Νίκος Μ. Χατζηγιαννάκης

Η γλώσσα Python σε βάθος

Περιλαμβάνει εισαγωγή
στην επιστήμη των υπολογιστών
και τον προγραμματισμό

Απαντήσεις ασκήσεων

ΚΛΕΙΔΑΡΙΘΜΟΣ

Ασκήσεις Κεφαλαίου 1

1.1

☐ Ένα μηχάνημα που μας λύνει μαθηματικά προβλήματα.
☐ Ένα μηχάνημα που σκέφτεται με μεγάλη ταχύτητα.
☑ Ένα μηχάνημα που εκτελεί απλές πράξεις αλλά με μεγάλη ταχύτητα.

1.2

Η επιστήμη των Η/Υ χωρίζεται σε δύο μεγάλα πεδία, το πεδίο του υλικού (hardware) και το πεδίο του λογισμικού (software).

- Ο τομέας του υλικού ασχολείται με τον εξοπλισμό των Η/Υ, δηλαδή με τη σχεδίαση των ηλεκτρονικών κυκλωμάτων και των περιφερειακών μονάδων που τον υποστηρίζουν.
- Το πεδίο του λογισμικού (software) ασχολείται με τον σχεδιασμό και την κατασκευή των προγραμμάτων που καθορίζουν τη λειτουργία του Η/Υ.

1.3

Τις μετατρέπει σε 1 και 0.

1.4

- Το bit είναι μια μονάδα πληροφορίας η οποία μπορεί να λάβει μόνο δύο τιμές. Την τιμή 1 ή την τιμή 0.
- Ένας συνδυασμός 8 bit ονομάζεται **byte** και μπορεί να δημιουργήσει 256 διαφορετικούς συνδυασμούς τού 0 και τού 1.

1.5

Με 4 bit είναι 16 (2^4) και με ένα byte, το οποίο αποτελείται από 8 bit, 256 (2^8).

1.6

Ας υποθέσουμε ότι έχουμε ένα στύλο και μία σημαία την οποία τοποθετούμε στο στύλο σε διαφορετικά ύψη. Ανάλογα με τη θέση της σημαίας στο στύλο μπορούμε να στείλουμε κάποιο μήνυμα. Π.χ. αν είναι στην κορυφή του στύλου να σημαίνει κάτι, αν είναι στο μέσον του κάτι άλλο και αν είναι στο κάτω μέρος κάτι διαφορετικό. Θεωρητικά, οι θέσεις που μπορεί να πάρει η σημαία στο στύλο είναι άπειρες. Αυτός ο τρόπος επικοινωνίας αντιστοιχεί στην αναλογική επικοινωνία. Αν τώρα αποφασίσουμε ότι ο στύλος μας ή θα έχει σημαία ή δεν θα έχει, δηλαδή μόνο δύο καταστάσεις, τότε μιλάμε για ψηφιακή επικοινωνία.

1.7

Ένα Petabyte (PB) = 1,000,000 GB

1.8

Τους μετατρέπει σε αριθμούς, σύμφωνα με κάποιο πρότυπο κωδικοποίησης, και τελικά σε ένα ή περισσότερα byte.

1.9

Έξι, ένα για κάθε χαρακτήρα.

1.10

Είναι το πλήθος των bit στο οποίο μετατρέπεται κάθε pixel της εικόνας.

1.11

Είναι 200x300x4 = 240,000 byte (τα 32 bit είναι 4 byte).

1.12

Είναι 96,000 x 120 x 3 = 34,560,000 byte = 34.56 MB. (96,000 δειγματοληψίες/sec * 120 sec * 3 byte (24bit).

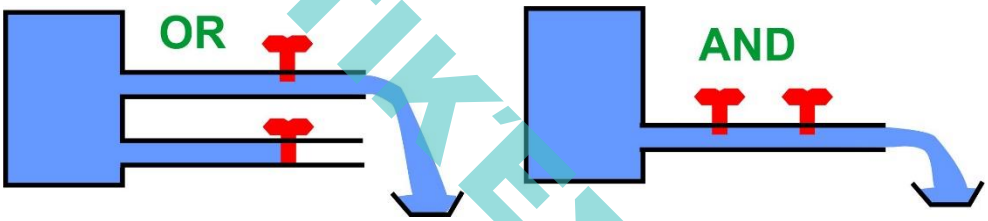
1.13

Εννοούμε τη δυνατότητα αναπαράστασης της πληροφορίας με λιγότερα byte χρησιμοποιώντας ειδικούς μαθηματικούς αλγορίθμους, με ή χωρίς απώλεια πληροφορίας.

1.14

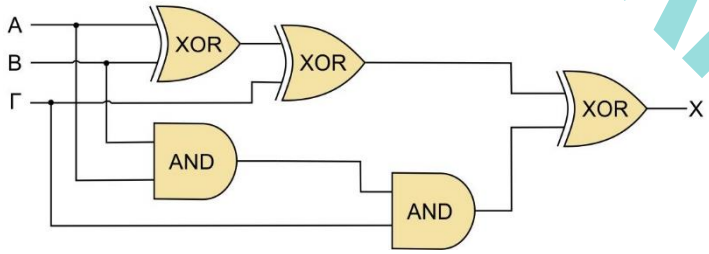
Μια λογική πύλη είναι ένα κύκλωμα το οποίο πραγματοποιεί μια λογική πράξη στις εισόδους του και παράγει μια έξοδο. Οι βασικές λογικές πύλες είναι τέσσερις: AND, OR, XOR και NOT. Οι πράξεις σε επίπεδο δυαδικών αριθμών πραγματοποιούνται με διάφορους συνδυασμούς αυτών των πυλών.

1.15



- Στη πρώτη περίπτωση (OR) αρκεί μια από τις δύο βρύσες να είναι ανοιχτή για να τρέξει νερό στο δοχείο.
- Στη πρώτη περίπτωση (AND) πρέπει και οι δύο βρύσες να είναι ανοιχτές για να τρέξει νερό στο δοχείο.

1.16



1.17

- Για την μόνιμη αποθήκευση των πληροφοριών
- Για τον μεγάλη χωρητικότητα
- Για τη μεταφορά πληροφοριών
- Για την τήρηση αντιγράφων ασφαλείας

1.18

Ότι χωράει 500 δισεκατομμύρια bytes. Δηλαδή ένα απλό κείμενο με 500,000,000,000 χαρακτήρες!!!

Ασκήσεις Κεφαλαίου 2

2.1

☐ Πρόβλημα είναι κάτι που δεν επιδέχεται λύσης.
☐ Η λύση ενός προβλήματος είναι πάντα γνωστή.
☑ Ένα πρόβλημα μπορεί να μην έχει λύση.
☑ Η ανάλυση ενός προβλήματος αναδεικνύει τη δομή του.
☑ Τα στάδια αντιμετώπισης ενός προβλήματος είναι, με τη σειρά, τα εξής: Ανάλυση-Κατανόηση-Επίλυση.
☑ Η επεξεργασία δεδομένων αποδίδει πληροφορίες.

2.2

☑ Ένας αλγόριθμος είναι η περιγραφή της λύσης ενός προβλήματος.
☐ Κάθε αλγόριθμος μπορεί να περιγραφεί με κώδικα.
☐ Ένας αλγόριθμος πρέπει κάποια στιγμή να σταματάει.
☑ Τα βήματα ενός αλγορίθμου πρέπει να είναι καθορισμένα.
☑ Ο αλγόριθμος «Μέτρα τα κόκκινα αυτοκίνητα που περνάνε έξω από το παράθυρο» είναι λανθασμένος.

2.3

19	*Τύπου int (ακέραιος αριθμός)*
12.7	*Τύπου float (πραγματικός αριθμός)*
1.0	*Τύπου float (πραγματικός αριθμός)*
"Python"	*Τύπου string (συμβολοσειρά - αλφαριθμητικό)*
'A'	*Τύπου char ή τύπου string (ανάλογα με τη γλώσσα)*
true	*Τύπου bool (λογική τιμή)*
'false'	*Τύπου string (συμβολοσειρά - αλφαριθμητικό)*
"123"	*Τύπου string (συμβολοσειρά - αλφαριθμητικό)*
""	*Τύπου string (συμβολοσειρά - αλφαριθμητικό)*

2.4

*Στις γλώσσες στατικού τύπου πρέπει να **δηλώνεται** ο τύπος μιας μεταβλητής ενώ αυτό δεν είναι απαραίτητο στις γλώσσες **δυναμικού** τύπου. Στις γλώσσες στατικού τύπου η δέσμευση μνήμης γίνεται στο στάδιο της **μεταγλώττισης** ενώ στις γλώσσες δυναμικού τύπου στο στάδιο της **εκτέλεσης**.*

2.5

☑ Πρέπει οπωσδήποτε να δηλώσουμε στο πρόγραμμά μας τον τύπο κάθε μεταβλητής που χρησιμοποιούμε.
☑ Η μεταβλητή δημιουργείται και δεσμεύει χώρο στη μνήμη κατά το στάδιο της μεταγλώττισης.
☑ Ο τύπος μιας μεταβλητής δεν μπορεί να μεταβληθεί κατά τη διάρκεια της εκτέλεσης του προγράμματος.
☑ Ο χώρος μνήμης που δεσμεύει μια μεταβλητή είναι συγκεκριμένος και καθορίζεται από τον τύπο της.
☐ Ο χώρος μνήμης που δεσμεύει μια μεταβλητή μπορεί να τροποποιηθεί κατά την εκτέλεση του προγράμματος.
☐ Η Python είναι μια γλώσσα στατικού τύπου.

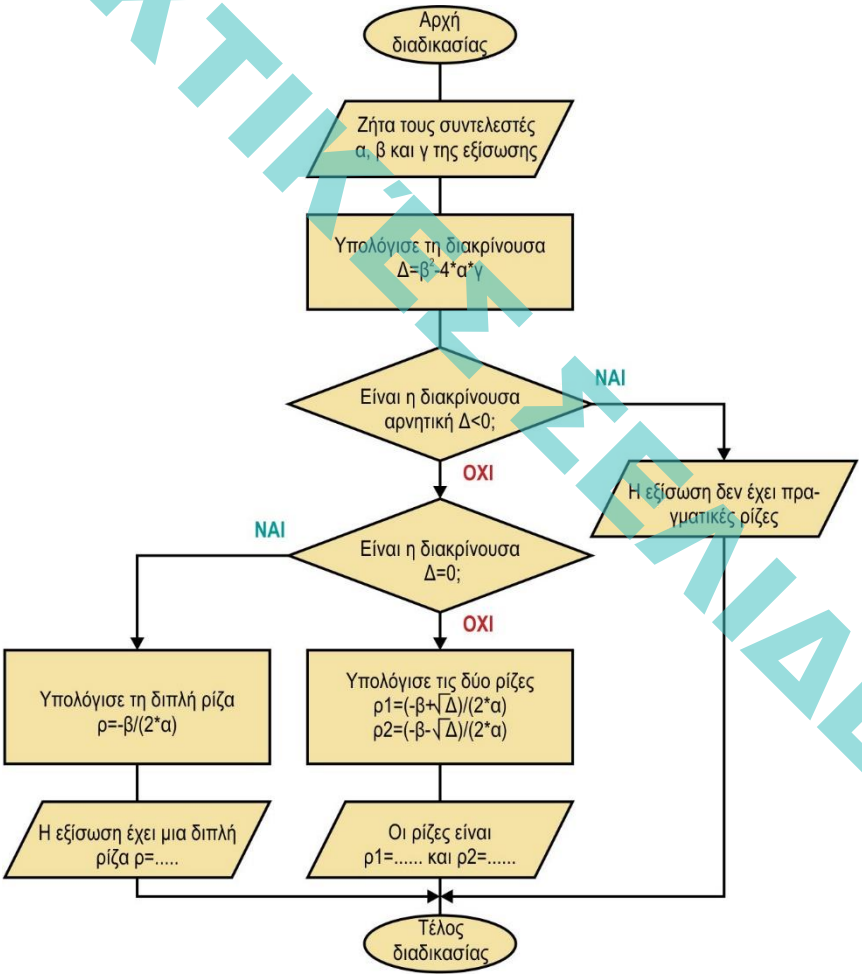
2.6

☑ Ο τύπος μιας μεταβλητής μπορεί να αλλάζει κατά την εκτέλεση του προγράμματος.
☐ Η μεταβλητή δημιουργείται και δεσμεύει χώρο στη μνήμη κατά το στάδιο της μεταγλώττισης.
☑ Ο τύπος μιας μεταβλητής καθορίζεται από τον τύπο της τιμής στην οποία αναφέρεται.
☑ Δεν χρειάζεται να δηλωθεί μια μεταβλητή, όπως γίνεται στις γλώσσες στατικού τύπου.
☑ Η μεταβλητή δημιουργείται δυναμικά την πρώτη φορά που θα της ανατεθεί κάποια τιμή.
☑ Η Python είναι μια γλώσσα δυναμικού τύπου.

2.7

☑ Η ανάλυση είναι το πιο σημαντικό στάδιο στον κύκλο ζωής ενός προγράμματος.
☐ Στο στάδιο του σχεδιασμού μιας εφαρμογής πρέπει να ακολουθηθούν πιστά τα βήματα που προέκυψαν κατά το στάδιο της ανάλυσης.
☑ Στο στάδιο της ανάπτυξης μιας εφαρμογής πρέπει να ακολουθηθούν πιστά τα βήματα που προέκυψαν κατά το στάδιο του σχεδιασμού.
☑ Η δοκιμή μιας εφαρμογής μπορεί να έχει ως αποτέλεσμα την επανεκκίνηση ενός νέου κύκλου ζωής του προγράμματος.
☑ Στο στάδιο της συντήρησης μπορούν να εντοπισθούν λογικά σφάλματα και να επιδιορθωθούν.

2.8



Ασκήσεις Κεφαλαίου 6

6.1

☐ Ο βρόχος while εκτελείται τουλάχιστον μία φορά.
☑ Στην εντολή while ο έλεγχος της συνθήκης γίνεται πριν από την εκτέλεση των προτάσεων του βρόχου.
☑ Η συνθήκη της while μπορεί να είναι οποιαδήποτε παράσταση επιστρέφει τιμή.
☑ Η πρόταση while 1: έχει ως αποτέλεσμα τη συνεχή εκτέλεση ενός βρόχου.
☑ Η Python διαθέτει μόνο δύο εντολές επανάληψης.
☑ Η εντολή while χρησιμοποιείται συνήθως όταν δεν γνωρίζουμε το πλήθος των επαναλήψεων.

6.2

```
g=1
a=1
ar=int(input('Δώσε έναν ακέραιο αριθμό:'))
while a<=ar:
   g=g*a
   a+=1
print(g)
```

Η αρχική τιμή της μεταβλητής **g** πρέπει να είναι 1.

Κάθε φορά στη μεταβλητή **g** ανατίθεται ως τιμή το γινόμενο της τιμής που ήδη έχει επί της νέας τιμής της μεταβλητής **a**.

Η τιμή της μεταβλητής **a** αυξάνεται κατά 1.

➢ Το πρόγραμμα ζητάει έναν ακέραιο αριθμό από το χρήστη και υπολογίζει το γινόμενο των αριθμών από το 1 μέχρι τον αριθμό που έδωσε. Για παράδειγμα αν ο χρήστης εισάγει τον αριθμό 5 το πρόγραμμα θα εμφανίσει το 120 (1*2*3*4*5).
➢ Αυτή είναι η μαθηματική έννοια του παραγοντικού (!). 5!=1*2*3*4*5.

6.3

```
ar=int(input('Δώσε έναν ακέραιο αριθμό:'))
z=m=0
while ar!=0:
   if ar%2==0:
      z+=1
   else:
      m+=1
   ar=int(input('Δώσε έναν ακέραιο αριθμό:'))
print(z,m)
```

Η μεταβλητή **z** αυξάνεται κατά 1 όταν ο αριθμός **ar** είναι ζυγός.

Η μεταβλητή **m** αυξάνεται κατά 1 όταν ο αριθμός **ar** είναι μονός.

➢ Το πρόγραμμα ζητάει συνέχεια ακέραιους αριθμούς και σταματάει όταν δοθεί ο αριθμός 0. Στο τέλος θα εμφανίσει το πλήθος των ζυγών και το πλήθος των μονών αριθμών.
➢ Η μεταβλητή m χρησιμοποιείται για την καταμέτρηση των μονών αριθμών, ενώ η μεταβλητή z χρησιμοποιείται για την καταμέτρηση των ζυγών αριθμών.

6.4

```
t=(12,15,5,8,10)
i=s=0
while i<len(t):
   s=s+t[i]
   i=i+1
print(s)
```

Η επαναληπτική λειτουργία θα σταματήσει όταν η τιμή της μεταβλητής **i** γίνει 5.

```
50
```

➢ Η συνάρτηση len(t) επιστρέφει το πλήθος των στοιχείων της πλειάδας t, δηλαδή το 5. Επομένως στη μεταβλητή i θα ανατεθούν οι τιμές από 0 μέχρι 4.
➢ Με την πρόταση s=s+t[i] θα προστεθούν στη μεταβλητή s ένας προς έναν οι αριθμοί της πλειάδας t. Στο τέλος η τιμή της μεταβλητής s θα είναι 50 (12+15+5+8+10) και θα εμφανιστεί στην οθόνη.

6.5

```
i=1
s=0
while i<=1000:
   s=s+i
   i=i+1
print(s)
```

Η μεταβλητή **i** θα πάρει τιμές από το 1 μέχρι το 1000. Οι τιμές αυτές προστίθενται στη μεταβλητή **s**.

```
500500
```

- Στη μεταβλητή i θα ανατεθούν οι τιμές από 0 μέχρι 4. Στη μεταβλητή s θα προστεθούν μια προς μια οι τιμές της μεταβλητής i .
- Στο τέλος η μεταβλητή s θα περιέχει το άθροισμα των αριθμών 1+2+3+4+….+1000.

6.6

☐ Ο βρόχος for εκτελείται τουλάχιστον μία φορά.
☑ Στην εντολή for χρησιμοποιείται πάντα μία μεταβλητή.
☐ Η τιμές που ανατίθενται στη μεταβλητή της for είναι πάντα αριθμητικές.
☑ Συνήθως η εντολή for χρησιμοποιείται με ένα αντικείμενο περιοχής τιμών.
☐ Η πρόταση for i in range(1,5): έχει ως αποτέλεσμα την εκτέλεση του βρόχου πέντε φορές.
☑ Η εντολή for χρησιμοποιείται όταν το πλήθος των επαναλήψεων είναι συγκεκριμένο.

6.7

```
ar=kp=0
for i in range(10):
   num=float(input())
   if num>=8.5:
      ar+=1
   elif num<5:
      kp+=1
print('Αριστούχοι={} Κόπηκαν={}'.format(ar,kp))
```

- Το πρόγραμμα ζητάει 10 πραγματικούς αριθμούς (π.χ βαθμούς). Στο τέλος θα εμφανίσει το πλήθος των αριθμών που είναι μεγαλύτεροι ή ίσοι από το 8.5 και το πλήθος των αριθμών που είναι μικρότεροι του 5.
- Η μεταβλητή ar χρησιμοποιείται για την καταμέτρηση των αριθμών >=8.5, ενώ η μεταβλητή kp χρησιμοποιείται για την καταμέτρηση των αριθμών <5.

6.8

Obj	Φορές
range(20)	*20*
range(1,20)	*19*
range(20,5,-5)	*3*
range(5,20,-5)	*0*
'Μυτιλήνη'	*8*
('Ψέριμος','Ανάφη','Ηρακλειά','Δονούσα')	*4*
{1,2,3,4,5}	*5*
[(1,2,3),(4,5),(6,7),(8,9)]	*4*
{'Χίος':12,'Λέσβος':44,'Λήμνος':89}	*3*
(123,'Μαρία',5,True,4.67)	*5*

6.9

```
z=0
for a in ('Ψέριμος','Ανάφη','Ηρακλειά','Δονούσα'):
   print(a)
   z=z+len(a)
print(z)
```

```
Ψέριμος
Ανάφη
Ηρακλειά
Δονούσα
26
```

- Στη μεταβλητή a θα ανατεθούν ένα προς ένα τα στοιχεία της πλειάδας, δηλαδή τα ονόματα των τεσσάρων νησιών του Αιγαίου. Επομένως, θα εμφανιστούν στην οθόνη αυτά τα ονόματα.
- Στη μεταβλητή z προστίθεται κάθε φορά το πλήθος των χαρακτήρων κάθε ονόματος. Στο τέλος η τιμή της μεταβλητής z θα είναι 26, όσο το σύνολο των χαρακτήρων των ονομάτων και των τεσσάρων νησιών.

6.10

```
vathmoi=[5.5,4,8,6.5,3,9.5,8.5,4,5,7]
k=s=0
for i in vathmoi:
   s=s+i
   if i<5:
      k+=1
print('Μέσος όρος=',s/10,'Κόπηκαν:',k)
```

- Στη μεταβλητή i θα ανατεθούν ένας προς έναν οι βαθμοί της λίστας vathmoi.
- Στη μεταβλητή s προστίθενται όλοι οι βαθμοί. Η μεταβλητή k αυξάνεται κατά 1 όταν ο βαθμός είναι μικρότερος του 5.

6.11

```
# Εμφάνιση χαρακτήρων στην ίδια γραμμή
for i in range(32,128):
   print(chr(i),end='')
```

Στη μεταβλητή **i** θα ανατεθούν τιμές από 32 μέχρι και 127.

Η συνάρτηση **chr(i)** επιστρέφει ως τιμή τον χαρακτήρα με κωδικό την τιμή της μεταβλητής **i**.

6.11b

```
# Διαφορετική λύση της άσκησης 6.11 με αλλαγή γραμμής κάθε 10 χαρακτήρες
for i in range(32,128):
   print(chr(i),end='')
   if (i-31)%10==0:
      print()
```

Στην περίπτωση που η μεταβλητή **i** πάρει τιμές 41,51,61,71,..., δηλαδή όταν η τιμή (i-31) είναι πολλαπλάσια του 10, αλλάζει γραμμή!

6.12

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

Στη μεταβλητή **i** θα ανατεθούν τιμές από το 1 μέχρι και το 100.

Στη μεταβλητή **s** προστίθενται όλα τα κλάσματα 1/i.

6.13

```
for x in range(-20,21,2):
   f=x**4-10*x**3+3
   print('f({})={}'.format(x,f))
```

Στη μεταβλητή **x** θα ανατεθούν τιμές από το -20 μέχρι και το 20 με βήμα 2.

Στη μεταβλητή **f** ανατίθεται η τιμή της παράστασης της εκφώνησης

6.14

```
for i in range(1,10):
   for j in range(1,i+1):
      print(j,end='')
   print()
```

```
1
12
123
1234
12345
123456
1234567
12345678
123456789
```

- Ο εσωτερικός βρόχος εκτελείται για τιμές του j από το 0 μέχρι το i (το οποίο καθορίζεται από τον εξωτερικό βρόχο).
- Επομένως, την πρώτη φορά που το i είναι 1 ο εσωτερικός βρόχος θα εμφανίσει μόνο το 1. Τη δεύτερη φορά που το i είναι 2 ο εσωτερικός βρόχος θα εμφανίσει το 1 και το 2. Την τρίτη φορά θα εμφανίσει το 1, το 2, και το 3, κ.ο.κ. Την τελευταία φορά, που το i θα είναι 9, ο εσωτερικός βρόχος θα εμφανίσει τους αριθμούς από το 1 μέχρι το 9.

6.15

```
k=4
for i in range(1,11,2):
   k+=1
   for j in range(1,5):
      k+=2
print('k=',k)
```

```
k= 49
```

- Ο εξωτερικός βρόχος θα εκτελεστεί πέντε φορές (όταν το i παίρνει τις τιμές 1, 3, 5, 7, και 9), ενώ ο εσωτερικός τέσσερις (όταν το j παίρνει τις τιμές 1, 2, 3, 4).
- Η πρόταση k+=1 θα εκτελεστεί πέντε φορές (επειδή ανήκει μόνο στον εξωτερικό βρόχο), ενώ η k+=2 είκοσι (5*4) φορές (επειδή ανήκει και στους δύο βρόχους).
- Κάθε φορά που εκτελείται η πρόταση k+=1, η μεταβλητή k αυξάνεται κατά 1, οπότε τελικά θα αυξηθεί κατά 5 αφού θα εκτελεστεί πέντε φορές. Κάθε φορά που εκτελείται η πρόταση k+=2, η k αυξάνεται κατά 2, οπότε τελικά θα αυξηθεί κατά 40 αφού θα εκτελεστεί είκοσι φορές.
- Η τελική τιμή του k θα είναι 4 + 5 + 40 = 49 (το 4 είναι η αρχική τιμή της k).

6.16

```
import math
s=0
for i in range(1,1000):
   s=s+1/(math.sqrt(i)+math.sqrt(i+1))
print('s=',s)
```

- Στη μεταβλητή i θα ανατεθούν οι τιμές από το 1 μέχρι το 999.

6.17 Να γραφεί πρόγραμμα το οποίο θα εμφανίζει στην οθόνη όλους τους χαρακτήρες από

- ☑ Η εντολές break και continue έχουν νόημα ύπαρξης μόνο μέσα στο σώμα ενός βρόχου while ή for.
- ☐ Η εντολή break διακόπτει την εκτέλεση του προγράμματος.
- ☑ Η εντολή continue παρακάμπτει τις επόμενες προτάσεις του βρόχου και συνεχίζει με την επόμενη επανάληψή του.
- ☐ Η εντολή continue έχει ως αποτέλεσμα τη συνεχή εκτέλεση ενός βρόχου while ή for.
- ☑ Στο σώμα ενός βρόχου while True: πρέπει να υπάρχει οπωσδήποτε μια εντολή break.

6.18

```
import random
a=1
while a!=0:
   a=random.randint(-100,100)
   if a>=50:
      continue
   print(a)
```

Η επαναληπτική διαδικασία θα σταματήσει όταν η στη μεταβλητή **a** ανατεθεί η τιμή 0.

Αν ο αριθμός είναι μεγαλύτερος ή ίσος με το 50, τότε η επόμενη **print()** παρακάμπτεται και η διαδικασία επαναλαμβάνεται.

- Στη μεταβλητή a ανατίθεται ένας τυχαίος αριθμός από το -100 έως το +100.
- Θα σταματήσει όταν ο τυχαίος αριθμός είναι 0.
- Ο τελευταίος αριθμός που θα δούμε είναι το 0. Δεν θα δούμε ποτέ αριθμούς μεγαλύτερους ή ίσους από το 50.

6.19

```
.....
for i in range(1,6):
   for j in range(1,6):
      if (i+j)%2==0: continue
      if i*j>=10: break
      print(i,j)
   print('=======')
print('Τέλος')
```

```
1 2
1 4
=======
2 1
2 3
=======
3 2
=======
4 1
=======
=======
Τέλος
```

6.20

```
fr=input('Δώσε μια φράση:')
for i in fr:
   if i in '0123456789':
      print(i,end='')
```

- Στη μεταβλητή i ανατίθενται ένας προς έναν οι χαρακτήρες της φράσης fr που πληκτρολόγησε ο χρήστης.
- Στην εντολή if ελέγχεται αν ο χαρακτήρας που έχει ανατεθεί ως τιμή του i, βρίσκεται μέσα στην κυριολεκτική συμβολοσειρά '0123456789'. Αν βρίσκεται τότε εμφανίζεται στην οθόνη, διαφορετικά όχι!

6.21

```
for i in range(1,101):
   print('{:4}'.format(i),end='')
   if i%10==0:
      print()
```

- Στη μεταβλητή i ανατίθενται οι ακέραιοι αριθμοί από το 1 μέχρι το 100.
- Στη συνέχεια, με την συνάρτηση print(), εμφανίζεται ο κάθε αριθμός σε τέσσερις θέσεις στην οθόνη χωρίς αλλαγή γραμμής.
- Στην εντολή if ελέγχεται αν η τιμή του i είναι πολλαπλάσιο του 10 και αν είναι αλλάζει γραμμή.

6.22

```
import math
for a in range(-10,11):
   for b in range(-10,11):
      if a==0 or b==0:
         continue
```

Στην περίπτωση που ένας από τους δύο συντελεστές είναι 0, συνεχίζουμε στην επόμενη επανάληψη.

```
d=b*b-4*a*3
if d>=0:
   r1=(-b+math.sqrt(d))/(2*a)
   r2=(-b-math.sqrt(d))/(2*a)
   print('a={} b={} r1={}  r2={}'.format(a,b,r1,r2))
else:
   print('a={} b={} δεν έχει πραγματικές ρίζες'.format(a,b))
```

Αν η διακρίνουσα είναι μεγαλύτερη από ή ίση με το 0 υπάρχουν πραγματικές ρίζες, διαφορετικά δεν υπάρχουν.

6.23

```
# Με ένθετους βρόχους for
for i in range(1,6):
   for j in range(1,i+1):
      print('*',end='')
   print()
```

Στη μεταβλητή **i** ανατίθενται οι αριθμοί από το 1 μέχρι το 5.

Στη μεταβλητή **j** ανατίθενται οι αριθμοί από το 1 μέχρι την τιμή της **i**, οπότε η πρόταση **print(...)** θα εκτελεστεί τόσες φορές όσες η τιμή της μεταβλητής **i**.

Επιβάλει αλλαγή γραμμής.

6.23b

```
# Διαφορετική λύση της άσκησης 6.23 με έναν μόνο βρόχο for
for i in range(1,6):
   print(i*'*')
```

Εμφανίζει τόσους αστερίσκους όση είναι η τιμή του **i**.

6.24

```
# Με ένθετους βρόχους for
gr=int(input('Δώσε το ύψος του ορθογωνίου :'))
st=int(input('Δώσε το πλάτος του ορθογωνίου :'))
for i in range(st):
   print('*',end='')
print()
for i in range(gr-2):
   print('*',end='')
   for j in range(st-2):
      print(' ',end='')
   print('*')
for i in range(st):
   print('*',end='')
```

Εμφανίζει την πρώτη γραμμή με αστερίσκους.

Εμφανίζει τις ενδιάμεσες γραμμές.

Εμφανίζει την τελευταία γραμμή με αστερίσκους.

6.24b

```
# Διαφορετική λύση της άσκησης 6.24 με έναν μόνο βρόχο for
gr=int(input('Δώσε το ύψος του ορθογωνίου :'))
st=int(input('Δώσε το πλάτος του ορθογωνίου :'))
print(st*'*')
for i in range(gr-2):
   print('*'+(st-2)*' '+'*')
print(st*'*')
```

Εμφανίζει την πρώτη γραμμή με αστερίσκους.

Εμφανίζει τις ενδιάμεσες γραμμές.

Εμφανίζει την τελευταία γραμμή με αστερίσκους.

6.25

```
# Κλασικός τρόπος με ένθετους βρόχους for
while True:
   gr=int(input('Δώσε το ύψος του δένδρου :'))
   if gr>=4 and gr<=20:
      break
   else:
      print('Η τιμή δεν είναι αποδεκτή, δώσε νέα τιμή')
for i in range(gr-2):
   for j in range(gr-2-i):
      print(' ',end='')
   for j in range(i*2+1):
      print('*',end='')
```

Στην περίπτωση που η τιμή είναι αποδεκτή, σταματάει η εκτέλεση του βρόχου **while**, διαφορετικά εμφανίζει κατάλληλο μήνυμα και ξαναζητάει τιμή.

Προσθέτει τα απαραίτητα κενά πριν από τους αστερίσκους

Εμφανίζει τους απαιτούμενους αστερίσκους.

```
   print()
for j in range(gr-2):
   print(' ',end='')
print('*')
```

Προσθέτει τα απαραίτητα κενά και εμφανίζει τον τελευταίο αστερίσκο.

6.25b

```
# «Πυθωνικός» τρόπος λύσης της άσκησης 6.25 με έναν μόνο βρόχο for
while True:
   gr=int(input('Δώσε το ύψος του δένδρου :'))
   if gr>=4 and gr<=20:
      break
   else:
      print('Η τιμή δεν είναι αποδεκτή, δώσε νέα τιμή')
for i in range(gr-2):
   print((gr-i)*' '+(2*i+1)*'*')
print(gr*' '+'*')
```

Εμφανίζει τα απαραίτητα κενά και τους αστερίσκους κάθε γραμμής

Εμφανίζει τα απαραίτητα κενά και τον τελευταίο αστερίσκο.

6.26

```
# Με ένθετους βρόχους for
for j in range(5,0,-1):
   for i in range(1,j+1):
      print(i,end='')
   print()
```

Στη μεταβλητή **j** ανατίθενται οι αριθμοί από το 5 μέχρι το 1.

Στη μεταβλητή **i** ανατίθενται οι αριθμοί από το 1 μέχρι την τιμή της **j**.

6.26b

```
# Διαφορετική λύση της άσκησης 6.26 με έναν μόνο βρόχο for
s='12345'
for j in range(5):
   print(s[0:5-j])
```

Κάθε φορά εμφανίζεται διαφορετικό τμήμα της συμβολοσειράς **s**.

6.27

```
s=1
for i in range(1,11):
   p=1
   for j in range(1,i+1):
      p=p*j
   s=s+1/p
print(s)
```

Στη μεταβλητή **i** ανατίθενται οι αριθμοί από το 1 μέχρι το 10.

Στη μεταβλητή **j** ανατίθενται οι αριθμοί από το 1 μέχρι την τιμή της **i**.

Στη μεταβλητή **p** υπολογίζεται κάθε φορά το **i!**, δηλαδή το γινόμενο **1*2*3*...*i**.

Στη μεταβλητή **s** αθροίζονται όλα τα κλάσματα **1/i!**.

6.28

```
# Πρόγραμμα-Α
s=10
for i in range(1,6,2):
   s=s+15
   for j in range(4,-1,-1):
      s=s-j
print(s)
```

```
25
```

Θα εκτελεστεί 3 φορές (για i 1,3 και 5), οπότε το **s** θα αυξηθεί κατά **45** (3*15).

Κάθε φορά που εκτελείται η εσωτερική εντολή **for**, το **s** θα μειώνεται κατά 10 (για j 4+3+2+1+0). Αυτό θα γίνει 3 φορές, οπότε το **s** συνολικά θα μειωθεί κατά **30**!

Επομένως, στο τέλος η τιμή του **s** θα είναι 25 (10 που είχε αρχικά +45-30)

```
# Πρόγραμμα-Β
k=10
for i in range(1,11,2):
   k=k-3;
   for j in range(1,5):
      k=k+j
print('k=',k)
```

```
k= 45
```

Θα εκτελεστεί 5 φορές (για i 1,3,5,7 και 9), οπότε το **k** θα μειωθεί κατά **15** (3*5).

Κάθε φορά που εκτελείται η εσωτερική εντολή **for**, το **k** θα αυξάνεται κατά 10 (για j 1+2+3+4). Αυτό θα γίνει 5 φορές, οπότε το **k** συνολικά θα αυξηθεί κατά **50**!

Επομένως, στο τέλος η τιμή του **k** θα είναι 45 (10 που είχε αρχικά -15+50)

6.45

- Το πρόγραμμα δημιουργεί αρχικά μια κενή λίστα στην οποία προσθέτει στη συνέχεια 37 στοιχεία με τιμή 0.
- Στη συνέχεια δημιουργεί 1000 τυχαίους αριθμούς στο διάστημα 0~36 και ανάλογα με τον αριθμό αυξάνει το αντίστοιχο στοιχείο της λίστας κατά 1.
- Στο τέλος, αν το πρώτο στοιχείο της λίστας είναι ο αριθμός 10 σημαίνει ότι 10 από τους τυχαίους αριθμούς ήταν το 0. Αν το δεύτερο στοιχείο της λίστας είναι ο αριθμός 23 σημαίνει ότι 23 από τους τυχαίους αριθμούς ήταν το 1. Αν το τελευταίο στοιχείο της λίστας είναι ο αριθμός 4 σημαίνει ότι 4 από τους τυχαίους αριθμούς ήταν το 36.

6.46

```
import math
Lst=[]
for i in range(5000):
   ar=random.randint(0,36)
   Lst.append(ar)
fores=1
maxfores=0
for i in range(1,5000):
   if Lst[i]==Lst[i-1]:
      fores=fores+1
      if fores>maxfores:
         maxfores=fores
         maxar=Lst[i]
      else:
      fores=1
print(maxfores,maxar)
```

Το πρόγραμμα δημιουργεί αρχικά μια κενή λίστα στην οποία προσθέτει στη συνέχεια 5000 τυχαίους αριθμούς στο διάστημα 0~36.

Αν το στοιχείο της λίστας είναι ίδιο με το προηγούμενο, αυξάνει τη μεταβλητή **fores** κατά 1. Αν η μεταβλητή πάρει τιμή μεγαλύτερη από τη **maxfores**, την αναθέτει στη **maxfores**.

Αν το στοιχείο της λίστας δεν είναι ίδιο με το προηγούμενο, αναθέτει ξανά 1 στη μεταβλητή **fores**, ξεκινώντας το μέτρημα από την αρχή.

Ασκήσεις Κεφαλαίου 13

13.1

- ☑ Οι μεταβλητές κλάσης είναι «κοινόχρηστες» σε όλα τα στιγμιότυπα της κλάσης.
- ☐ Οι μεταβλητές κλάσης ορίζονται και αυτές μέσα στη μέθοδο δόμησης __init__().
- ☑ Μπορούμε να προσπελάζουμε τις μεταβλητές κλάσης είτε με χρήση του ονόματος της κλάσης είτε μέσω οποιουδήποτε αντικειμένου της.
- ☑ Μπορούμε να αναθέσουμε τιμή σε μια μεταβλητή κλάσης μέσω ενός στιγμιότυπου της κλάσης.
- ☑ Για να δημιουργήσουμε μια μέθοδο κλάσης και όχι στιγμιότυπου, πρέπει να χρησιμοποιήσουμε τον διακοσμητή @classmethod.
- ☐ Μπορούμε να προσπελάζουμε μεθόδους κλάσης μόνο με χρήση του ονόματος της κλάσης και όχι μέσω των αντικειμένων της.
- ☑ Οι μέθοδοι κλάσης μπορούν να προσπελάζουν μόνο μέλη κλάσης και όχι στιγμιότυπου.
- ☐ Στην πρώτη παράμετρο μιας μεθόδου κλάσης μεταβιβάζεται το αντικείμενο στο οποίο εφαρμόζεται.
- ☑ Οι στατικές μέθοδοι είναι μέθοδοι κλάσης οι οποίες όμως δεν μπορούν να προσπελάζουν τα υπόλοιπα μέλη της κλάσης.
- ☐ Στην πρώτη παράμετρο μιας στατικής μεθόδου κλάσης μεταβιβάζεται η κλάση στην οποία εφαρμόζεται.
- ☑ Οι στατικές μέθοδοι κλάσης ορίζονται με χρήση του διακοσμητή @staticmethod.

13.2

```
import random

class triangle:
   def __init__(self,b,y):
      self.basi=b
      self.ypsos=y
   def emvado(self):
      return self.basi*self.ypsos/2
   def show(self):
      print('B =',self.basi,' Y =',self.ypsos,' E =',self.emvado())

# Κυρίως πρόγραμμα
Lst=[]
sum=0
for i in range(10):
    Lst.append(triangle(random.randint(10,100),random.randint(10,100)))
for i in range(10):
    Lst[i].show()
    sum=sum+Lst[i].emvado()
print('Συνολικό εμβαδό:',sum)
```

Δημιουργεί μια κενή λίστα και την αναθέτει στη μεταβλητή **Lst**.

Δημιουργεί ένα αντικείμενο-triangle με τυχαίες τιμές (μεταξύ του 10 και του 100) για τη βάση και το ύψος του και το προσθέτει ως στοιχείο στη λίστα **Lst**. Αυτό γίνεται 10 φορές, οπότε στη λίστα προστίθενται δέκα τέτοια αντικείμενα.

Εμφανίζει τα στοιχεία και των δέκα στοιχείων (αντικειμένων triangle) της λίστας **Lst**.

Προσθέτει το εμβαδόν όλων των αντικειμένων-τριγώνων της λίστας **Lst** και το αναθέτει στη μεταβλητή **sum**.

13.3

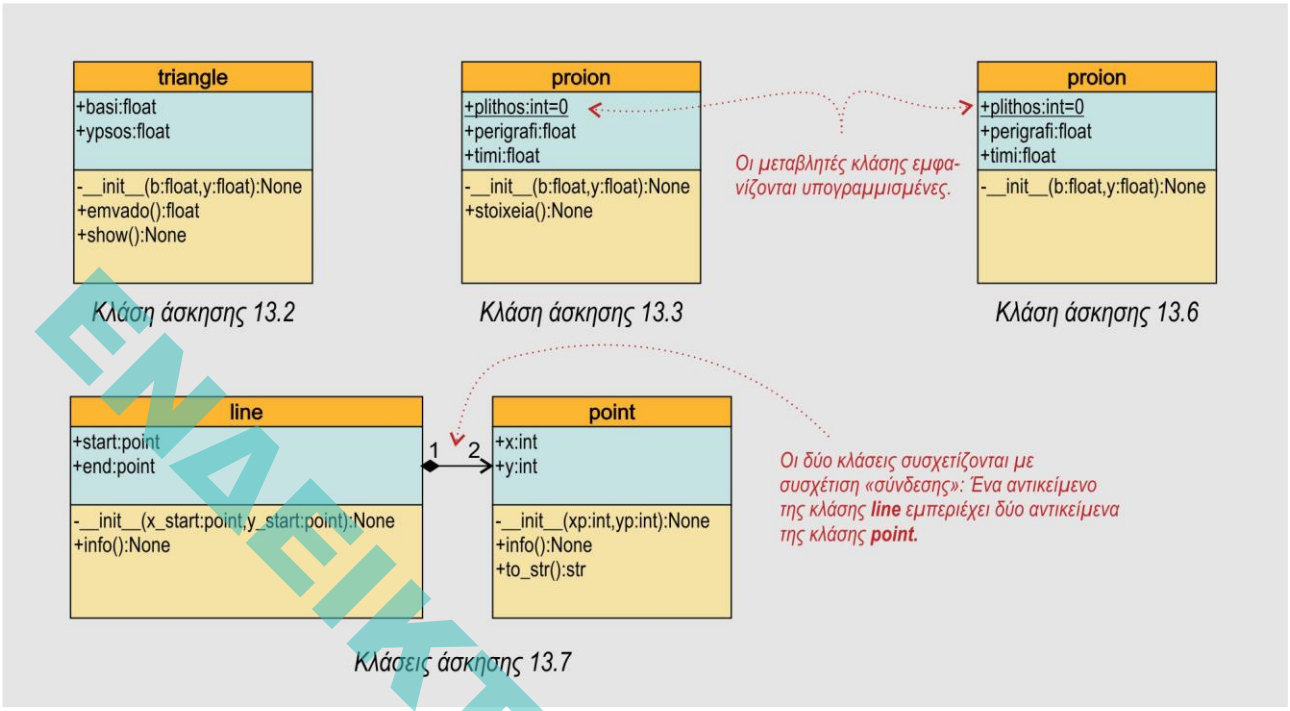
```
class proion:
   plithos=0
   def __init__(self,p,t):
      self.perigrafi=p
      self.timi=t
      proion.plithos+=1
      print('Ένα',self.perigrafi,'δημιουργήθηκε')

   def stoixeia(self):
      print(self.perigrafi,'με τιμή',self.timi)
```

Ορίζεται η μεταβλητή κλάσης **plithos** με αρχική τιμή 0.

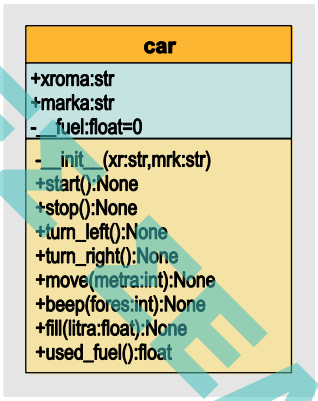
Κάθε φορά που δημιουργείται ένα νέο αντικείμενο της κλάσης, η μεταβλητή κλάσης **plithos** αυξάνεται κατά 1.

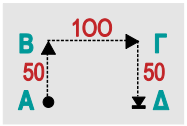
13.11



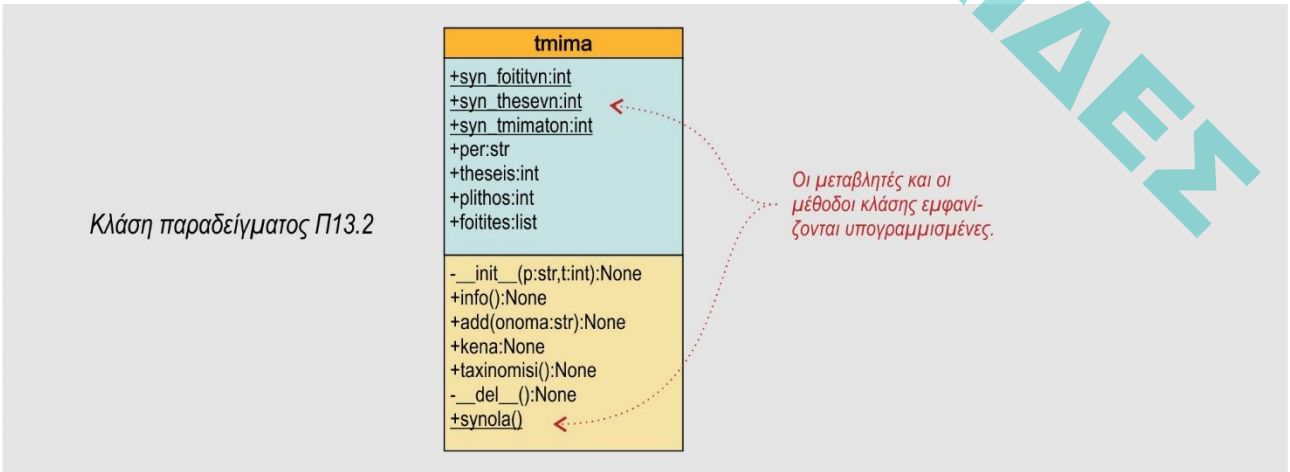
13.12

```
mycar=car('κόκκινο','Toyota')
mycar.fill(50)
mycar.start()
mycar.move(50)
mycar.beep(2)
mycar.turn_right()
mycar.move(100)
mycar.beep(1)
mycar.turn_right()
mycar.move(50)
mycar.stop()
print('Κάψαμε ',mycar.used_fuel(),'λίτρα')
```





13.13



13.14

```
class proion:
   def __init__(self,p,t):
      self.perigrafi=p
      self.timi=t
      print('Ένα',self.perigrafi,'δημιουργήθηκε')

   def stoixeia(self):
      print(self.perigrafi,'με τιμή',self.timi)

   def set(self,s):
      if type(s) is str:
         self.perigrafi=s
      elif type(s) is int or type(s) is float:
         self.timi=s

# Κυρίως πρόγραμμα
p1=proion('πουκάμισο',45)
p1.stoixeia()
p1.set('ρολόι')
p1.stoixeia()
p1.set(200)
p1.stoixeia()
```

```
Ένα πουκάμισο δημιουργήθηκε
πουκάμισο με τιμή 45
ρολόι με τιμή 45
ρολόι με τιμή 200
```

Η μέθοδος **set()** συμπεριφέρεται ως «υπερφορτωμένη», διότι ανάλογα με τον τύπο των ορισμάτων με τα οποία καλείται, εκτελεί διαφορετική λειτουργία. Όταν κληθεί με όρισμα μια συμβολοσειρά, την αναθέτει ως τιμή της μεταβλητής στιγμιότυπου **perigrafi** ενός αντικειμένου της κλάσης **proion**. Αν κληθεί με όρισμα έναν ακέραιο ή έναν πραγματικό αριθμό, τον αναθέτει ως τιμή της μεταβλητής στιγμιότυπου **timi** ενός αντικειμένου της κλάσης.

Αναθέτει ως περιγραφή του αντικειμένου **p1** το "ρολόι".

Αναθέτει ως τιμή του αντικειμένου **p1** το 200.

13.15

*Ο κανόνας **LEGB** σημαίνει ότι ο διερμηνευτής της Python αναζητά τα ονόματα με την εξής σειρά: Πρώτα στον τοπικό χώρο ονομάτων (**L**ocal), μετά στον περιβάλλοντα χώρο ονομάτων (**E**nclosing), στη συνέχεια στον καθολικό χώρο ονομάτων (**G**lobal), και τέλος στον ενσωματωμένο χώρο ονομάτων (**B**uilt-in). Στην περίπτωση που το όνομα δεν εντοπιστεί σε κανέναν από τους παραπάνω τέσσερις χώρους ονομάτων, η Python εγείρει ανάλογο σφάλμα.*

13.16

*Οι **περιβάλλοντες** (Enclosing) και οι **τοπικοί** (Local) χώροι ονομάτων, διότι αυτοί δημιουργούνται κατά την κλήση και καταστρέφονται κατά την επιστροφή συναρτήσεων (ή μεθόδων).*