**ΜΕΡΟΣ 1: Εισαγωγή στον Αντικειμενοστραφή Προγραμματισμό (OOP) σε Python**

Ο **Αντικειμενοστραφής Προγραμματισμός (Object-Oriented Programming – OOP)** είναι ένας τρόπος οργάνωσης του κώδικα ώστε να μοιάζει με τον πραγματικό κόσμο.
Αντί να γράφουμε απλές εντολές, δημιουργούμε **αντικείμενα** που έχουν:

- **Ιδιότητες (attributes)** → δεδομένα
- **Μεθόδους (methods)** → λειτουργίες / τι μπορεί να κάνει το αντικείμενο

---

◆ **Γιατί OOP;**

- Κάνει τον κώδικα **πιο οργανωμένο**
- Μπορούμε να δημιουργήσουμε **πολλά αντικείμενα** χωρίς να ξαναγράφουμε τον ίδιο κώδικα
- Είναι **ευκολότερο στη συντήρηση**
- Χρησιμοποιείται σε μεγάλες εφαρμογές, AI, web apps, games κ.λπ.

---

◆ **Βασικές έννοιες OOP**

**1 Class (Κλάση)**

Το "σχέδιο" από το οποίο φτιάχνουμε αντικείμενα.

**2 Object (Αντικείμενο)**

Η πραγματική "υλοποίηση" της κλάσης.
Παράδειγμα: κλάση = "Σκύλος", αντικείμενο = ένας συγκεκριμένος σκύλος.

**3 Encapsulation (Ενθυλάκωση)**

Προστατεύει τα δεδομένα μέσα στο αντικείμενο.

**4 Inheritance (Κληρονομικότητα)**

Μια κλάση μπορεί να κληρονομήσει λειτουργίες από άλλη, π.χ.
`Dog` → κληρονομεί από `Animal`.

**5 Polymorphism (Πολυμορφισμός)**

Ίδιο όνομα μεθόδου – διαφορετική συμπεριφορά.

---

◆ **Μικρό παράδειγμα σε Python**

```
class Animal:
    def speak(self):
        return "..."

class Dog(Animal):
    def speak(self):
        return "Woof!"

dog = Dog()
print(dog.speak())
```

Αυτό δείχνει:

- Κληρονομικότητα
- Πολυμορφισμό
- Χρήση αντικειμένων

**Επεξήγηση Προγράμματος: Λήψη και Αποθήκευση Εικόνων από Ιστοσελίδα (Web Scraping με Python)**

Το πρόγραμμα αυτό κατεβάζει όλες τις εικόνες από μια ιστοσελίδα και τις αποθηκεύει σε έναν φάκελο στον υπολογιστή.
Χρησιμοποιεί τις βιβλιοθήκες:

- **requests** → κατέβασμα δεδομένων από το internet
- **BeautifulSoup** → ανάλυση HTML
- **os** → διαχείριση φακέλων και αρχείων
- **urljoin** → σωστή ένωση συνδέσμων

---

◆ **1. Εισαγωγή Βιβλιοθηκών**

```
import os
import requests
from bs4 import BeautifulSoup
from urllib.parse import urljoin
```

- `os` χρησιμοποιείται για δημιουργία φακέλων και διαχείριση αρχείων.
- `requests` για λήψη περιεχομένου από ιστοσελίδες.
- `BeautifulSoup` για εύρεση και ανάλυση στοιχείων HTML.
- `urljoin` για σωστή δημιουργία πλήρους URL μιας εικόνας.

---

◆ **2. Ορισμός της ιστοσελίδας**

```
url = "https://www.python.org"
```

Η μεταβλητή `url` περιέχει τη διεύθυνση της σελίδας από την οποία θα κατέβουν οι εικόνες.

---

### ◆ 3. Λήψη HTML κώδικα της σελίδας

```
r = requests.get(url)
soup = BeautifulSoup(r.text, "html.parser")
```

- `requests.get(url)` κατεβάζει τον HTML κώδικα.
- `BeautifulSoup` μετατρέπει τον HTML σε αντικείμενο που μπορεί να αναλυθεί.

---

### ◆ 4. Δημιουργία φακέλου για τις εικόνες

```
os.makedirs("images", exist_ok=True)
```

Αν δεν υπάρχει φάκελος με όνομα **images**, δημιουργείται.
Αν υπάρχει, δεν εμφανίζεται σφάλμα (`exist_ok=True`).

---

### ◆ 5. Εύρεση όλων των στοιχείων `<img>`

```
for img in soup.find_all("img"):
```

Η εντολή βρίσκει όλα τα tags εικόνων στη σελίδα.

---

### ◆ 6. Ανάγνωση του link της εικόνας

```
src = img.get("src")
if not src:
    continue
```

- Παίρνει το περιεχόμενο του `src`, δηλαδή τη διεύθυνση της εικόνας.
- Αν δεν υπάρχει `src`, προχωράει στο επόμενο.

---

### ◆ 7. Δημιουργία πλήρους URL της εικόνας

```
img_url = urljoin(url, src)
```

Κάποιες εικόνες έχουν **σχετικό URL** (π.χ. `/images/logo.png`).
Η `urljoin` τα μετατρέπει σε **πλήρη URL**, π.χ.:

```
https://www.python.org/images/logo.png
```

---

### ◆ 8. Δημιουργία ονόματος αρχείου

```
filename = img_url.split("/")[-1]
```

Παίρνει το τελευταίο μέρος του URL, που είναι και το όνομα της εικόνας.

---

### ◆ 9. Κατέβασμα της εικόνας

```
data = requests.get(img_url).content
```

Κατεβάζει τα bytes της εικόνας.

---

### ◆ 10. Αποθήκευση της εικόνας στον φάκελο

```
with open(os.path.join("images", filename), "wb") as f:
    f.write(data)
```

- Ανοίγει ένα αρχείο σε μορφή **binary** (`"wb"`).
- Γράφει μέσα τα δεδομένα της εικόνας.

---

### ◆ 11. Μήνυμα επιτυχίας

```
print("✓ Κατέβηκε:", filename)
```

Εμφανίζει το όνομα κάθε εικόνας που κατέβηκε.

---

### ◆ 12. Χειρισμός σφαλμάτων

```
except:
    pass
```

Αν προκύψει σφάλμα (π.χ. μη διαθέσιμο link), το πρόγραμμα συνεχίζει χωρίς να σταματά.

---

✓ **Τι κάνει συνολικά το πρόγραμμα**

1. Συνδέεται σε μια ιστοσελίδα.
2. Εντοπίζει όλες τις εικόνες.
3. Μετατρέπει κάθε link εικόνας σε πλήρες URL.
4. Κατεβάζει τις εικόνες.
5. Τις αποθηκεύει σε φάκελο που ονομάζεται **images**.

**νάλυση Προγράμματος: Εξαγωγή Τίτλων από Ιστοσελίδα (Web Scraping με Python)**

Το παρακάτω πρόγραμμα κατεβάζει τον HTML κώδικα μιας ιστοσελίδας και εμφανίζει όλους τους τίτλους που βρίσκονται σε στοιχεία `<h1>`, `<h2>` και `<h3>`.

---

◆ **1. Εισαγωγή βιβλιοθηκών**

```
import requests
from bs4 import BeautifulSoup
```

- **requests**: βιβλιοθήκη για λήψη περιεχομένου από ιστοσελίδες.
- **BeautifulSoup**: εργαλείο για ανάλυση και εύρεση στοιχείων μέσα σε HTML κώδικα.

---

◆ **2. Ορισμός της ιστοσελίδας**

```
url = "https://news.google.com"
```

Η μεταβλητή `url` κρατά τη διεύθυνση του site από το οποίο θα ληφθούν οι πληροφορίες.
Μπορεί να αλλαχθεί σε οποιαδήποτε άλλη σελίδα.

---

◆ **3. Λήψη HTML κώδικα της σελίδας**

```
r = requests.get(url)
```

Η `requests.get` στέλνει HTTP αίτημα και επιστρέφει την απάντηση του server.
Στη μεταβλητή `r.text` βρίσκεται το HTML περιεχόμενο της σελίδας.

---

### ◆ 4. Ανάλυση HTML με BeautifulSoup

```
soup = BeautifulSoup(r.text, "html.parser")
```

Το `soup` γίνεται ένα δέντρο HTML που επιτρέπει εύκολη αναζήτηση στοιχείων όπως `<h1>`, `<p>`, `<div>` κτλ.

---

### ◆ 5. Εύρεση όλων των τίτλων

```
titles = soup.find_all(["h1", "h2", "h3"])
```

Η `find_all` βρίσκει όλα τα στοιχεία της σελίδας που είναι είτε:

- `<h1>`
- `<h2>`
- `<h3>`

Έτσι καλύπτονται οι περισσότεροι τίτλοι και υπότιτλοι.

---

### ◆ 6. Εμφάνιση των τίτλων στην οθόνη

```
for t in titles:
    print("•", t.get_text(strip=True))
```

Για κάθε στοιχείο τίτλου:

- `t.get_text(strip=True)` επιστρέφει το καθαρό κείμενο χωρίς HTML
- Το πρόγραμμα εμφανίζει τον τίτλο με κουκκίδα •

Παράδειγμα εξόδου:

```
• Latest news
• Breaking stories
• World events
```

---

### ✓ Τι κάνει συνολικά το πρόγραμμα

1. Συνδέεται σε μια ιστοσελίδα.
2. Κατεβάζει τον HTML κώδικα.
3. Εντοπίζει όλους τους τίτλους `<h1>`, `<h2>`, `<h3>`.
4. Εμφανίζει το κείμενό τους στην οθόνη.

**Ανάλυση Προγράμματος: Αποθήκευση Καθαρού Κειμένου από Ιστοσελίδα**

Το πρόγραμμα κατεβάζει μια ιστοσελίδα, αφαιρεί όλο τον HTML κώδικα και αποθηκεύει μόνο το **καθαρό κείμενο** σε αρχείο.

---

### ◆ 1. Εισαγωγή βιβλιοθηκών

```
import requests
from bs4 import BeautifulSoup
```

- **requests** → χρησιμοποιείται για λήψη ιστοσελίδων μέσω HTTP.
- **BeautifulSoup** → αναλύει τον HTML κώδικα και επιτρέπει την εξαγωγή καθαρού κειμένου.

---

### ◆ 2. Ορισμός URL και λήψη της σελίδας

```
url = "https://www.python.org"
r = requests.get(url)
```

- Η μεταβλητή `url` περιέχει τη διεύθυνση της σελίδας που θα αναλυθεί.
- Η `requests.get()` κατεβάζει το περιεχόμενο της σελίδας.
- Στο `r.text` υπάρχει ο HTML κώδικας.

---

### ◆ 3. Ανάλυση του HTML

```
soup = BeautifulSoup(r.text, "html.parser")
```

Το περιεχόμενο μετατρέπεται σε μορφή που επιτρέπει εύκολη εξαγωγή κειμένου.

---

### ◆ 4. Εξαγωγή καθαρού κειμένου από την ιστοσελίδα

```
text = soup.get_text(separator="\n", strip=True)
```

Το `get_text()`:

- αφαιρεί όλο τον HTML κώδικα, αφήνοντας μόνο κείμενο
- προσθέτει αλλαγές γραμμής (`separator="\n"`) ανάμεσα στα κομμάτια κειμένου
- αφαιρεί κενά στην αρχή και στο τέλος (`strip=True`)

Έτσι προκύπτει καθαρό, διαβάσιμο κείμενο.

---

### ◆ 5. Αποθήκευση του κειμένου σε αρχείο

```
with open("page.txt", "w", encoding="utf-8") as f:
    f.write(text)
```

- Δημιουργείται (ή αντικαθίσταται) το αρχείο **page.txt**.
- Το περιεχόμενο γράφεται σε UTF-8 ώστε να υποστηρίζονται ελληνικά και ειδικοί χαρακτήρες.
- Η χρήση του `with` εξασφαλίζει ότι το αρχείο θα κλείσει σωστά.

---

### ◆ 6. Μήνυμα ολοκλήρωσης

```
print("✓ Το καθαρό κείμενο αποθηκεύτηκε στο page.txt")
```

Ενημερώνει τον χρήστη ότι το αρχείο δημιουργήθηκε με επιτυχία.

---

### ✓ Τι κάνει συνολικά το πρόγραμμα

1. Κατεβάζει μια ιστοσελίδα.
2. Αφαιρεί πλήρως τον HTML κώδικα.
3. Κρατά μόνο το καθαρό κείμενο.
4. Το αποθηκεύει σε αρχείο **page.txt**.

**Ανάλυση Προγράμματος: Παρακολούθηση Αλλαγών σε Ιστοσελίδα**

Το πρόγραμμα ελέγχει αν μια ιστοσελίδα **αλλάζει περιεχόμενο**.
Κάνει επαναλαμβανόμενα αιτήματα και συγκρίνει την τρέχουσα έκδοση της σελίδας με την προηγούμενη.

---

### ◆ 1. Εισαγωγή βιβλιοθηκών

```
import requests
import time
import hashlib
```

- **requests** → για λήψη της ιστοσελίδας
- **time** → για καθυστέρηση ανάμεσα στους ελέγχους
- **hashlib** → για δημιουργία hash, ώστε να συγκρίνουμε εύκολα τις αλλαγές

---

### ◆ 2. Ορισμός URL και εκτύπωση ενημερωτικού μηνύματος

```
url = "https://www.python.org"
print("Παρακολουθώ την σελίδα:", url)
```

Το πρόγραμμα θα ελέγχει την συγκεκριμένη σελίδα.
Η διεύθυνση μπορεί να αλλαχθεί σε οποιοδήποτε site.

---

### ◆ 3. Μεταβλητή αποθήκευσης προηγούμενου hash

```
old_hash = ""
```

Το `old_hash` θα κρατά την προηγούμενη «υπογραφή» της σελίδας.
Χρησιμοποιείται για να συγκρίνουμε αν το περιεχόμενο άλλαξε.

---

### ◆ 4. Ατέρμων βρόχος παρακολούθησης

```
while True:
```

Το πρόγραμμα τρέχει συνεχώς, ελέγχοντας την ιστοσελίδα ξανά και ξανά.

---

### ◆ 5. Λήψη τρέχοντος HTML περιεχομένου

```
r = requests.get(url)
```

Η σελίδα κατεβαίνει εκ νέου σε κάθε επανάληψη.

---

### ◆ 6. Δημιουργία hash του περιεχομένου

```
new_hash = hashlib.md5(r.text.encode("utf-
8")).hexdigest()
```

- `r.text` → το HTML της σελίδας
- `encode("utf-8")` → μετατροπή σε bytes
- `hashlib.md5()` → δημιουργεί μια «υπογραφή» του περιεχομένου
- `hexdigest()` → την επιστρέφει σε μορφή κειμένου

Αν έστω **ένας χαρακτήρας** αλλάξει στο HTML, το hash θα γίνει τελείως διαφορετικό.

---

### ◆ 7. Σύγκριση προηγούμενου και νέου hash

```
if old_hash and new_hash != old_hash:
    print("⚠ Η ιστοσελίδα ΑΛΛΑΞΕ!")
else:
    print("✓ Καμία αλλαγή...")
```

**Τι σημαίνουν οι δύο περιπτώσεις:**

- **`old_hash and new_hash != old_hash`**
  Αν υπάρχει προηγούμενο hash ΚΑΙ το νέο είναι διαφορετικό → η σελίδα άλλαξε.
- **`else`**
  Στην πρώτη επανάληψη ή όταν δεν υπάρχει αλλαγή → μήνυμα ότι όλα είναι ίδια.

---

### ◆ 8. Ενημέρωση του old_hash

```
old_hash = new_hash
```

Από εδώ και πέρα, αυτό θα είναι το σημείο αναφοράς.

---

### ◆ 9. Χρονική καθυστέρηση

```
time.sleep(10)
```

Το πρόγραμμα περιμένει **10 δευτερόλεπτα** πριν ξανακάνει έλεγχο.
Το διάστημα μπορεί να αλλάξει.

---

### ✓ Τι πετυχαίνει συνολικά το πρόγραμμα

1. Κατεβάζει την ιστοσελίδα.
2. Μετατρέπει το περιεχόμενο σε hash.
3. Το συγκρίνει με την προηγούμενη έκδοση.
4. Αν αλλάξει κάτι στο HTML → εμφανίζεται προειδοποίηση.
5. Επαναλαμβάνει τη διαδικασία κάθε 10 δευτερόλεπτα.