

ΘΕΜΑΤΑ & ΛΥΣΕΙΣ ΕΞΕΤΑΣΕΩΝ

ΣΕΠΤΕΜΒΡΙΟΣ 2025

Διάρκεια εξέτασης: 2 ώρες

Ερώτημα 1 (1.5 μονάδες)

Τι θα εμφανίσει το καθένα από τα παρακάτω τμήματα κώδικα; Δικαιολογήστε την απάντησή σας **εν συντομία**:

2.1)

```
i = 8
while i<=12:
    print(i)
    i+=3
```

Θα εμφανίσει:

8

11

Το i θα πάρει τιμές 8, 11, 14 το 14 είναι >12
οπότε θα σταματήσει πριν το τυπώσει

2.2)

```
a = [2,4,5,2]
b = a
del b[3]
print(a[-1]+b[-1])
```

Θα εμφανίσει:

10

Το a και το b συνδέονται by reference.
σβήνοντας το τέταρτο στοιχείο του b θα
σβηστεί και αυτό του a. Άρα και το a και το b
θα είναι [2,4,5]. Οπότε 5+5=10.

2.3)

```
d = "Η Python είναι εύκολη!"
d[22]=";"
print(d)
```

Θα εμφανίσει:

Error.

Δεν μπορούμε να τροποποιήσουμε έναν
χαρακτήρα ενός string, είναι immutable.

2.4)

```
a=["wind","fire","air"]
a.sort()
print(a[0]+"y")
```

Θα εμφανίσει:

airy

Με την ταξινόμηση το air μπαίνει πρώτο στη
λίστα. Η πρόσθεση string (+) κάνει συνένωση.

2.5)

```
a = []
b = 40
while b % 2 == 0:
    a.append(b)
    b = b/2
print(a)
```

Θα εμφανίσει:

[40 20.0 10.0]

Το % είναι το mod, οπότε 5%2==1, και εκεί
θα σταματήσει η επανάληψη

2.6)

```
a = 5
b = 7
c = 12
print(not c<10 or a>6 and b>5)
```

Θα εμφανίσει:

True

(not c<10)→True, (a>6 and b>5)→False, (True or
False)→True)

Ερώτημα 2 (1.5 μονάδες)

Να γραφεί κώδικας σε Python που:

- 1) Θα ζητάει πενήντα ακέραιους αριθμούς από το χρήστη
- 2) Θα αποθηκεύει σε μια λίστα μόνο όσους από αυτούς διαιρούνται ακριβώς με το 3
- 3) Στο τέλος, θα εμφανίζει τους **μοναδικούς** αριθμούς της λίστας **χωρίς να επαναλαμβάνει τους ίδιους**. Δηλαδή αν η λίστα περιέχει τους [3, 9, 6, 9] θα εμφανίζει τους 3, 9, 6.
- 4) Θα εμφανίζει το άθροισμα των μοναδικών αριθμών αυτών. (Στο παράδειγμα: 18)

Απάντηση (ενδεικτική):

```
L=[]
for i in range(50):
    a = int(input("Δώσε έναν ακέραιο:"))
    if(a % 3 == 0):
        L.append(a)
print(set(L))
print(sum(set(L)))
```

Ερώτημα 3 (1 μονάδα)

Δίνεται μια λίστα **A** που περιέχει string με γυναικεία ονόματα. Να γραφεί μια γραμμή κώδικα *list comprehension* (“υπολογιζόμενες λίστες”) που να δημιουργεί μια νέα λίστα **B** που θα περιέχει τα ονόματα της λίστας που έχουν μήκος πάνω από 5 χαρακτήρες, μαζί με το άρθρο “Η”.

Π.χ. αν **A**=[“Μαρία”, “Δήμητρα”, “Ιόλη”, “Νικολέττα”, “Νεφέλη”],

η **B** θα είναι [“Η Δήμητρα”, “Η Νικολέττα”, “Η Νεφέλη”].

Δε χρειάζεται εντολή εισόδου, θεωρήστε ότι η **A** υπάρχει ήδη.

Απάντηση:

```
b=["Η "+x for x in A if len(x)>5]
```

Ερώτημα 4 (2 μονάδες)

A) Να γραφεί μια μέθοδος σε Python με όνομα **mult_list** που θα δέχεται ως είσοδο μια λίστα **L** και έναν αριθμό **n**, και θα επιστρέφει μια νέα λίστα που θα περιέχει τα στοιχεία της **L** πολλαπλασιασμένα επί το **n**. Δηλαδή αν δώσουμε στη **mult_list** την [3,4,2] και το 3 θα επιστρέψει: [9, 12, 6].

B) Να γραφεί μια μέθοδος σε Python με όνομα **double_and_select** που θα δέχεται ως είσοδο μια λίστα, θα διπλασιάζει (δηλ. θα πολλαπλασιάζει επί 2) τις τιμές των στοιχείων της με τη χρήση της **mult_list**, και θα εμφανίζει μόνο όσα από τα διπλασιασμένα στοιχεία είναι μικρότερα του 20. Δηλαδή, αν δώσουμε στην **double_and_select** την [4, 14, 9] θα εμφανίσει [8,18].

Γ) Καλέστε την **double_and_select** με είσοδο τη λίστα [4,14,9]

Απάντηση (ενδεικτική):

Α)

```
def multiply_list(L,n):  
    # Δημιουργώ την κενή λίστα που θα επιστρέψω  
    L_m = []  
    # Για κάθε στοιχείο της L...  
    for l in L:  
        # ...το προσαρτώ στην L_m πολλαπλασιασμένο επί n  
        L_m.append(l*n)  
    # Εναλλακτικά, θα μπορούσα να κάνω το ίδιο με comprehension:  
    L_m = [l*n for l in L]  
    return(L_m)
```

Β)

```
def double_and_select(L):  
    # Καλώ την multiply_list με τη λίστα και το 2  
    L_2 = multiply_list(L,2)  
    #Εμφανίζω τα στοιχεία του αποτελέσματος που είναι <20  
    for l in L_2:  
        if (l<20):  
            print(l)  
    # Εναλλακτικά, θα μπορούσα να κάνω το ίδιο με concatenation:  
    print([l for l in multiply_list(L,2) if l<20])  
    # Και με τους δυο τρόπους η μέθοδος τυπώνει τα αποτελέσματά της  
    # και δεν επιστρέφει τίποτα
```

Γ)

```
# Καλώ τη συνάρτηση  
l = double_and_select([4,14,9])
```

Ερώτημα 5 (1 μονάδα)

Δίνεται το παρακάτω τμήμα κώδικα:

```
a=2  
def my_fun(a,c):  
    a=a+2  
    d=a*c  
    return d  
print(a)  
b = my_fun(2,c=4)  
print(a)  
print(b)  
print(c)
```

Τι θα εμφανίσει κάθε μία από τις τέσσερις εντολές **print**; Εξηγήστε **σύντομα** την απάντησή σας.

Απάντηση:

2, παίρνει τιμή στην αρχή του κώδικα

2 ξανά. Η αλλαγή της τοπικής μεταβλητής **a** μέσα στην **my_fun** δεν επηρεάζει την εξω **a**

16, $(2+2)*4$ είναι η έξοδος της **my_fun** που ανατίθεται στη **b**

Error, η **c** δεν υπάρχει στο κυρίως τμήμα κώδικα αλλά μόνο ως παράμετρος μέσα στη συνάρτηση

Ερώτημα 6 (2 μονάδες)

Σχεδιάζουμε ένα παιχνίδι σε Python, με αντικειμενοστραφή προγραμματισμό.

A) Να σχεδιαστεί μια κλάση **Character** με μια ιδιότητα **health** και μια ιδιότητα **speed** με default αρχική τιμή το **10**. Η **Character** θα έχει επίσης μια μέθοδο **take_damage** που θα δέχεται μια τιμή **damage**, και θα την αφαιρεί από τη **health** του χαρακτήρα.

B) Να σχεδιαστεί επίσης μια υποκλάση της **Character** με όνομα **Player**, που θα ξεκινάει με **health=100** και την προεπιλεγμένη **speed**, και θα έχει επιπλέον μια ιδιότητα **level** με default τιμή **1**. Θα έχει μια επιπλέον μέθοδο με όνομα **level_up()**, η οποία απλώς θα ανεβάζει το **level** κατά **1** και το **health** κατά **10**.

Απάντηση:

```
class Character:
```

```
    def __init__(self, health, speed = 10):
```

```
        self.health = health
```

```
        self.speed = speed
```

```
    def take_damage(self, damage):
```

```
        self.health = self.health - damage
```

```
class Player(Character):
```

```
    def __init__(self, level = 1):
```

```
        Character.__init__(health=100)
```

```
        self.level = level
```

```
    def level_up(self):
```

```
        self.level +=1
```

```
        self.health +=10
```

Ερώτημα 7 (1.5 μονάδες)

Δίνεται ένα λεξικό με όνομα **grades** που περιέχει ως κλειδιά τους α/μ των φοιτητών/τριών ενός έτους, και ως τιμές τους βαθμούς τους. Να γραφεί κώδικας σε Python που

A) Να εμφανίζει τη μέση τιμή της βαθμολογίας του έτους

B) Να εμφανίζει τους α/μ των φοιτητών/τριών των οποίων ο βαθμός είναι υψηλότερος από τη μέση τιμή.

Γ) Να εμφανίζει τη διάμεση τιμή της βαθμολογίας του έτους. Αν το πλήθος των φοιτητών/τριων είναι περιττό τότε πρόκειται για τη μεσαία τιμή, αλλιώς αν το πλήθος είναι άρτιο υπολογίζουμε τη

μέση τιμή των δύο μεσαίων τιμών. Π.χ. αν οι βαθμοί είναι [4, 5, 7, 7, 9] η διάμεση τιμή είναι 7, ενώ αν είναι [3, 4, 6, 7, 9, 10] η μέση τιμή είναι 6.5.

Απάντηση (ενδεικτική):

```
# Κρατάω μόνο τους βαθμούς και τους μετατρέπω σε λίστα
g = list(grades.values())
# Υπολογίζω τη μέση τιμή των βαθμών
mo = sum(g)/len(g)
print(mo)
# Σαρώνω τη λίστα και εμφανίζω τα keys που αντιστοιχούν σε
# βαθμό>mo
for am in grades.keys():
    if grades[am]>mo:
        print(am)

# Ταξινομώ τους βαθμούς για να βρω το μεσαίο (διάμεση τιμή)
g.sort()
# Αν είναι μονό το πλήθος
if len(g)%2!=0:
    # Η μεσαία θέση είναι στρογγυλοποίηση προς τα κάτω του πλήθους/2
    # π.χ. αν το πλήθος είναι 5, η μεσαία θέση είναι η 2 (αφού
    αρχίζω από το 0)
    print(g[int(len(g)/2)])
else:
    # Αν το πλήθος είναι μονό, παίρνω το μέσο όρο των δύο μεσαίων
    # π.χ. αν είναι 6, παίρνω τις θέσεις 2 και 3
    print((g[int(len(g)/2)-1]+g[int(len(g)/2)])/2)
```

Χρήσιμες μέθοδοι της Python (**δεν είναι απαραίτητο να τις χρησιμοποιήσετε όλες**):

len(): μήκος (πλήθος τιμών) ενός αντικειμένου

sum(): άθροισμα τιμών ενός αντικειμένου

list.append(x): επέκταση λίστας με νέο στοιχείο

list.sort(*, key=None, reverse=False): ταξινόμηση λίστας in-place

str.split(sep=None, maxsplit=-1): διαχωρισμός string

str.replace(old, new[, count]): αντικατάσταση περιεχομένου string

dictionary.values()

dictionary.keys()