Εισαγωγή στον Προγραμματισμό Η/Υ

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

Τμήμα Πληροφορικής, Ιόνιο Πανεπιστήμιο

4. Λίστες

Πίνακες (λίστες) στην Python

Οι πίνακες (λίστες) αποτελούν θεμελιώδη δομή δεδομένων και συνδέουν την αλγοριθμική σκέψη με τον προγραμματισμό.

Στόχοι Μαθήματος:

- Κατανόηση της έννοιας του πίνακα
- Εφαρμογή πινάκων στην Python
- Υλοποίηση βασικών αλγορίθμων με πίνακες
- Εξάσκηση σε πρακτικά παραδείγματα

Τι είναι Πίνακας?

- Συλλογή στοιχείων του ίδιου τύπου
- Κάθε στοιχείο προσπελαύνεται με δείκτη (index)
- Στην Python οι πίνακες υλοποιούνται με λίστες (lists)

4. Λίστες

Δημιουργία Λίστας στην Python

```
numbers = [10, 20, 30, 40]
names = ["Anna", "Nikos", "Maria"]
```

- Οι λίστες ορίζονται με αγκύλες [ ]
- Μπορούν να περιέχουν διάφορους τύπους δεδομένων
- Σε αντίθεση με άλλες γλώσσες, οι λίστες στην Python δεν απαιτούν δήλωση τύπου.

4. Λίστες

Δημιουργία Λίστας στην Python

Χαρακτηριστικό	Λίστα (Python)	Πίνακας (C)
Τύπος στοιχείων	Μπορεί να περιέχει **στοιχεία διαφορετικών τύπων** (π.χ. int, str, float μαζί)	Όλα τα στοιχεία πρέπει να είναι **του ίδιου τύπου** (π.χ. int a[5];)
Μέγεθος	**Δυναμικό**: μεγαλώνει ή μικραίνει αυτόματα με append(), remove(), κλπ.	**Στατικό**: το μέγεθος πρέπει να καθοριστεί **εκ των προτέρων**
Δήλωση	numbers = [1, 2, 3]	int numbers[3] = {1, 2, 3};
Μνήμη	Αποθηκεύεται ως **αντικείμενο υψηλού επιπέδου** με δείκτες σε άλλα αντικείμενα	Αποθηκεύεται σε **συνεχή περιοχή μνήμης** (χαμηλού επιπέδου)
Πρόσβαση	Μέσω δείκτη (index): numbers[0]	Μέσω δείκτη ή αριθμητικής δείκτη *(numbers + i)
Ενσωματωμένες λειτουργίες	Έχει πολλές μεθόδους: append, insert, sort, count, reverse, ...	Δεν έχει καμία έτοιμη λειτουργία: πρέπει να γραφτείς ο ίδιος ο αλγόριθμος
Έλεγχος ορίων (bounds checking)	Η Python **ελέγχει αυτόματα** τα όρια και δίνει σφάλμα αν προσπελαστεί μη έγκυρος δείκτης	Η C **δεν ελέγχει** τα όρια: μπορεί να γίνει πρόσβαση σε λάθος διεύθυνση (undefined behavior)
Απόδοση	Πιο αργή (λόγω δυναμικότητας & ασφάλειας)	Πιο γρήγορη (λόγω χαμηλού επιπέδου και άμεσης πρόσβασης στη μνήμη)

4. Λίστες

```
numbers = [1, 2, 3]
numbers.append(4)
print(numbers)
```

```
int numbers[4] = {1, 2, 3, 4};
// σταθερό μέγεθος - Δεν μπορούμε να κάνουμε "append" εύκολα
```

Δημιουργία Λίστας στην Python

Χαρακτηριστικό	Λίστα (Python)	Πίνακας (C)
Τύπος στοιχείων	Μπορεί να περιέχει **στοιχεία διαφορετικών τύπων** (π.χ. int, str, float μαζί)	Όλα τα στοιχεία πρέπει να είναι **του ίδιου τύπου** (π.χ. int a[5];)
Μέγεθος	**Δυναμικό**: μεγαλώνει ή μικραίνει αυτόματα με append(), remove(), κλπ.	**Στατικό**: το μέγεθος πρέπει να καθοριστεί **εκ των προτέρων**
Δήλωση	numbers = [1, 2, 3]	int numbers[3] = {1, 2, 3};
Μνήμη	Αποθηκεύεται ως **αντικείμενο υψηλού επιπέδου** με δείκτες σε άλλα αντικείμενα	Αποθηκεύεται σε **συνεχή περιοχή μνήμης** (χαμηλού επιπέδου)
Πρόσβαση	Μέσω δείκτη (index): numbers[0]	Μέσω δείκτη ή αριθμητικής δείκτη *(numbers + i)
Ενσωματωμένες λειτουργίες	Έχει πολλές μεθόδους: append, insert, sort, count, reverse, ...	Δεν έχει καμία έτοιμη λειτουργία: πρέπει να γραφτείς ο ίδιος ο αλγόριθμος
Έλεγχος ορίων (bounds checking)	Η Python **ελέγχει αυτόματα** τα όρια και δίνει σφάλμα αν προσπελαστεί μη έγκυρος δείκτης	Η C **δεν ελέγχει** τα όρια: μπορεί να γίνει πρόσβαση σε λάθος διεύθυνση (undefined behavior)
Απόδοση	Πιο αργή (λόγω δυναμικότητας & ασφάλειας)	Πιο γρήγορη (λόγω χαμηλού επιπέδου και άμεσης πρόσβασης στη μνήμη)

- Οι λίστες της Python είναι ευέλικτες και εύχρηστες, ιδανικές για γρήγορη ανάπτυξη και διδασκαλία αλγοριθμικής.
- Οι πίνακες της C είναι γρήγοροι και προβλέψιμοι, αλλά απαιτούν αυστηρή διαχείριση μνήμης και περισσότερη προσοχή.

4. Λίστες

Χειρισμός Λίστας στην Python

```
numbers = [10, 20, 30, 40]
names = ["Anna", "Nikos", "Maria"]
```

Πρόσβαση σε Στοιχεία

```
print(numbers[0])    # 10
print(numbers[-1])   # 40
```

Οι δείκτες ξεκινούν από το 0. και μπορούμε να χρησιμοποιήσουμε και αρνητικούς δείκτες.

Τροποποίηση Στοιχείων

```
numbers[1] = 25
print(numbers)  # [10, 25, 30, 40]
```

Οι λίστες είναι μεταβλητές δομές (mutable).

4. Λίστες

Χειρισμός Λίστας στην Python

```
numbers = [10, 20, 30, 40]
names = ["Anna", "Nikos", "Maria"]
```

- **Βασικές Λειτουργίες Λίστας**

```
numbers.append(50)
numbers.insert(2, 15)
numbers.remove(25)
```

- **Επανάληψη με for**

```
for x in numbers:
    print(x)
```

4. Λίστες

Χειρισμός Λίστας στην Python

```
numbers = [10, 20, 30, 40]
names = ["Anna", "Nikos", "Maria"]
```

- **Υπολογισμοί σε Πίνακα**

```
total = sum(numbers)
average = total / len(numbers)
print("Μέσος όρος:", average)
```

- **Αναζήτηση Στοιχείου**

```
target = 30
found = False
for x in numbers:
    if x == target:
        found = True
        break
```

4. Λίστες

Χειρισμός Λίστας στην Python

```
numbers = [10, 20, 30, 40]
names = ["Anna", "Nikos", "Maria"]
```

- **Υπολογισμοί σε Πίνακα**

```
total = sum(numbers)
average = total / len(numbers)
print("Μέσος όρος:", average)
```

- **Αναζήτηση Στοιχείου**

```
target = 30
found = False
for x in numbers:
    if x == target:
        found = True
        break
```

```
if 30 in numbers:
    print("Βρέθηκε!")
```

4. Λίστες

Χειρισμός Λίστας στην Python

```
numbers = [10, 20, 30, 40]
names = ["Anna", "Nikos", "Maria"]
```

- **Ταξινόμηση Λίστας**

```
numbers.sort()
numbers.reverse()
```

- **Bubble Sort (Αλγοριθμική Προσέγγιση)**

```
for i in range(len(numbers)-1):
for j in range(len(numbers)-1-i):
    if numbers[j] > numbers[j+1]:
        numbers[j], numbers[j+1] = numbers[j+1], numbers[j]
```

4. Λίστες

- **Χειρισμός Λίστας στην Python**

```
numbers = [10, 20, 30, 40]
names = ["Anna", "Nikos", "Maria"]
```

- **Ταξινόμηση Λίστας**

```
numbers.sort()
numbers.reverse()
```

- **Ελάχιστο και Μέγιστο**

```
max_value = max(numbers)
min_value = min(numbers)
```

- **Bubble Sort (Αλγοριθμική Προσέγγιση)**

```
for i in range(len(numbers)-1):
for j in range(len(numbers)-1-i):
    if numbers[j] > numbers[j+1]:
        numbers[j], numbers[j+1] = numbers[j+1], numbers[j]
```

Παράδειγμα Εφαρμογής Bubble Sort

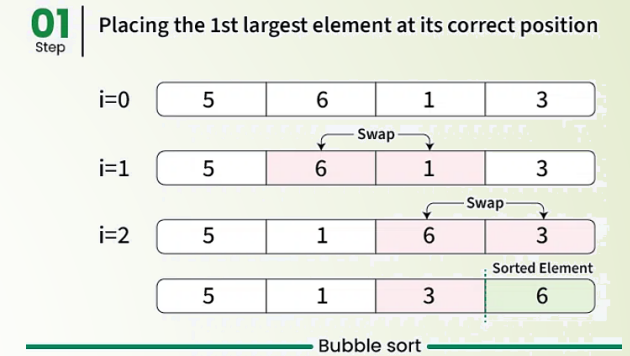
- Είσοδος: [5, 2, 9, 1]
- Διαδικασία: εναλλαγή γειτονικών στοιχείων
- Έξοδος: [1, 2, 5, 9]

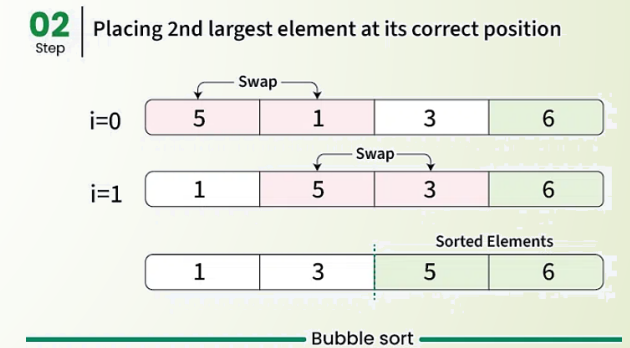
4. Λίστες

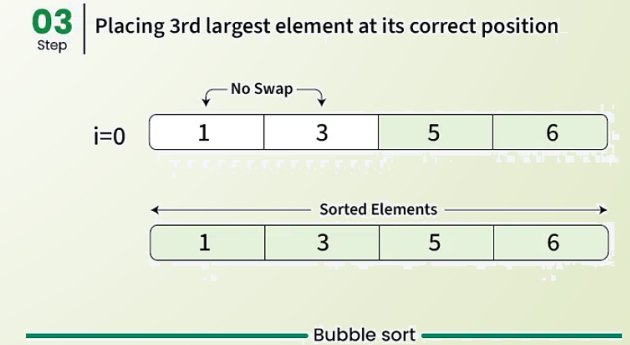
Χειρισμός Λίστας στην Python

Bubble Sort

```
def bubbleSort(arr):
    n = len(arr)

    # Traverse through all array elements
    for i in range(n):
        swapped = False

        # Last i elements are already in place
        for j in range(0, n-i-1):

            # Traverse the array from 0 to n-i-1
            # Swap if the element found is greater
            # than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        if (swapped == False):
            break

# Driver code to test above
if __name__ == "__main__":
    arr = [64, 34, 25, 12, 22, 11, 90]

    bubbleSort(arr)

    print("Sorted array:")
    for i in range(len(arr)):
        print("%d" % arr[i], end=" ")
```







4. Λίστες

Χειρισμός Λίστας στην Python

- **Πίνακες 2 Διαστάσεων (προσπελαύνουμε στοιχεία matrix[i][j])**

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

4. Λίστες

Χειρισμός Λίστας στην Python

- **Πίνακες 2 Διαστάσεων (προσπελαύνουμε στοιχεία matrix[i][j])**

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

- **Πρόσβαση σε 2D Πίνακα**

```
for row in matrix:
    for value in row:
        print(value, end=" ")
    print()
```

4. Λίστες

Χειρισμός Λίστας στην Python

- **Παράδειγμα: Άθροισμα Πίνακα**

```
total = 0
for row in matrix:
    for value in row:
        total += value
```

- **… εισαγωγή στο NumPy**

```
import numpy as np
a = np.array([1, 2, 3, 4])
print(a.mean())
```

4. Λίστες

- **Σύγκριση Python List vs NumPy Array**

Ιδιότητα	List	NumPy Array
Ευελιξία	Υψηλή	Περιορισμένη
Ταχύτητα	Μέτρια	Πολύ γρήγορη
Τύπος στοιχείων	Μικτός	Ενιαίος

- **Παράδειγμα NumPy Πίνακα 2D**

```
matrix = np.array([[1,2,3],[4,5,6]])
print(matrix[1,2])
```

4. Λίστες

- **Παράδειγμα Εφαρμογής:**

1. Δημιουργήστε λίστα 10 αριθμών και βρείτε το άθροισμα.
2. Εμφανίστε τα στοιχεία σε φθίνουσα σειρά.
3. Υπολογίστε το πλήθος των άρτιων στοιχείων.

4. Λίστες

Παράδειγμα Εφαρμογής:

1. Δημιουργήστε λίστα 10 αριθμών και βρείτε το άθροισμα.
2. Εμφανίστε τα στοιχεία σε φθίνουσα σειρά.
3. Υπολογίστε το πλήθος των άρτιων στοιχείων.

```
# 1. Δημιουργία λίστας 10 αριθμών και υπολογισμός αθροίσματος
numbers = [3, 7, 12, 5, 9, 1, 8, 4, 11, 2]
print("Άθροισμα:", sum(numbers))

# 2. Εμφάνιση των στοιχείων σε φθίνουσα σειρά
desc = sorted(numbers, reverse=True)
print("Φθίνουσα σειρά:", desc)

# 3. Υπολογισμός πλήθους άρτιων στοιχείων
even_count = 0
for x in numbers:
    if x % 2 == 0:
        even_count += 1
print("Πλήθος άρτιων:", even_count)
```

4. Λίστες

- **Παράδειγμα Εφαρμογής:**

1. Πίνακας βαθμών φοιτητών
2. Υπολογισμός μέσου όρου ανά φοιτητή
3. Εμφάνιση ονομάτων με βαθμό > 8

4. Λίστες

Παράδειγμα Εφαρμογής:

1. Πίνακας βαθμών φοιτητών
2. Υπολογισμός μέσου όρου ανά φοιτητή
3. Εμφάνιση ονομάτων με βαθμό > 8

```
# Πίνακας φοιτητών και βαθμών
students = ["Άννα", "Γιώργος", "Νίκος", "Μαρία"]
grades = [8.5, 9.2, 7.8, 9.8]

# 1. Υπολογισμός μέσου όρου ανά φοιτητή
for i in range(len(students)):
    print(f"{students[i]}: {grades[i]}")

# 2. Εμφάνιση όσων έχουν βαθμό > 8
print("\nΦοιτητές με βαθμό > 8:")
for i in range(len(students)):
    if grades[i] > 8:
        print(students[i])
```

4. Λίστες

- **Παράδειγμα Εφαρμογής:**

1. Υλοποιήστε τον αλγόριθμο αναζήτησης και ταξινόμησης χωρίς έτοιμες συναρτήσεις.

4. Λίστες

Παράδειγμα Εφαρμογής:

1. Υλοποιήστε τον αλγόριθμο αναζήτησης και ταξινόμησης χωρίς έτοιμες συναρτήσεις.

```
# Λίστα τιμών
values = [5, 2, 9, 1, 7, 3]

# 1. Γραμμική αναζήτηση
target = 7
found = False
for i in range(len(values)):
    if values[i] == target:
        print(f"Το στοιχείο {target} βρέθηκε στη θέση {i}")
        found = True
        break
if not found:
    print("Δεν βρέθηκε")

# 2. Bubble sort
for i in range(len(values)-1):
    for j in range(len(values)-1-i):
        if values[j] > values[j+1]:
            values[j], values[j+1] = values[j+1], values[j]

print("Ταξινομημένος πίνακας:", values)
```

4. Λίστες

Παράδειγμα Εφαρμογής:

1. Ψευδοκώδικας → Python

```
Διάβασε Ν
Για i από 1 μέχρι Ν
    Διάβασε Α[i]
Τέλος_Επανάληψης
```

4. Λίστες

Παράδειγμα Εφαρμογής:

1. Ψευδοκώδικας → Python

```
Διάβασε Ν
Για i από 1 μέχρι Ν
    Διάβασε Α[i]
Τέλος_Επανάληψης
```

```
# Εισαγωγή αριθμών σε λίστα από τον χρήστη
N = int(input("Πόσους αριθμούς θέλεις να δώσεις; "))

A = []
for i in range(N):
    num = int(input(f"Δώσε τον αριθμό {i+1}: "))
    A.append(num)

print("Η λίστα σου είναι:", A)
```

4. Λίστες

Συχνά Λάθη…

1. Λανθασμένη πρόσβαση σε δείκτη εκτός ορίων
2. Μη σωστή χρήση append()
3. Αντιγραφή λίστας με αναφορά αντί αντιγράφου

```
# Παράδειγμα: IndexError
numbers = [10, 20, 30]
# print(numbers[3])   # Σφάλμα: δείκτης εκτός ορίων

# Παράδειγμα: Αντιγραφή λίστας με αναφορά
a = [1, 2, 3]
b = a          # Τώρα δείχνει στην ίδια λίστα!
b.append(4)
print(a)       # [1, 2, 3, 4]

# Σωστός τρόπος αντιγραφής
c = a.copy()
c.append(5)
print("a:", a)
print("c:", c)
```

4. Λίστες

Καλές Πρακτικές…

1. Χρήση len() αντί σταθερών ορίων
2. Καθαρή ονοματοδοσία μεταβλητών
3. Τεκμηρίωση και σχόλια στον κώδικα

```
# Χρήση len() αντί για σταθερές
grades = [8, 9, 10, 7]
for i in range(len(grades)):
    print(f"Φοιτητής {i+1}: {grades[i]}")

# Καθαρή ονοματοδοσία
def average(values):
    return sum(values) / len(values)

print("Μέσος όρος:", average(grades))

# Τεκμηρίωση και σχόλια
# Υπολογίζει τον μέσο όρο βαθμών φοιτητών
```

4. Λίστες

Καλές Πρακτικές…

1. Τι επιστρέφει η len();
2. Πώς προσπελαύνουμε το τελευταίο στοιχείο μιας λίστας;
3. Ποια είναι η διαφορά λίστας και NumPy πίνακα;

```
# Βασικές λειτουργίες λίστας
nums = [4, 8, 2, 10]
nums.append(5)
nums.sort()
print("Ταξινομημένα:", nums)

# Μέσος όρος
print("Μέσος όρος:", sum(nums)/len(nums))

# Αναζήτηση στοιχείου
if 10 in nums:
    print("Το 10 υπάρχει στη λίστα!")
```

4. Λίστες

Καλές Πρακτικές…

1. Τι επιστρέφει η len();
2. Πώς προσπελαύνουμε το τελευταίο στοιχείο μιας λίστας;
3. Ποια είναι η διαφορά λίστας και NumPy πίνακα;

```
# Quiz - εκτελέστε και ελέγξτε τις απαντήσεις σας!

nums = [2, 4, 6, 8, 10]

# 1. Τι επιστρέφει η len(nums);
print(len(nums))  # → 5

# 2. Πώς προσπελαύνουμε το τελευταίο στοιχείο;
print(nums[-1])   # → 10

# 3. Ποια η διαφορά λίστας και NumPy πίνακα;
import numpy as np
a = np.array(nums)
print("Μέσος όρος με NumPy:", a.mean())
```

4. Λίστες

Άσκηση

Να γραφεί πρόγραμμα σε Python που: Διαβάζει τον αριθμό των φοιτητών και των μαθημάτων, εισάγει σε πίνακα (2D λίστα) τις βαθμολογίες κάθε φοιτητή, και υπολογίζει / εμφανίζει (τον μέσο όρο κάθε φοιτητή, τον μέσο όρο κάθε μαθήματος, ποιοι φοιτητές έχουν περάσει όλα τα μαθήματα (βαθμός ≥ 5), και ποιος έχει τον υψηλότερο μέσο όρο.

4. Λίστες

Άσκηση

Να γραφεί πρόγραμμα σε Python που: Διαβάζει τον αριθμό των φοιτητών και των μαθημάτων, εισάγει σε πίνακα (2D λίστα) τις βαθμολογίες κάθε φοιτητή, και υπολογίζει / εμφανίζει (τον μέσο όρο κάθε φοιτητή, τον μέσο όρο κάθε μαθήματος, ποιοι φοιτητές έχουν περάσει όλα τα μαθήματα (βαθμός ≥ 5), και ποιος έχει τον υψηλότερο μέσο όρο.

Ανάλυση Αλγορίθμου

1.Δομή επανάληψης for μέσα σε for → εισαγωγή και επεξεργασία πίνακα 2D
2.Δομή ελέγχου if → έλεγχος για βαθμούς κάτω από 5
3.Λίστες λιστών (2D πίνακες) → αποθήκευση δεδομένων
4.Χρήση break μέσα σε εμφωλευμένη δομή → έλεγχος συνθηκών
5.Χρήση max() και index() → εύρεση φοιτητή με το υψηλότερο αποτέλεσμα

```python
# Είσοδος δεδομένων
students = int(input("Δώσε αριθμό φοιτητών: "))
courses = int(input("Δώσε αριθμό μαθημάτων: "))

grades = []  # 2D λίστα (λίστα από λίστες)

# Εισαγωγή βαθμολογιών (nested loops)
for i in range(students):
    print(f"\nΦοιτητής {i+1}")
    row = []  # βαθμοί ενός φοιτητή
    for j in range(courses):
        mark = float(input(f"  Βαθμός μαθήματος {j+1}: "))
        row.append(mark)
    grades.append(row)

# Υπολογισμός μέσου όρου κάθε φοιτητή
averages = []
for i in range(students):
    avg = sum(grades[i]) / len(grades[i])
    averages.append(avg)
    print(f"Μέσος όρος φοιτητή {i+1}: {avg:.2f}")

# Υπολογισμός μέσου όρου κάθε μαθήματος (nested for)
print("\nΜέσος όρος ανά μάθημα:")
for j in range(courses):
    total = 0
    for i in range(students):
        total += grades[i][j]
    print(f"Μάθημα {j+1}: {total / students:.2f}")

# Εύρεση φοιτητών που πέρασαν όλα τα μαθήματα
print("\nΦοιτητές που πέρασαν όλα τα μαθήματα:")
for i in range(students):
    passed_all = True
    for mark in grades[i]:
        if mark < 5:
            passed_all = False
            break
    if passed_all:
        print(f"Φοιτητής {i+1}")

# Εύρεση φοιτητή με τον υψηλότερο μέσο όρο
max_avg = max(averages)
best_student = averages.index(max_avg) + 1
print(f"\nΦοιτητής {best_student} έχει τον υψηλότερο μέσο όρο: {max_avg:.2f}")
```