

Εισαγωγή στον Προγραμματισμό Η/Υ

Διδάσκοντες: Ι. Πολενάκης, Κ. Σκιαδόπουλος, Δ. Μουρατίδου

Τμήμα Πληροφορικής, Ιόνιο Πανεπιστήμιο



3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'While'

- Η εντολή 'while' εκτελεί μια επαναλαμβανόμενη εκτέλεση ενός μπλοκ κώδικα (μπλοκ-1 στο παρακάτω παράδειγμα) εφόσον μια λογική συνθήκη (λογική έκφραση) είναι True.
- Μπλοκ & Εσοχή - σημαντικοί συντακτικοί κανόνες της Python:
 - a) Τα μπλοκ κώδικα στην Python προηγούνται με (':') που τοποθετείται στο τέλος της εντολής που τα περικλείει.
 - b) Τα μπλοκ έχουν εσοχή (συνήθως 4 κενά) στα δεξιά . Λάβετε υπόψη ότι η αναγνώριση στον κώδικα Python είναι υποχρεωτική . Έτσι, ο κώδικας χωρίς σωστή εσοχή ενδέχεται να μην εκτελεστεί σωστά και να επιστρέψει σφάλματα χρόνου εκτέλεσης.

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'While' - Παραδείγματα Εκτέλεσης

```
a = 0
b = 10
while a < b:                # note the ':'
    a += 1                  # note the block indentation of 4 spaces
    print(a, end = ' ')
else:                       # note the ':'
    print('\nEnd of loop')  # note the block indentation of 4 spaces

print('This is the first code line after the while loop')
```

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'While' - Παραδείγματα Εκτέλεσης

```
lines = 1
maxlines = 10
while lines <= maxlines:
    print(lines*'*')
    lines +=1
```

```
*
**
***
****
*****
*****
*****
*****
*****
*****
*****
```

3. Δομές Επανάληψης

► Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'While' - Παραδείγματα Εκτέλεσης

```
import random

guess = tries = 0
num = None

while guess != num:
    tries += 1
    num = int(random.randint(1,5))
    guess = int(input('Can you guess the random number? (1 to 5): '))
    print('Random number is:', num)

else:
    print('Correct guess after', tries, 'tries!')
```

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For'

- Ο βρόχος 'for' είναι μια σημαντική εντολή προγραμματισμού στην Python, καθώς υλοποιεί ορισμένες βασικές ιδέες και δομές της γλώσσας.
- Γενικά, ένα επαναλήψιμο αντικείμενο είναι οποιοδήποτε αντικείμενο που επιστρέφει μια ακολουθία τιμών, μία προς μία, στη μεταβλητή `iterator`. Για κάθε νέα τιμή που επιστρέφεται, εκτελείται μια νέα επανάληψη βρόχου `for`. Όταν ολοκληρωθεί η ακολουθία τιμών, η εκτέλεση του βρόχου ολοκληρώνεται.
- Πιο τεχνικά μιλώντας, ένα επαναλήψιμο αντικείμενο είναι ένα αντικείμενο που διαθέτει μια μέθοδο `__next__`, η οποία επιστρέφει το επόμενο στην ακολουθία στοιχείο του επαναλήψιμου αντικειμένου κάθε φορά που καλείται.
- Η μεταβλητή `iterator` είναι απλώς ένα όνομα συνδεδεμένο με την τιμή που επιστρέφεται από το `iterable`.

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Παραδείγματα Εκτέλεσης

```
asum = 0
for i in [1,2,3,4,5]:
    asum += i
    print(i, asum)

print('for loop is now over')
```

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Παραδείγματα Εκτέλεσης

```
for i in ['spam', 'eggs', 'ham']:      # List of strings as iterable object in the loop
    print(i, end=' ')
```

```
print('\n')
```

```
m = 'spam'
```

```
for ch in m:      # String as iterable object
    print(ch)
```

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Παραδείγματα Εκτέλεσης

```
alist = ['a', 'b', 'c', 'd', 'e']  
ch = ''  
for i in alist:  
    ch += i  
print(ch)
```

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Παραδείγματα Εκτέλεσης

```
alist = ['a', 'b', 'c']
```

```
i = 0
```

```
s = ''
```

```
for ch in alist:
```

```
    i += 1
```

```
    s = s+i*ch
```

```
print(s)
```

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Παραδείγματα Εκτέλεσης

```
MIN = 1
MAX = 10
for i in range(MIN, MAX+1):
    for j in range(MIN, MAX+1):
        print('\t', i*j, end='')    # try both prints and observe
        the difference
        #print('{:8d}'.format(i*j), end='')
    print()
```

3. Δομές Επανάληψης

► Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Παραδείγματα Εκτέλεσης

```
MIN = 1
MAX = 10
for i in range(MIN, MAX+1):
    for j in range(MIN, MAX+1):
        print('\t', i*j, end='')    # try both prints and observe
        the difference
        #print('{:8d}'.format(i*j), end='')
    print()
```

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Περί Εύρους

- Ο βρόχος 'for' απαιτεί μια ισχυρή επαναλήψιμη δομή που μπορεί να επιστρέψει την απαραίτητη ακολουθία τιμών γρήγορα, αξιόπιστα και αποτελεσματικά στη μνήμη. Μια τέτοια δομή παρέχεται από τη συνάρτηση **range()**
- Πώς λειτουργεί η εμβέλεια (range)?
 - Η συνάρτηση 'range' επιστρέφει (μία προς μία) μια ακολουθία τιμών στο διάστημα μεταξύ των τιμών 'start' και 'end':
 - έναρξη : η πρώτη τιμή της ακολουθίας· εάν δεν οριστεί, τότε θεωρείται '0'
 - τέλος : η τελευταία τιμή του διαστήματος ΑΛΛΑ ΔΕΝ περιλαμβάνεται στην ακολουθία
 - βήμα (προαιρετικό): εάν περιλαμβάνεται, ορίζει το βήμα στο οποίο παράγονται οι τιμές ξεκινώντας από την «έναρξη» και καταλήγοντας στην τιμή που βρίσκεται πλησιέστερα στο «τέλος», αλλά χωρίς να την υπερβαίνει

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Περί Εύρους

- Ο βρόχος 'for' απαιτεί μια ισχυρή επαναλήψιμη δομή που μπορεί να επιστρέψει την απαραίτητη ακολουθία τιμών γρήγορα, αξιόπιστα και αποτελεσματικά στη μνήμη. Μια τέτοια δομή παρέχεται από τη συνάρτηση `range()`
- Παραδείγματα ...

```
x = range(1, 10)
```

```
for i in x:
```

```
    print(i, end=' ')
```

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Περί Εύρους

- Ο βρόχος 'for' απαιτεί μια ισχυρή επαναλήψιμη δομή που μπορεί να επιστρέψει την απαραίτητη ακολουθία τιμών γρήγορα, αξιόπιστα και αποτελεσματικά στη μνήμη. Μια τέτοια δομή παρέχεται από τη συνάρτηση **range()**
- Παραδείγματα ...

```
for x in range(10, 5, -1):  
    print(x, end=' ')
```

3. Δομές Επανάληψης

➤ Εντολές επανάληψης - Δομές επανάληψης (iteration statements)

Βρόχος 'For' - Περί Εύρους

- Ο βρόχος 'for' απαιτεί μια ισχυρή επαναλήψιμη δομή που μπορεί να επιστρέψει την απαραίτητη ακολουθία τιμών γρήγορα, αξιόπιστα και αποτελεσματικά στη μνήμη. Μια τέτοια δομή παρέχεται από τη συνάρτηση `range()`
- Παραδείγματα ...

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

3. Δομές Επανάληψης

➡ Εντολές επανάληψης

Δομές επανάληψης (iteration statements)

Επαναληπτική εκτέλεση κώδικα

- **while** - Επανάληψη ενέργειας όσο η συνθήκη παραμένει True

```
>>> studyPctg = 20
>>> while studyPctg < 65:
    studyPctg = studyPctg + 20
>>> studyPctg
```

80

- **for** - Επανάληψη ενέργειας για κάθε αντικείμενο σε μια ακολουθία αντικειμένων

```
>>> for character in 'Η Python τα σπάει': print(character, end=' ')
'Η  P y t h o n   τ α   σ π ά ε ι '
```

3. Δομές Επανάληψης

► Παράδειγμα Εφαρμογής...

```
# Ζητάμε από τον χρήστη έναν αριθμό
n = int(input("Δώσε έναν θετικό αριθμό: "))

# Έλεγχος εγκυρότητας με if
if n <= 0:
    print("Ο αριθμός πρέπει να είναι θετικός!")
else:
    print(f"\nΥπολογισμός για αριθμούς από το 1 μέχρι το {n}:\n")

    # Μεταβλητές για το άθροισμα και τα άρτια
    sum_total = 0
    even_numbers = []

    # Χρήση while για να αθροίσουμε τους αριθμούς
    i = 1
    while i <= n:
        sum_total += i
        i += 1

    print(f"Το άθροισμα των αριθμών από 1 έως {n} είναι: {sum_total}\n")

    # Χρήση for για να βρούμε τους άρτιους αριθμούς
    print("Οι άρτιοι αριθμοί είναι:")
    for i in range(1, n + 1):
        if i % 2 == 0:          # if statement
            even_numbers.append(i)
            print(i)

    print(f"\nΣυνολικά βρέθηκαν {len(even_numbers)} άρτιοι αριθμοί.")
```

1. Ζητάει από τον χρήστη έναν αριθμό.
2. Αν είναι **μη θετικός**, εμφανίζει μήνυμα λάθους (χρήση if).
3. Αν είναι θετικός:
4. Με **while** υπολογίζει το άθροισμα από το 1 έως το n.
5. Με **for** και **if** βρίσκει όλους τους άρτιους αριθμούς.
6. Εμφανίζει τα αποτελέσματα.

3. Δομές Επανάληψης

➤ Παράδειγμα Εφαρμογής...

Σύντομη παρένθεση ... Εισαγωγή στις ΛΙΣΤΕΣ!!!

Τι είναι λίστα;

- Μια λίστα είναι μια ομαδοποίηση τιμών κάτω από ένα κοινό όνομα.
- Κάθε τιμή λέγεται στοιχείο (element).

```
fruits = ["μήλο", "μπανάνα", "κεράσι"]  
numbers = [1, 2, 3, 4, 5]  
mixed = ["Python", 3.14, True]
```

- Πρόσβαση σε στοιχεία: Μπορούμε να έχουμε πρόσβαση στα στοιχεία μιας λίστας με δείκτες (indices).
Οι δείκτες ξεκινούν από το 0!

```
fruits = ["μήλο", "μπανάνα", "κεράσι"]  
print(fruits[0]) # μήλο  
print(fruits[2]) # κεράσι  
print(fruits[-1]) # κεράσι
```

- Οι λίστες μπορούν να περιέχουν αριθμούς, κείμενα, ή ακόμα και άλλες λίστες!

3. Δομές Επανάληψης

► Παράδειγμα Εφαρμογής...

Σύντομη παρένθεση ... Εισαγωγή στις ΛΙΣΤΕΣ!!!

Τι είναι λίστα;

- Μια λίστα είναι μια ομαδοποίηση τιμών κάτω από ένα κοινό όνομα.
- Κάθε τιμή λέγεται στοιχείο (element).

```
for i in fruits:  
    print(i)
```

```
fruits = ["μήλο", "μπανάνα", "κεράσι"]  
numbers = [1, 2, 3, 4, 5]  
mixed = ["Python", 3.14, True]
```

Στοιχεία: Μπορούμε να έχουμε πρόσβαση στα στοιχεία μιας λίστας με δείκτες (indices).
Μόνο από το 0!

```
fruits = ["μήλο", "μπανάνα", "κεράσι"]  
print(fruits[0]) # μήλο  
print(fruits[2]) # κεράσι  
print(fruits[-1]) # κεράσι
```

- Οι λίστες μπορούν να περιέχουν αριθμούς, κείμενα, ή ακόμα και άλλες λίστες!

3. Δομές Επανάληψης

➤ Παράδειγμα Εφαρμογής...

Σύντομη παρένθεση ... Εισαγωγή στις ΛΙΣΤΕΣ!!!

Τι είναι λίστα;

- Μια λίστα είναι μια ομαδοποίηση τιμών κάτω από ένα κοινό όνομα.
- Κάθε τιμή λέγεται στοιχείο (element).

```
for i in fruits:  
    print(i)
```

Στοιχεία: Μπορούμε να έχουμε πρόσβαση στα στοιχεία μιας λίστας με δείκτες (indices).
Μόνο από το 0!

```
fruits = ["μήλο", "μπανάνα", "κεράσι"]  
print(fruits[0]) # μήλο  
print(fruits[2]) # κεράσι  
print(fruits[-1]) # κεράσι
```

- Οι λίστες μπορούν να περιέχουν αριθμούς, κείμενα, ή ακόμα και άλλες λίστες!

Κάποιες ενσωματωμένες συναρτήσεις για λίστες:

Συνάρτηση	Περιγραφή	Παράδειγμα
len()	Πλήθος στοιχείων	len(fruits)
max()	Μεγαλύτερο στοιχείο	max(numbers)
min()	Μικρότερο στοιχείο	min(numbers)
sum()	Άθροισμα αριθμών	sum(numbers)
sorted()	Ταξινόμηση	sorted(numbers)

3. Δομές Επανάληψης

➤ Παράδειγμα Εφαρμογής...

Σύντομη παρένθεση ... Εισαγωγή στις ΛΙΣΤΕΣ!!!

Τι είναι λίστα;

- Μια λίστα είναι μια ομαδοποίηση τιμών κάτω από ένα κοινό όνομα.
- **Εμφωλευμένες Λίστες (Nested Lists)**

Μια λίστα μπορεί να περιέχει άλλες λίστες!

Παράδειγμα: `matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`

Αναφορά στοιχείου με διπλό index: `print(matrix[0][2])`

```
numbers = []
```

```
for i in range(5):
```

```
    num = float(input("Δώσε έναν αριθμό: "))
```

```
    numbers.append(num)
```

```
average = sum(numbers) / len(numbers)
```

```
print(f"Μέσος όρος: {average:.2f}")
```

3. Δομές Επανάληψης

➡ Παράδειγμα Εφαρμογής...

```
# Αρχικοποίηση λίστας για τους βαθμούς
grades = []

# Ζητάμε τον αριθμό των μαθητών
n = int(input("Πόσοι μαθητές υπάρχουν; "))

# Χρησιμοποιούμε for loop για να εισάγουμε βαθμούς
for i in range(n):
    grade = float(input(f"Δώσε τον βαθμό του μαθητή {i + 1}: "))
    grades.append(grade)

# Υπολογισμοί
average = sum(grades) / len(grades)
max_grade = max(grades)
min_grade = min(grades)

# Μετράμε πόσοι πέρασαν (βαθμός ≥ 10)
passed = 0
for g in grades:
    if g >= 10:
        passed += 1

failed = len(grades) - passed

# Εμφάνιση αποτελεσμάτων
print("\n📊 Αποτελέσματα:")
print(f"Μέσος όρος: {average:.2f}")
print(f"Υψηλότερος βαθμός: {max_grade}")
print(f"Χαμηλότερος βαθμός: {min_grade}")
print(f"Πέρασαν: {passed}")
print(f"Κόπηκαν: {failed}")
```

- Χρησιμοποιούμε τη λίστα grades για να αποθηκεύσουμε όλους τους βαθμούς.
- Με ένα for i in range(n) ζητάμε επαναληπτικά εισαγωγή βαθμών από τον χρήστη.
- Με ενσωματωμένες συναρτήσεις sum(), max(), min() υπολογίζουμε συνολικά στοιχεία.
- Ένα δεύτερο for loop μετρά πόσοι έχουν βαθμό ≥ 10 (δηλαδή πέρασαν).
- Εκτυπώνουμε τα αποτελέσματα με ωραία μορφοποίηση.

3. Δομές Επανάληψης

➡ Παράδειγμα Εφαρμογής...

```
# λίστες για προϊόντα και τιμές
products = []
prices = []

print("👋 Καλώς ήρθες στο καλάθι αγορών!")
print("Πληκτρολόγησε 'τέλος' για να ολοκληρώσεις τις αγορές.\n")

while True:
    product = input("Όνομα προϊόντος: ")

    # Αν ο χρήστης γράψει 'τέλος', σταματάμε το loop
    if product.lower() == "τέλος":
        break

    # Ζητάμε τιμή προϊόντος
    try:
        price = float(input(f"Τιμή για το '{product}': "))
        # Αποθηκεύουμε δεδομένα
        products.append(product)
        prices.append(price)
    except ValueError:
        print("⚠️ Παρακαλώ δώσε έγκυρη τιμή!\n")

# Αν δεν μπήκαν προϊόντα
if len(products) == 0:
    print("\nΔεν προστέθηκαν προϊόντα στο καλάθι.")
else:
    total = sum(prices)
    average = total / len(prices)
    max_price = max(prices)
    max_index = prices.index(max_price)
    expensive_product = products[max_index]

    print("\n📋 Αναφορά Αγορών:")
    print("-----")
    for i in range(len(products)):
        print(f"{products[i]} - {prices[i]:.2f}€")

    print("-----")
    print(f"💎 Σύνολο αγορών: {total:.2f}€")
    print(f"💎 Μέση τιμή προϊόντος: {average:.2f}€")
    print(f"💎 Πιο ακριβό προϊόν: {expensive_product} ({max_price:.2f}€")
```

- Το πρόγραμμα ζητά συνεχώς όνομα προϊόντος με ένα while True βρόχο.
- Αν ο χρήστης γράψει "τέλος", το πρόγραμμα σταματά με break.
- Για κάθε προϊόν, ζητάμε τιμή και τη βάζουμε σε λίστα.
- Αν δοθεί μη έγκυρη τιμή, εμφανίζεται μήνυμα λάθους
Η try-except ΘΑ ΔΟΥΜΕ ΑΡΓΟΤΕΡΑ ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ.
- Στο τέλος, γίνεται υπολογισμός:
 1. Σύνολο τιμών (sum)
 2. Μέση τιμή (average)
 3. Πιο ακριβό προϊόν (με χρήση max() και index())

3. Δομές Επανάληψης

- Άσκηση: Να αποθηκεύσουμε τις βαθμολογίες πολλών μαθητών σε 3 μαθήματα (π.χ. Μαθηματικά, Φυσική, Πληροφορική) και να υπολογίσουμε:
 1. Τον μέσο όρο κάθε μαθητή,
 2. Τον μέσο όρο κάθε μαθήματος,
 3. Τον καλύτερο μαθητή

3. Δομές Επανάληψης

➤ Άσκηση: Να αποθηκεύσουμε τις βαθμολογίες πολλών μαθητών σε 3 μαθήματα (π.χ. Μαθηματικά, Φυσική, Πληροφορική) και να υπολογίσουμε:

1. Τον μέσο όρο κάθε μαθητή,
2. Τον μέσο όρο κάθε μαθήματος,
3. Τον καλύτερο μαθητή

```
# Λίστα με τα ονόματα των μαθητών
students = ["Άννα", "Γιώργος", "Μαρία", "Νίκος"]
```

```
# Δισδιάστατη λίστα με βαθμούς για 3 μαθήματα
# Κάθε υπολίστα αντιστοιχεί σε έναν μαθητή
grades = [
```

```
    [18, 15, 17], # Άννα
    [12, 14, 11], # Γιώργος
    [19, 20, 18], # Μαρία
    [10, 13, 12]  # Νίκος
]
```

```
# --- Υπολογισμός μέσου όρου κάθε μαθητή ---
print("\n📊 Μέσος όρος ανά μαθητή:")
print("-----")
```

```
for i in range(len(students)): # Εξωτερικός βρόχος: κάθε μαθητής
    total = 0
    for j in range(len(grades[i])): # Εσωτερικός βρόχος: κάθε μάθημα
        total += grades[i][j]
    avg = total / len(grades[i])
    print(f"{students[i]} → {avg:.2f}")
```

```
# --- Υπολογισμός μέσου όρου ανά μάθημα ---
print("\n📊 Μέσος όρος ανά μάθημα:")
print("-----")
```

```
num_subjects = len(grades[0])

for j in range(num_subjects): # Κάθε μάθημα (στήλη)
    total = 0
    for i in range(len(students)): # Κάθε μαθητής (γραμμή)
        total += grades[i][j]
    avg = total / len(students)
    print(f"Μάθημα {j + 1} → {avg:.2f}")
```

```
# --- Εύρεση καλύτερου μαθητή ---
averages = []
for i in range(len(students)):
    avg = sum(grades[i]) / len(grades[i])
    averages.append(avg)
```

```
max_avg = max(averages)
best_index = averages.index(max_avg)
```

```
print("\n🏆 Καλύτερος μαθητής:")
print("-----")
print(f"{students[best_index]} με μέσο όρο {max_avg:.2f}")
```

3. Δομές Επανάληψης

➤ Άσκηση: Να αποθηκεύσουμε τις βαθμολογίες πολλών μαθητών σε 3 μαθήματα (π.χ. Μαθηματικά, Φυσική, Πληροφορική) και να υπολογίσουμε:

1. Τον μέσο όρο κάθε μαθητή,
2. Τον μέσο όρο κάθε μαθήματος,
3. Τον καλύτερο μαθητή

```
# Λίστα με τα ονόματα των μαθητών
students = ["Άννα", "Γιώργος", "Μαρία", "Νίκος"]
```

```
# Δισδιάστατη λίστα με βαθμούς για 3 μαθήματα
# Κάθε υπολίστα αντιστοιχεί σε έναν μαθητή
grades = [
```

```
    [18, 15, 17], # Άννα
    [12, 14, 11], # Γιώργος
    [19, 20, 18], # Μαρία
    [10, 13, 12]  # Νίκος
]
```

```
# --- Υπολογισμός μέσου όρου κάθε μαθητή ---
print("\n📊 Μέσος όρος ανά μαθητή:")
print("-----")
```

```
for i in range(len(students)): # Εξωτερικός βρόχος: κάθε μαθητής
    total = 0
    for j in range(len(grades[i])): # Εσωτερικός βρόχος: κάθε μάθημα
        total += grades[i][j]
    avg = total / len(grades[i])
    print(f"{students[i]} → {avg:.2f}")
```

```
# --- Υπολογισμός μέσου όρου ανά μάθημα ---
print("\n📊 Μέσος όρος ανά μάθημα:")
print("-----")
```

```
num_subjects = len(grades[0])

for j in range(num_subjects): # Κάθε μάθημα (στήλη)
    total = 0
    for i in range(len(students)): # Κάθε μαθητής (γραμμή)
        total += grades[i][j]
    avg = total / len(students)
    print(f"Μάθημα {j + 1} → {avg:.2f}")
```

```
# --- Εύρεση καλύτερου μαθητή ---
averages = []
for i in range(len(students)):
    avg = sum(grades[i]) / len(grades[i])
    averages.append(avg)
```

```
max_avg = max(averages)
best_index = averages.index(max_avg)

print("\n🏆 Καλύτερος μαθητής:")
print("-----")
print(f"{students[best_index]} με μέσο όρο {max_avg:.2f}")
```

3. Δομές Επανάληψης

➤ Άσκηση: Να αποθηκεύσουμε τις βαθμολογίες πολλών μαθητών σε 3 μαθήματα (π.χ. Μαθηματικά, Φυσική, Πληροφορική) και να υπολογίσουμε:

1. Τον μέσο όρο κάθε μαθητή,
2. Τον μέσο όρο κάθε μαθήματος,
3. Τον καλύτερο μαθητή

```
# Λίστα με τα ονόματα των μαθητών
students = ["Αννα", "Γιώργος", "Μαρία", "Νίκος"]
```

```
# Δισδιάστατη λίστα με βαθμούς για 3 μαθήματα
# Κάθε υπολίστα αντιστοιχεί σε έναν μαθητή
```

```
grades = [
    [18, 15, 17], # Άννα
    [12, 14, 11], # Γιώργος
    [19, 20, 18], # Μαρία
    [10, 13, 12]  # Νίκος
]
```

```
# --- Υπολογισμός μέσου όρου κάθε μαθητή ---
```

```
print("\n 📊 Μέσος όρος ανά μαθητή:")
print("-----")
```

```
for i in range(len(students)): # Εξωτερικός βρόχος: κάθε μαθητής
    total = 0
    for j in range(len(grades[i])): # Εσωτερικός βρόχος: κάθε μάθημα
        total += grades[i][j]
    avg = total / len(grades[i])
    print(f"{students[i]} → {avg:.2f}")
```

```
# --- Υπολογισμός μέσου όρου ανά μάθημα ---
```

```
print("\n 📊 Μέσος όρος ανά μάθημα:")
print("-----")
```

```
num_subjects = len(grades[0])
```

```
for j in range(num_subjects): # Κάθε μάθημα (στήλη)
    total = 0
    for i in range(len(students)): # Κάθε μαθητής (γραμμή)
        total += grades[i][j]
    avg = total / len(students)
    print(f"Μάθημα {j + 1} → {avg:.2f}")
```

```
# --- Εύρεση καλύτερου μαθητή ---
```

```
averages = []
for i in range(len(students)):
    avg = sum(grades[i]) / len(grades[i])
    averages.append(avg)
```

```
max_avg = max(averages)
best_index = averages.index(max_avg)
```

```
print("\n 🏆 Καλύτερος μαθητής:")
print("-----")
print(f"{students[best_index]} με μέσο όρο {max_avg:.2f}")
```

3. Δομές Επανάληψης

➤ Άσκηση: Να αποθηκεύσουμε τις βαθμολογίες πολλών μαθητών σε 3 μαθήματα (π.χ. Μαθηματικά, Φυσική, Πληροφορική) και να υπολογίσουμε:

1. Τον μέσο όρο κάθε μαθητή,
2. Τον μέσο όρο κάθε μαθήματος,
3. Τον καλύτερο μαθητή

```
# Λίστα με τα ονόματα των μαθητών  
students = ["Άννα", "Γιώργος", "Μαρία", "Νίκος"]
```

```
# Δισδιάστατη λίστα με βαθμούς για 3 μαθήματα  
# Κάθε υπολίστα αντιστοιχεί σε έναν μαθητή  
grades = [  
    [18, 15, 17], # Άννα  
    [12, 14, 11], # Γιώργος  
    [19, 20, 18], # Μαρία  
    [10, 13, 12]  # Νίκος  
]
```

```
# --- Υπολογισμός μέσου όρου κάθε μαθητή ---  
print("📊 Μέσος όρος ανά μαθητή:")  
print("-----")
```

```
for i in range(len(students)): # Εξωτερικός βρόχος: κάθε μαθητής  
    total = 0  
    for j in range(len(grades[i])): # Εσωτερικός βρόχος: κάθε μάθημα  
        total += grades[i][j]  
    avg = total / len(grades[i])  
    print(f"{students[i]} → {avg:.2f}")
```

```
# --- Υπολογισμός μέσου όρου ανά μάθημα ---  
print("\n📊 Μέσος όρος ανά μάθημα:")  
print("-----")
```

```
num_subjects = len(grades[0])  
  
for j in range(num_subjects): # Κάθε μάθημα (στήλη)  
    total = 0  
    for i in range(len(students)): # Κάθε μαθητής (γραμμή)  
        total += grades[i][j]  
    avg = total / len(students)  
    print(f"Μάθημα {j + 1} → {avg:.2f}")
```

```
# --- Εύρεση καλύτερου μαθητή ---  
averages = []  
for i in range(len(students)):  
    avg = sum(grades[i]) / len(grades[i])  
    averages.append(avg)
```

```
max_avg = max(averages)  
best_index = averages.index(max_avg)
```

```
print("\n🏆 Καλύτερος μαθητής:")  
print("-----")  
print(f"{students[best_index]} με μέσο όρο {max_avg:.2f}")
```

3. Δομές Επανάληψης

- Άσκηση: Να αποθηκεύσουμε τις βαθμολογίες πολλών μαθητών σε 3 μαθήματα (π.χ. Μαθηματικά, Φυσική, Πληροφορική) και να υπολογίσουμε:

1. Τον μέσο όρο κάθε μαθητή,
2. Τον μέσο όρο κάθε μαθήματος,
3. Τον καλύτερο μαθητή

- ✓ Η δισδιάστατη λίστα grades είναι σαν ένας πίνακας:
κάθε γραμμή = μαθητής,
κάθε στήλη = μάθημα.
- ✓ Ο εξωτερικός for διατρέχει τις γραμμές (μαθητές).
- ✓ Ο εσωτερικός for διατρέχει τις στήλες (βαθμούς κάθε μαθητή).
- ✓ Χρησιμοποιούμε ενσωματωμένες συναρτήσεις όπως sum(), max(), index() για απλότητα.

```
# Λίστα με τα ονόματα των μαθητών
students = ["Αννα", "Γιώργος", "Μαρία", "Νίκος"]
```

```
# Δισδιάστατη λίστα με βαθμούς για 3 μαθήματα
# Κάθε υπολίστα αντιστοιχεί σε έναν μαθητή
grades = [
```

```
    [18, 15, 17], # Άννα
    [12, 14, 11], # Γιώργος
    [19, 20, 18], # Μαρία
    [10, 13, 12]  # Νίκος
]
```

```
# --- Υπολογισμός μέσου όρου κάθε μαθητή ---
print("\n 📊 Μέσος όρος ανά μαθητή:")
print("-----")
```

```
for i in range(len(students)): # Εξωτερικός θρόχος: κάθε μαθητής
    total = 0
    for j in range(len(grades[i])): # Εσωτερικός θρόχος: κάθε μάθημα
        total += grades[i][j]
    avg = total / len(grades[i])
    print(f"{students[i]} → {avg:.2f}")
```

```
# --- Υπολογισμός μέσου όρου ανά μάθημα ---
print("\n 📊 Μέσος όρος ανά μάθημα:")
print("-----")
```

```
num_subjects = len(grades[0])
```

```
for j in range(num_subjects): # Κάθε μάθημα (στήλη)
    total = 0
    for i in range(len(students)): # Κάθε μαθητής (γραμμή)
        total += grades[i][j]
    avg = total / len(students)
    print(f"Μάθημα {j + 1} → {avg:.2f}")
```

```
# --- Εύρεση καλύτερου μαθητή ---
averages = []
for i in range(len(students)):
    avg = sum(grades[i]) / len(grades[i])
    averages.append(avg)
```

```
max_avg = max(averages)
best_index = averages.index(max_avg)
```

```
print("\n 🏆 Καλύτερος μαθητής:")
print("-----")
print(f"{students[best_index]} με μέσο όρο {max_avg:.2f}")
```