ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΟΙΚΟΝΟΜΙΑΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

**ΜΑΘΗΜΑ: ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ Η/Υ**

ΔΙΔΑΣΚΩΝ: Μ. ΖΑΜΠΟΓΛΟΥ

# ΘΕΜΑΤΑ & ΛΥΣΕΙΣ ΕΞΕΤΑΣΕΩΝ

## ΙΑΝΟΥΑΡΙΟΣ 2026

Διάρκεια εξέτασης: 2 ώρες

### **Ερώτημα 1** (2 μονάδες)

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

1.1)

```
s = "Python2026"
print(s[6:8] + s[-1])
```

***Θα εμφανίσει:***

**206**

"20"+"6"="206"

1.2)

```
s = "Exam2026"
print(s[6::-2])
```

***Θα εμφανίσει:***

**22aE**

Από τον έβδομο χαρακτήρα ως την αρχή, με βήμα -2

1.3)

```
a = 7
b = 2
print(a // b + a % b)
```

***Θα εμφανίσει:***

**4**.

3+1=4.

1.4)

```
def check(n):
    return n > 5 and n % 2 == 0
print(check(8))
print(check(7))
```

***Θα εμφανίσει:***

**True**

**False**

Και το 8 και το 7 είναι > 5, όμως το 8 είναι άρτιο ενώ το 7 περιττό.

1.5)

```
a = [1, 2]
b = [3, 4]
c = a + b
print(c * 2)
```

***Θα εμφανίσει:***

**[1, 2, 3, 4, 1, 2, 3, 4]**

Η πρόσθεση λιστών κάνει συνένωση, και ο πολλαπλασιασμός κάνει συνένωση με τον εαυτό της.

1.6)

```
s = "Python"
s[0] = "C"
print(s.lower())
```

***Θα εμφανίσει:***

**Error**

Τα strings είναι immutable, η δεύτερη εντολή προκαλεί σφάλμα και ο κώδικας δε φτάνει ποτέ στην τρίτη.

1.7)

```
a = [1, 3, 1, 2]
b = set(a)
print(sum(a))
```

***Θα εμφανίσει:***

**7**

Η **set()** δημιουργεί ένα σύνολο **b** με τιμή **{1,2,3}**, αλλά η **a** παραμένει **[1,3,1,2]** με άθροισμα **7**.

1.8)

```
M = [[1, 2, 3], [4, 5], [7, 8, 9]]
for i in range(len(M)):
  print(M[i][2])
```

***Θα εμφανίσει:***

**3**
**Error**

Το δεύτερο στοιχείο της **M** έχει μόνο 2 στοιχεία.

1.9)

```
x = [10, 20, 30]
y = x
y.append(40)
x[0] = 5
print(x)
```

***Θα εμφανίσει:***

**[5, 20, 30, 40]**

Το x και το y συνδέονται by reference. Άρα και η προσθήκη του 40 και η αντικατάσταση του 10 με 5 θα γίνουν πάνω στην ίδια λίστα.

1.10)

```
a = [5, 2, 4, 12]
while len(a)>0:
  del(a[0])
  print(a[0])
```

***Θα εμφανίσει:***

**2**
**4**
**12**
**Error**

Πρώτα διαγράφει το πρώτο στοιχείο και έπειτα εμφανίζει το πρώτο (που απομένει). Στην τελευταία επανάληψη, θα διαγράψει το 12 και η λίστα θα είναι κενη.

1.11)

```
x = [1,2,3,4,5,6,7,8,9,10]
for i in x:
  if i%3!=0:
    continue
  print(i)
```

***Θα εμφανίσει:***

**3**
**6**
**9**

Όταν το x δε διαιρείται ακριβώς με το 3, η continue μας πάει κατευθείαν στην επόμενη επανάληψη.

1.12)

```
d = {4:12, 1:-4, 3:21, 10:2}
for i in sorted(d.keys()):
  if d[i]<10:
    print(i)
```

***Θα εμφανίσει:***

**1**
**10**

Εμφανίζει με αύξουσα σειρά τα κλειδιά που αντιστοιχούν σε τιμή είναι μικρότερη του 10

## **Ερώτημα 2** (1.5 μονάδες)

Να γραφεί κώδικας σε Python που:

1) Θα ζητάει 25 πραγματικούς αριθμούς από το χρήστη και θα τις αποθηκεύει σε μια λίστα
2) Θα δημιουργεί μια νέα λίστα που θα περιέχει το τετράγωνο κάθε αριθμού
3) Θα εμφανίζει τη μέση τιμή των τετραγώνων
4) Θα εμφανίζει τη διάμεση τιμή των τετραγώνων
5) Θα εμφανίζει όλους τους αριθμούς που το τετράγωνό τους είναι μικρότερο από τη διάμεση τιμή

*Για να υπολογίσουμε τη διάμεση τιμή, ταξινομούμε τα στοιχεία σε σειρά και παίρνουμε το μεσαίο.*

***Απάντηση** (ενδεικτική)**:***

```
N=[]
for i in range(25):
  n = float(input("Δώσε έναν αριθμό:"))
  N.append(n)

S=[]
for n in N:
  S.append(n**2)

print(sum(S)/25)

S.sort()
med=S[12] %Αφού έχει 25 στοιχεία, παίρνω το 13ο.
print(med)

for n in N:
  if n**2<med:
     print(n)
```

## **Ερώτημα 3** (1 μονάδα)

Δίνεται η παρακάτω εντολή list comprehension:

```
B = [x**2 for x in range(10) if x % 3 == 0]
```

Να γραφεί ισοδύναμος κώδικας Python με δομές ελέγχου και επανάληψης.

***Απάντηση:***

```
B=[]
for x in range(10):
  if x % 3 == 0:
     B.append(x**2)
```

## **Ερώτημα 4** (1.5 μονάδες)

Να γραφεί κώδικας σε Python που:

Α) Θα ζητάει μια φράση από το χρήστη

Β) Θα τη χωρίζει σε επιμέρους λέξεις χρησιμοποιώντας το κενό ως διαχωριστικό

Γ) Θα δημιουργεί ένα λεξικό το οποίο θα έχει ως κλειδί τον πρώτο χαρακτήρα κάθε λέξης, και ως τιμή τον τελευταίο χαρακτήρα της αντίστοιχης λέξης

Δ) Θα εμφανίζει το πλήθος λέξεων της πρότασης, και το πλήθος στοιχείων του λεξικού. Μπορεί να συμβεί ποτέ οι δυο αριθμοί να είναι διαφορετικοί; Σε ποια περίπτωση;

```
a = input("Δώσε μια φράση:")
# Χωρίζουμε την πρόταση στα κόμματα
s=a.split()
# Δημιουργούμε και γεμίζουμε το λεξικό
d={}
for w in s:
  d[w[0]]=w[-1]
# Εμφανίζουμε τα δυο πλήθη
print(len(s),len(d))
```

Τα δύο πλήθη θα είναι διαφορετικά, όταν υπάρχουν στη φράση παραπάνω από μία λέξεις που αρχίζουν από το ίδιο γράμμα. Τα κλειδιά του λεξικού πρέπει να είναι μοναδικά, οπότε όταν συναντάει πρώτο γράμμα που έχει συναντήσει ήδη, θα αντικαθιστά το παλιό.

## **Ερώτημα 5** (2 μονάδες)

Α) Να γραφεί μια συνάρτηση σε Python με όνομα **`calc_fpa()`** που θα δέχεται ως είσοδο έναν αριθμό και το ποσοστό του ΦΠΑ (default τιμή **`0.24`**) και θα επιστρέφει τον αριθμό προσαυξημένο κατά το ποσοστό ΦΠΑ. Δηλαδή αν δώσω **`100`** και **`0.23`**, να επιστρέφει **`123`**.

Β) Να γραφεί μια άλλη συνάρτηση με όνομα **`paraggelia18()`** που θα δέχεται μια λίστα από αριθμούς, θα καλεί την **`calc_fpa()`** για τον καθένα με ΦΠΑ **`0.18`**, και θα επιστρέφει μια νέα λίστα με όσους από αυτούς είναι μεγαλύτεροι του **`100`** μετά την προσαύξηση.

Γ) Να κληθεί η **`paraggelia18()`** με είσοδο τη λίστα **`[40, 120, 80, 260]`**.

(Στις εξετάσεις η εκφώνηση έγραφε από λάθος *"Να κληθεί η **`calc_fpa()`**"*. Και οι δύο απαντήσεις θεωρήθηκαν σωστές, παρότι η **`calc_fpa()`** με είσοδο λίστα θα προκαλέσει error.)

***Απάντηση*** *(ενδεικτική)**:***

Α)

```
def calc_fpa(price,fpa=0.24):
      return price*(1+fpa)
```

Β)

```
def paraggelia18(L):
  prices = []
  for p in L:
     p_fpa = calc_fpa(p,0.18)
     if p_fpa>100:
        prices.append(p)
  return prices
```

Γ)

```
p = paraggelia18([40,120,80,260])
```

## **Ερώτημα 6** (1.5 μονάδα)

Δίνεται το παρακάτω τμήμα κώδικα:

```
def sum_half(n):
  print(n)
  if n<10:
    return n
  else:
    return n + sum_half(n/2)

print(sum_half(40))
print(n)
```

Τι θα εμφανίσει κάθε μία εντολή **`print`**; Εξηγήστε **σύντομα** την απάντησή σας.

***Απάντηση:***

**`40`**, η **`n`** παίρνει τιμή από την εντολή **`divide(40)`**

**`20`**, καθώς καλείται από την **`divide(n/2)`** με **`n=40`**

**`10`**, ομοίως

**`5`**, ομοίως

(*Η αναδρομή τελειώνει εδώ, η 5 επιστρέφεται και αθροίζεται με αυτό που επιστρέφει η κάθε προηγούμενη κλήση: 5+10+20+40=75 οπότε η εξωτερική* ***`print(divide(40))`*** *εμφανίζει τελικά:*)

**`75`**

*Και τέλος η τελευταία* ***`print(n)`*** *εμφανίζει:*

**`Error`** γιατί η μεταβλητή **`n`** υπάρχει μόνο μέσα στο namespace της συνάρτησης **`divide()`**

## **Ερώτημα 7** (1.5 μονάδα)

Σχεδιάστε τις παρακάτω κλάσεις για ένα σύστημα αποθήκης:

1. Μια κλάση **`Product`** με ιδιότητες **`name`** και **`price`** που θα αρχικοποιούνται ως παράμετροι κατά τη δημιουργία, και μια μέθοδο **`apply_discount(percent)`**
2. που θα μειώνει την τιμή (price) κατά το αντίστοιχο ποσοστό.
3. Μια υποκλάση **`Electronic`** που θα κληρονομεί από την **`Product`** και έχει μια επιπλέον ιδιότητα **`warranty`** (έτη εγγύησης) με προεπιλεγμένη (default) τιμή το **`2`**.
4. Μια κλάση **`Warehouse`** με α) μια ιδιότητα **`stock`** που θα αρχικοποιείται ως άδεια λίστα και β) μια μέθοδο **`add_item(item)`** που θα δέχεται ένα αντικείμενο και θα το προσθέτει στο τέλος της **`stock`**. Υλοποιήστε την κατάλληλη magic function ώστε η **`len()`** σε μια **`Warehouse`** να επιστρέφει το πλήθος αντικειμένων της **`stock`** της.

***Απάντηση:***

```
class Product:
  def __init__(self, name, price):
    self.name = name
    self.price = price
  def apply_discount(self, percent):
    self.price -= self.price*percent

class Electronic(Product):
  def __init__(self, name, price, warranty=2):
    super().__init__(name, price)
    self.warranty = warranty

class Warehouse():
  def __init__(self):
    self.stock = []
  def add_item(self,item):
    self.stock.append(item)
  def __len__(self):
    return len(self.stock)
```

## **Ερώτημα 8 (**1.5 μονάδες)

Δίνεται ο παρακάτω κώδικας Python:

```
import numpy as np
rng = np.random.default_rng()
r = rng.random((400,50))*15
```

Να συνεχιστεί ώστε

Α) Να δημιουργεί έναν πίνακα NumPy ίσων διαστάσεων με τον **r**, που να περιέχει τυχαίους ακέραιους από 0 έως 15 και να τον αθροίζει με τον **r**. Να αποθηκεύει το άθροισμα σε έναν πίνακα **s**.

Β) Να εμφανίζει τη μέση τιμή κάθε στήλης του **s**

Γ) Να αντικαθιστά τα στοιχεία του **s** που είναι μικρότερα από 15 με -1

Δ) Να μετράει πόσες φορές εμφανίζεται το -1 στον **s** μετά την αντικατάσταση.

Ε) Περίπου τι τιμή θα περιμένατε να επιστρέψει το ερώτημα Δ); Δικαιολογήστε την απάντησή σας.

***Απάντηση*** *(ενδεικτική)**:***

```
import numpy as np
rng = np.random.default_rng()
r = rng.random((400,50))*15
# Δημιουργώ πίνακα τυχαίων ακεραίων
t = rng.integers(0,16,(400,50))
# Αθροίζω στον s
s = r + t
# Μέση τιμή ανά στήλη
print(np.mean(s,0))
# Αντικατάσταση στοιχείων με το -1
s[s<15]=-1
# Πλήθος στοιχείων == -1
print(np.sum(s==-1))
```

Θα περίμενα η εντολή να επιστρέψει περίπου 10.000, καθώς αφενός ο **t** περιέχει ακέραιες τιμές στο [0,15] αλλά και ο **r** παίρνει πραγματικές τιμές στο [0,1) πολλαπλασιασμένες ×15, οπότε τελικά θα περιέχει πραγματικές τιμές στο [0,15). Άρα το άθροισμά τους θα περιέχει τιμές στο [0,30), οι μισές περίπου εκ των οποίων θα είναι <15. Αφού ο s είναι 400x50 θα έχει 20.000 τιμές, οπότε οι μισές θα είναι 10.000.

---

Χρήσιμες μέθοδοι της Python:

**`list.append(`*`x`*`)`:** επέκταση λίστας με νέο στοιχείο (in-place)

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

**`sorted(iterable)`:** ταξινόμηση iterable (επιστρέφει τιμή)

**`str.split(`*`sep=None, maxsplit=-1`*`)`:** επιστρέφει μια λίστα με το string διαχωρισμένο στους χαρακτήρες **sep**. Αν το **sep** είναι **None** (default), το διαχωρίζει σε όλους τους χαρακτήρες κενού.

**`str.lower()`:** επιστρέφει το string σε lower case

**`numpy.random.default_rng(seed=None)`:** Δημιουργεί μια γεννήτρια τυχαίων αριθμών (κλάση `numpy.random.Generator`)

**`numpy.random.Generator.random(size=None)`:** Δημιουργεί έναν πίνακα τυχαίων πραγματικών αριθμών ομοιόμορφης κατανομής στο διάστημα **`[0,1)`** με διαστάσεις που ορίζει το tuple **`size`**

**`numpy.random.Generator.integers(low, high=None, size=None)`:** Δημιουργεί έναν πίνακα ακέραιων αριθμών ομοιόμορφης κατανομής στο **`[low,high)`** με διαστάσεις που ορίζει το tuple **`size`**. Αν **`high==None`** τότε οι αριθμοί είναι στο **`[0,low)`**.

**`numpy.mean(a, axis=None, …)`:** Υπολογίζει τη μέση τιμή των στοιχείων του πίνακα a κατά μήκος της διάστασης **`axis`**. Αν **`axis==None`** τότε την υπολογίζει για όλον τον πίνακα.

**`numpy.sum(a, axis=None, …)`:** Όπως παραπάνω, για το άθροισμα των στοιχείων.