Μαθητεία ΕΠΑΛ
Σημειώσεις Εργαστηριακού Μαθήματος

Ενότητα
«Αντικειμενοστρεφής Προγραμματισμός»

Πάλλας Αναστάσιος – Καθηγητής Πληροφορικής ΠΕ86

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον Νίκο Μαυρογενειάδη,
μαθητή μου και φοιτητή του Τμήματος Πληροφορικής του Πανεπιστημίου Δυτικής Αττικής
για την πολύ ουσιαστική συμβολή του στην συγγραφή του παρόντος.
Νίκο σου εύχομαι Υγεία & Πάντα Επιτυχίες!

Αντικειμενοστρεφής Προγραμματισμός (Object Oriented Programming)

Περί Αντικειμενοστρεφούς Προγραμματισμού

Τι είναι ο Αντικειμενοστρεφής Προγραμματισμός (Object Oriented Programming);

Παρότι που στην πλειονότητα των γλωσσών προγραμματισμού χρησιμοποιούνται οι λεγόμενοι τύποι δεδομένων, όπως για παράδειγμα, ο ακέραιος (integer), ο πραγματικός (float ή double), ο χαρακτήρας (char ή string), κ.α., η ανάγκη **αναπαράστασης των αντικειμένων του πραγματικού κόσμου** στο προγραμματισμό, οδήγησε σ' αυτό που ονομάζουμε «αντικειμενοστρεφής προγραμματισμός». Με τον αντικειμενοστρεφή προγραμματισμό όχι μόνο επιτυγχάνουμε την βελτίωση της απόδοση των εφαρμογών/προγραμμάτων, αλλά κυρίως την ευκολότερη ανάπτυξή τους απ' την πλευρά των προγραμματιστών, σε σύγκριση με τις παραδοσιακές γλώσσες «διαδικαστικού προγραμματισμού» (procedural languages).

Ο αντικειμενοστρεφής προγραμματισμός δεν περιορίζεται απλώς στη χρήση μιας διαφορετικής γλώσσας προγραμματισμού με κάποιες νέες δυνατότητες, αλλά είναι μια **διαφορετική φιλοσοφία αντίληψης και σχεδιασμού των εφαρμογών/προγραμμάτων**. Οι **κλάσεις** (classes) και τα **αντικείμενα** (objects) είναι οι κυρίαρχες έννοιες στις οποίες βασίζεται ο αντικειμενοστρεφής προγραμματισμός.

Τι είναι η Κλάση (Class);

Η **κλάση** είναι μια έννοια η οποία προσδιορίζει μια κατηγορία αντικειμένων και ταυτόχρονα περιγράφει τα χαρακτηριστικά και τις λειτουργίες ή τις ενέργειες που πραγματοποιούνται σ' αυτά. Τα χαρακτηριστικά της κλάσης ονομάζονται **ιδιότητες (properties)** και μπορεί να είναι απλοί ή αρκετά σύνθετοι τύποι δεδομένων, ενώ οι λειτουργίες/ενέργειες που πραγματοποιούνται σε μια κλάση ονομάζονται **μέθοδοι (methods)**.

Τι είναι το Αντικείμενο (Object);

Ένα αντικείμενο μιας κλάσης αποκτά όλα τα χαρακτηριστικά και τις λειτουργίες της κλάσης στην οποία ανήκει. **Ένα αντικείμενο θα λέγαμε ότι αποτελεί ένα στιγμιότυπο (instance) μιας κλάσης.** Οι ιδιότητες και οι μέθοδοι καλούνται **πάντα** σε συνδυασμό με κάποιο αντικείμενο.

Προσοχή! Δεν πρέπει να συγχέουμε την έννοια της κλάσης με αυτή του αντικείμενου. Η κλάση αποτελεί ως μια απλουστευμένη μετάφραση το «καλούπι» για την παραγωγή «αντικειμένων» του ίδιου τύπου/ομάδας.

Τι είναι η «Αρχικοποίηση ενός Αντικειμένου» (Object Initialization);

Αρχικοποίηση αντικειμένου (initialization) ονομάζεται η διαδικασία κατά την οποία **"δίνουμε" αρχικές τιμές σ' ένα αντικείμενο κατά την δημιουργία του**. Υπάρχουν διάφοροι τρόποι, ανάλογα με την γλώσσα προγραμματισμού, για το πώς θα πραγματοποιηθεί αυτό, ο πιο συνηθισμένος είναι με την κλήση ενός λεγόμενου «κατασκευαστή» (constructor).

Χαρακτηριστικά του Αντικειμενοστρεφή Προγραμματισμού

Ενθυλάκωση (Encapsulation)

Η ενθυλάκωση είναι μία από τις θεμελιώδεις έννοιες στον αντικειμενοστρεφή προγραμματισμό. Περιγράφει την έννοια της «προστασίας» των δεδομένων και των μεθόδους που έχουν πρόσβαση σε δεδομένα εντός μιας κλάσης. Στην ουσία, η «ενθυλάκωση» **θέτει περιορισμούς στην άμεση πρόσβαση σε ιδιότητες και μεθόδους** και μπορεί να αποτρέψει την τυχαία ή από λάθος τροποποίηση των δεδομένων. Για την αποφυγή τυχαίας αλλαγής, η ιδιότητα ενός αντικειμένου μπορεί να αλλάξει μόνο μέσω των μεθόδων ενός αντικειμένου. Σε αρκετές γλώσσες αυτό υλοποιείται μέσω των ιδιωτικών (private) μεταβλητών/ιδιοτήτων (private variables) και των «προστατευμένων» μεθόδων (protected methods).

Κληρονομικότητα (Inheritance)

Κληρονομικότητα είναι η δυνατότητα δημιουργίας μιας νέας κλάσης βασιζόμενη σε μια ήδη υπάρχουσα κλάση την γονική (parent class). Η νέα κλάση ή αλλιώς υπο-κλάση κληρονομεί **όλα ή κάποια** από τα χαρακτηριστικά και τις λειτουργίες της γονικής κλάσης, αλλά μπορεί επίσης να ορίζει και δικά της επιπλέον χαρακτηριστικά και λειτουργίες.

Πολυμορφισμός (Polymorphism)

Πολυμορφισμός είναι η δυνατότητα που μας προσφέρουν οι αντικειμενοστρεφείς γλώσσες προγραμματισμού στο χειρισμό **διαφορετικού τύπων δεδομένων** με την χρήση «κοινή» διεπαφής, δηλαδή μέσω κοινής μεθόδου. Στην υλοποίηση αυτό πραγματοποιείται συνήθως **μέσω της «υπερφόρτωσης» (overloading)** τελεστών (operators) και μεθόδων (methods) ώστε να συμπεριφέρεται με διαφορετικό τρόπο ανάλογα με τα δεδομένα που του αλληλεπιδρούν.

Ο ακριβείς τρόπος με τον οποίο υλοποιούνται τα παραπάνω χαρακτηριστικά που αφορούν τον αντικειμενοστρεφή προγραμματισμό ενδέχεται να διαφέρουν από γλώσσα σε γλώσσα.

Python

Η Python είναι μια **αντικειμενοστρεφής γλώσσα προγραμματισμού γενικού σκοπού** (general purpose object-oriented programming language) της οποία τα κύρια χαρακτηριστικά είναι η δυναμική διαχείριση, η απλή σύνταξη και η ταχύτητα εκτέλεσης. Τα αρχεία της γλώσσα Python έχουν κατάληξη "**.py**". Η Python συνεχώς εξελίσσεται και βελτιώνεται. Μια από τις σημαντικότερες εκδόσεις είναι η έκδοση 2.x, ενώ αυτή την στιγμή βρισκόμαστε στην έκδοση 3.8. **Τα προγράμματα και τα παραδείγματα μας έχουν γραφτεί όλα στην έκδοση 2.7** για να υπάρχει ταύτισή τους με τις ερωτήσεις και τις ασκήσεις πιστοποίησης.

Μεταβλητές

Μέσω των μεταβλητών, μας δίνεται η δυνατότητα να αποθηκεύουμε τιμές τις οποίες μπορούμε στην συνέχεια να χρησιμοποιήσουμε. Συγκεκριμένα, θα λέγαμε ότι μεταβλητή είναι μια θέση μνήμης την οποία έχουμε δεσμεύσει και της έχουμε δώσει ένα όνομα. Το περιεχόμενο αυτής της θέσης μνήμης αποτελεί την τιμή αυτής της μεταβλητής.
Αποδεκτά ονόματα μεταβλητών είναι αυτά για τα οποία ισχύει:

- Το όνομα μιας μεταβλητής πρέπει να ξεκινάει με γράμμα ή κάτω παύλα, **δεν μπορεί να ξεκινάει με αριθμό**.
- Το όνομα μπορεί να περιέχει μόνο αλφαριθμητικούς χαρακτήρες και τον χαρακτήρα κάτω παύλα (A-z, 0-9 και _).
- Τα ονόματα των μεταβλητών είναι **case sensitive**, δηλ. οι μεταβλητές age, Age και AGE είναι τρεις διαφορετικές μεταβλητές!

Εκχώρηση Τιμών

Η εκχώρηση τιμών στην Python μπορεί να πραγματοποιηθεί μια αρκετούς τρόπους.

Παράδειγμα

```
x, y, z = "Orange", "Banana", "Cherry"
a = 5
b = 143L
p = 3.1415
name = 'Nikolas'

# εμφάνιση των μεταβλητών
print x, y
print b, a
print name
```

Τύποι Δεδομένων

Παρακάτω παραθέτουμε μερικούς από τους τύπους δεδομένων που υπάρχουν στην Python.

Τύπος Δεδομένων	Παράδειγμα
int	1000
long	1000L
float	1000.12
complex	1000 + 12j
str	‘apple’
bool	True

Στην προηγούμενη ενότητα είδαμε ότι για να δηλώσουμε μια μεταβλητή, αρκεί να εκχωρήσουμε μια τιμή σε αυτή **χωρίς** όμως ορίσουμε τον τύπο της μεταβλητής. Αυτό συμβαίνει, γιατί η Python βασίζεται στην έννοια του **duck typing**!

Στις γλώσσες προγραμματισμού που χρησιμοποιούν την έννοια του duck typing, **ο τύπος ενός αντικειμένου εξαρτάται από το ίδιο το αντικείμενο** και καθορίζεται από το ποιες μεθόδους υλοποιεί, επομένως αυτά τα συστήματα έχουν εξ ορισμού «ασφαλείς» τύπους, αφού μπορούν να κληθούν μόνο μέθοδοι που πραγματικά υλοποιεί ένα αντικείμενο.

Άμα δω ένα πουλί που περπατάει σαν πάπια και ακούγεται σαν πάπια,
τότε πρέπει να είναι πάπια!

Τελεστές

Αριθμητικοί Τελεστές

Τελεστής	Περιγραφή	Παράδειγμα
+	Πρόθεση	x+y
-	Αφαίρεση	x-y
*	Πολλαπλασιασμός	x*y
/	Διαίρεση	x/y
%	Υπόλοιπο	x%y
**	Δύναμη	x**y
//	Διαίρεση με στρογγυλοποίηση στον πλησιέστερο ακέραιο αριθμό.	x//y

Τελεστές Εκχώρησης

Τελεστής	Παράδειγμα	Ισοδύναμο
=	x=5	x=5
+=	x+=3	x = x + 3
-=	x-=3	x = x - 3
*=	x*=3	x = x * 3
/=	x/=3	x = x / 3
%=	x%=3	x = x % 3
//=	x//=3	x = x // 3
**=	x**3	x = x ** 3
&=	x&=3	x = x & 3
|=	x|=3	x = x | 3
^=	x^=3	x = x ^ 3
>>=	x>>=3	x = x >> 3
<<=	x<<=3	x = x << 3

Τελεστές Σύγκρισης

Τελεστής	Περιγραφή	Παράδειγμα
==	Ισότητα	x == y
!=	Διάφορο	x != y
>	Μεγαλύτερο	x > y
<	Μικρότερο	x < y
>=	Μεγαλύτερο ή ίσο	x >= y
<=	Μικρότερο ή ίσο	x <= y

Λογικοί Τελεστές

Τελεστής	Περιγραφή	Παράδειγμα
and	Επιστρέφει True αν και οι δύο συνθήκες είναι True.	x < 5 and x < 10
or	Επιστρέφει True αν **τουλάχιστον** μια από τις συνθήκες είναι True.	x < 5 or x < 4
not	Επιστρέφει το αντίστροφο αποτέλεσμα της συνθήκης.	not(x < 5 and x < 10)

Συνάρτηση print

Μέσω της print μπορούμε να εμφανίσουμε τιμές στο standard output.

```
1.  print "Hello There!"
2.  x = 10 + 10
3.  print "The value of x is", x  #εμφανίζει The value of x is 20
```

Αν θέλουμε να χρησιμοποιήσουμε Ελληνικά στο πρόγραμμα μας, χρειάζεται να τοποθετήσουμε την παρακάτω γραμμή στην αρχή του κώδικα.
 # -*- coding: utf-8 -*-

Block κώδικας και εντολές if … then … else

Στις γλώσσες προγραμματισμού, όταν θέλουμε να **ομαδοποιήσουμε** για κάποιο σκοπό **ένα πλήθος εντολών** του λεγόμενου «μπλοκ εντολών» (block) χρησιμοποιούμε κάποιο σύμβολο για την έναρξη και την λήξη, για παράδειγμα σε αρκετές γλώσσες όπως στην Javascript και στην PHP χρησιμοποιούνται τα άγκιστρα { και }. Στην Python τα πράγματα είναι λίγο διαφορετικά! Για να ομαδοποιήσουμε ένα συγκεκριμένο σύνολο εντολών χρειάζεται να υπάρχει η **κατάλληλη εσοχή**. Λέγοντας «εσοχή», εννοούμε το πλήθος των κενών στην αρχή της γραμμής.

Η σύνταξη της εντολής if… then … else είναι η ακόλουθη:

```
if συνθήκη1:
   εντολή
elif συνθήκη2:
   εντολή
elif συνθήκη3:
   εντολή
else:
   εντολή
```

Παράδειγμα 1

```
1.  a = 33
2.  b = 200
3.  if b > a:
4.    print "b is greater than a"
```

Παρατηρούμε ότι η εντολή print έχει εσοχή 4 κενών χαρακτήρων (ή 1 χαρακτήρα tab). Αυτό συμβαίνει γιατί η print θα εκτελεστεί μόνο αν ισχύει η συνθήκη στην if. Αν θέλαμε να έχουμε περισσότερες εντολές στην περίπτωση που ισχύει η συνθήκη, θα έπρεπε όλες οι εντολές να είναι κάτω από την if με το ίδιο μήκος εσοχής.

Παράδειγμα 2

```
1.  a = 33
2.  b = 200
3.  if b > a:
4.  print "b is greater than a" # ΛΑΘΟΣ! Δεν έχει εσοχή!
```

Στο παράδειγμα 2, το print δεν έχει κατάλληλη εσοχή μετά την if, με αποτέλεσμα να μην υπάρχει εντολή να εκτελεστεί αν ισχύει η συνθήκη. Αυτό αποτελεί **συντακτικό λάθος στην Python!**

Παράδειγμα 3

```
1.  v = 100
2.  if v == 200:
3.     print "1 - Got a true expression value"
4.     print v
5.  elif v == 150:
6.     print "2 - Got a true expression value"
7.     print v
8.  elif v == 100:
9.     print "3 - Got a true expression value"
10.    print v
11. else:
12.    print "4 - Got a false expression value"
13.    print v
14.
15. print "Good bye!"
```

Διερεύνηση! Τι θα εμφανίσει το παραπάνω τμήμα κώδικα;

Εντολές Επανάληψης

Εντολή επανάληψης	Διάρκεια εκτέλεσης
while συνθήκη: εντολές	Εκτελείται όσο η συνθήκη στην while είναι αληθής.
for στοιχείο **in** ακολουθία: εντολές	Εκτελείται μέχρι να διασχίσουμε όλη την ακολουθία.

Παράδειγμα 1

```
1.  for letter in 'Python':     # First Example
2.     print 'Current Letter :', letter
3.
4.  fruits = ['banana', 'apple',  'mango']
5.  for fruit in fruits:        # Second Example
6.     print 'Current fruit :', fruit
7.
8.  print "Good bye!"
```

Αποτέλεσμα εκτέλεσης

```
1.  Current Letter : P
2.  Current Letter : y
3.  Current Letter : t
4.  Current Letter : h
5.  Current Letter : o
6.  Current Letter : n
7.  Current fruit : banana
8.  Current fruit : apple
9.  Current fruit : mango
10. Good bye!
```

Παράδειγμα 2

```
1.  x = range(3, 20, 2)
2.  for n in x:
3.    print n
```

Αποτέλεσμα εκτέλεσης

```
1.  3
2.  5
3.  7
4.  9
5.  11
6.  13
7.  15
8.  17
9.  19
```

Παράδειγμα 3

```
1.  fruits = ['banana', 'apple',  'mango']
2.  for index in range(len(fruits)):
3.     print 'Current fruit :', fruits[index]
4.
5.  print "Good bye!"
```

Διερεύνηση!
α) Τι περιέχει κάθε φορά η μεταβλητή index;
β)Τι θα εμφανίσει το παραπάνω πρόγραμμα;

Παράδειγμα 4

```
1.  count = 0
2.  while (count < 9):
3.     print 'The count is:', count
4.     count = count + 1
5.
6.  print "Good bye!"
```

Αποτέλεσμα εκτέλεσης

```
1.  The count is: 0
2.  The count is: 1
3.  The count is: 2
4.  The count is: 3
5.  The count is: 4
6.  The count is: 5
7.  The count is: 6
8.  The count is: 7
9.  The count is: 8
10. Good bye!
```

Η συνάρτηση range() επιστρέφει μια ακολουθία αριθμών, η οποία ξεκινάει από το 0 (προκαθορισμένη αρχική τιμή) και αυξάνεται κατά 1 (προκαθορισμένο βήμα) και σταματάει πριν από έναν συγκεκριμένο αριθμό.

Σύνταξη: range(*start, stop, step*)

Παράδειγμα 5

```
1.  x = range(6)
2.  for n in x:
3.    print n
```

Αποτέλεσμα εκτέλεσης

```
1.  0
2.  1
3.  2
4.  3
5.  4
6.  5
```

Δομές Δεδομένων

Στην Python υλοποιούνται διάφορες δομές δεδομένων οι οποίες μπορούν να χρησιμοποιηθούν από τους προγραμματιστές για αποθήκευση απλών τύπων δεδομένων ή αντικειμένων. Στην Python ονομάζονται **συλλογές** (collections).

Collection Type	Παράδειγμα
list	[‘apple’, ‘ball’, ‘ball’]
tuple	(‘apple’, ‘ball’, ‘ball’)
dict	{‘fruit’ : ‘apple’, ‘toy’:’ball’}
set	{‘apple’, ‘ball’}

Η δομή **tuple**, αν και μοιάζει με το **list**, διαφέρει στο ότι είναι «**αμετάβλητη**» (immutable)**,** δηλαδή απ’ την στιγμή που θα δημιουργηθεί, δεν μπορεί να αλλάξει κάποια τιμή της.

Εμφάνιση Λίστας

```
1.  thislist = ["apple", "banana", "cherry"]
2.  print thislist
```

Πρόσβαση σε Τιμή μιας Λίστας

```
1.  thislist = ["apple", "banana", "cherry"]
2.  print thislist[1] #εμφανίζει banana
3.  print thislist[-1] #εμφανίζει cherry
```

Εμφάνιση Πολλών Τιμών μιας Λίστας

```
1.  thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
2.  print thislist[2:5] #εμφανίζει cherry orange kiwi
3.  print thislist[:4]   #εμφανίζει apple banana cherry orange
4.  print thislist[2:]   #εμφανίζει από το cherry μέχρι το τέλος
5.  print thislist[-4:-1] #εμφανίζει orange kiwi melon
```

Αλλαγή Τιμής σε Λίστα

```
1.  thislist = ["apple", "banana", "cherry"]
2.  thislist[1] = "cake"
3.  print thislist
```

Επαναληπτική Διαδικασία σε Λίστα

```
1.  thislist = ["apple", "banana", "cherry"]
2.  for x in thislist:
3.    print x
```

Έλεγχος για το αν υπάρχει μια Τιμή σε Λίστα

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print "Yes, 'apple' is in the fruits list"
```

Μέγεθος Λίστας

```
thislist = ["apple", "banana", "cherry"]
print len(thislist)
```

Προσθήκη Τιμής στο Τέλος μιας Λίστας

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print thislist
```

Προσθήκη Τιμής σε Συγκεκριμένη Θέση στην Λίστα

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print thislist
```

Αφαίρεση Τιμής από Λίστα

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print thislist
```

Παράδειγμα διαχείρισης λίστας στην Python

```
numbers = [1, 2, 3, 5, 8, 11, 18, 21]
odd_numbers = []
for num in numbers:
    if num % 2 != 0:
        odd_numbers.append(num)
print "the odd numbers are:", odd_numbers
```

Σκοπός του παραδείγματος είναι από μία λίστα με αριθμούς να τοποθετήσουμε σε μια άλλη λίστα τους **περιττούς** αριθμούς (1, 3, 5, κλπ).

1. Δημιουργούμε και αρχικοποιούμε την λίστα numbers με τυχαίους αριθμούς.
2. Δημιουργούμε την λίστα odd_numbers η οποία είναι κενή.
3. Επανάληψη τύπου for στην Python. Σε κάθε επανάληψη, η μεταβλητή num έχει ένα στοιχείο απ' την λίστα numbers.
4. Στην κάθε επανάληψη ελέγχουμε αν το υπόλοιπο της ακέραιας διαίρεσης με το 2 είναι διαφορετικό απ' το μηδέν, δηλαδή αν ο αριθμός είναι περιττός. Ο ίδιος έλεγχος θα μπορούσε να γίνει ως «num % 2 == 1».
5. Αν η συνθήκη είναι αληθής, τότε τοποθετούμε τον αριθμό στην λίστα odd_numbers χρησιμοποιώντας την μέθοδο **append**.
6. Στο τέλος της επανάληψης εμφανίζουμε την λίστα πίνακα odd_numbers.

Οι λίστες στην Python είναι «αντικείμενα», με αποτέλεσμα να διαθέτουν ιδιότητες και μεθόδους. Για παράδειγμα, η μέθοδος append, είναι μια **build-in** μέθοδος της κλάσης **list**. Παρατηρούμε ότι μετά το αντικείμενο, χρησιμοποιούμε τον τελεστή "**.**" για να έχουμε πρόσβαση είτε στις ιδιότητες ή στις μεθόδους του αντικειμένου.

Συναρτήσεις

Παράδειγμα ορισμού συνάρτησης στην Python

```
1.  def a_function(arg1, arg2="default", *args, **kwargs):
2.      """
3.          Comment στην αρχή μιας συνάρτησης.
4.      """
5.      print "arg1:", arg1              # arg1: απλή παράμετρος
6.      print "arg2:", arg2              # arg2: παράμετρος με προκαθορισμένη τιμή
7.      print "args:", args              # ένα tuple με τις επιπλέον παραμέτρους
8.      print "kwargs:", kwargs          # ένα dict για τις named παραμέτρους
```

Παρατηρούμε ότι **τα ορίσματα δεν έχουν τύπους**. Αυτό συμβαίνει γιατί στην Python ακολουθούμε και εδώ τον κανόνα "duck typing"! Ας δούμε μερικά παραδείγματα με το πως θα μπορούσαμε **να κάνουμε κλήση** της παραπάνω συνάρτησης.

Παράδειγμα 1

```
1.  a_function(10) # Κλήση της συνάρτησης με παράμετρο την τιμή 10
2.
3.  Αποτελέσματα:
4.  arg1: 10
5.  arg2: default
6.  args: ()
7.  kwargs: {}
```

Παράδειγμα 2

```
1.  a_function(10, "ten") # Κλήση της συνάρτησης με παραμέτρους 10 και "ten"
2.
3.  Αποτελέσματα:
4.  arg1: 10
5.  arg2: ten
6.  args: ()
7.  kwargs: {}
```

Παράδειγμα 3

```
1.  a_function(10, 20, 30, 40)
2.
3.  Αποτελέσματα:
4.  arg1: 10
5.  arg2: 20
6.  args: (30, 40)
7.  kwargs: {}
```

Παράδειγμα 4

```
a_function(10, "twenty", arg3=30, arg4="forty")

Αποτελέσματα:
arg1: 10
arg2: twenty
args: ()
kwargs: {'arg3': 30, 'arg4': 'forty'}
```

Παράδειγμα 5

```
a_function(arg2="twenty", arg1=10, arg3=30, arg4="forty")

Αποτελέσματα:
arg1: 10
arg2: twenty
args: ()
kwargs: {'arg3': 30, 'arg4': 'forty'}
```

Από την στιγμή που δώσουμε μια **named** παράμετρο, τότε και **όλες οι υπόλοιπες** θα πρέπει να είναι επίσης named παράμετροι.

Παράδειγμα συνάρτησης η οποία δέχεται 2 αριθμούς και **επιστρέφει** τον μεγαλύτερο.

```
def mymax(num1, num2):
    if num1 > num2:
        return num1
    elif num1 < num2:
        return num2
    else:
        return

a = 10
b = 20
result = mymax(a, b)
print result
```

Χρειάζεται προσοχή στην εμβέλεια (scope) των μεταβλητών. Οι μεταβλητές που δηλώνονται μέσα σε μια συνάρτηση δεν είναι ορατές εκτός αυτής, δηλαδή έχουν όπως λέμε «τοπική» εμβέλεια (local scope).

Τυχόν μεταβλητές στην Python οι οποίες δηλώνονται **εκτός κάποιου block**, **συνάρτησης, μεθόδου,** κ.λπ. τότε λέμε ότι έχουν **καθολική εμβέλεια** (**global scope)** οπότε είναι προσπελάσιμες από οποιοδήποτε τμήμα του προγράμματος.

Κλάσεις & Αντικείμενα

Οτιδήποτε στην Python αποτελεί ένα αντικείμενο. Ο τρόπος να δημιουργήσουμε αντικείμενα είναι μέσω των κλάσεων. Παρακάτω ακολουθεί μια δήλωση μια κλάσης **Person** που αντιπροσωπεύει ένα άτομο και έχει ως ιδιότητες το όνομα και το επίθετο του. Επίσης έχει 2 μεθόδους (χωρίς την __init__). Μια μέθοδο **full_name** η οποία επιστρέφει το όνομα και το επίθετο του Person και την **__str__** η οποία και αυτή επιστρέφει το όνομα και το επίθετο του Person, αλλά με ένα διαφορετικό μήνυμα.

```
1.  class Person(object):
2.      def __init__(self, first, last):
3.          self.first = first
4.          self.last = last
5.      def full_name(self):
6.          return "%s %s" % (self.first, self.last)
7.      def __str__(self):
8.          return "Person: " + self.full_name()
```

1. Η κλάση δηλώνεται με το αναγνωριστικό **class** και ακολουθεί το όνομα. Όλες οι κλάσεις που δημιουργούμε, πρέπει να κληρονομούν (θα μιλήσουμε παρακάτω γι' αυτό) την κλάση object. Γι' αυτό μετά το όνομα της κλάσης, σε παρενθέσεις, γράφουμε object, δηλαδή η κλάση Person κληρονομεί την object.
2. Η μέθοδος δηλώνεται με το αναγνωριστικό **def** μέσα στην κλάση. Ακολουθεί το όνομα, και στην συνέχεια οι παρενθέσεις με παραμέτρους. Η μέθοδος με το όνομα __init__ είναι η μέθοδος αρχικοποιήσεις ή με άλλα λόγια ο constructor της κλάσης. Οι παράμετροι της μεθόδου αυτής, δηλώνουν και τις ιδιότητες της κλάσης. **Δεν χρειάζεται να δηλώσουμε ξεχωριστά τις ιδιότητες στην κλάση.** Η πρώτη παράμετρος είναι η **self** (δεν είναι αναγκαστικό να την ονομάσουμε έτσι, καλό είναι όμως να μην αλλάζουμε το όνομα, αφού αυτό το όνομα χρησιμοποιείται ευρεία από τους προγραμματιστές της python). Η παράμετρος αυτής είναι μια αναφορά στο αντικείμενο που καλεί την μέθοδο, **γι' αυτό και υπάρχει ως πρώτη παράμετρος σε κάθε μέθοδο της κλάσης**.
3. Στην γραμμή αυτή, αρχικοποιούμε τις ιδιότητες του αντικειμένου με αυτές που περνιούνται ως παράμετρο.
4.
5. Δηλώνουμε την μέθοδο full_name η οποία δεν δέχεται κάποια παράμετρο (εκτός της self που όπως αναφέραμε είναι default η πρώτη παράμετρος για κάθε μεθόδο, χωρίς να χρειάζεται να περάσουμε εμείς κάποια παράμετρο).
6. Η μέθοδος full_name επιστρέφει ένα string. Το πρώτο %s αντιστοιχείται στο self.first που είναι το όνομα του Person. Το δεύτερο %s αντιστοιχείται στο self.last που είναι το επώνυμο του Person.
7. Η μέθοδος __str__ η οποία γίνεται **override** από την κλάση object (θα δούμε παρακάτω τι σημαίνει ακριβώς αυτό).
8. Επιστρέφει ένα μήνυμα που εμφανίζει αρχικά το μήνυμα "Person: " και δίπλα από αυτό το όνομα που έχουμε δώσει στο αντικείμενο καλώντας την full.name(). Παρατηρούμε ότι μέσα σε μια μέθοδο, μπορούμε να καλέσουμε και άλλες μεθόδους του ίδιου αντικειμένου μέσω της self.

Ας δούμε πως δημιουργούμε ένα αντικείμενο τύπου Person και πως καλούμε τις μεθόδους του.

```
1.  myperson = Person('Clark', 'Kent')
2.  print myperson #Person: Clark Kent
3.  print myperson.first #Clark
4.  print myperson.full_name() #Clark Kent
5.  print Person.full_name(myperson) #Clark Kent
```

Παρατηρούμε ότι όταν καλούμε μια μέθοδο μέσω ενός αντικειμένου δεν χρειάζεται να περάσουμε καμία παράμετρο στο self.

Μια εναλλακτική λύση είναι να καλέσουμε **μέσω της κλάσης** (όχι μέσω ενός αντικειμένου). Σε αυτή την περίπτωση δεν γνωρίζει η Python ποιο αντικείμενο είναι το self. Γι' αυτό χρειάζεται να το περάσουμε εμείς ως παράμετρο (γραμμή 5).

Βλέπουμε ότι στην γραμμή 3 εμφανίζεται η ιδιότητα first του αντικειμένου person. Μπορούμε αν θέλουμε και να την τροποποιήσουμε. Πολλές φορές αυτό **δεν είναι επιτρεπτό**. Δηλαδή, δεν θέλουμε να δίνουμε την δυνατότητα στον χρήστη να μπορεί να έχει πρόσβαση σε ιδιότητες ή μεθόδους του αντικειμένου (χαρακτηριστικό του αντικειμενοστρεφούς προγραμματισμού που ονομάζεται **ενθυλάκωση**). Στην Python δεν υπάρχει κάποιος ειδικός μηχανισμός ώστε να το κάνουμε αυτό, χρησιμοποιούμε όμως μια τεχνική, στην οποία τοποθετούμε πριν από την ιδιότητα ή την μέθοδο ή μία ή δύο κάτω παύλες πριν από το όνομα της μεθόδου (ή ιδιότητας). Αυτές οι ιδιότητες/μέθοδοι ονομάζονται **Ιδιωτικές (Private)**.

```
1.  class Person(object):
2.      def __init__(self, first, last):
3.          self.__first = first
4.          self.last = last
5.      def __full_name(self):
6.          return "%s %s" % (self.first, self.last)
7.      def __str__(self):
8.          return "Person: " + self.full_name()
9.
10. person = Person('nikolas', 'mavrogeneiadis')
11. print person.last  #swsto!
12. print person.full_name()  #lathos!
13. print person.first  #lathos!
```

Οι **ιδιωτικές** ιδιότητες και μέθοδοι, μπορούν να χρησιμοποιηθούν μόνο εσωτερικά εντός της κλάσης, αλλά **όχι έξω από αυτή**.

Γενικά ισχύει ότι τα ονόματα όλων των **private μεθόδων και ιδιοτήτων** ξεκινούν με διπλή κάτω παύλα. Οι private μέθοδοι και ιδιότητες δεν μπορούν να αλλάξουν «εκτός» της κλάσης **και ΔΕΝ κληρονομούνται στις υποκλάσεις**.

Μοτίβο	Παράδειγμα	Σημασία
Μονή κάτω παύλα στην αρχή	`_var`	Χρησιμοποιείται για να προειδοποιήσει τον προγραμματιστή ότι αυτή η ιδιότητα δεν πρέπει να αλλάζει εκτός κλάσης (private ιδιότητα).
Μονή κάτω παύλα στο τέλος	`var_`	Χρησιμοποιείται στην περίπτωση που θέλουμε να δηλώσουμε μια μεταβλητή με όνομα ίδιο με κάποιο keyword της Python. Με τον τρόπο αυτό δεν δημιουργούνται conflicts.
Διπλή κάτω παύλα στην αρχή	`__var`	Η python διαχειρίζεται τα ονόματα τέτοιων μεταβλητών με τέτοιο τρόπο, ώστε οι μεταβλητές να μην είναι προσβάσιμες εκτός κλάσης.
Διπλή κάτω παύλα στην αρχική και στο τέλος	`__var__`	Με αυτό τον τρόπο δηλώνονται ειδικοί μέθοδοι και μεταβλητές από την Python. Πρέπει να αποφεύγεται να δίνουμε τέτοιου είδους ονόματα σε δικές μας δηλώσεις!
Μονή κάτω παύλα	`_`	Χρησιμοποιείται για να δώσουμε ένα προσωρινό όνομα σε προσωρινές ή ασήμαντες μεταβλητές (“don’t care”).

Σε μια κλάση, μπορούμε να έχουμε μεθόδους, οι οποίες μπορούν να καλεστούν χωρίς κάποιο αντικείμενο (και χωρίς να χρειάζεται να περάσουμε κάποιο αντικείμενο ως παράμετρο). Αυτή η μέθοδος **ονομάζεται «στατική μέθοδος» (static method)**. Παρακάτω βλέπουμε τους δύο τρόπους που μπορούμε να καλέσουμε μια στατική μέθοδο.

Τρόπος #1: Κλήση μέσω της staticmethod()

```
1.  class Calculator:
2.
3.      def addNumbers(x, y):
4.          return x + y
5.
6.  # create addNumbers static method
7.  Calculator.addNumbers = staticmethod(Calculator.addNumbers)
8.
9.  print 'Product:', Calculator.addNumbers(15, 110)
```

Τρόπος #2: Κλήση μέσω της δήλωσης @staticmethod

```
class Calculator:

    # create addNumbers static method
    @staticmethod
    def addNumbers(x, y):
        return x + y

print 'Product:', Calculator.addNumbers(15, 110)
```

Αντί του @staticmethod, μπορούμε να χρησιμοποιήσουμε το @classmethod, το οποίο στην πρώτη παράμετρο **(cls)** περνάμε την κλάση και όχι το αντικείμενο.

Παράδειγμα

```
class Circle:
    objN = 0
    def __init__(self, r):
        self.r = r
        Circle.objN = Circle.objN+1
    @classmethod
    def instnum(cls):
        return cls.objN

c1 = Circle(5)
c2 = Circle(6)
print Circle.instnum()  # εμφανίζει 2
```

Εκτός από **στατικές μεθόδους**, μπορούμε να έχουμε και **στατικές μεταβλητές**. Οι μεταβλητές αυτές δηλώνονται εσωτερικά στην κλάση, αλλά εκτός μεθόδων.

Παράδειγμα

```
class Fruits(object):
    count = 0  #static variable
    def __init__(self, name, count):
        self.name = name
        self.count = count
        Fruits.count = Fruits.count + count


apples = Fruits("apples", 3)
pears = Fruits("pears", 4)
print apples.count
print pears.count
print Fruits.count
```

Μπορούμε αν χρειάζεται, σε συγκεκριμένα αντικείμενα μιας κλάσης, να προσθέσουμε νέες ιδιότητες. Οι ιδιότητες αυτές ισχύουν μόνο για το συγκεκριμένο αντικείμενο και όχι για όλα τα αντικείμενα της κλάσης. Οπότε αν στις μεθόδους έχουμε τέτοιες ιδιότητες, θα πρέπει να προσέξουμε να τις καλούμε μόνο με αντικείμενα που έχουν αυτές τις μεθόδους.

Παράδειγμα

```
1.  class Employee:
2.      def __init__(self):
3.          self.age = 25
4.      def myprint(self):
5.          print self.name
6.
7.
8.  john = Employee()  # Create an empty employee record
9.  mary = Employee()
10.
11. # Fill the fields of the record
12. john.name = 'John Doe'
13. john.dept = 'computer lab'
14. john.salary = 1000
15.
16. print john.salary
17. print john.age
18. print john.myprint()
19. #print mary.myprint()  λάθος!
```

Κληρονομικότητα

Κάθε κλάση που δημιουργούμε στην Python κληρονομεί αυτομάτως την **built-in** κλάση object. Τι εννοούμε λέγοντας κληρονομεί; Ότι κάθε ιδιότητα και μέθοδος που είναι public στην κλάση object, γίνεται αυτόματα μέλος και της κλάσης που εμείς δημιουργούμε. Η κλάση object είναι μια κλάση που μας προσφέρει κάποιες χρήσιμες «προκαθορισμένες» μεθόδους από την Python και κληρονομούνται αυτόματα. Αν θέλουμε να δημιουργήσουμε μια μέθοδο η οποία **έχει το ίδιο όνομα** με μια μέθοδο που ήδη κληρονομήθηκε από μια άλλη κλάση, τότε στην πραγματικότητα «αντικαθιστούμε» την μέθοδο και θα καλείτε αυτή, έναντι αυτής που κληρονομήσαμε, αυτό ονομάζεται **override**. Οι στατικές μέθοδοι, κληρονομούνται επίσης.

Μπορούμε επίσης, να κληρονομήσουμε και άλλες κλάσεις. **Έστω η κλάση Person που δημιουργήσαμε παραπάνω.** Μπορούμε να φτιάξουμε μια άλλη κλάση, η οποία θα κληρονομεί την κλάση αυτή.

```
1.  class SuperHero(Person):
2.      def __init__(self, first, last, nick):
3.          super(SuperHero, self).__init__(first, last)
4.          self.nick = nick
5.      def nick_name(self):
6.          return "I am %s" % self.nick
```

1. Εδώ η κλάση SuperHero κληρονομεί την κλάση Person (η οποία με την σειρά της όπως είπαμε κληρονομεί την object, οπότε αφού η Person έχεις κληρονομήσει τις μεθόδους της object, αυτές τις μεθόδους τις κληρονομεί και η SuperHero!
2. Κάνουμε **αντικατάσταση** (override) την __init__ όπως και σε κάθε κλάση για να υλοποιήσουμε τον constructor. Ο constructor δέχεται 2 παραμέτρους για το Person και μια νέα μέθοδο την **nick** για την SuperHero.
3. Εδώ χρησιμοποιείται η μέθοδος super για να μπορέσουμε να καλέσουμε την init της Person. Η πρώτη παράμετρος της super είναι το όνομα της κλάσης που υλοποιούμε και το δεύτερο αντικείμενο που αρχικοποιείται (self). Καλούμε την __init__ ώστε να αρχικοποιήσουμε τις 2 παραμέτρους από την Person.
4. Αρχικοποιούμε την επιπλέον παράμετρο για την SuperHero.
5. Η κλάση SuperHero έχει μια νέα μέθοδο με το όνομα nick_name η οποία επιστρέφει ένα string με την ιδιότητα nick.

Ας δούμε ένα παράδειγμα που μπορούμε να υλοποιήσουμε μια τέτοια κλάση.

```
1.  p = SuperHero("Clark", "Kent", "Superman")
2.  print p.nick_name() # I am Superman
3.  print p.full_name() # Clark Kent
```

Πολυμορφισμός

Ο πολυμορφισμός είναι η δυνατότητα να μπορούμε να επιλέγουμε **δυναμικά** ποια μέθοδο θα κληθεί από μια σειρά μεθόδων **με το ίδιο όνομα**, ανάλογα με το αντικείμενο που την καλεί και το πλήθος των παραμέτρων της.

Παράδειγμα 1

```
1.  class Square(object):
2.      def draw(self):
3.          print "I draw a square"
4.
5.  class Circle(object):
6.      def draw(self):
7.          print "I draw a circle"
8.
9.  shapes = [Square(), Circle()]
10. for shape in shapes:
11.     shape.draw()
```

Στο παραπάνω πρόγραμμα δηλώνουμε αρχικά δύο κλάσεις. Μια κλάση Square και μια κλάση Circle. Κάθε μια έχει μια μέθοδο με το όνομα draw. Στην γραμμή 10 δηλώνουμε έναν πίνακα ο οποίος έχει δύο αντικείμενα. Το πρώτο είναι ένα αντικείμενο τύπου Square και το δεύτερο είναι ένα αντικείμενο τύπου Circle. Στις γραμμές 11 και 12 καλούμε επαναληπτικά για κάθε αντικείμενο την draw. Επειδή η Python λειτουργεί **δυναμικά** μπορούμε να χρησιμοποιήσουμε το παραπάνω χωρίς να χρειάζεται να δημιουργήσουμε μια νέα κλάση, ας πούμε Shape.

Παράδειγμα 2

```
1.  class Shape(object):
2.      def __init__(self, base_message):
3.          self.base_message = base_message
4.      def draw(self):
5.          print self.base_message
6.
7.  class Square(Shape):
8.      def __init__(self, base_message, child_message):
9.          super(Square, self).__init__(base_message)
10.         self.child_message = child_message
11.     def draw(self):
12.         print self.base_message, ',', self.child_message
13.
14. class Circle(Shape):
15.     def __init__(self, base_message, child_message):
16.         super(Circle, self).__init__(base_message)
17.         self.child_message = child_message
18.     def draw(self):
19.         print self.base_message, ',', self.child_message
20.
21. shape1 = Square('Base1', 'I am a Sqaure!')
22. shape2 = Circle('Base2', 'I am a Circle!')
23. shapes = [shape1, shape2]
24. for shape in shapes:
25.     shape.draw()
```

Python Tkinter

Η **Tkinter** είναι η πρότυπη βιβλιοθήκη της Python για την υλοποίηση γραφικών διεπαφών χρήστη (User Interfaces), όπως για παράδειγμα η δημιουργία desktop εφαρμογών σε Windows ή σε άλλα λειτουργικά συστήματα.

Η εισαγωγή της βιβλιοθήκης Tkinter στο πρόγραμμα γίνεται με μία από τις παρακάτω εντολές ανάλογα την έκδοση της Python.

```
1.  from Tkinter import *  # Εισαγωγή της βιβλιοθήκης Tkinter στην έκδοση 2.7
2.
3.  import tkinter         # Εισαγωγή της βιβλιοθήκης Tkinter σε νεότερες εκδόσεις 3.x
4.
```

Ασκήσεις Πιστοποίησης

25. ***Δραστηριότητα:***

1. Δημιουργήστε μία νέα κλάση με όνομα roloi και ιδιότητες wr, lep, deft που αντιστοιχούν στις ώρες, τα λεπτά και τα δευτερόλεπτα.
2. Φτιάξτε για την κλάση έναν κατασκευαστή που θα δέχεται σαν παραμέτρους τις ώρες (0-23), τα λεπτά (0-59) και τα δευτερόλεπτα (0-59) και θα τα τοποθετεί στις αντίστοιχες ιδιότητες, αφού γίνει έλεγχος της εγκυρότητας των τιμών. Αν μία τιμή δεν είναι έγκυρη να μπαίνει στην αντίστοιχη ιδιότητα το 0.
3. Φτιάξτε μία μέθοδο που θα εμφανίζει την ώρα με τη μορφή ω:λ:δ.
4. Φτιάξτε μία μέθοδο που θα επιστρέφει το πλήθος των δευτερολέπτων που αντιστοιχούν στην ώρα που δείχνει το ρολόι.
5. Δημιουργήστε ένα νέο αντικείμενο ρολογιού, και εμφανίστε στην οθόνη την ώρα που δείχνει καθώς και το πλήθος των δευτερολέπτων.

```
1.  class roloi(object):
2.      def __init__(self, wr, lep, deft):
3.          if wr >= 0 and wr <= 23:
4.              self.wr = wr
5.          else:
6.              self.wr = 0
7.          if lep >= 0 and lep <= 59:
8.              self.lep = lep
9.          else:
10.             self.lep = 0
11.         if deft >= 0 and deft <= 59:
12.             self.deft = deft
13.         else:
14.             self.deft = 0
15.
16.     def showTime(self):
17.         print str(self.wr) + ":" + str(self.lep) + ":" + str(self.deft)
18.
19.     def totalDef(self):
20.         return self.wr*60*60 + self.lep*60 + self.deft
21.
22.
23. myroloi = roloi(10, 26, 30)
24. myroloi.showTime()
25. print myroloi.totalDef()
```

26. *Δραστηριότητα:*

1. Δημιουργήστε μία νέα κλάση με όνομα metr και μία ιδιωτική (private) ιδιότητα την i.
2. Δημιουργήστε επίσης δύο ιδιότητες κλάσης με όνομα ano και kato και δώστε στην μεταβλητή ano τιμή μεγαλύτερη από αυτή της kato.
3. Φτιάξτε έναν κατασκευαστή για την κλάση που θα παίρνει σαν παράμετρο την τιμή του μετρητή και θα την αποδίδει στην ιδιότητα i, αφού πρώτα ελέγξει ότι βρίσκεται μεταξύ του πάνω και κάτω ορίου. Αν αυτό δεν συμβαίνει θα καταχωρεί στην i την τιμή του πλησιέστερου ορίου.
4. Δημιουργείστε μία μέθοδο auxisi που θα αυξάνει κατά 1 την τιμή του μετρητή εφόσον αυτή δεν ξεπερνά το πάνω όριο.
5. Δημιουργείστε μία μέθοδο meiosi που θα μειώνει κατά 1 την τιμή του μετρητή εφόσον αυτή δεν ξεπερνά το κάτω όριο.
6. Δημιουργήστε μια μέθοδο που θα εκτυπώνει την τιμή του μετρητή.
7. Δημιουργήστε ένα νέο αντικείμενο μετρητή, αυξήστε την τιμή του και εμφανίστε την.

```
1.  class metr(object):
2.      def __init__(self, i):
3.          self.kato = 10
4.          self.ano = 15
5.          if i >= self.kato and i <= self.ano:
6.              self.__i = i
7.          else:
8.              def_from_ano = abs(self.ano - i)
9.              def_from_kato = abs(self.kato - i)
10.             if def_from_ano >= def_from_kato:
11.                 self.__i = self.kato
12.             else:
13.                 self.__i = self.ano
14.
15.     def auxisi(self):
16.         temp = self.__i + 1
17.         if temp <= self.ano:
18.             self.__i = self.__i + 1
19.
20.     def meiosi(self):
21.         temp = self.__i - 1
22.         if temp => self.kato:
23.             self.__i = self.__i - 1
24.
25.     def printCounter(self):
26.         print "Counter: ", self.__i
27.
28.
29. my_metr = metr(11)
30. my_metr.auxisi()
31. my_metr.printCounter()
```

27. *Δραστηριότητα:*

1. Δημιουργήστε μία νέα κλάση για ορθογώνια παραλληλόγραμμα με όνομα paral και δύο ιδιότητες, τις mikos και platos.
2. Δημιουργήστε ένα κατασκευαστή που θα δέχεται σαν παραμέτρους δύο τιμές και θα τις εισάγει στις αντίστοιχες ιδιότητες. Αν κάποια από τις τιμές είναι μικρότερη του 0, να εισάγει το 0.
3. Δημιουργήστε μια μέθοδο με όνομα emvado που θα επιστρέφει το εμβαδό του παραλληλογράμμου (μήκος x πλάτος).
4. Δημιουργήστε μια νέα κλάση για ορθογώνια παραλληλεπίπεδα με όνομα pepipedo που θα κληρονομεί από την κλάση paral και επιπλέον θα ορίζει την ιδιότητα ypsos.
5. Δημιουργήστε ένα κατασκευαστή για την νέα κλάση που θα δέχεται σαν παραμέτρους τρεις τιμές και θα τις εισάγει στις αντίστοιχες ιδιότητες. Για την εισαγωγή των δύο πρώτων θα πρέπει να χρησιμοποιεί τον κατασκευαστή της κλάσης paral. Όπως και πριν να γίνεται έλεγχος εγκυρότητας και για το ύψος.
6. Δημιουργήστε μια νέα μέθοδο με όνομα ogos που θα επιστρέφει τον όγκο του παραλληλεπίπεδου. Ο όγκος είναι το εμβαδόν της βάσης επί το ύψος. Για το εμβαδό της βάσης θα πρέπει να χρησιμοποιηθεί η μέθοδος emvado της κλάσης paral
7. Δημιουργείστε ένα νέο παραλληλεπίπεδο και εμφανίστε στην οθόνη τον όγκο του

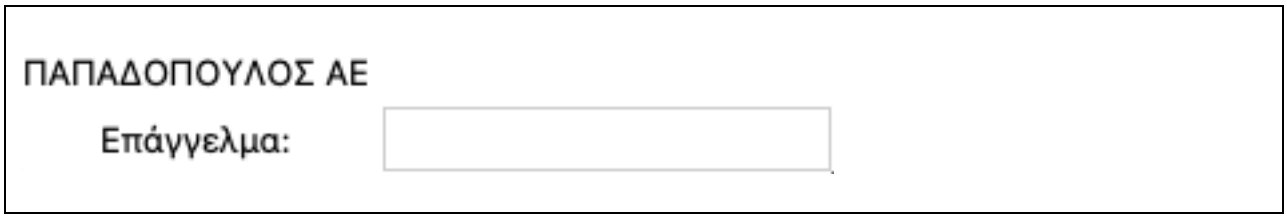
```
1.  class paral(object):
2.      def __init__(self, mikos, platos):
3.          if mikos < 0:
4.              self.mikos = 0
5.          else:
6.              self.mikos = mikos
7.          if platos < 0:
8.              self.platos = platos
9.          else:
10.             self.platos = platos
11.
12.     def emvado(self):
13.         return self.platos * self.mikos
14.
15. class pepipedo(paral):
16.     def __init__(self, mikos, platos, upsos):
17.         super(pepipedo, self).__init__(mikos, platos)
18.         if upsos < 0:
19.             self.upsos = 0
20.         else:
21.             self.upsos = upsos
22.
23.     def ogos(self):
24.         return self.emvado()*self.upsos
25.
26.
27. my_cube = pepipedo(2, 3, 10)
28. print my_cube.ogos()
```

34. *Δραστηριότητα:*

Χρησιμοποιήστε το άρθρωμα (module) Tkinter της Python, για να τη σελίδα διεπαφής της εταιρείας σας με τα εξής στοιχεία:

1. Να εμφανίζεται στην αρχή της σελίδας διεπαφής, το όνομα της της εταιρείας σας, με κεφαλαίους ελληνικούς χαρακτήρες (το όνομα παραμένει συνεχώς μέχρι να κλείσει η σελίδα).
2. Να εμφανίζεται μια φόρμα μέσω της οποίας να εισάγεται το επάγγελμα του χρήστη.

```
1.  # -*- coding: utf-8 -*-
2.
3.  from Tkinter import *  # Εισαγωγή βιβλιοθήκης Tkinter
4.  master = Tk()  # Δημιουργία παραθύρου
5.  Label(master, text='ΠΑΠΑΔΟΠΟΥΛΟΣ ΑΕ').grid(row=0)  # Καθορισμός Label. Εμφανίζεται στην
     πρώτη γραμμή και πρώτη στήλη.
6.  Label(master, text='Επάγγελμα:').grid(row=1)  # Εμφανίζεται στην δεύτερη γραμμή πρώτη σ
    τήλη
7.  e1 = Entry(master)  # Text input μιας γραμμής από τον χρήστη.
8.  e1.grid(row=1, column=1)  # Εμφανίζεται στην δεύτερη γραμμή και δεύτερη στήλη
9.  mainloop() # συνάρτηση η οποία εμφανίζει το παράθυρο και το κρατάει ανοιχτό
```



35. *Δραστηριότητα:*

Η εταιρεία σας έχει ήδη μια μισοτελειωμένη σελίδα στην οποία εμφανίζεται το λογότυπό της και μια φόρμα για εισαγωγή του επαγγέλματος του χρήστη. Χρησιμοποιήστε το άρθρωμα (module) Tkinter της Python, για να προσθέσετε στη σελίδα διεπαφής της εταιρείας σας τα εξής στοιχεία:

1. Να δημιουργήσετε πλήκτρο (κουμπί) το οποίο όταν πατηθεί σταματά την εισαγωγή χαρακτήρων του επαγγέλματος (τερματισμός προγράμματος)
2. Να δημιουργήσετε πλήκτρο (κουμπί) το οποίο όταν πατηθεί να εμφανίζει το επάγγελμά που πληκτρολογήθηκε
3. Να δημιουργήσετε ένα πλήκτρο (κουμπί), με κείμενο «ΕΞΟΔΟΣ ΑΠΟ ΤΗ ΣΕΛΙΔΑ ΜΑΣ», που όταν πατηθεί κλείνει τη σελίδα της εταιρείας σας

```
5.  # -*- coding: utf-8 -*-
6.
7.  from Tkinter import *  # Εισαγωγή βιβλιοθήκης Tkinter Python 2.7!
8.  import tkMessageBox
9.
10.
11. def b1function():
12.     state = e1['state']
13.     if state == 'normal':
14.         e1.configure(state='disabled')
```

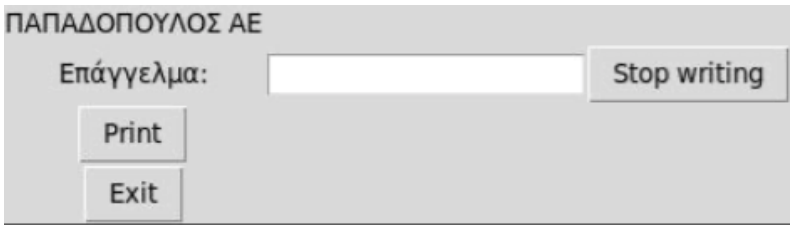
```
    elif state == 'disabled':
        e1.configure(state='normal')

def b2function():
    tkMessageBox.showinfo("Επάγγελμα", "Το επάγγελμα που πληκτρολογήσατε είναι: " + e1.get().encode('utf-8'))

def close_window():
    master.destroy()  # κλείσιμο εφαρμογής


master = Tk()  # Δημιουργία παραθύρου
l1 = Label(master, text='ΠΑΠΑΔΟΠΟΥΛΟΣ ΑΕ')  # Καθορισμός Label.
l1.grid(row=0)  # Εμφανίζεται στην πρώτη γραμμή και πρώτη στήλη.
l2 = Label(master, text='Επάγγελμα:')
l2.grid(row=1)  # Εμφανίζεται στην δεύτερη γραμμή πρώτη στήλη
e1 = Entry(master)  # Text input μιας γραμμής από τον χρήστη.
e1.grid(row=1, column=1)  # Εμφανίζεται στην δεύτερη γραμμή και δεύτερη στήλη
b1=Button(master, text='Stop writing', command=b1function)
b1.grid(row=1, column=2)
b2=Button(master, text='Print', command=b2function)
b2.grid(row=2, column=0)
b3=Button(master, text='Exit', command=close_window)
b3.grid(row=3, column=0)


mainloop()  # συνάρτηση η οποία εμφανίζει το παράθυρο και το κρατάει ανοιχτό
```



Ερωτήσεις Πιστοποίησης

103. Χαρακτηρίστε τις ακόλουθες προτάσεις ως Σωστές (Σ) ή Λανθασμένες (Λ).

α. Ο αντικειμενοστραφής προγραμματισμός δίνει προτεραιότητα στις διαδικασίες έναντι των εννοιών
β. Στον αντικειμενοστραφή προγραμματισμό οι μέθοδοι αντιπροσωπεύουν τις ενέργειες που μπορεί να κάνει ένα αντικείμενο
γ. Στον αντικειμενοστραφή προγραμματισμό οι ιδιότητες ενός αντικειμένου κωδικοποιούνται με χρήση μεταβλητών
δ. Στον αντικειμενοστραφή προγραμματισμό μια μέθοδος μπορεί να τροποποιήσει την τιμή των ιδιοτήτων ενός αντικειμένου

104. Χαρακτηρίστε τις ακόλουθες προτάσεις ως Σωστές (Σ) ή Λανθασμένες (Λ).

α. Μια Κλάση είναι ένα σύνολο αντικειμένων ίδιου τύπου
β. Κληρονομικότητα ονομάζουμε την δυνατότητα απόδοσης αρχικών τιμών στις ιδιότητες αντικειμένων που κατασκευάζουμε
γ. Ένας κατασκευαστής είναι μια μέθοδος που χρησιμοποιείται για τη δημιουργία νέων αντικειμένων και την απόδοση αρχικών τιμών στις ιδιότητές τους.
δ. Με τον όρο πολυμορφισμός εννοούμε τη δυνατότητα δημιουργίας αντικειμένων που οι ιδιότητές τους να έχουν διαφορετική τιμή

105. Ποιος είναι ο σωστός τρόπος για τον ορισμό στην Python μιας κλάσης με όνομα Clock και ιδιότητες h, m, s;

α.
```
class Clock:
  def init (self,h,m,s):
    Κώδικας κατασκευαστή
```
β.
```
class Clock:
  def __init__(self,h,m,s):
    Κώδικας κατασκευαστή
```
γ.
```
class Clock:
  __init__(self,h,m,s):
    Κώδικας κατασκευαστή
```
δ.
```
class Clock:
  def __create__(self,h,m,s):
    Κώδικας κατασκευαστή
```

106. Ποιος είναι ο σωστός τρόπος να δημιουργήσουμε ένα νέο αντικείμενο για την παρακάτω κλάση;

```
class Circle:
  def __init__(self,x,y,r):
    self.x=x
    self.y=y
    self.r=r
```

α. `c1=Circle(self,10,10,15)`
β. `Circle(c1,10,10,15)`
γ. `c1=Circle(10,10)`
δ. `c1=Circle(10,10,15)`

107. Δοθέντος του ορισμού της κλάσης Circle και της δημιουργίας του αντικειμένου c1, με ποια έκφραση θα εμφανιστεί στην οθόνη το εμβαδό του αντικειμένου που υπολογίζεται από τη μέθοδο area();

```
class Circle:
  def __init__(self,r):
    self.r=r
  def area(self):
    return 3.1415*self.r*self.r

c1=Circle(5)
```

α. `print Circle.c1.area()`
β. `print c1.area()`
γ. `print area(c1)`
δ. `print area(Circle.c1)`

108. Ποιες από τις παρακάτω προτάσεις που αφορούν τις στατικές μεθόδους είναι Σωστές και ποιες Λανθασμένες;

α. Μπορούμε να χρησιμοποιήσουμε το διακριτικό @staticmethod στον ορισμό τους
β. Απαιτούν σαν πρώτη παράμετρο το αντικείμενο (self)
γ. Μπορούν να κληρονομηθούν από υποκλάσεις
δ. Έχουν το ίδιο αποτέλεσμα ανεξάρτητα του ποιο στιγμιότυπο τις καλεί.

109. Χαρακτηρίστε ως Σωστές (Σ) ή Λανθασμένες (Λ). τις ακόλουθες προτάσεις.

α. Με την κληρονομικότητα υλοποιείται μια ιεραρχία αντικειμένων από το γενικότερο προς το ειδικότερο
β. Μια κλάση που κληρονομεί ιδιότητες και μεθόδους από μία άλλη λέγεται υποκλάση
γ. Στιγμιότυπα των υπερκλάσεων μπορούν να χρησιμοποιηθούν εκεί που απαιτούνται στιγμιότυπα των υποκλάσεων
δ. Μία υπερκλάση περιέχει τις μεθόδους και τις ιδιότητες μιας υποκλάσης και μπορεί επιπλέον να ορίσει και τις δικές της

110. Με την έκφραση: Class A(B):

α. Δημιουργούμε ένα δεύτερο όνομα (B) για την ήδη υπάρχουσα κλάση A.
β. Δημιουργούμε ένα δεύτερο όνομα (A) για την ήδη υπάρχουσα κλάση B.
γ. Ορίζεται η κλάση B σαν υποκλάση της κλάσης A.
δ. Ορίζεται η κλάση A σαν υποκλάση της κλάσης B.

111. Τι θα εμφανίσει στην οθόνη το ακόλουθο πρόγραμμα;

```
class A(object):
  def __init__(self,x):
    self.x = x

class B(A):
  def __init__(self,x,y):
    super(B,self).__init__(y)
    self.y = x

b = B(1,2)
print b.x,',', b.y
```

α. 2 , 1
β. 1 , 2
γ. 0 , 1
δ. 1 , 0

112. Τι θα εμφανίσει στην οθόνη το ακόλουθο πρόγραμμα;

```
class A(object):
  def __init__(self):
    self.x = 1
  def mod(self):
    self.x = 10

class B(A):
  def __init__(self):
    super(B,self).__init__()
  def mod(self):
    self.x=self.x+1
    return self.x

a = B()
print a.mod()
```

α. 11
β. 1
γ. 2
δ. 10

113. Ποια έννοια περιγράφει ο ακόλουθος ορισμός;

«Η δυνατότητα των αντικειμενοστραφών μεταγλωττιστών/διερμηνευτών να αποφασίζουν δυναμικά ποια, από της ομώνυμες μεθόδους, είναι κατάλληλη να κληθεί ανάλογα με το αντικείμενο που την καλεί ή τον τύπο και τον αριθμό των παραμέτρων της»

α. Ενθυλάκωση
β. Κληρονομικότητα
γ. Αρχικοποίηση
δ. Πολυμορφισμός

114. Ποια από τα παρακάτω είναι πλεονεκτήματα της ενθυλάκωσης;

α. Επιτρέπει τη χρήση αντικειμένων χωρίς να είναι γνωστός ο τρόπος υλοποίησής τους
β. Επιτρέπει των έλεγχο των τιμών που παίρνουν οι ιδιότητες των αντικειμένων
γ. Επιτρέπει την άμεση αλλαγή των ιδιοτήτων των αντικειμένων
δ. Επιτρέπει τη χρήση των αντικειμένων μόνο μέσα των μεθόδων τους

115. Ποιες από τις ιδιότητες της ακόλουθης κλάσης μπορούν να τροποποιηθούν άμεσα από κώδικα που βρίσκεται εκτός της κλάσης;

```
class A():
   def __init__(self):
      self.x = 0
      self.__y = 0
      self.__k=0
      self.z=0

o = A()
```

α. Οι x και y
β. Οι y και z
γ. Οι k και y
δ. Οι x και z

130. Συμπληρώστε από τις παρακάτω τη λέξη που λείπει.

Η _____ είναι η πρότυπη βιβλιοθήκη της Python για την υλοποίηση γραφικών διεπαφών

α. GUI
β. TKINTER
γ. APIs
δ. ATM

135. Αντιστοιχίστε κάθε γράμμα της στήλης Α με το αριθμό της σωστής πρότασης από τη στήλη Β. Δεν θα χρησιμοποιηθούν όλες οι προτάσεις της στήλης Β.

ΣΤΗΛΗ Α: μέθοδοι κλάσης URL (JAVA)	ΣΤΗΛΗ Β (περιγραφή λειτουργίας)
α. getProtocol( )	1. Επιστρέφει το όνομα του σελιδοδείκτη της σελίδας
β. getHost( )	2. Επιστρέφει το όνομα της σελίδας
γ. getFile( )	3. Επιστρέφει τον αριθμό της θύρας
δ. getRef( )	4. Επιστρέφει το όνομα του δικτυακού τόπου
	5. Επιστρέφει το όνομα του πρωτοκόλλου

136. Αντιστοιχίστε όλους τους αριθμούς των προτάσεων της στήλης Β στα γράμματα της στήλης Α. Δεν θα χρησιμοποιηθούν όλα τα στοιχεία της στήλης Α.

ΣΤΗΛΗ Α: κλάσεις JAVA	ΣΤΗΛΗ Β (περιγραφή λειτουργίας)
α. ServerSocket	1. Σύνδεση από την πλευρά του πελάτη
β. Socket	2. Σύνδεση από την πλευρά του διακομιστή
γ. ClientSocket	3. Ακούει ("διαβάζει") πακέτα δεδομένων και όχι συνδέσεις

137. Συμπληρώστε τις λέξεις που λείπουν, αντιστοιχίζοντας τους αριθμούς των κενών με τα γράμματα των λέξεων. Δεν θα χρησιμοποιηθούν όλες οι λέξεις.

Στα UDP Datagrams η επικοινωνία που αναπτύσσεται είναι _____(1), δηλαδή δεν υπάρχει κάποια–_____(2) μεταξύ των αποστολέα και του παραλήπτη. Κάθε πακέτο που στέλνουμε πρέπει να περιέχει την διεύθυνση _____(3).

α. URL
β. ασύγχρονη
γ. σταθερή σύνδεση
δ. IP
ε. ασταθής σύνδεση

140. Χαρακτηρίστε τις ακόλουθες προτάσεις ως Σωστές (Σ) ή Λανθασμένες (Λ).

α. Το Ant (Another Neat Tool), του Eclipse, είναι εργαλείο αυτοματοποίησης της διαδικασίας του χτισίματος μιας εφαρμογής μόνο για τη JAVA.
β. Το μειονέκτημα του Ant (Another Neat Tool), του Eclipse, είναι η μικρή ταχύτητα ανάπτυξης μιας Java εφαρμογής.
γ. Το Ant (Another Neat Tool), του Eclipse, καθιστά ταχύτατη την προσθήκη νέων βιβλιοθηκών.
δ. Το Ant (Another Neat Tool), του Eclipse, συγκεντρώνει μεταγλωττίζει τον κώδικά μας και «πακετάρει» όλα τα αρχεία που χρειάζονται σε ένα τελικό αρχείο.

178. Ποιο είναι το λάθος στο ακόλουθο τμήμα κώδικα;

```
class Clock:
  def __init__(self,h,m,s):
    self.h=h
    self.m=m
  def print_time(self):
    print self.h,':',self.m, ':',self.s

c=Clock(9,35,41)
c.print_time()
```

α. Κατά τη δημιουργία του αντικειμένου δεν θα πρέπει να δοθεί η τελευταία παράμετρος.
β. Στον κατασκευαστή δεν θα πρέπει να δοθεί σαν παράμετρος το s.
γ. Στον κατασκευαστή δεν αρχικοποιείται η ιδιότητα s, αν και αυτή χρησιμοποιείται από την μέθοδο print_time().
δ. Κατά τη δημιουργία του αντικειμένου λείπει η λέξη self σαν πρώτη παράμετρος.

185. Ποιο θα είναι το αποτέλεσμα της εκτέλεσης του ακόλουθου τμήματος κώδικα;

```
class Counter:
  def __init__(self,i):
    self.i=i
  def displ(self):
    print self.i

c=Counter()
c.displ()
```

α. Θα εκτελεσθεί χωρίς να εμφανίσει κάτι στην οθόνη.
β. Θα εμφανίσει την τιμή 0.
γ. Θα εμφανίσει μήνυμα λάθους.
δ. Θα εμφανίσει την τιμή 0 και στη συνέχεια μήνυμα λάθους.

186. Επιλέξτε την σωστή έκφραση με την οποία θα υλοποιήσετε για την κλάση Circle μία μέθοδο που θα μετρά το πλήθος των στιγμιότυπων της κλάσης.

```
class Circle:
  objN=0
  def __init__(self,r):
    self.r=r
    Circle.objN=Circle.objN+1
```

α.
```
@classmethod
def instnum(cls):
  return cls.objN
```

β.
```
@classmethod
def instnum(self):
  return cls.objN
```

γ.
```
@staticmethod
def instnum(cls):
  return cls.objN
```

δ.
```
@staticmethod
def instnum():
  return cls.objN
```

187. Ποια η διαφορά στον ορισμό της κλάσης test με τους ακόλουθους δύο τρόπους;

```
i. Class test:
ii. Class test(object):
```

α. Δεν υπάρχει διαφορά.
β. Με τον πρώτο τρόπο δεν θα μπορούμε να δημιουργήσουμε υποκλάσεις της test.
γ. Με τον πρώτο τρόπο οι υποκλάσεις της test δεν θα μπορούν να καλούν τις μεθόδους της υπερκλάσης.
δ. Με τον πρώτο τρόπο οι υποκλάσεις της test κληρονομούν μόνο τις ιδιότητες και όχι τις μεθόδους της κλάσης test.

188. Συμπληρώστε τη γραμμή κώδικα που θα πρέπει να εισαχθεί στη γραμμή [7] για να γίνει κλήση του κατασκευαστή της κλάσης Α.

```
1. class A(object):
2.     def __init__(self,i):
3.         self.i = i
4.
5. class B(A):
6.     def __init__(self,i):
7.
8.         self.j = j
```

α . `super(B).__init__(i)`

β . `super(B,self).__init__(i)`

γ . `A.__init__(i)`

δ . `A.__init__(i,j)`

190. Ποια από τα παρακάτω είναι πλεονεκτήματα της ενθυλάκωσης;

α. Επιτρέπει τη χρήση αντικειμένων χωρίς να είναι γνωστός ο τρόπος υλοποίησής τους.
β. Επιτρέπει των έλεγχο των τιμών που παίρνουν οι ιδιότητες των αντικειμένων.
γ. Επιτρέπει την άμεση αλλαγή των ιδιοτήτων των αντικειμένων.
δ. Επιτρέπει τη χρήση των αντικειμένων μόνο μέσα των μεθόδων τους.

191. Τι θα εμφανισθεί με την εκτέλεση του ακόλουθου τμήματος κώδικα;

```
class A:

    def __init__(self):
        self.__b = 1

    def getb(self):
        return self.__b

a = A()
a.__b=15
print a.__b,',', a.getb()
```

α. 15,1
β. 1,15
γ. 1,1
δ. 15,15

208. Στον παρακάτω ελλιπή κώδικα JAVA λαμβάνονται δεδομένα από μια σύνδεση URL, τα οποία εμφανίζονται στην οθόνη γραμμή-γραμμή (κλάση URLConnection). Αντιστοιχίστε τους αριθμούς των γραμμών που έχουν κενό με τα γράμματα των λέξεων. Δεν θα χρησιμοποιηθούν όλες οι λέξεις

```
1. URL wikipedia = _______ URL("http://www.wikipedia.org/");
2. URLConnection wk = wikipedia.openConnection();
3. BufferedReader in = new BufferedReader(new InputStreamReader(____.getIn
   putStream()));
4. String inputLine;
5. _________ ((inputLine = in.readLine()) != null)
6. System.out.println(inputLine);
7. in.close();
```

α. wk
β. new
γ. open
δ. while

209. Στον παρακάτω ελλιπή κώδικα JAVA γίνεται μια ασύγχρονη σύνδεση (κλάση socket) για αποστολή δεδομένων στο διακομιστή, από τη μεριά του πελάτη). Αντιστοιχίστε τους αριθμούς των γραμμών που έχουν κενό με τα γράμματα των λέξεων. Δεν θα χρησιμοποιηθούν όλες οι λέξεις.

```
1. Socket socket = new _____(IPaddress, 8128);
2. PrintWriter out = CreateWriter(socket);
3.
4. String message =  in._____ ();
5. while (!message.equals("JAVA"));
6.          .println(message);
7. message = in.readLine(); }
```

α. out
β. new
γ. Socket
δ. readLine

210. Στον παρακάτω ελλιπή κώδικα JAVA γίνεται μια ασύγχρονη σύνδεση για αμφίδρομη επικοινωνία (κλάση DatagramSocket) για αποστολή δεδομένων στο διακομιστή και αναμονής απάντησης από αυτόν. Αντιστοιχίστε τους αριθμούς των γραμμών που έχουν κενό με τα γράμματα των λέξεων. Δεν θα χρησιμοποιηθούν όλες οι λέξεις.

```
1. DatagramSocket server = new DatagramSocket( );
2. int size = 512;
3. _____ packet;
4. packet = new DatagramPacket( message, _____, address, port );
5. server.send( _____);
6. server.receive( packet );
```

α. packet
β. DatagramPacket
γ. size
δ. address

211. Αντιστοιχίστε κάθε γράμμα της στήλης Α με το αριθμό της σωστής πρότασης από τη στήλη Β. Δεν θα χρησιμοποιηθούν όλες οι προτάσεις της στήλης Β.

ΣΤΗΛΗ Α: Εργαλεία ολοκλήρωσης εφαρμογής	ΣΤΗΛΗ Β (περιγραφή λειτουργίας)
α. JUnit	1. Αυτόματη τεκμηρίωση σημαντικού τμήματος της εφαρμογής
β. javadoc	2. Αυτοματοποίηση των δοκιμών ελέγχου
γ. Maven	3. Αυτοματοποίηση της διαδικασίας διόρθωσης των συντακτικών λαθών της εφαρμογής
	4. Αυτοματοποίηση της διαδικασίας του χτισίματος της εφαρμογής

Βιβλιογραφία - Πηγές

- Ο επίσημος δικτυακός τόπος της Python
  https://www.python.org/

- W3schools - Python
  https://www.w3schools.com/python/

- Python Tutor – Visualize your code and get live help
  http://www.pythontutor.com/visualize.html#mode=edit

- JavaTpoint – Python Tutorial
  https://www.javatpoint.com/python-tutorial

- Python Tutorial – A Comprehensive Guide to Learn Python Programming
  https://www.edureka.co/blog/python-tutorial/

- Βιβλίο Μαθητή Γ' Τάξη ΕΠΑΛ - Προγραμματισμός Υπολογιστών με Python
  http://iep.edu.gr/images/IEP/EPISTIMONIKI_YPIRESIA/Epist_Monades/B_Kyklos/Tee/2017/GEpal/G_Epal_Progr_Ypologiston_Simeioseis_Mathiti_2h_ekdosi_2017.pdf

- Βιβλίο: Ειδικά Θέματα στον Προγραμματισμό
  http://iep.edu.gr/images/IEP/EPISTIMONIKI_YPIRESIA/Epist_Monades/B_Kyklos/Tee/2016/GEpal/2016_GEpal_SpPr_Net.pdf

- ΕΟΠΠΕΠ - Θέματα Εξετάσεων Πιστοποίησης Αποφοίτων Μεταλυκειακού Έτους - Τάξης Μαθητείας των ΕΠΑ.Λ. της ειδικότητας "Τεχνικός Εφαρμογών Πληροφορικής"
  https://www.eoppep.gr/images/EPAL/epal_texnikos_plirofo.pdf

Video

- Top 10 Programming Languages In 2020 | Best Programming Languages To Learn In 2020 | Edureka (11 min)
  https://www.youtube.com/watch?v=mUxS-35qO44

- Σειρά 43 μαθημάτων! - Python greek, μαθήματα στα ελληνικά 1 (Εισαγωγή)
  https://www.youtube.com/watch?v=_lSTjapkwl4&list=PLlae17WKTjHnNivSY3r7ZWwzIpr8EcQp2&index=1

- Python Tutorial for Beginners | Python Programming Language Tutorial | Python Training | Edureka (1h 46min)
  https://www.youtube.com/watch?v=N0lxfilGfak

Απαντήσεις

103. Λάθος, Σωστό, Σωστό, Σωστό
104. Λάθος, Λάθος, Σωστό, Λάθος
105. β
106. δ
107. β
108. Σωστό, Λάθος, Σωστό, Σωστό
109. Σωστό, Σωστό, Λάθος, Λάθος
110. Λάθος, Λάθος, Λάθος, Σωστό
111. α
112. γ
113. δ
114. β, δ (ίδια με την 190)
115. δ

130. β
135. α5, β4, γ2, δ1
136. α2, β3, γ1
137. 1 → β, 2 → γ, 3 → δ
140. α →Σωστό, β→ Λάθος, γ → Σωστό, δ→Σωστό

178. γ
185. γ
186. α
187. γ
188. β
189. α
190. β, δ (ίδια με την 114)
191. α

208. 1→β, 2→α, 5→δ
209. 1→γ, 4→δ, 6→α
210. 3→β, 4→γ, 5→α
211. α→2, β→1, γ→4