Ουρά
- **8.5** Πλειάδες
- **8.6** Λεξικά
- **8.7** Εισαγωγή στους γράφους και τα δέντρα
- **8.8** Δραστηριότητες
- **8.9** Ερωτήσεις

Σύνοψη

Στο κεφάλαιο αυτό παρουσιάστηκαν οι βασικές δομές δεδομένων της **Python** που είναι ενσωματωμένες στη γλώσσα και χωρίζονται σε δυο κατηγορίες. Αυτές που επιτρέπουν την τροποποίηση των περιεχομένων τους **(immutables)**, όπως οι *λίστες* και τα *λεξικά* και αυτές που δεν το επιτρέπουν, όπως οι *συμβολοσειρές* και οι *πλειάδες*. Στη συνέχεια, με τη βοήθεια της δομής της λίστας υλοποιήθηκαν οι δομές δεδομένων της στοίβας και της ουράς και παρουσιάστηκαν κάποιες εφαρμογές τους.

Τέλος παρουσιάστηκε η δομή δεδομένων του γράφου και του δέντρου.

Σημείωση: Το κεφάλαιο αυτό είναι μερικά εντός διδακτέας και εξεταστέας ύλης.

Θεωρητική ανασκόπηση - Συμβολοσειρές

Μια συμβολοσειρά στην **Python** είναι ουσιαστικά μια λίστα χαρακτήρων με σταθερό μέγεθος. Η αρίθμηση των δεικτών ξεκινάει από το 0, όπως συμβαίνει στη **C** και στην **Java**, ενώ παρέχεται η δυνατότητα αρνητικής δεικτοδότησης.

0	1	2	3	4	5	6	7
'P'	**'Y'**	**'T'**	**'H'**	**'O'**	**'N'**	**' '**	**'3'**
-8	-7	-6	-5	-4	-3	-2	-1

Μπορούμε να δημιουργήσουμε την παραπάνω συμβολοσειρά με διάφορους τρόπους. Ένας από αυτούς δίνεται παρακάτω:

```
word = "PYTHON" + "  " + "3"
```

Δραστηριότητες - Ασκήσεις

Δραστηριότητα 1.

Ανοίξτε το διερμηνευτή της **Python**, δώστε τις παρακάτω εντολές και παρατηρήστε τα αποτελέσματα. Στη συνέχεια περιγράψτε τη λειτουργία των τελεστών **+, in, not in** και των συναρτήσεων **len(), str(), int()**.

```
>>> W = "MONTY PYTHON"
>>> w[0] + w[1] + w[2] + w[8]
>>> len( w )
>>> 123 + "123"
>>> str(123) + "123"
>>> 123 + int("123")
>>> "PYTHON" in w
>>> vowels = "aeiou"
>>> 'e' in vowels
>>> 'p' not in vowels
>>> for letter in vowels:
        print letter
```

Δραστηριότητα 2.

Να γράψετε μια συνάρτηση η οποία θα ελέγχει αν μια συμβολοσειρά αποτελεί ηλεκτρονική διεύθυνση αλληλογραφίας ελληνικού ιστότοπου, δηλαδή περιέχει το σύμβολο **'@'**, δεν περιέχει κενά και έχει κατάληξη **'.gr'**.

Δραστηριότητα 3.

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει λέξεις από το πληκτρολόγιο και θα μετράει και θα εμφανίζει πόσες λέξεις ξεκινούν από το γράμμα Α (κεφαλαίο ή μικρό). Όταν δοθεί λέξη που να τελειώνει σε 'Ω' ή 'ω' θα τερματίζει.

Δραστηριότητα 4.

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει λέξεις από το πληκτρολόγιο και θα τις ενώνει σε μια μεγάλη πρόταση, την οποία στη συνέχεια θα εμφανίζει στην οθόνη. Οι λέξεις θα χωρίζονται μεταξύ τους με κενά και η πρόταση θα τελειώνει με τελεία '.' .

Δραστηριότητα 5.

Να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει μια λέξη και θα ελέγχει αν είναι παλινδρομική, δηλαδή αν διαβάζεται το ίδιο και αντίστροφα, για παράδειγμα **radar**, **madam**.

Θεωρητική ανασκόπηση - Λίστες

Εισαγωγή

Όπως αναφέρθηκε σε μια λίστα στην **Python**, η αρίθμηση των δεικτών ξεκινάει από το 0, όπως συμβαίνει στη C και στην Java, ενώ παρέχεται η δυνατότητα αρνητικής δεικτοδότησης.

0	1	2	3	4	5	6	7
1	**1**	**2**	**3**	**5**	**8**	**13**	**21**
-8	-7	-6	-5	-4	-3	-2	-1

```
a=[1,1,2,5,8,13,21]
print a
```

Εμφανίζεται στην οθόνη `[1, 1, 2, 5, 8, 13, 21]`

Η **Python** παρέχει μεγάλη ποικιλία τελεστών και μεθόδων επεξεργασίας λιστών.

Δραστηριότητα 6.

Δίνονται τα παρακάτω προγράμματα σε **Python**. Τι πιστεύετε ότι θα εμφανιστεί στην οθόνη μετά την εκτέλεσή τους; Καταγράψτε τις απαντήσεις σας. Στη συνέχεια επαληθεύστε τις απαντήσεις αυτές, εκτελώντας τα προγράμματα στο προγραμματιστικό περιβάλλον **IDLE** της **Python**.

```
x = [21, 23, 25, 27]
y = [5, 6, 7, 8]
z = x + y
print z
# Απάντηση: .....

print z[ 1 ]
# Απάντηση: .....

z[ 0 ] = 45
print z
# Απάντηση: .....

a = [ x , y ]
print a
print a[ 1 ][ 2 ]
# Απάντηση: .....
```

Δραστηριότητα 7.

Δίνεται το παρακάτω πρόγραμμα. Τι πιστεύετε ότι θα εμφανιστεί στην οθόνη μετά την εκτέλεσή του; Επαληθεύστε το αποτέλεσμα δοκιμάζοντας το πρόγραμμα στο προγραμματιστικό περιβάλλον της **Python**.

```
daysofweek= [ "Δευτέρα","Τρίτη","Τετάρτη","Πέμπτη" ] +
                ["Παρασκευή", "Σάββατο","Κυριακή"]
print "θα έχω πολύ ελεύθερο χρόνο την ", daysofweek[ 6 ]
```

Δραστηριότητα 8.

Πειραματιστείτε και δώστε στη συνέχεια την περιγραφή - τεκμηρίωση των παρακάτω συναρτήσεων.

```
def print_grades(grades):
    for grade in grades:
        print grade

def grades_sum(grades):
    total = 0
    for grade in grades:
        total += grade
    return total

def grades_average(grades):
    sum_of_grades = grades_sum(grades)
    average = sum_of_grades / float(len(grades))
    return average
```

Δραστηριότητα 9.

Ανοίξτε το διερμηνευτή της **Python**, δώστε τις παρακάτω εντολές και παρατηρήστε τα αποτελέσματα. Να περιγράψετε τη λειτουργία των τελεστών **+**, **in** , των μεθόδων **pop** και **append** και των συναρτήσεων **len** και **range** .

```
>>> w = "MONTY PYTHON"
>>> fib = [1, 1, 2, 3, 5, 8, 13, 21]
>>> fib = fib + [34]
>>> fib = [0] + fib
>>> print fib
>>> last = fib.pop()
>>> print fib, last
>>> fib.append(55)
>>> print fib
>>> 5 in fib
>>> len( fib )
>>> fib.pop() ; len( fib )
>>> range( 10 )
>>> range( 0, 10 )
>>> len( range( 10 ) )
>>> range( 1, 10 )
>>> range( 1, 10, 2 )
>>> range( 10, 0, -2 )
>>> 100 not in range( 1, 10 )
```

Δραστηριότητα 10.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο θα ζητάει από το χρήστη έναν θετικό ακέραιο αριθμό Ν, θα δημιουργεί μια λίστα με όλους τους αριθμούς από το 1 έως και το Ν και στη συνέχεια θα εμφανίζει το άθροισμα των αριθμών της λίστας.

Δραστηριότητα 11.

Το παρακάτω πρόγραμμα δημιουργεί μια λίστα με όλα τα θετικά πολλαπλάσια του **3** που είναι μικρότερα του **1000**. Να το εκτελέσετε στο **IDLE** και να διορθώσετε τυχόν λάθη, ώστε να βγάζει το σωστό αποτέλεσμα.

```
list3 = [ ]
for i in range(1,1001, 3):
    list3 = list3 + i
```

Δραστηριότητα 12.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο θα διαβάζει από το πληκτρολόγιο αριθμούς και θα τους αποθηκεύει σε μια λίστα. Η εισαγωγή των αριθμών θα σταματάει όταν δοθεί ένας αρνητικός αριθμός. Στη συνέχεια το πρόγραμμα θα εμφανίζει:

1) Πόσοι αριθμοί δόθηκαν.
2) Τους αριθμούς σε αντίστροφη σειρά από αυτή που δόθηκαν.

Δραστηριότητα 13.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο να διαβάζει από το πληκτρολόγιο 12 ακέραιους αριθμούς και τους αποθηκεύει σε μια λίστα. Στη συνέχεια να υπολογίζει και να εμφανίζει:

1) Το μέσο όρο των 12 αριθμών.
2) Το μεγαλύτερο αριθμό από τους 12.
3) Το πλήθος των θετικών αριθμών.

Ενδεικτική Λύση

```
# -*- coding: utf-8 -*-

number_list = []
for i in range(12):
    number = input("δώσε έναν αριθμό = ")
    number_list.append(number)

# Χρησιμοποιούμε έναν αθροιστή s στον οποίο
# προσθέτουμε κάθε φορά τον επόμενο αριθμό στη
# λίστα.
s = 0
for number in number_list:
    s = s + number
average = s / len(number_list)
print "μέσος όρος = ", average

maximum = number_list[0] #αρχικοποίηση με το 1ο στοιχείο
for i in range(1,len(number_list)):
    if number_list[i] > maximum :
        maximum = number_list[i]
print "μέγιστο = ", maximum

# Κάθε φορά που βρίσκουμε έναν θετικό αυξάνουμε
# κατά 1 το μετρητή positives.
```

```
positives = 0
for number in number_list:
    if number > 0 :
        positives = positives + 1
print "θετικοι = ", positives
```

Μπορούμε να διασχίσουμε μια λίστα List με δυο τρόπους:

```
for index in range( len( List ) ):
    print List[ index ]
```

```
for item in List :
    print item
```

Δραστηριότητα 14.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο θα διαβάζει αριθμούς από το πληκτρολόγιο μέχρι το άθροισμά τους να ξεπεράσει το 1000, θα τους αποθηκεύει σε μια λίστα και στη συνέχεια θα υπολογίζει και θα εμφανίζει:

1) Το άθροισμά τους.
2) Τη μέγιστη τιμή που έχει στοιχείο της λίστας.
3) Πόσες φορές εμφανίζεται αυτή η μέγιστη τιμή.

Ενδεικτική Λύση

```
# -*- coding: utf-8 -*-

number_list = [ ]
Sum = 0
while Sum <= 1000:
    number = input("δώσε έναν αριθμό = ")
    number_list.append(number)
    Sum = Sum + number
```

```
maximum = number_list[0]
for item in number_list:
    if item > maximum :
        maximum = item
print "μέγιστο = ", maximum

# Κάθε φορά που βρίσκουμε έναν αριθμό ίσο με τη μέγιστη
# τιμή αυξάνουμε κατά 1 το μετρητή counter.
counter = 0
for item in number_list:
    if item == maximum :
        counter = counter + 1
number_list = [ ]
```

Δραστηριότητα 15.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο θα διαβάζει αριθμούς από το πληκτρολόγιο, μέχρι να δοθεί ο αριθμός 0. Στη συνέχεια με τη χρήση λίστας θα εμφανίζει τους αριθμούς σε αντίστροφη σειρά από αυτή με την οποία τους διάβασε.

Δραστηριότητα 16.

Να αναπτύξετε μια συνάρτηση **reverseList(L)** σε **Python**, η οποία επιστρέφει τη δοσμένη λίστα L αντεστραμμένη, όπως φαίνεται στο παρακάτω παράδειγμα

```
>>> reverseList( [ 1, 1, 2, 3 ,5, 8, 13 ,21 ] )
[ 21, 13, 8, 5 ,3, 2, 1 ,1 ]
```

Δραστηριότητα 17.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο, δεδομένης μια λίστας ακέραιων αριθμών, θα διαχωρίζει τους αριθμούς σε δύο νέες λίστες, μία για τους θετικούς και μία για τους αρνητικούς.

Δραστηριότητα 18.

Στα επόμενα παραδείγματα προσπαθήστε πρώτα να μαντέψετε τι θα εμφανιστεί μετά την **print()** και έπειτα εκτελέστε τον κώδικα στο προγραμματιστικό περιβάλλον **IDLE Python**.

```
alist = ['a','b','c','d']
ch = ''
for i in alist:
    ch += i
print(ch)

list1 = ['a','b','c']
i = 0
s1=''
for ch in list1:
    i +=2
    s1 = s1+i*ch
print(s1)
```

Δραστηριότητα 19. Στοίβα - Ουρά

Να υλοποιήσετε τις παρακάτω λειτουργίες της δομής δεδομένων "Στοίβα" συμπληρώνοντας κατάλληλα τις συναρτήσεις. Θυμίζουμε ότι στη στοίβα η *εισαγωγή* και η Ι γίνονται από το ένα άκρο της λίστας. Μπορείτε να χρησιμοποιήσετε τις μεθόδους **append**, **pop** της λίστας και τη συνάρτηση **len**.

```
def push(stack, item) :                 def isEmpty(stack) :
    ____________________                    ____________________
def pop(stack) :                        def createStack( ) :
    ____________________                    ____________________
```

Δραστηριότητα 20.

Προσπαθήστε να "μαντέψετε" τα αποτελέσματα (όπου υπάρχουν) που εμφανίζονται στις παρακάτω εντολές. Επαληθεύστε εκτελώντας τον κώδικα.

```
>>> a=[5,8,13,21,34]
>>> a.append(55)
>>> print( a )
.....
>>> a.pop()
55
>>> print( a )
.....
>>> a.pop(2)
.....
>>> print( a )
.....
```

Δραστηριότητα 21.

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο θα διαβάζει αριθμούς μέχρι να δοθεί το 0 και στη συνέχεια θα εμφανίζει τους αριθμούς στην αντίστροφη σειρά από αυτή που τους διάβασε, με τη χρήση στοίβας.

Δραστηριότητα 22.

Να δώσετε την υλοποίηση της δομής δεδομένων ουρά, χωρίς τη χρησιμοποίηση των λειτουργιών **append** και **pop**(θέση) των λιστών.

def enqueue(queue, item) : _____	def isEmpty(queue) : _____
def dequeue(queue) : _____	def createQueue( ) : _____

Στη συνέχεια να γράψετε ένα πρόγραμμα το οποίο θα διαβάζει αριθμούς μέχρι να δοθεί η τιμή **None** και θα τους αποθηκεύει σε μια ουρά. Μετά θα τους εξάγει από την ουρά και θα τους αποθηκεύει σε μια στοίβα. Αφού τοποθετηθούν όλοι στη στοίβα θα τους απωθεί και θα τους εμφανίζει έναν - έναν στην οθόνη. Τι παρατηρείτε; Τι καταφέρατε με τον παραπάνω αλγόριθμο;

Παρατήρηση

Στις παραπάνω ασκήσεις ζητείται να μη χρησιμοποιήσετε έτοιμες συναρτήσεις της **Python**, όπως **sum**, **max**, **min**, **sorted**, **reversed** κ.ο.κ., οι οποίες δεν αναφέρονται στο βιβλίο και έτσι δεν εξυπηρετούνται οι διδακτικοί στόχοι του μαθήματος στη φάση αυτή. Ζητείται λοιπόν εδώ, αν θέλετε να τις χρησιμοποιήσετε, να δώσετε και δική σας υλοποίηση.

Σημειώνεται ότι το εύρος χρήσης έτοιμων στοιχείων μιας γλώσσας καθορίζεται τόσο από τους διδακτικούς στόχους κάθε κεφαλαίου όσο και από την ίδια την εκφώνηση μιας άσκησης.

ΚΕΦΑΛΑΙΟ 9 Εφαρμογές σε γλώσσα προγραμματισμού με χρήση API

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιαστούν ορισμένα θεωρητικά στοιχεία των διεπαφών και στη συνέχεια θα γίνει υλοποίηση μικρών Προγραμμάτων-εφαρμογών με την βιβλιοθήκη **Tkinter**. Η Tkinter είναι η πρότυπη βιβλιοθήκη της Python για την υλοποίηση γραφικών διεπαφών **(Graphical User Interface, GUI)**. Με τον τρόπο αυτό θα έχουμε την ευκαιρία να δούμε πώς μπορούμε να κάνουμε τα προγράμματά μας πιο ελκυστικά και φιλικά προς το χρήστη, χρησιμοποιώντας μια βιβλιοθήκη με έτοιμα τμήματα κώδικα.

Η συνεισφορά της μεγάλης κοινότητας προγραμματιστών που υποστηρίζει τη γλώσσα, με συλλογές έτοιμου κώδικα για πάρα πολλές εφαρμογές, αποτελεί ένα ισχυρό πλεονέκτημα της γλώσσας προγραμματισμού Python. Οι περισσότερες από αυτές τις εφαρμογές ανήκουν στην κατηγορία του ελεύθερου λογισμικού και είναι γνωστές ως Διεπαφές Προγραμματισμού Εφαρμογών (Application Programming Interfaces-APIs) ή και ως βιβλιοθήκες λογισμικού.

Ωστόσο, ο βασικός σκοπός αυτού του κεφαλαίου δεν είναι να μάθουμε ένα συγκεκριμένο **API**, ούτε φυσικά να "αποστηθίσουμε" κάποιες βασικές συναρτήσεις των βιβλιοθηκών. Σκοπός του είναι να αντιληφθούμε ότι, κατά την ανάπτυξη μιας εφαρμογής, είναι καλή πρακτική η αξιοποίηση τυχόν διαθέσιμων τμημάτων έτοιμου κώδικα που ανήκουν σε κάποιο API και να πειραματιστούμε με τις δυνατότητες μιας τέτοιας βιβλιοθήκης.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- περιγράφουμε τις βασικές αρχές της επικοινωνίας ανθρώπου υπολογιστή
- χρησιμοποιούμε περιβάλλοντα, **Application Program Interfaces (APIs)** και βιβλιοθήκες για την ανάπτυξη και τροποποίηση εφαρμογών λογισμικού στη γλώσσα προγραμματισμού **Python**.

Λέξεις κλειδιά
Επικοινωνία ανθρώπου υπολογιστή, διεπαφή χρήστη, διεπαφές προγραμματισμού εφαρμογών API, βιβλιοθήκες λογισμικού.

Διδακτικές Ενότητες

9. Εφαρμογές σε γλώσσα προγραμματισμού με χρήση API
   - 9.1 Επικοινωνία ανθρώπου-υπολογιστή και διεπαφή χρήστη
     - 9.1.1 Παραδείγματα - Εφαρμογές
   - 9.2 Γενικές αρχές σχεδίασης διεπαφής
   - 9.3 Η βιβλιοθήκη **Tkinter** για ανάπτυξη γραφικών διεπαφών **GUI** στην **Python**
   - 9.4 Δραστηριότητες
     - 9.4.1 Εργαστηριακές ασκήσεις
   - 9.5 Σύντομα Θέματα για συζήτηση στην τάξη

   Ερωτήσεις - Ασκήσεις

Σύνοψη κεφαλαίου
Στην ενότητα αυτή αρχικά προσεγγίσαμε τις βασικές αρχές σχεδίασης διεπαφής. Στη συνέχεια είχαμε την ευκαιρία να υλοποιήσουμε απλά παραδείγματα ανάπτυξης διεπαφών, χρησιμοποιώντας τη βιβλιοθήκη **Tkinter** της γλώσσας **Python**. Βιβλιοθήκη που μας προσφέρει πολλές δυνατότητες για να κατασκευάσουμε γραφικές διεπαφές. Μέσα από τη δημιουργία μηνυμάτων ανατροφοδότησης, κουμπιών για την εκτέλεση κάποιων διεργασιών και πεδίων εισόδων για την εισαγωγή δεδομένων, διερευνήσαμε πώς να κάνουμε τα προγράμματά μας πιο φιλικά προς το χρήστη δίνοντας τη δυνατότητα αλληλεπίδρασης με το περιβάλλον των εφαρμογών μας.

Σημείωση: Το κεφάλαιο αυτό είναι εκτός διδακτέας και εξεταστέας ύλης.

ΚΕΦΑΛΑΙΟ 10 Βάσεις δεδομένων

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό προσεγγίζονται θέματα που αφορούν στη δημιουργία και διαχείριση Βάσεων Δεδομένων μέσα από εντολές της γλώσσας **Python**.

Ένας συνηθισμένος τρόπος αποθήκευσης δεδομένων στον προγραμματισμό είναι αυτός που χρησιμοποιεί *αρχεία*. Τον τρόπο αυτό τον γνωρίσαμε τόσο στη Β' τάξη όσο και σε προηγούμενα κεφάλαια αυτού του βιβλίου. Ο τρόπος αυτός είναι ικανοποιητικός μόνο, όταν τα δεδομένα είναι σχετικά περιορισμένα σε μέγεθος, απλά και χρησιμοποιούνται από λίγους χρήστες. Όταν τα δεδομένα είναι πολύπλοκα και υπάρχουν πολλοί χρήστες, τότε η αποθήκευσή τους σε ένα μόνο αρχείο ή σε πολλά αρχεία παρουσιάζει πολλά προβλήματα, τα οποία λύνουν οι Βάσεις Δεδομένων.

Μια Βάση Δεδομένων-ΒΔ **(Data Base)**, είναι μια συλλογή δεδομένων με υψηλό βαθμό οργάνωσης. Αναπόσπαστο τμήμα μιας βάσης δεδομένων αποτελεί ένα πακέτο προγραμμάτων λογισμικού που ονομάζεται *Σύστημα Διαχείρισης Βάσης Δεδομένων* - ΣΔΒΔ **(Data Base Management System)**, το οποίο επιτρέπει τη δημιουργία της ΒΔ και τη διαχείριση των δεδομένων που περιέχει, κρύβοντας την πολυπλοκότητά της από το χρήστη.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- περιγράφουμε τους βασικούς τύπους δεδομένων που χρησιμοποιεί η βιβλιοθήκη της **Python**, **sqlite3**, για τη δημιουργία και διαχείριση ΒΔ
- περιγράφουμε το Μοντέλο Δεδομένων και τα επίπεδά του
- χρησιμοποιούμε τις βασικές εντολές για τη δημιουργία μιας ΒΔ
- χρησιμοποιούμε τις βασικές εντολές για τη δημιουργία ενός πίνακα σε μια ΒΔ
- χρησιμοποιούμε τις βασικές εντολές για την εισαγωγή, τροποποίηση και διαγραφή εγγραφών σε ένα πίνακα
- χρησιμοποιούμε παραλλαγές της εντολής SELECT για την αναζήτηση στοιχείων σε μια ΒΔ.
- διαγράφουμε έναν πίνακα.

Λέξεις κλειδιά

Βάση δεδομένων, μοντέλο δεδομένων, σχεσιακό μοντέλο δεδομένων, **sqlite3**, εισαγωγή, τροποποίηση και διαγραφή δεδομένων, πίνακας.

Διδακτικές Ενότητες

10. Βάσεις δεδομένων
    - 10.1 Αναφορά στο Μοντέλο Δεδομένων
    - 10.2 Εισαγωγή στη διαχείριση Βάσεων Δεδομένων με προγραμματισμό
        - 10.2.1 Η γλώσσα **SQL**
        - 10.2.2 Η βιβλιοθήκη **SQLite** της **Python**
        - 10.3 Δημιουργία ή σύνδεση με μια Βάση Δεδομένων στην **Python**
    - 10.4 Εισαγωγή, ενημέρωση και διαγραφή δεδομένων
        - 10.4.1 Ενημέρωση δεδομένων
        - 10.4.2 Διαγραφή πίνακα
    - 10.5 Αναζήτηση και ταξινόμηση δεδομένων
    - 10.6 Δραστηριότητες
    - 10.7 Ερωτήσεις

    Εργαστηριακές ασκήσεις

Σύνοψη

Στο κεφάλαιο αυτό γνωρίσαμε τη χρήση της βιβλιοθήκης **sqlite3**, η οποία επιτρέπει τη χρήση εντολών της **SQL** μέσα σε προγράμματα **Python**. Παρουσιάστηκε το σχεσιακό μοντέλο δεδομένων που χρησιμοποιείται ευρέως στις βάσεις δεδομένων και οι βασικές εντολές της **sqlite3** για τη δημιουργία μιας ΒΔ. Στη συνέχεια παρουσιάστηκε η δημιουργία ενός ή περισσοτέρων πινάκων μέσα στη ΒΔ, καθώς και οι εντολές που είναι απαραίτητες για τη διαχείριση μιας βάσης δεδομένων. Οι εντολές αυτές είναι η **INSERT**, για την εισαγωγή δεδομένων σε πίνακα, η εντολή **UPDATE**, για την τροποποίηση στοιχείων ενός πίνακα και η εντολή **DELETE**, για τη διαγραφή των στοιχείων ενός πίνακα. Παρουσιάστηκαν οι κυριότερες ίσως λειτουργίες σε μια ΒΔ, όπως η αναζήτηση και εμφάνιση στοιχείων της ΒΔ, λειτουργίες που βασίζονται σε παραλλαγές της εντολής **SELECT**. Τέλος, θα πρέπει να σημειωθεί ότι τα παραδείγματα των Βάσεων Δεδομένων που χρησιμοποιήθηκαν στο κεφάλαιο, εξυπηρετούν εκπαιδευτικούς σκοπούς και δεν είναι οι βέλτιστες λύσεις.

Σημείωση: Το κεφάλαιο αυτό είναι εκτός διδακτέας και εξεταστέας ύλης.

ΚΕΦΑΛΑΙΟ 11 Αντικειμενοστρεφής Προγραμματισμός

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Στο κεφάλαιο αυτό αναπτύσσονται οι έννοιες που αφορούν τον αντικειμενοστρεφή προγραμματισμό.

Στη Β' τάξη ΕΠΑ.Λ., καθώς και στην αρχή του παρόντος διδακτικού υλικού, είχε γίνει μια σύντομη αναφορά στον αντικειμενοστρεφή προγραμματισμό με παραδείγματα. Στο κεφάλαιο αυτό θα γίνει εμβάθυνση στις σχετικές έννοιες και εξάσκηση στη δημιουργία κλάσεων και αντικειμένων με χρήση της **Python**.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- αναγνωρίζουμε και να κατονομάζουμε χαρακτηριστικά του αντικειμενοστρεφούς προγραμματισμού
- περιγράφουμε τις διαφορές του αντικειμενοστρεφούς προγραμματισμού σε σχέση με το δομημένο προγραμματισμό
- δημιουργούμε απλά αντικείμενα και κλάσεις
- διακρίνουμε την έννοια της κλάσης από εκείνη του αντικειμένου
- αναγνωρίζουμε το ρόλο της κληρονομικότητας και του πολυμορφισμού στη δημιουργία επαναχρησιμοποιήσιμου κώδικα.

Λέξεις κλειδιά

Αντικειμενοστρεφής προγραμματισμός, κλάση, αντικείμενο, μέθοδος, κληρονομικότητα, πολυμορφισμός, προγραμματισμός οδηγούμενος από γεγονότα.

Διδακτικές Ενότητες

11. Αντικειμενοστρεφής Προγραμματισμός
- **11.1** Αντικείμενα και Κλάσεις
- **11.2** Στιγμιότυπα (αυτόματη αρχικοποίηση αντικειμένων)
- **11.3** Ιδιότητες και Μέθοδοι
- **11.4** Χαρακτηριστικά: Κληρονομικότητα, Πολυμορφισμός
  - **11.4.1** Κληρονομικότητα

Σύνοψη

Στο κεφάλαιο αυτό παρουσιάστηκαν οι βασικές αρχές του Αντικειμενοστρεφούς Προγραμματισμού. Γνωρίσαμε τη βασική ορολογία και μάθαμε να δημιουργούμε κλάσεις και να χρησιμοποιούμε αντικείμενα. Τέλος παρουσιάστηκαν οι έννοιες της κληρονομικότητας και της ενθυλάκωσης και πώς τα γεγονότα καθορίζουν τη συμπεριφορά των αντικειμένων.

Σημείωση: Το κεφάλαιο αυτό είναι μερικά εντός διδακτέας και εξεταστέας ύλης.

Δραστηριότητες - Ασκήσεις

Δραστηριότητα 1

Να υλοποιήσετε την δομή δεδομένων "Στοίβα" ως μια κλάση αξιοποιώντας τις τεχνικές του αντικειμενοστρεφούς προγραμματισμού.

```
class Stack :
    def ____init____(self ) :
        self.items = [ ]
    def push(self, item) :
        ____________________
    def pop(self) :
        ____________________
    def isEmpty(self) :
        ____________________
```

Να υλοποιήσετε την εφαρμογή *Αντιστροφή αριθμών* που περιγράφεται στις σελίδες 143-144 του βιβλίου, χρησιμοποιώντας την αντικειμενοστρεφή έκδοση της στοίβας που υλοποιήσατε. Ποιες διαφορές παρατηρείτε ανάμεσα στις δυο υλοποιήσεις; Ποια προτιμάτε εσείς; Γιατί; Ποια μέθοδος της κλάσης **Stack** έχει υποκαταστήσει την συνάρτηση **createStack**;

Δραστηριότητα 2.

Να ορίσετε την κλάση **Student** η οποία περιέχει τα παρακάτω δεδομένα:

- το όνομα του μαθητή**(name)**
- μια λίστα με τους βαθμούς του μαθητή **(grades)**
- μια μεταβλητή κλάσης η οποία μετράει πόσα αντικείμενα υπάρχουν κάθε στιγμή **(students_count)**

και τις παρακάτω μεθόδους:

- κατασκευαστής **(constructor)** αντικειμένων της κλάσης μαθητής **(___init___)**
- εμφανίζει μήνυμα "καλημέρα" και το όνομα του μαθητή
- υπολογίζει και επιστρέφει το μέσο όρο των βαθμών του μαθητή **(average)**
- εμφανίζει το περιεχόμενο της μεταβλητής *πλήθος_μαθητών*, δηλαδή είναι μέθοδος κλάσης και μπορεί να κληθεί και με το όνομα της κλάσης.

Στη συνέχεια να γράψετε ένα πρόγραμμα το οποίο:

4) Δημιουργεί **5** αντικείμενα **student** με στοιχεία συμμαθητών σας.
5) Αποθηκεύει τα αντικείμενα αυτά σε μια λίστα.
6) Διατρέχει τη λίστα και εμφανίζει τα ονόματα όσων έχουν μέσο όρο πάνω από 18.
7) Εμφανίζει το όνομα του μαθητή με το μεγαλύτερο μέσο όρο.

Δραστηριότητα 3.

Να σχεδιάσετε και να υλοποιήσετε μια ιεραρχία κλάσεων, όπου οι κλάσεις Τετράγωνο **(Square)**, Κύκλος **(Circle)** και Ορθογώνιο **(Rectangle)** κληρονομούν από την κλάση Σχήμα **(Schema)**. Να ορίσετε όλα τα δεδομένα και τις μεθόδους που θεωρείτε απαραίτητα για κάθε κλάση. Κάθε κλάση πρέπει να έχει και μια μέθοδο Εμβαδό **(getArea)** που θα υπολογίζει και θα επιστρέφει το εμβαδόν του αντίστοιχου σχήματος.

Δραστηριότητα 4.

Δίνεται το παρακάτω πρόγραμμα σε Python:

```
class Teacher:

count = 0

def ___init___ (self, name, subject):
    self.name = name
    self.subject = subject
    Teacher.count += 1

def displayCount(self):
    print "All Teachers ", Teacher.count

def displayTeacher(self):
    print "Name : ", self.name, ", Subject: ", self.subject

godel = Teacher("Turing", "Logic")
dijkstra = Employee("Dijkstra", "Computer Science")
godel.displayT eacher()
dijkstra.displayTeacher( )
Teacher.displayCount( )
```

Να καταγράψετε:

1) τον κατασκευαστή της κλάσης
2) τις ιδιότητες της κλάσης
3) τις μεθόδους της κλάσης
4) τα στιγμιότυπα κάθε κλάσης.

Δραστηριότητα 5.

Δίνεται το παρακάτω πρόγραμμα:

```
class Customer:
    # Ένας πελάτης με καταθετικό λογαριασμό σε μια τράπεζα.
    # Οι πελάτες έχουν τις παρακάτω ιδιότητες:
    # name: όνομα του πελάτη
    # balance: το υπόλοιπο του λογαριασμού του πελάτη.

    def __init__ ( self, name, balance = 0.0 ):
        # επιστρέφει ένα αντικείμενο της κλάσης Customer
        # με όνομά name και αρχικό υπόλοιπο 0.0
        self.name = name
        self.balance = balance

    def withdraw(self, amount):
        # Επιστρέφει το νέο υπόλοιπο του λογαριασμού μετά την
        # ανάληψη του ποσού *amount*

        self.balance -= amount
        return self.balance

    def deposit(self, amount):
        # Επιστρέφει το νέο υπόλοιπο του λογαριασμού μετά την
        # κατάθεση του ποσού *amount*
        self.balance += amount
        return self.balance
```

Να γράψετε εντολές που να κάνουν τα εξής:

1) Να δημιουργούν δύο αντικείμενα της κλάσης **Customer**, **John** και **Mary**.
2) Να γίνεται κατάθεση του ποσού 1400 ευρώ στο λογαριασμό του **John**.
3) Να γίνεται ανάληψη 300 ευρώ από το λογαριασμό του **John**.
4) Να γίνεται ανάληψη 400 ευρώ από το λογαριασμό της **Mary**.
5) Εκτελέστε το ολοκληρωμένο πρόγραμμα στο περιβάλλον της **Python** και μελετήστε τα αποτελέσματα.

ΚΕΦΑΛΑΙΟ 12 Εισαγωγή στην Υπολογιστική Σκέψη

Στοιχεία από το Βιβλίο Μαθητή

Εισαγωγή

Η σύγχρονη εποχή χαρακτηρίζεται από ραγδαίες τεχνολογικές εξελίξεις, που επιφέρουν σημαντικές αλλαγές σε διαφόρους τομείς της οικονομικής, εργασιακής και κοινωνικής ζωής. Καθημερινά, ακόμα και σε απλές ασχολίες, αυξάνεται η απαίτηση χρήσης πληροφοριακών συστημάτων, υπολογιστικών συσκευών και υπηρεσιών του Διαδικτύου. Σε ένα μεταβαλλόμενο τεχνολογικά περιβάλλον, ο σύγχρονος πολίτης καλείται να αναπτύξει ικανότητες και γνώσεις, που να του επιτρέπουν να αξιοποιεί κατά τον καλύτερο δυνατό τρόπο την τεχνολογία της εποχής του.

Στο κεφάλαιο αυτό θα αναπτυχθεί μια σημαντική -για τη σύγχρονη εποχή- νοητική διαδικασία οργάνωσης και επίλυσης προβλήματος: η *υπολογιστική σκέψη*, που ενισχύει το συνδυασμό των ψηφιακών τεχνολογιών με τις ανθρώπινες ιδέες. Μέσα από απλά παραδείγματα θα προσεγγίσουμε τα βασικά χαρακτηριστικά της *υπολογιστικής σκέψης* και θα εξοικειωθούμε με τις σχετικές έννοιες.

Διδακτικοί στόχοι

Μετά τη μελέτη του κεφαλαίου θα μπορούμε να:

- περιγράφουμε την έννοια της υπολογιστικής σκέψης (ΥΣ)
- αναφέρουμε τα χαρακτηριστικά της ΥΣ
- εξηγούμε τη σημασία της ΥΣ στην επίλυση προβλημάτων της καθημερινής ζωής
- αναλύουμε ένα υπολογιστικό πρόβλημα, περιγράφοντας τις βασικές διαδικασίες με τις οποίες αντιμετωπίζεται, όπως τη διάσπασή του σε απλούστερα, τη λογική οργάνωση των δεδομένων, την αναγνώριση προτύπων, την αναγνώριση πιθανών λύσεων, την περιγραφή της λύσης του με αλγοριθμικό τρόπο, τη γενίκευση της λύσης του.

Λέξεις κλειδιά

Υπολογιστική Σκέψη, επίλυση προβλήματος, διάσπαση προβλήματος, αφαίρεση, οπτική αναπαράσταση δεδομένων, αλγοριθμική λύση, γενίκευση λύσης.

Διδακτικές ενότητες

12. Εισαγωγή στην Υπολογιστική Σκέψη
- **12.1** Η έννοια της Υπολογιστικής Σκέψης
- **12.2** Χαρακτηριστικά της Υπολογιστικής Σκέψης
- **12.3** Υπολογιστική σκέψη και επίλυση προβλημάτων
- **12.4** Δραστηριότητες κεφαλαίου
  - Δραστηριότητα 1. Διαχείριση προβλημάτων
  - Δραστηριότητα 2. Γενίκευση
  - Δραστηριότητα 3. Γενίκευση
- **12.5** Ερωτήσεις - Ασκήσεις

Σύνοψη

Στο κεφάλαιο αυτό ασχοληθήκαμε με τα χαρακτηριστικά της υπολογιστικής σκέψης στην επίλυση προβλημάτων. Δόθηκε, με τη βοήθεια παραδειγμάτων, έμφαση στη σημασία της υπολογιστικής σκέψης στην επίλυση προβλημάτων της καθημερινής ζωής.

Σημείωση

Το κεφάλαιο αυτό είναι εκτός διδακτέας και εξεταστέας ύλης.

Παραρτήματα

13 Παραρτήματα

13.1 Ενδεικτικό διαγώνισμα

Ενδεικτικό Διαγώνισμα

Εξεταζόμενο Μάθημα: Προγραμματισμός Υπολογιστών
Σύνολο Σελίδων: Τρεις (3)

ΘΕΜΑ Α

Α1. Να χαρακτηρίσετε τις προτάσεις που ακολουθούν, γράφοντας στο τετράδιό σας, δίπλα στο γράμμα που αντιστοιχεί σε κάθε πρόταση τη λέξη Σωστό, αν η πρόταση είναι σωστή ή τη λέξη Λάθος, αν η πρόταση είναι λανθασμένη.

1. Η εντολή **while** σταματάει όταν η συνθήκη γίνει ψευδής.
2. Ο λογικός τύπος στην **Python** έχει μόνο δύο τιμές, την **True** και την **False**.
3. Η εντολή **print "99" + "1"** θα εμφανίσει **100**.
4. Η δυαδική αναζήτηση βρίσκει το ζητούμενο πολύ πιο αργά από ότι η σειριακή αναζήτηση.
5. Μπορούμε να αλλάξουμε το μέγεθος μιας λίστας προσθέτοντας ή αφαιρώντας στοιχεία.

Μονάδες 10

Α2. Να γράψετε στο τετράδιό σας τους αριθμούς α,β από τη στήλη Α και 1, 2, 3, 4 από τη στήλη Β, έτσι ώστε να αντιστοιχίσετε τη σωστή λειτουργία σε κάθε δομή δεδομένων.

Στήλη Α	**Στήλη Β**
α. Στοίβα	**1.** Εισαγωγή
	2. Ώθηση
β. Ουρά	**3.** Απώθηση
	4. Εξαγωγή

Μονάδες 8

Α3. Σας δίνεται το παρακάτω τμήμα προγράμματος στη γλώσσα **Python**:

```
a = [3,2,4]
b = [1] + a
sum = b[1] + b[2]
pow = sum ** a[1]
print sum, pow
```

Να γράψετε στο τετράδιό σας τι θα εμφανιστεί στην οθόνη μετά την εκτέλεση των παραπάνω εντολών.

Μονάδες 6

Α4. Σας δίνονται **x=10** και **y=20**. Να γράψετε στο τετράδιό σας τον αριθμό κάθε μιας από τις παρακάτω εκφράσεις και δίπλα την τιμή (**True** ή **False**) που προκύπτει από την εκτέλεση των πράξεων.

1) (x < y) or (x**2 > y)
2) not (x==y) and (2*x == y)

Μονάδες 4

Α5. Να μεταφέρετε και να συμπληρώσετε στο τετράδιό σας τον παρακάτω πίνακα με τα αποτελέσματα των πράξεων μεταξύ δυο λογικών μεταβλητών **P** και **Q**.

P	Q	P and Q	P or Q	not (P and Q)
True	True			
True	True			
True	True			
True	True			

Μονάδες 12

ΘΕΜΑ Β

Δίνεται το παρακάτω τμήμα αλγορίθμου στο οποίο έχουν αριθμηθεί οι γραμμές:

Τμήμα Αλγορίθμου	Πίνακας Τιμών

```
1: X = 1
2: D = 3
3: print X, D
4: for iter in range(5):
5: X = X + D
6: D = D + 2
7: print X
```

Αρ. Γραμμής	iter	X	D	Έξοδος
1		1		
2			3	
3				1 3
4	0			
…	…	…	…	

Β1. Να μεταφέρετε τον πίνακα στο τετράδιό σας και να το συμπληρώσετε εκτελώντας το παραπάνω τμήμα αλγορίθμου, όπου για κάθε εντολή που εκτελείται, να γράψετε σε νέα γραμμή του πίνακα τον αριθμό της γραμμής και το αποτέλεσμα της εντολής, είτε πρόκειται για την νέα τιμή κάποιας μεταβλητής είτε για το μήνυμα που εμφανίζεται στην οθόνη.

Μονάδες 12

Β2. Να ξαναγράψετε το παραπάνω τμήμα αλγορίθμου χρησιμοποιώντας τη δομή επανάληψης **while** της **Python**, έτσι ώστε να εμφανίζει τις ίδιες τιμές.

Μονάδες 8

ΘΕΜΑ Γ

Η κλίμακα Τορίνο είναι μία κλίμακα επικινδυνότητας πιθανής πρόσπτωσης ή σύγκρουσης ουρανίου σώματος με τον πλανήτη Γη. Η Κλίμακα Τορίνο φέρει δέκα διαβαθμίσεις σε ακέραιους αριθμούς αρχής γενομένης από το 0, ως κατάσταση ανύπαρκτου κινδύνου ή αμελητέου, καταλήγοντας στο 10 ως κατάσταση τελείας καταστροφής του ανθρώπινου πολιτισμού. Οι διαβαθμίσεις αυτές κατανέμονται στις παρακάτω βασικές ζώνες:

Χρώμα	Κλίμακα	Επικινδυνότητα
Λευκό	0	Κανένας Κίνδυνος
Πράσινο	1	Σύνηθες Επίπεδο
Κίτρινο	2 - 4	Αστρονομικού Ενδιαφέροντος
Πορτοκαλί	5 - 7	Εν Δυνάμει Απειλή
Κόκκινο	8 - 10	Βέβαιη Σύγκρουση

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο για κάθε ένα από 30 ουράνια σώματα θα:

Γ1. διαβάζει τον αριθμό επικινδυνότητας

Μονάδες 4

Γ2. εμφανίζει τη ζώνη επικινδυνότητας του ουράνιου σώματος και το αντίστοιχο χρώμα

Μονάδες 8

Γ3. εμφανίζει το πλήθος των ουρανίων σωμάτων που δεν αποτελούν κανέναν κίνδυνο

Μονάδες 4

Γ4. εμφανίζει το μήνυμα "Κίνδυνος σύγκρουσης", αν υπάρχει έστω και ένα ουράνιο σώμα που βρίσκεται στην υψηλότερη κατηγορία επικινδυνότητας.

Μονάδες 4

ΘΕΜΑ Δ

Να γράψετε ένα πρόγραμμα σε **Python** το οποίο για κάθε μαθητή που δίνει πανελλήνιες εξετάσεις θα:

Δ1. Διαβάζει το όνομα, τον τελικό βαθμό και την πόλη και θα αποθηκεύει τα δεδομένα στις λίστες **name**, **grade**, **city**.

Μονάδες 4

Δ2. σταματάει η είσοδος των δεδομένων όταν δοθεί αντί για όνομα το γράμμα "Τ"

Μονάδες 4

Δ3. εμφανίζει το μέσο όρο των μαθητών από τα Ιωάννινα

Μονάδες 6

Δ4. εμφανίζει τα ονόματα των μαθητών που έχουν βαθμό μεγαλύτερο από τον παραπάνω μέσο όρο.

Μονάδες 6

13.2 Ενδεικτικά Φύλλα εργασίας

Φύλλο εργασίας 1 (στις εντολές εκχώρησης, τύπους δεδομένων, αριθμητικοί - σχεσιακοί - λογικοί τελεστές)

Α. Τι θα εμφανίσουν οι παρακάτω εντολές εκχώρησης στη γλώσσα προγραμματισμού **Python**;

Εντολές εκχώρησης	**Αποτελέσματα**
a = 5 **print** a **print** a + a a=a+10 **print** a a=a *0.1 **print** a **print** a*100	
onoma = "Μυρτώ" **print** onoma	
x=y =5 **print** x,y	

x,y,tmhma=1,2,'Γ Πληροφορική' **print** x,y, tmhma **print** 'Καλημέρα ' + tmhma **print** tmhma * 2	
logikh =**True** **print** logikh	
mathites = 20 mathites == 10 mathites == 10 **or** mathites == 20 not mathites == 10 mathites <= 10 mathites >= 20 mathites != 10	
c=7 **type**(c) c=7.8 **type**(c) c ='Καλημέρα' **type**(c) c=False **type**(c)	

Β. Τι θα εμφανίσουν οι παρακάτω αριθμητικές πράξεις αν εκτελεστούν στο Ολοκληρωμένο Περιβάλλον Ανάπτυξης Προγραμμάτων (**IDLE**) της **Python** (να κάνετε αναλυτικά τις πράξεις);

Πράξεις	**Αποτελέσματα**
2+3*4 =	
2**2 + 4/2 - 3*4 =	
34/10 =	
34%10 =	
34.0/10 =	
34.0 % 10 =	

Γ. Ερωτήσεις ανάπτυξης που αφορούν στη γλώσσα προγραμματισμού **Python**.

1. Μπορεί η ίδια μεταβλητή να αλλάζει τύπο στην **Python**;

_____

2. Είναι συντακτικά σωστή η εντολή: x,y=**input**(),**input**()

_____

α. Αν ναι, πώς γίνεται η εισαγωγή τιμών;

_____

3. Κυκλώστε ποιες από τις επόμενες τιμές λογικών τύπων είναι συντακτικά σωστές: **true, True, TRUE, False, false, FALSE.**

4. Η εντολή **print** είναι συντακτικά σωστή και αν γραφτεί και **Print** ή **PRINT**.

Φύλλο εργασίας 2 (πράξεις στις λογικές εκφράσεις)

Α. Να βρείτε την τιμή των παρακάτω λογικών εκφράσεων, αν A = **True**, B = **False** και C = **True** (να αναγράφονται αναλυτικά οι πράξεις).

1) A **and** B	
2) A **and** (**not** B)	
3) (**not** A) **and** B	
4) A **and** B **and** C	
5) A **and** (**not** B) **and** C	
6) (**not** A) **or** B	
7) A **or** B **or** C	

Β. Να βρεθεί το αποτέλεσμα των παρακάτω εκφράσεων, αν είναι A = 2, B = 3 και C = Αληθής.

1) (A > B) **and** C	
2) (A != B) **or not** (A == B) **and** C	
3) (A > 1) **and not** C	
4) (A != 3) **and** ((B - A) > 0 **or not** C)	
5) (A > 0) **and** (A <= 2)	
6) not (B == 3)	
7) (B == 3) **and** C	

Γ. Να συμπληρώσετε τον παρακάτω πίνακα, κάνοντας τις λογικές πράξεις για τις ακόλουθες τιμές των μεταβλητών (να γράψετε αναλυτικά τις πράξεις).

ΛΟΓΙΚΗ ΕΚΦΡΑΣΗ	a = 2, b = 3	a = 5, b = 3	a = 0, b = 9
1) a==2 **and** b==3			
2) a > 2 **and** b < 3			
3) a == 3 **or not** (b>2)			
4) a == b **or** a != b			
5) a == b **and** a != b			

Φύλλο εργασίας 3 (Δομή ακολουθίας)

1. Να γραφούν οι ακόλουθες αλγεβρικές εκφράσεις σε μορφή αποδεκτή από την **Python**.

$$\frac{x + 3y}{x - 2y}$$

$$x(y + z(5 - k))$$

$$\frac{xy}{x + y}$$

$$\frac{x + \frac{y}{z}}{2x}$$

2. Ποιες από τις παρακάτω εκφράσεις αποδίδουν σωστά το αποτέλεσμα της παράστασης

$$\frac{1}{7 - y} \bullet 15$$

α. 15 / (7 - y)
β. 15 / 7 - y
γ. (1 / 7 - y) * 15
δ. 1 / (7 - y) * 15
ε. 1 / ((7 - y) * 15)

3. Να αναπτύξετε ένα πρόγραμμα σε **Python** το οποίο θα:

1) διαβάζει δύο αριθμούς
2) υπολογίζει το άθροισμα, τη διαφορά, το γινόμενο και τον Μέσο Όρο των αριθμών.
3) εμφανίζει τη λέξη «Άθροισμα» ακολουθούμενη από το άθροισμα, τη λέξη «Γινόμενο» ακολουθούμενη από το γινόμενο, τη λέξη «Διαφορά» ακολουθούμενη από τη διαφορά και τη λέξη «ΜΟ» ακολουθούμενη από το ΜΟ.

4. Να αναπτύξετε ένα πρόγραμμα σε **Python** το οποίο θα:

1) διαβάζει τις ώρες εργασίας ενός ωρομίσθιου υπαλλήλου, την ωριαία αποζημίωση, και το ποσοστό κρατήσεων για ασφάλιση.

2) υπολογίζει τις καθαρές αποδοχές του υπαλλήλου και τις κρατήσεις από την ασφάλιση.

3) εμφανίζει τη λέξη «Καθαρές αποδοχές =» και τις καθαρές αποδοχές του υπαλλήλου και τη λέξη «Ασφάλιση=» ακολουθούμενη από τις κρατήσεις για την ασφάλιση.

5. Να αναπτύξετε ένα πρόγραμμα σε **Python** το οποίο θα:

1) διαβάζει δύο αριθμούς και θα τους «αποθηκεύει» σε δύο μεταβλητές **a, b.**

2) εμφανίζει για κάθε μεταβλητή το όνομα και την τιμή της

3) αντιμεταθέτει τις τιμές των μεταβλητών και θα ξαναεμφανίζει για κάθε μεταβλητή το όνομα και τη νέα τιμή της.

Φύλλο εργασίας 4 (Δομή απλής επιλογής)

1. Σε ένα πρόγραμμα, για το ΟΝΟΜΑ, το ΤΜΗΜΑ, την ΗΛΙΚΙΑ και το ΜΙΣΘΟ των υπαλλήλων μιας εταιρίας χρησιμοποιούνται αντίστοιχες μεταβλητές.

i. Για τα παραπάνω δεδομένα να γράψετε αντίστοιχα ονόματα μεταβλητών.

ii. Να γράψετε τις παρακάτω συνθήκες χρησιμοποιώντας τα ονόματα των μεταβλητών, τους σχεσιακούς τελεστές της **Python** και λογικούς τελεστές.

1) ΗΛΙΚΙΑ μεγαλύτερη από **25**
2) ΗΛΙΚΙΑ όχι μεγαλύτερη από **30** και ΜΙΣΘΟ όχι μικρότερο από **2000**
3) ΟΝΟΜΑ "ΑΝΤΩΝΙΟΥ" και δουλεύουν στο τμήμα "ΜΗΧΑΝΟΓΡΑΦΗΣΗ"
4) ΜΙΣΘΟ όχι μικρότερο από **1000** και δεν δουλεύουν στο τμήμα "ΓΡΑΜΜΑΤΕΙΑ"
5) ΜΙΣΘΟ εκτός του διαστήματος **[1000, 1500)** και ΗΛΙΚΙΑ εντός του διαστήματος **(20, 30]**.

2. Τι θα εμφανιστεί κατά την εκτέλεση του διπλανού προγράμματος αν δοθεί από το πληκτρολόγιο η τιμή:
α. 4
β. 5

```
x=input()
if x>4:
   x=x+1
print x
```

3. Τι θα εμφανιστεί κατά την εκτέλεση του διπλανού προγράμματος, αν δοθούν από το πληκτρολόγιο οι τιμές:
α. 2 και 7
β. 7 και 2

```
a,b = input(), input()
if a<b:
   a=b-a
   b=b-a
print a,b
```

4. Δίνεται το διπλανό πρόγραμμα σε **Python**. Μετά την εκτέλεση του ποιες θα είναι οι τιμές των μεταβλητών a, b, c που θα εμφανιστούν αν:
α. a=5 και b=10
β. a = 10 και b = 5

```
a, b= input(),input()
a=a-b
b=a+b
c=b
if a>c:
   c=a
print a,b,c
```

5. Δίνεται το διπλανό πρόγραμμα σε **Python**. Να παρουσιαστεί ο πίνακας τιμών των μεταβλητών και οι τιμές που θα εμφανιστούν.

```
x=2
y=pow(x,3) - 1
z=2*x+y-1
if y>abs(2*x-z):
    y=z/x
    z=x**2
x=x-1
print x,y,z
```

6. Να σχηματίσετε τον πίνακα τιμών του διπλανού προγράμματος, αν εισαχθούν οι τιμές **5,2,9,1.**

```
x,y = input(), input()
x= abs(x-y)
y=abs(x-y)
if y%x <= 3:
    x=input('Δώσε τιμή για το x: ')
    y=y + x/2
if y<x:
    x=x-y/5
    y=input(Δώσε τιμή για το y: ')
print x,y
```

7. Να αναπτύξετε πρόγραμμα σε **Python** που θα διαβάζει έναν αριθμό και θα εμφανίζει την απόλυτη τιμή του, με δύο τρόπους: **i)** με απλή επιλογή και, **ii)** με τη χρήση συνάρτησης.

8. Να αναπτυχθεί πρόγραμμα σε **Python** που θα διαβάζει τρεις αριθμούς και θα εκτυπώνει το μικρότερο από αυτούς.

9. Μια οικογένεια κατανάλωσε **X Kwh** (κιλοβατώρες) ημερήσιου ρεύματος και **Y Kwh** νυχτερινού ρεύματος. Το κόστος ημερήσιου ρεύματος είναι **0,4** ευρώ ανά **Kwh** και του νυχτερινού **0,25** ευρώ ανά **Kwh**. Να αναπτύξετε ένα πρόγραμμα σε **Python** το οποίο:

α) να διαβάζει τα X, Y

β) να υπολογίζει και να εμφανίζει το συνολικό κόστος της κατανάλωσης ρεύματος της οικογένειας

γ) να εμφανίζει το μήνυμα ΥΠΕΡΒΟΛΙΚΗ ΚΑΤΑΝΑΛΩΣΗ, αν το συνολικό κόστος είναι μεγαλύτερο από **300** ευρώ.

10. Σε τρεις πόλεις ενός νομού καταγράφηκαν την ίδια ώρα οι θερμοκρασίες θ1, θ2, θ3. Να αναπτύξετε πρόγραμμα σε **Python** που:

α) Να διαβάζει τις τρεις θερμοκρασίες.

β) Να υπολογίζει και να εμφανίζει τη μέση τιμή των παραπάνω θερμοκρασιών.

γ) Να εμφανίζει το μήνυμα "ΚΑΥΣΩΝΑΣ", αν η μέση τιμή είναι μεγαλύτερη των 38 βαθμών Κελσίου.

Φύλλο εργασίας 5 (Δομή σύνθετης επιλογής)

1. Ερωτήσεις πολλαπλής επιλογής

1. Αν μετά την εκτέλεση του διπλανού προγράμματος προκύπτει **a = 0** και **b = 3**, τι τιμές θα μπορούσαν να έχουν τα **x** και **y**;
α. x = 7, y = 2
β. x = 4, y = 3
γ. x = 3, y = 5
δ. x = 9, y = 3.

```
if (x % y < x / y) :
    a = 0
    b = 0
else:
    a = x / y
    b = x % y
```

2. Δίνεται το παρακάτω τμήμα προγράμματος:

```
if a >= 0 or a<= 20:
    print a
else:
    print "λάθος"
```

Για ποιες τιμές του **a** θα εκτυπωθεί το μήνυμα "λάθος";
α. 0 < α < 20
β. δεν θα εκτυπωθεί
γ. α < 20
δ. α > 0
ε. για όλες
στ. α < 0 και α > 20

2. Τι θα εμφανιστεί κατά την εκτέλεση του διπλανού προγράμματος σε **Python** αν δοθεί από το πληκτρολόγιο η τιμή:
α. 4
β. 7.

```
a=input()
if a>=4:
    a=a+1
else:
    a=a-1
print a
```

3. Τι θα εμφανιστεί κατά την εκτέλεση του διπλανού προγράμματος σε **Python** αν δοθούν από το πληκτρολόγιο οι τιμές:
α) 7 και 7
β) 10 και 7
γ) 7 και 10.

```
a, b=input(), input()
if a<b:
    a=b-a
else:
    a=a-b
print a,b
```

4. Τι θα εμφανιστεί κατά την εκτέλεση του διπλανού προγράμματος σε **Python** αν δοθούν από το πληκτρολόγιο οι τιμές:
α) 7 και 7
β) 10 και 15.

```
a, b=input(), input()
if a==b:
    a=a-b
else:
    a=-1
print a,b
```

5. Δίνεται το διπλανό πρόγραμμα σε **Python**. Μετά την εκτέλεση του ποιες θα είναι οι τιμές των μεταβλητών **a** και **b** που θα εμφανιστούν για:
α) a = 6 και b = 3
β) a = 4 και b = 4

```
a, b=input(), input()
if a>5:
    b=2*a+b
else:
    a=2*b+a
print a, b
```

6. Δίνεται το παρακάτω πρόγραμμα. Κάθε εντολή περιέχει ένα ή δύο κενά που το καθένα αντιστοιχεί σε σταθερά (αριθμό), μεταβλητή ή τελεστή. Επίσης δίνεται ο πίνακας τιμών όπου παρουσιάζεται το αποτέλεσμα που έχει η εκτέλεση του προγράμματος. Να συμπληρώσετε τα κενά.

Πρόγραμμα **Python**	x	y	z
1. x = _____	1		
2. _____ = x - 1		0	
3. z = _____ * x			2
4. if x ____ y :			
5. print x			
6. _____:			
7. print z			
9. ____ ← 2 * (y - ____) + z	0		
10. y ← x ____ y		0	
11. ____ ← x + z			2
12. print _____	Θα εκτυπωθεί: 1, 0,0, 2		

7. Να αναπτύξετε πρόγραμμα σε **Python** το οποίο θα διαβάζει τις δικαιολογημένες απουσίες Δ και τις αδικαιολόγητες απουσίες Α ενός μαθητή και θα εμφανίζει το μήνυμα «ΠΡΟΑΓΕΤΑΙ» αν το άθροισμα των απουσιών είναι μικρότερο από 115 ή το μήνυμα «ΑΠΟΡΡΙΠΤΕΤΑΙ» σε διαφορετική περίπτωση.

8. Να αναπτύξετε πρόγραμμα σε **Python** το οποίο θα διαβάζει το χρόνο ενός δρομέα σε δευτερόλεπτα και θα τον μετατρέπει σε ώρες, λεπτά και δευτερόλεπτα. Για παράδειγμα, αν ο χρόνος του είναι 85 δευτερόλεπτα, να εκτυπώνει: 1 λεπτό, 25 δευτερόλεπτα και όχι, 0 ώρες, 1 λεπτό, 25 δευτερόλεπτα (μόνο οι ώρες δεν πρέπει να εμφανίζονται αν είναι 0).

9. Να αναπτύξετε πρόγραμμα σε **Python** που θα διαβάζει έναν αριθμό (θεωρούμε ότι είναι θετικός) και θα εκτυπώνει μήνυμα σχετικά με το αν είναι άρτιος ή περιττός και επιπλέον, αν είναι άρτιος να εκτυπώνει το τριπλάσιο του και αν είναι περιττός να εκτυπώνει το διπλάσιο.

Φύλλο εργασίας 6 (Δομή πολλαπλής και εμφωλευμένης επιλογής)

1. Ερωτήσεις πολλαπλής επιλογής

i. Για ποιες τιμές του **a** θα εκτυπωθεί η τιμή 3;
   α. για 0<= α <= 5
   β. ποτέ δεν θα εκτυπωθεί
   γ. α> 0 ή α < 5
   δ. α < -1000.

ii. Για ποιες τιμές του **a** θα εκτυπωθεί η τιμή 2;
   α. για α > 20
   β. ποτέ δεν θα εκτυπωθεί
   γ. για α> 5
   δ. για α > 100.

```
a=input()
if a<0 or a>5:
    print 1
elif a>20:
    print 2
else:
    print 3
```

2. Τι θα εμφανιστεί κατά την εκτέλεση του διπλανού προγράμματος σε **Python** αν δοθούν από το πληκτρολόγιο οι τιμές
α) 7 και 7
β) 8 και 7
γ) 7 και 8.

```
a, b=input(), input()
if a<b:
    a=b-a
elif a>b:
    b=a-b
else:
    a = b * 2
    b = a + 2
print a,b
```

3. Δίνεται ο διπλανός αλγόριθμος σε ψευδογλώσσα.
Να γράψετε το αντίστοιχο πρόγραμμα **Python**.

```
Αλγόριθμος ΑΣΚΗΣΗ
Κ ← 23
Διάβασε Λ
Αν Κ>Λ τότε
Εμφάνισε "ΕΝΑ"
αλλιώς_αν Κ < Λ τότε
Εμφάνισε "ΔΥΟ"
αλλιώς
Εμφάνισε "ΤΡΙΑ"
Τέλος_αν
Τέλος ΑΣΚΗΣΗ
```

4. Δίνεται το διπλανό πρόγραμμα σε **Python**. Να σημειώσετε τις τιμές που θα εκτυπωθούν.

```
a =15
b = a ** 2
c = (a+b) / 2
if (c % 2 == 0) or c >=25:
    c = c - 20
    if not (c > a):
        a = b % a
    else:
        c = c / 2
else:
    a = a + c
    c = c + b
print a,b,c
a = a / b + c
b = b % c + a
c = c % a + b
print a,b,c
```

5. Να αναπτύξετε πρόγραμμα σε **Python** που θα διαβάζει έναν αριθμό **a** και θα εμφανίζει το μήνυμα «ΘΕΤΙΚΟΣ», «ΑΡΝΗΤΙΚΟΣ», «ΜΗΔΕΝ» ανάλογα με την τιμή του.

6. Να αναπτύξετε πρόγραμμα σε **Python** το οποίο θα διαβάζει δύο αριθμούς Α και Β και θα εμφανίζει το μήνυμα :
« Α ΜΕΓΑΛΥΤΕΡΟΣ ΑΠΟ Β» αν Α>Β
Ή
« Α ΜΙΚΡΟΤΕΡΟΣ ΑΠΟ Β» αν Α<Β
Ή
« Α ΙΣΟΣ ΜΕ Β» αν Α=Β

7. Να αναπτυχθεί πρόγραμμα σε **Python** που θα διαβάζει τέσσερις αριθμούς και θα εκτυπώνει το μικρότερο. Επιπλέον αν το τελευταίο του ψηφίο είναι το 0 ή το 1 ή το 2, να εμφανίζει το διπλάσιό του, αλλιώς το τριπλάσιό του.

8. Σε τρεις διαφορετικούς αγώνες πρόκρισης για την Ολυμπιάδα του Σύδνεϋ στο άλμα εις μήκος ένας αθλητής πέτυχε τις επιδόσεις a, b, c. Να αναπτύξετε πρόγραμμα σε **Python** το οποίο:

α) να διαβάζει τις τιμές των επιδόσεων a, b, c
β) να υπολογίζει και να εμφανίζει τη μέση τιμή των παραπάνω τιμών
γ) να εμφανίζει το μήνυμα «ΠΡΟΚΡΙΘΗΚΕ», αν η παραπάνω μέση τιμή είναι μεγαλύτερη των 8 μέτρων
δ) να εμφανίζει τη μεγαλύτερη επίδοση
ε) σε περίπτωση που η μεγαλύτερη επίδοση είναι μεγαλύτερη από 8.95 μέτρα να εμφανίζει το μήνυμα «ΠΑΓΚΟΣΜΙΟ ΡΕΚΟΡ».

11. Να αναπτύξετε πρόγραμμα σε **Python** το οποίο θα διαβάζει το βαθμό ενός μαθητή και θα εμφανίζει ένα μήνυμα αντίστοιχο με την επίδοσή του.

Βαθμός μαθητή	Επίδοση μαθητή
[0,9.5)	Απορρίπτεται
[9.5, 12)	Μέτρια
[12, 15)	Καλά
[15, 18.5)	Πολύ καλά
[18.5, 20]	Άριστα

Σε κάθε άλλη περίπτωση να εμφανίζει "Λάθος βαθμός" (Να μην χρησιμοποιήσετε σύνθετες λογικές εκφράσεις)

12. Μια εταιρεία ταχυδρομικών υπηρεσιών εφαρμόζει για τα έξοδα αποστολής ταχυδρομικών επιστολών εσωτερικού και εξωτερικού, χρέωση σύμφωνα με τον παρακάτω πίνακα:

Βάρος επιστολής σε γραμμάρια	Χρέωση εσωτερικού σε ευρώ	Χρέωση εξωτερικού σε ευρώ
από 0 έως και 500	2,0	4,8
από 500 έως και 1000	3,5	7,2
από 1000 έως και 2000	4,6	11,5

Για παράδειγμα τα έξοδα αποστολής μιας επιστολής βάρους 800 γραμμαρίων και προορισμού εσωτερικού είναι 3,5 ευρώ.

Να γράψετε πρόγραμμα σε **Python** το οποίο:

α. Να διαβάζει το βάρος της επιστολής.

β. Να διαβάζει τον προορισμό της επιστολής. Η τιμή «ΕΣ» δηλώνει προορισμό εσωτερικού και η τιμή «ΕΞ» δηλώνει προορισμό εξωτερικού.

γ. Να υπολογίζει τα έξοδα αποστολής ανάλογα με τον προορισμό και το βάρος της επιστολής.

δ. Να εκτυπώνει τα έξοδα αποστολής.

13. Ένα Internet Cafe έχει την ακόλουθη κλιμακωτή πολιτική χρέωσης:

Χρόνος	Χρέωση
Τα πρώτα 30 λεπτά	0,50 €
Τα επόμενα 30 λεπτά	0,40 €
Υπόλοιπος χρόνος	0,1 € ανά λεπτό

Να αναπτύξετε πρόγραμμα σε **Python** που θα διαβάζει το χρόνο που κάποιος πελάτης έκανε χρήση των υπηρεσιών του Internet Café και θα εκτυπώνει τη χρέωση.

14. Ένα τυπογραφείο χρεώνει κλιμακωτά τους πελάτες του ως εξής:

Αριθμός Βιβλίων	Χρέωση ανά βιβλίο
1-100	3 ευρώ
101-300	2.5 ευρώ
301 και πάνω	1.5 ευρώ

Να γραφεί πρόγραμμα σε **Python** που θα υπολογίζει συνολικά τι πρέπει να πληρώσει κάποιος που θέλει να εκτυπώσει κάποια βιβλία.

15. Μια εταιρεία κινητής τηλεφωνίας ακολουθεί ανά μήνα την πολιτική χρέωσης που φαίνεται στον παρακάτω πίνακα:

Πάγιο 10 ευρώ	
Χρόνος τηλεφωνημάτων (δευτερόλεπτα)	**Χρονοχρέωση** (ευρώ/δευτερόλεπτα)
1 - 500	0,007
501 - 800	0,004
801 και άνω	0,003

Η χρονοχρέωση είναι κλιμακωτή. Να αναπτύξετε πρόγραμμα σε **Python** το οποίο να:

α. διαβάζει τη χρονική διάρκεια των τηλεφωνημάτων ενός συνδρομητή σε διάστημα ενός μήνα.

β. υπολογίζει τη μηνιαία χρέωση του συνδρομητή.

γ. τυπώνει τη λέξη "ΧΡΕΩΣΗ" και τη μηνιαία χρέωση του συνδρομητή.

17. Τα ταχυδρομικά τέλη για προορισμούς εσωτερικού ("ΕΣ") και εξωτερικού ("ΕΞ") υπολογίζονται κλιμακωτά ως εξής:

Βάρος φακέλων	Χρέωση "ΕΣ"	Χρέωση "ΕΞ"
1 - 20 γραμμάρια	0,05 € / γραμμάριο	0,07 € / γραμμάριο
21 - 150 γραμμάρια	0,12 € / γραμμάριο	0,14 € / γραμμάριο
151 και πάνω	0,01 € / γραμμάριο	0,03 € / γραμμάριο

Στις παραπάνω τιμές υπάρχει προσαύξηση ΦΠΑ 23%. Να αναπτύξετε πρόγραμμα σε **Python** που θα διαβάζει το βάρος ενός φακέλου, τον προορισμό του και θα εμφανίζει τη χρέωση.

13.3 Βασικές Αναφορές

(Η τελευταία προσπέλαση στις ηλεκτρονικές πηγές έγινε τον Οκτώβριο του 2016)

Python Tutorial
http://python-tutorial-greek.readthedocs.org/en/latest/oop_general.html

Tutorial από το δικτυακό τόπο της γλώσσας Python για την γνωριμία με τη γλώσσα Python - Δομή Ελέγχου και Δομές Επανάληψης https://docs.Python.org/2/tutorial/controlflow.htm

Αβούρης Ν. (2000) «Εισαγωγή στην επικοινωνία ανθρώπου υπολογιστή», εκδ. Δίαυλος.

ΑΠΣ_Τομέα_Πληρ_ΕΠΑΛ (2015). Αναλυτικά Προγράμματα Σπουδών του μαθήματος Γενικής Παιδείας «Εισαγωγή στις Αρχές της Επιστήμης των Η/Υ» της Β' και Γ' τάξης Ημερήσιων και Γ' και Δ' τάξης Εσπερινών ΕΠΑ.Λ. και των μαθημάτων ειδικοτήτων του Τομέα Πληροφορικής της Ομάδας Προσανατολισμού Τεχνολογικών Εφαρμογών των τάξεων Β' και Γ Ημερήσιων και Β', Γ και Δ' Εσπερινών ΕΠΑ.Λ., ΦΕΚ 2010. 16/9/2015.

Αρχές *Προγραμματισμού Υπολογιστών*, (2015). Διδακτικό υλικό των Αράπογλου Α., Βραχνός Ε., Κανίδης Ε., Μακρυγιάννης Π., Μπελεσιώτης Β., Τζήμας Δ, ISBN 978-960-06-5141-6

ΒΙΚΙΠΑΙΔΕΙΑ Αντικειμενοστρεφής Προγραμματισμός
https://el.wikipedia.org/wiki/Αντικειμενοστρεφής_προγραμματισμός

Βικιπαίδεια, υλικό και χρήσιμοι σύνδεσμοι, με έλεγχο ποιότητας,
http://el.wikipedia.org/wiki/Python

Δικτυακός κόμβος υποστήριξης με πλούσιο υλικό πολυμέσων για τη διδασκαλία της γλώσσας Python, http://www.Pythonschool.net/

Δικτυακός κόμβος υποστήριξης της γλώσσας Python, https://www.Python.org/

Δικτυακός κόμβος υποστήριξης της γλώσσας Python,
http://www.pythonschool.net και ειδικότερα των Βάσεων Δεδομένων
http://www.pythonschool.net/category/databases.html

Δικτυακός τόπος εκπαίδευσης στη γλώσσα Python:
https://www.codecademy.com/learn/python

Δικτυακός τόπος με πλούσιο σχετικό υλικό http://www.python- course.eu/course.php

Οδηγός για τον Εκπαιδευτικό για το Πρόγραμμα Σπουδών του Μαθήματος «Πληροφορική» Γ’ Τάξης Γενικού Λυκείου, στο πλαίσιο του έργου «ΝΕΟ ΣΧΟΛΕΙΟ (Σχολείο 21ου αιώνα) - Νέο Πρόγραμμα Σπουδών», Υποέργο 9: «Εκπόνηση Προγραμμάτων Σπουδών Γενικού Λυκείου, Μουσικών και Καλλιτεχνικών Λυκείων», Υ.ΠΟ.ΠΑΙ.Θ, Ινστιτούτο Εκπαιδευτικής Πολιτικής (Ι.Ε.Π), Ιανουάριος 2015.

Οδηγός εκπαιδευτικού - Κεφάλαιο 1, ΈΡΓΟ «Νέο Σχολείο (Σχολείο 21ου αιώνα) - Νέο Πρόγραμμα Σπουδών» (κωδ. ΟΠΣ: 295450/Υποέργο 9/ Δράση: Εκπόνηση των Προγραμμάτων Σπουδών και Οδηγού Εκπαιδευτικού Μαθήματος «Πληροφορική» Γ' τάξης Γενικού Λυκείου - ΦΕΚ 189/23-1 - 2015).

ΠΣ_Γ_ΓΕΛ (2015). Πρόγραμμα Σπουδών και Οδηγός Εκπαιδευτικού Μαθήματος «Πληροφορική» Γ' τάξης Γενικού Λυκείου (ΦΕΚ189/23-1 -2015) (ΠΣ_Γ_ΓΕΛ, 2015). *Έργο ΙΕΠ «Νέο Σχολείο (Σχολείο 21ου αιώνα) - Νέο Πρόγραμμα Σπουδών» (κωδ. ΟΠΣ: 295450/0ριζόντια Πράξη στις 8 Π.Σ., 3 Π.Στ. Εξ., 2 Π.Στ. Εισ./Υποέργο 9 Δράση: Εκπόνησης των Προγραμμάτων Σπουδών και Οδηγού Εκπαιδευτικού Μαθήματος «Πληροφορική» Γ' τάξης Γενικού Λυκείου. Μέλη (αξιολογητές, συντονιστής, εμπειρογνώμονες): (εδώ αλφ/κά-αναλυτικά στο κάθε υλικό/παραδοτέα): Αράπογλου Αριστείδης, εκπαιδευτικός ΠΕ19, υπ. ΚΕ.ΠΛΗ.ΝΕ.Τ - Βαρζάκας Παναγιώτης, μέλος Ε.Π. -Α.Τ.Ε.Ι. - Βραχνός Ευριπίδης, εκπαιδευτικός ΠΕ19 - Κανίδης Ευάγγελος, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής - Λέκκα Δήμητρα, εκπαιδευτικός ΠΕ19 - Μαραγκός Κωνσταντίνος, εκπαιδευτικός ΠΕ19 - Μαυρίδης Ιωάννης, μέλος ΔΕΠ - Μπελεσιώτης Βασίλειος, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής - Παπαδάκης Σπυρίδων, Σχολικός Σύμβουλος ΠΕ19-Πληροφορικής - Τζήμας Δημήτριος, εκπαιδευτικός ΠΕ19.*

Συγγραφείς_Προγραμματισμός. (2016). Ομάδες συγγραφής διδακτικού υλικού μαθήματος Προγραμματισμός, Γ' ΕΠΑ.Λ., Τομέας Πληροφορικής, ΑΔΑ ΩΓΣΝΟΞΛΔ-Τ6Ψ και 6ΥΓΖΟΞΛΔ- ΛΨΥ, διαθέσιμες στο https://et.diavgeia.gov.gr/

Ωρολόγιο Πρόγραμμα (2015). των μαθημάτων Γενικής Παιδείας της Α' τάξης Εσπερινών ΕΠΑ.Λ. και των Β', Γ' τάξεων Ημερησίων και Α', Β' και Γ' τάξεων Εσπερινών ΕΠΑ.Λ. ανά Ειδικότητα Τομέα Ομάδας Προσανατολισμού, ΦΕΚ1053/2015.

Βάσει του ν. 3966/2011 τα διδακτικά βιβλία του Δημοτικού, του Γυμνασίου, του Λυκείου, των ΕΠΑ.Λ. και των ΕΠΑ.Σ. τυπώνονται από το ΙΤΥΕ - ΔΙΟΦΑΝΤΟΣ και διανέμονται δωρεάν στα Δημόσια Σχολεία. Τα βιβλία μπορεί να διατίθενται προς πώληση, όταν φέρουν στη δεξιά κάτω γωνία του εμπροσθόφυλλου ένδειξη «ΔΙΑΤΙΘΕΤΑΙ ΜΕ ΤΙΜΗ ΠΩΛΗΣΗΣ». Κάθε αντίτυπο που διατίθεται προς πώληση και δεν φέρει την παραπάνω ένδειξη θεωρείται κλεψίτυπο και ο παραβάτης διώκεται σύμφωνα με τις διατάξεις του άρθρου 7 του νόμου 1129 της 15/21 Μαρτίου 1946 (ΦΕΚ 1946,108, Α').

Κωδικός Βιβλίου: 0-24-0602
ISBN 978-960-06-5889-7