1- Mi az algoritmus?

A) Egy programozási nyelv neve
B) Egy lépések sorozata egy probléma megoldására
C) Egy fájltípus
D) Egy program végrehajtása

Megoldás

A helyes válasz:
B) Egy lépések sorozata egy probléma megoldására

Magyarázat:

- Az **algoritmus** egy pontosan meghatározott **lépéssorozat**, amelyet egy adott probléma megoldására dolgozunk ki. ☑
- **A)** Nem programozási nyelv (ilyen például a Python vagy a Java lenne). ❌
- **C)** Nem fájltípus (fájltípus például .txt, .jpg). ❌
- **D)** Nem a program végrehajtása maga, hanem a végrehajtandó lépések terve. ❌

2. Mit jelent a szekvencia?

A) Utasítások párhuzamos végrehajtása
B) Utasítások véletlenszerű végrehajtása
C) Utasítások egymás utáni végrehajtása
D) Utasítások rekurzív végrehajtása

Megoldás

A helyes válasz:
C) Utasítások egymás utáni végrehajtása

Magyarázat:

- A **szekvencia** azt jelenti, hogy az utasítások **sorban, egymás után, meghatározott sorrendben** hajtódnak végre. ☑
- **A)** Párhuzamos végrehajtás más fogalom (pl. szálkezelés – threading). ❌
- **B)** Véletlenszerű végrehajtás nem jellemző szekvenciára. ❌
- **D)** Rekurzió ismételt önhívást jelent, nem szekvenciális végrehajtást. ❌

🔘 **Egyszerű szekvencia példa:**

python

```
print("Első lépés")
print("Második lépés")
print("Harmadik lépés")
```

Kimenet:

css

```
Első lépés
Második lépés
Harmadik lépés
```

(A program **sorrendben** hajtja végre az utasításokat.)

Egy egyszerű összehasonlító táblázat a három alapvető vezérlési szerkezetről: **szekvencia**, **elágazás**, **ciklus**.

Vezérlési szerkezetek összehasonlítása Pythonban

Fogalom	Leírás	Példa kóddal	Mikor használjuk?
Szekvencia	Utasítások egymás után, sorrendben történő végrehajtása	print("1. lépés") print("2. lépés")	Ha mindent egymás után, változtatás nélkül kell végrehajtani.
Elágazás (döntés)	Feltétel alapján más utat választunk	if x > 0: print("Pozitív") else: print("Nem pozitív")	Ha a program különböző helyzetekben másképp kell viselkedjen.
Ciklus (ismétlés)	Egy műveletet többször végrehajtunk	for i in range(3): print(i)	Ha valamit ismételni kell (pl. lista bejárása, számolás).

Gyors magyarázat:

- **Szekvencia** = minden lépést egymás után, változtatás nélkül végzünk el.
- **Elágazás** = ha egy feltétel igaz vagy hamis, akkor más művelet történik.
- **Ciklus** = egy vagy több műveletet **többször** végrehajtunk.

3. Melyik NEM adattípus a Pythonban?

A) Integer
B) List
C) Switch
D) String

Megoldás

A helyes válasz:

C) Switch
Magyarázat:

- **A) Integer**: egész szám adattípus Pythonban → például 5 ☑
- **B) List**: lista adattípus Pythonban → például [1, 2, 3] ☑
- **C) Switch**: **nem** adattípus Pythonban.
  (A "switch" inkább vezérlési szerkezet néhány más nyelvben, de Pythonban ilyen adattípus nincs.) ✖
- **D) String**: szöveg adattípus Pythonban → például "Hello" ☑

🧩 Alapvető adattípusok Pythonban

Típus neve	Példa	Leírás
Integer (int)	5, -10, 0	Egész számok
Float (float)	3.14, -0.5	Tizedes törtszámok
String (str)	"Hello", 'Python'	Szövegek, karakterláncok
Boolean (bool)	True, False	Igaz/hamis logikai érték
List (list)	[1, 2, 3], ['apple', 'banana']	Több elem tárolása sorrendben
Tuple (tuple)	(1, 2, 3), ('a', 'b')	Nem módosítható (immutable) lista
Set (set)	{1, 2, 3}, {'apple', 'banana'}	Egyedi elemek halmaza
Dictionary (dict)	{'name': 'John', 'age': 30}	Kulcs–érték párok gyűjteménye

🧠 Megjegyzés:

- **Integer** és **Float**: számokkal dolgozunk.
- **String**: szövegekkel dolgozunk.
- **Boolean**: logikai döntésekhez használjuk.
- **List**, **Tuple**, **Set**, **Dictionary**: **összetett adattípusok**, amelyek több adatot tudnak tárolni.

4. Melyek bejárható objektumok a Pythonban?

A) lista (list)
B) szótár (dictionary)
C) integer
D) karakterlánc (string)

Megoldás

A helyes válaszok:

☑ **A) lista (list)**
☑ **B) szótár (dictionary)**
☑ **D) karakterlánc (string)**

Magyarázat:

- **lista (list)**: Igen, bejárható elemről elemre egy for ciklussal.
- **szótár (dictionary)**: Igen, bejárható a kulcsokon (vagy kulcs-érték párokon) keresztül.
- **karakterlánc (string)**: Igen, karakterenként végigjárható.
- **integer**: ❌ **Nem bejárható**, mert egy szám nem sorozat vagy gyűjtemény.

🔘 Egyszerű példák:

python

```
# lista bejárása
for item in [1, 2, 3]:
    print(item)

# szótár bejárása (kulcsok)
for key in {'name': 'Alice', 'age': 25}:
    print(key)

# szótár bejárása (kulcs-érték párok)
for key, value in {'name': 'Alice', 'age': 25}.items():
    print(key, value)

# string bejárása (karakterenként)
for char in "Python":
    print(char)
```

5. Mi a while-ciklus lényege?

A) Végtelen ismétlés
B) Feltétel nélküli ismétlés
C) Feltételes ismétlés, elöl tesztel
D) Csak egyszer hajtódik végre

Megoldás

A helyes válasz:
C) Feltételes ismétlés, elöl tesztel

Magyarázat:

- A **while-ciklus** addig ismétli a benne lévő utasításokat, **amíg a feltétel igaz**.
- **Elöl tesztel**, tehát **minden ciklus előtt** ellenőrzi a feltételt.
- Ha a feltétel már az elején hamis, akkor **egyszer sem** hajtódik végre.

🟩 Egyszerű példa a while-ciklusra:

python

```
i = 0
while i < 5:
    print(i)
    i += 1
```

Kimenet:

0
1
2
3
4

(Itt i folyamatosan nő, és a ciklus csak addig fut, amíg i < 5 igaz.)

Mi NEM igaz:

- **A)** Végtelen ismétlés csak akkor van, ha a feltétel **sosem lesz hamis** (de ez nem a while-ciklus lényege).
- **B)** Feltétel **nélkül** a while nem működik – mindig kell feltétel.
- **D)** Nem csak egyszer fut le – akár többször is, ha a feltétel teljesül.

6. A Pythonban az eljárás és a függvény külön nyelvi elem.

• Igaz
• Hamis

Megoldás

A helyes válasz:
Hamis

Magyarázat:

- A Pythonban **nincs külön nyelvi elemként** elválasztva az **eljárás** (procedure) és a **függvény** (function).
- **Minden függvény** a def kulcsszóval kezdődik, **és akár ad vissza értéket, akár nem**, ugyanúgy van kezelve.
- Ha egy függvény **nem ad vissza értéket**, akkor automatikusan a None értéket adja vissza.

🔘 Példa:
python

```
# Eljárás jellegű (nem ad vissza semmit)
def greet():
    print("Hello!")

# Függvény jellegű (visszaad egy értéket)
def add(a, b):
    return a + b
```

Mindkettőt ugyanúgy hozod létre és hívod meg.

Összefoglalva:

- Pythonban **nincs külön eljárás és függvény**.
- **Minden** def-fel létrehozott egység **függvény**, akár visszatérési értékkel, akár anélkül.

7. Mit tárol a változó?

A) Csak szöveges adatot
B) Egyetlen adatot, amely lehet bármilyen típusú
C) Csak számot
D) Csak ciklusokat

Megoldás

A helyes válasz:
B) Egyetlen adatot, amely lehet bármilyen típusú

Magyarázat:

- A **változó** egy név, amely egy **adatot tárol** a programban.
- Ez az adat **bármilyen típusú** lehet: szám, szöveg, lista, szótár, logikai érték stb.
- Egy változó **egyszerre egy értéket** tartalmaz (de ez lehet például egy lista is, ami több elemet tartalmaz).

🟤 Egyszerű példák:

python

```
# Számot tárol
x = 42

# Szöveget tárol
y = "Hello"
# Logikai értéket tárol
z = True

# Listát tárol
lista = [1, 2, 3]
```

⚡ Mi NEM igaz:

- **A)** Csak szöveges adatot? ❌ Nem, bármilyet lehet.
- **C)** Csak számot? ❌ Nem, szöveg, logikai érték, lista stb. is lehet.
- **D)** Csak ciklusokat? ❌ Ciklus nem adat, hanem vezérlési szerkezet.

8. A break utasítás kilép a ciklusból.

• Igaz
• Hamis

Megoldás

A helyes válasz:

☑ **Igaz**

Magyarázat:

A **break** utasítás **azonnal kilépteti** a programot a **legbelső ciklusból**, **akkor is, ha a ciklus feltétele még igaz** lenne.

Példa:

python

```
for i in range(5):
    if i == 3:
        break
    print(i)
```

Kimenet:

0
1
2

Amikor i == 3, a break lefut, és a ciklus **megszakad**.

9. Mit ad vissza a len() függvény?

A) Egy lista legnagyobb elemét
B) Egy objektum hosszát
C) Egy lista első elemét
D) Egy string utolsó karakterét

Megoldás

A helyes válasz:

☑ **B) Egy objektum hosszát**

▧ **Magyarázat:**

A **len()** függvény **visszaadja egy objektum (pl. lista, string, tuple, szótár) hosszát**, vagyis hogy **hány elemet tartalmaz**.

🔍 **Példák:**

python

```
# Lista hossza
print(len([1, 2, 3, 4]))     # Kimenet: 4

# Sztring hossza (karakterek száma)
print(len("Python"))         # Kimenet: 6

# Szótár hossza (kulcsok száma)
print(len({"a": 1, "b": 2}))  # Kimenet: 2
```

❌ **Mi NEM igaz:**

- **A)** legnagyobb elem → max() adja vissza ❌
- **C)** első elem → indexeléssel: lista[0] ❌
- **D)** utolsó karakter → pl. szöveg[-1] ❌

10. Mely függvények számolnak összegzést a listában?
A) sum()
B) min()
C) max()
D) append()

Megoldás

A kérdésedre a helyes válasz:

A) sum()

Magyarázat:

- **sum()**: összeadja a lista elemeit → **összegzést végez** ☑
- **min()**: a lista legkisebb elemét adja vissza → **nem összegez** ❌
- **max()**: a lista legnagyobb elemét adja vissza → **nem összegez** ❌
- **append()**: elemet fűz hozzá a listához → **nem összegez** ❌

A) sum()
Összeadja az összes számot a listában:
python

```
numbers = [1, 2, 3, 4, 5]
total = sum(numbers)
print(total)

# Kimenet: 15
```

B) min()
Megkeresi a lista legkisebb elemét:
python

```
numbers = [1, 2, 3, 4, 5]
minimum = min(numbers)
```

```
print(minimum)

# Kimenet: 1
```

C) max()
Megkeresi a lista legnagyobb elemét:
python

```
numbers = [1, 2, 3, 4, 5]
maximum = max(numbers)
print(maximum)

# Kimenet: 5
```

D) append()
Hozzáad egy elemet a lista végéhez:
python

```
numbers = [1, 2, 3, 4, 5]
numbers.append(6)
print(numbers)

# Kimenet: [1, 2, 3, 4, 5, 6]
```

Összefoglalva: **csak a sum() végez valódi összeadást**.

11. A for-ciklusban a ciklusváltozó automatikusan végighalad az objektum elemein.

• Igaz
• Hamis

Megoldás

A kérdésedre a helyes válasz:

☑ **Igaz**

▧ **Magyarázat:**
A **for-ciklus** Pythonban **automatikusan végighalad** az **iterálható objektum** (pl. lista, sztring, tuple, szótár) **elemein**, és minden lépésben az aktuális elemet hozzárendeli a **ciklusváltozóhoz**.

🔍 **Példa:**
python

```
gyümölcsök = ["alma", "banán", "körte"]

for gyümölcs in gyümölcsök:
    print(gyümölcs)
```

Kimenet:
nginx

```
alma
banán
körte
```

🔁 A gyümölcs ciklusváltozó **automatikusan végigmegy** az alma, banán, körte értékeken.

12. Mi a függvény visszatérési értéke?

A) Az eljárás neve
B) A függvény által visszaadott érték
C) A függvény paramétere
D) Egy változó típusa

Megoldás

A kérdésedre a helyes válasz:

☑ **B) A függvény által visszaadott érték**

▧ **Magyarázat:**
A **függvény visszatérési értéke** az az adat, amit a függvény **return** utasítással **visszaad** a hívónak.
Ez lehet szám, szöveg, lista, szótár, logikai érték – **bármi**, amit a függvény számít ki vagy előállít.

🔍 **Példa:**
python

```
def osszeg(a, b):
    return a + b

eredmeny = osszeg(3, 4)
print(eredmeny)  # Kimenet: 7
```

Itt:

- **osszeg()** a függvény
- **7** a **visszatérési érték**, mert a + b = 3 + 4

❌ **Mi nem helyes:**

- **A)** Eljárás neve: a függvény neve, nem az eredménye ❌
- **C)** Paraméter: bemeneti adat, nem visszatérési érték ❌
- **D)** Változó típusa: ez a type()-pel nézhető meg, nem függvény visszatérés ❌

13. Melyek listaműveletek?

A) append()
B) keys()
C) count()
D) strip()

Megoldás

A helyes válaszok:

☑ **A) append()**
☑ **C) count()**

Magyarázat:

Művelet	Típus	Listán működik?	Leírás
append()	listaművelet	☑ Igen	Hozzáad egy elemet a lista végéhez
count()	listaművelet	☑ Igen	Megszámolja, hogy egy adott elem hányszor szerepel a listában
keys()	szótárművelet	✗ Nem	Szótár kulcsait adja vissza (pl. my_dict.keys())
strip()	sztringművelet	✗ Nem	Sztring elejéről és végéről levág szóközöket (pl. " szöveg ".strip())

🔘 **Példa:**

python

```
my_list = [1, 2, 2, 3]

# append
my_list.append(4)
print(my_list)  # [1, 2, 2, 3, 4]

# count
print(my_list.count(2))  # 2
```

📋 Leggyakoribb listaműveletek Pythonban

Művelet	Leírás	Példa	Kimenet / Hatás
append(x)	Elem hozzáadása a lista végére	lista.append(5)	[1, 2, 3, 5]
insert(i, x)	Elem beszúrása adott indexre	lista.insert(1, 99)	[1, 99, 2, 3]
remove(x)	Első előfordulás törlése	lista.remove(2)	[1, 3]
pop(i)	Elem eltávolítása adott helyről (alapból a végéről)	lista.pop()	[1, 2], visszaadott érték: 3
count(x)	Megszámolja, hányszor szerepel az adott elem	lista.count(2)	2
index(x)	Visszaadja az első előfordulás indexét	lista.index(3)	2
sort()	Rendezés növekvő sorrendben	lista.sort()	[1, 2, 3]
reverse()	Megfordítja a lista elemeinek sorrendjét	lista.reverse()	[3, 2, 1]
extend([x, y])	Több elem hozzáadása egy másik listából	lista.extend([4, 5])	[1, 2, 3, 4, 5]
copy()	Másolat készítése a listáról	másolat = lista.copy()	Új, független lista

📝 Legfontosabb sztringműveletek Pythonban

Művelet	Leírás	Példa	Eredmény
upper()	Kisbetűkből nagybetűket csinál	"hello".upper()	"HELLO"
lower()	Nagybetűkből kisbetűket csinál	"HELLO".lower()	"hello"
capitalize()	Csak az első betű nagy	"hello".capitalize()	"Hello"
title()	Minden szó első betűje nagy lesz	"python is fun".title()	"Python Is Fun"
strip()	Szóközök (vagy karakterek) eltávolítása a sztring elejéről és végéről	" hello ".strip()	"hello"
lstrip() / rstrip()	Bal/jobb oldali szóközök eltávolítása	" hello".lstrip()	"hello"
replace(old, new)	Részlet kicserélése másikra	"alma".replace("a", "e")	"elme"
split(separator)	Feldarabolja a sztringet listaelemekké	"a,b,c".split(",")	["a", "b", "c"]
join(list)	Listaelemeket fűz össze egy sztringgé	" ".join(["Python", "is", "fun"])	"Python is fun"
find(substring)	Első előfordulás pozícióját adja vissza	"banana".find("na")	2

Művelet	Leírás	Példa	Eredmény
startswith(x)	Megnézi, hogy az adott szöveg az x résszel kezdődik-e	"python".startswith("py")	True
endswith(x)	Megnézi, hogy az adott szöveg az x résszel végződik-e	"script.py".endswith(".py")	True
len(s)	Megadja a sztring hosszát (karakterszámát)	len("hello")	5
in	Megnézi, hogy szerepel-e egy részlet a sztringben	"a" in "apple"	True

---

🔍 **Extra tipp**: A sztringek **nem módosíthatók (immutable)**, tehát minden művelet **új sztringet** hoz létre.

Legfontosabb szótárműveletek (dictionary)

Művelet	Leírás	Példa	Kimenet / Hatás
dict[key]	Érték lekérése kulcs alapján	d["név"]	'Anna'
dict.get(key[, default])	Biztonságos lekérés (hiba nélkül, ha nincs ilyen kulcs)	d.get("kor", 0)	25 vagy 0
dict.keys()	Összes kulcs	d.keys()	dict_keys(['név', 'kor'])
dict.values()	Összes érték	d.values()	dict_values(['Anna', 25])
dict.items()	Kulcs–érték párok listája	d.itms()	dict_items([('név', 'Anna'), ('kor', 25)])
dict.update(másik_dict)	Frissíti/megváltoztatja a szótárt másik szótár alapján	d.update({'kor': 30})	'kor' értéke 30 lesz
dict.pop(key)	Törli a megadott kulcsot és visszaadja az értékét	d.pop("kor")	25 (és a kulcs törlődik)
dict.popitem()	Törli és visszaadja az **utoljára hozzáadott** kulcs–érték párt	d.popitem()	('név', 'Anna')
dict.clear()	Kiüríti a szótárt	d.clear()	{} lesz belőle
key in dict	Megnézi, hogy szerepel-e a kulcs	'név' in d	True vagy False
len(dict)	Megadja, hány kulcs–érték pár van a szótárban	len(d)	2

14. Mi a szótár kulcs-érték párokból álló adatszerkezet angol neve?

A) list
B) dictionary
C) range
D) method

Megoldás

A helyes válasz:

B) dictionary

Magyarázat:

- A **dictionary** (szótár) egy **kulcs–érték párokból álló adatszerkezet** Pythonban.
- Például:
  python

  person = {"name": "Alice", "age": 30}
  Itt "name" és "age" a kulcsok, "Alice" és 30 az értékek.

🔍 **A többi válasz:**

- **A) list**: sorrendezett elemek gyűjteménye (indexeléssel) ❌
- **C) range**: számtartomány-generátor (pl. range(5) → 0-tól 4-ig) ❌
- **D) method**: egy objektumhoz tartozó függvény (pl. append() lista esetén) ❌

15. A range() függvény egy szöveges adatot állít elő.

• Igaz
• Hamis

Megoldás

A helyes válasz:

❌ **Hamis**

▧ **Magyarázat:**
A **range()** függvény **nem szöveges**, hanem **egész számok sorozatát állítja elő** egy speciális, **iterálható objektumban**.

16. Mit jelent a sorozatszámítás?

A) Egy sorozat elemeinek feltétel nélküli kiválasztása
B) Egy sorozat elemeinek összeadása vagy összeszorzása
C) Egy sorozatból csak a legkisebb elem kiválasztása
D) Egy sorozat megduplázása

Megoldás

A helyes válasz:

☑ **A helyes válasz: B) Egy sorozat elemeinek összeadása vagy összeszorzása**

▧ **Magyarázat – Sorozatszámítás:**

A **sorozatszámítás** egy **alapvető algoritmustípus**, amely során:

- Egy **sorozat elemein végzünk műveleteket** (pl. összeadás, szorzás),
- A cél: egy **összesített érték** meghatározása (összeg, szorzat, átlag stb.)

🔍 **Példa: Összegzés (összeadás)**

python

```
lista = [1, 2, 3, 4, 5]
osszeg = 0

for szam in lista:
    osszeg += szam

print(osszeg)  # Kimenet: 15
```

❌ **Mi nem helyes:**

- **A)**: nem csak kiválasztás, hanem számolás is történik ❌
- **C)**: legkisebb elem kiválasztása az **minimumkiválasztás**, nem sorozatszámítás ❌
- **D)**: a sorozat megduplázása más típusú művelet (nem összesítés) ❌

17. Mi az enumerate() függvény feladata?

A) Fájlok olvasása
B) Egy iterálható objektum elemeinek indexeléssel való bejárása
C) Egy lista elemeinek törlése
D) Fájlba írás

Megoldás

A helyes válasz:

☑ **A helyes válasz: B) Egy iterálható objektum elemeinek indexeléssel való bejárása**

▧ **Magyarázat – enumerate() függvény:**

Az **enumerate()** egy beépített Python függvény, amely:

- Egy **iterálható objektum** (pl. lista, sztring) elemein megy végig,
- És **minden elemhez hozzárendeli annak indexét is** (sorszámát),
- Így kényelmesen **egyszerre kapod meg az indexet és az értéket**.

🔍 **Példa:**

python

```
nevek = ["Anna", "Béla", "Cili"]

for i, nev in enumerate(nevek):
    print(i, nev)
```

Kimenet:

css

```
0 Anna
1 Béla
2 Cili
```

❌ **Mi nem igaz:**

- **A)** fájlok olvasása: azt az open() és read() végzi ❌
- **C)** listaelemek törlése: pl. remove(), del, pop() kell hozzá ❌
- **D)** fájlba írás: pl. write() vagy writelines() kell hozzá ❌

18. A szótár kulcsai nem ismétlődhetnek.

• Igaz
• Hamis

Megoldás

A helyes válasz:

☑ **Igaz**

Magyarázat:
A **szótár (dictionary)** kulcsai **egyediek** kell legyenek Pythonban – **nem ismétlődhetnek**.
Ha ugyanazt a kulcsot többször adod meg, **az utolsó érték felülírja az előzőt**.

Példa:
python

```
d = {"név": "Anna", "név": "Béla"}
print(d)
```

Kimenet:

python

```
{'név': 'Béla'}
```

👉 A "név" kulcs kétszer szerepelt, de csak **az utolsó érték** ("Béla") maradt meg.

Összefoglalva:

- ☑ Kulcsok **nem ismétlődhetnek**
- ☑ Értékek **ismétlődhetnek** (az lehet akár ugyanaz)

19. Mit csinál a strip() függvény?

A) Levágja a szóközöket a szöveg végéről
B) Levágja a speciális karaktereket a szöveg elejéről is
C) Megszünteti az összes sortörést a programban
D) Csak számokat ad vissza

Megoldás

A helyes válasz:

☑ **A helyes válasz: B) Levágja a speciális karaktereket a szöveg elejéről is**

▧ **Magyarázat: strip() függvény**

A **strip()** egy **sztringfüggvény**, amely:

- **Alapértelmezésben**: **eltávolítja a szóközöket** a **szöveg elejéről és végéről**
- **Paraméterrel**: **eltávolítja az összes megadott karaktert** (nemcsak szóközt), **a szöveg elejéről és végéről**

🔍 **Példák:**

python

```
# Szóközök levágása
s = "   Python   "
print(s.strip())  # Kimenet: "Python"

# Speciális karakterek eltávolítása
s = "...Hello!!!"
print(s.strip(".!"))  # Kimenet: "Hello"
```

❌ **Mi NEM igaz:**

- **A) → Csak** a végéről nem vág – **mindkét oldalról** vág ❌
- **C) →** Nem törli a sortöréseket mindenhol, csak ha azok az elején vagy végén vannak ❌
- **D) →** Nem ad vissza számokat, hanem **szövegeken dolgozik** ❌

20. A Python OOP nyelvként is használható.

• Igaz
• Hamis

Megoldás

A helyes válasz:

☑ **Igaz**

▧ **Magyarázat:**

A **Python** egy **többparadigmás nyelv**, ami azt jelenti, hogy:

- **struktúrált** (pl. függvényalapú),
- **objektum-orientált (OOP)**,
- és részben **funkcionális** programozási stílusban is használható.

◇ **Objektum-orientált programozás (OOP) Pythonban:**

Python teljes mértékben támogatja az OOP alapelveit:

☑ **Osztályok (class)**
☑ **Objektumok (példányok)**
☑ **Konstruktor (__init__)**
☑ **Öröklés, polimorfizmus, enkapszuláció**

🔍 **Egyszerű OOP példa:**

python

```
class Ember:
    def __init__(self, nev):
        self.nev = nev

    def koszon(self):
        print(f"Szia, {self.nev}!")

p1 = Ember("Anna")
p1.koszon()  # Kimenet: Szia, Anna!
```