ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΟΙΚΟΝΟΜΙΑΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ
ΜΑΘΗΜΑ: ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ Η/Υ
ΔΙΔΑΣΚΩΝ: Μ. ΖΑΜΠΟΓΛΟΥ

ΘΕΜΑΤΑ & ΛΥΣΕΙΣ ΕΞΕΤΑΣΕΩΝ

ΙΟΥΝΙΟΣ 2026

Διάρκεια εξέτασης: 2 ώρες
Μέγιστη βαθμολογία 10.5 μονάδες

Ερώτημα 1 (2 μονάδες)

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

1.1)

```
a = [10, 20, 30, 40]
while len(a) > 0:
    print(a[0])
```

Θα εμφανίσει:

```
10
10
10
...
```

Για πάντα

1.2)

```
x = [1, 2, 3]
y = x
y.append(4)
x[0] = 9
print(y, x)
```

Θα εμφανίσει:

```
[9, 2, 3, 4], [9, 2, 3, 4]
```

Το **x** και το **y** συνδέονται by reference, οπότε και η προσθήκη του **4** και η αλλαγή του πρώτου στοιχείου γίνονται στην ίδια λίστα.

1.3)

```
def f(n):
    return n > 3 and n % 2 != 0
print(f(5))
print(f(4))
```

Θα εμφανίσει:

```
True
False
```

Και το **5** και το **4** είναι **>3**, όμως το **5** είναι περιττό (**True**) ενώ το 4 άρτιο (**False**).

1.4)

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

Θα εμφανίσει:

```
3 10
```

Η **set()** κρατά τις μοναδικές τιμές **{1,2,3}** άρα το **len(b)** είναι 3. Η **a** δεν αλλάζει, άθροισμα 10.

1.5)

```
d = "Python rocks"
d[0] = "J"
print(d)
```

Θα εμφανίσει:

```
Error
```

Τα strings είναι immutable· δεν μπορούμε να τροποποιήσουμε χαρακτήρα τους.

1.6)

```
s = "Programming2026"
print(s[1:4] + s[-2])
```

Θα εμφανίσει:

```
rog2
```

Το **s[1:4]** είναι **"rog"** και το **s[-2]** είναι **"2"**. Η πρόσθεση string κάνει συνένωση

Ερώτημα 2 (2 μονάδες)

Να γραφεί κώδικας σε Python που:

1) Θα ζητάει σαράντα ακέραιους αριθμούς από το χρήστη
2) Θα αποθηκεύει σε μια λίστα μόνο όσους από αυτούς διαιρούνται ακριβώς με το **4**
3) Στο τέλος, θα εμφανίζει τους μοναδικούς αριθμούς της λίστας χωρίς να επαναλαμβάνει τους ίδιους. Δηλαδή αν η λίστα περιέχει τους **[8, 4, 12, 4, 8]** θα εμφανίζει τους **8**, **4**, **12**.
4) Θα εμφανίζει το άθροισμα των μοναδικών αυτών αριθμών. (Στο παράδειγμα: **24**)

Απάντηση *(ενδεικτική)**:***

```
L = []
for i in range(40):
    a = int(input("Δώσε έναν ακέραιο:"))
    if a % 4 == 0:
        L.append(a)
print(set(L))
print(sum(set(L)))
```

Ερώτημα 3 (1 μονάδα)

Δίνεται μια λίστα Α που περιέχει string με ανδρικά ονόματα. Να γραφεί μια γραμμή κώδικα list comprehension (“υπολογιζόμενες λίστες”) που να δημιουργεί μια νέα λίστα Β που θα περιέχει τα ονόματα της Α που έχουν μήκος πάνω από **4** χαρακτήρες, μαζί με το άρθρο **"Ο "**.
Π.χ. αν **A=["Νίκος", "Άρης", "Δημήτρης", "Ίων", "Πέτρος"]**, η **B** θα είναι **["Ο Νίκος", "Ο Δημήτρης", "Ο Πέτρος"]**. Δε χρειάζεται εντολή εισόδου, θεωρήστε ότι η **A** υπάρχει ήδη.

Απάντηση:

```
B = ["Ο " + x for x in A if len(x) > 4]
```

Ερώτημα 4 (2 μονάδες)

1) Να γραφεί μια συνάρτηση σε Python με όνομα **add_to_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** αυξημένα κατά **n**. Δηλαδή αν δώσουμε στην **add_to_list** την **[3,4,2]** και το **2** θα επιστρέψει: **[5, 6, 4]**.
2) Να γραφεί μια συνάρτηση σε Python με όνομα **boost_and_select** που θα δέχεται ως είσοδο μια λίστα, θα αυξάνει τις τιμές των στοιχείων της κατά **10** με τη χρήση της **add_to_list**, και θα εμφανίζει μόνο όσα από τα αυξημένα στοιχεία είναι μεγαλύτερα

του 15. Δηλαδή, αν δώσουμε στην **`boost_and_select`** την **`[2, 8, 6]`** θα εμφανίσει **`[18, 16]`**.

3) Καλέστε την boost_and_select με είσοδο τη λίστα **`[2, 8, 6]`**.

Απάντηση *(ενδεικτική)**:***

A)

```
def add_to_list(L, n):
    # Δημιουργώ την κενή λίστα που θα επιστρέψω
    L_n = []
    # Για κάθε στοιχείο της L το προσαρτώ αυξημένο κατά n
    for l in L:
        L_n.append(l + n)
    # Εναλλακτικά με comprehension:
    L_n = [l + n for l in L]
    return L_n
```

B)

```
def boost_and_select(L):
    # Καλώ την add_to_list με τη λίστα και το 10
    L_b = add_to_list(L, 10)
    # Εμφανίζω τα στοιχεία του αποτελέσματος που είναι > 15
    print([l for l in L_b if l > 15])
    # Η μέθοδος τυπώνει το αποτέλεσμά της και δεν επιστρέφει τίποτα
```

B)

```
boost_and_select([2, 8, 6])
```

Ερώτημα 5 (1.5 μονάδα)

Δίνεται το παρακάτω τμήμα κώδικα:

```
x = 3
y = 2
def my_fun(x, y):
    x = x * 2
    z = x - y
    return z
b = my_fun(5, y=3)
print(x)
print(b)
print(z)
print(y)
```

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

Απάντηση:

3, η x στο κυρίως πρόγραμμα δεν επηρεάζεται από την τοπική x της συνάρτησης.

7, η συνάρτηση εκτελείται για x=5 και y=3: x γίνεται 10, z = 10−3 = 7, που ανατίθεται στη b.

`Error`, η z υπάρχει μόνο μέσα στο namespace της my_fun και δεν είναι ορατή στο κυρίως πρόγραμμα.

Τίποτα, η print(y) δε θα εκτελεστεί ποτέ λόγω του Error.

Ερώτημα 6 (2 μονάδες)

Σχεδιάζουμε ένα σύστημα διαχείρισης οχημάτων με αντικειμενοστραφή προγραμματισμό.

1. Να σχεδιαστεί μια κλάση **`Vehicle`** με μια ιδιότητα **`brand`** που θα δίνεται ως παράμετρος κατά τη δημιουργία, και μια ιδιότητα **`speed`** με default αρχική τιμή το **`0`**. Η **`Vehicle`** θα έχει επίσης μια μέθοδο **`accelerate`** που θα δέχεται μια τιμή **`amount`** και θα αυξάνει την **`speed`** του οχήματος κατά **`amount`**.
2. Να σχεδιαστεί επίσης μια υποκλάση της **`Vehicle`** με όνομα **`ElectricCar`**, που θα έχει επιπλέον μια ιδιότητα **`battery`** με default τιμή **`100`**. Θα έχει μια επιπλέον μέθοδο με όνομα **`charge()`**, η οποία απλώς θα θέτει την **`battery`** στο **`100`**.

Απάντηση:

```
class Vehicle:
    def __init__(self, brand, speed=0):
        self.brand = brand
        self.speed = speed
    def accelerate(self, amount):
        self.speed = self.speed + amount

class ElectricCar(Vehicle):
    def __init__(self, brand, battery=100):
        super().__init__(brand)
        self.battery = battery
    def charge(self):
        self.battery = 100
```