09-algoritmizálás-01

1. Melyik állítás írja le legjobban az algoritmus fogalmát?

Helyes válasz: c) Egy probléma megoldásának lépésenkénti leírása ☑

Magyarázat:

- Az algoritmus egy **lépések sorozata**, amely egy adott probléma megoldására szolgál.
- Nemcsak számítógéppel hajtható végre – lehet recept, útvonalterv is.

Miért nem jók a többiek?

- a) „Egy programozási nyelv egyik eleme” – az algoritmus nem nyelvi elem, hanem elméleti fogalom.
- b) „Egy számítógépes alkalmazás” – ez inkább egy **program**, nem algoritmus.
- d) „Hardverkomponens működésének szabályzata” – ez **hardware specifikáció**, nem algoritmus.

---

2. Mit jelent az "IDE" rövidítés?

Helyes válasz: b) Integrated Development Environment ☑

Magyarázat:

- Az **IDE** egy fejlesztői környezet, amely **szerkesztőt, fordítót, hibakeresőt** és egyéb eszközöket egyesít.

Miért nem jók a többiek?

- a) „International...” – ilyen nem létezik a programozásban.
- c) „Intelligent Debugging Engine” – nem létező kifejezés.
- d) „Interactive Data Editor” – adatkezelésre utal, de nem fejlesztői környezet.

---

3. Mi a Python nyelv egyik fő jellemzője?

Helyes válasz: b) Értelmezett nyelv, soronként hajtja végre az utasításokat ☑

Magyarázat:

- A Python **interpretált nyelv**, nem szükséges előre lefordítani.
- Az utasításokat a **Python interpreter** futtatja sorban.

Miért nem jók?

- a) A bináris fájlt a fordító nyelvek készítik (pl. C, C++).
- c) Python nem **csak** OOP – procedurális és funkcionális is lehet.
- d) Python **több platformon is fut**, nem csak Windows-on.

---

4. Melyik programozási nyelv NEM tekinthető magas szintű nyelvnek?

Helyes válasz: b) Assembly ☑

Magyarázat:

- Az Assembly nyelv **alacsony szintű**, közel áll a gépi kódhoz.
- Nehezen olvasható, platformfüggő.

A többiek magas szintűek:

- a) C++: magas szintű, objektumorientált.
- c) Python: magas szintű, egyszerű szintaxis.
- d) Java: magas szintű, platformfüggetlen.

---

5. Melyik utasítás használható Pythonban adatbekérésre?

Helyes válasz: b) input() ☑

Magyarázat:

- `input()` lehetővé teszi a felhasználótól való szöveges adatbekérést.

Miért nem jók?

- a) `get_input()` nem létezik a Pythonban.
- c) `read()` fájlkezeléshez tartozik.
- d) `scanf()` a C nyelv utasítása.

---

6. Melyik állítás igaz az elágazásra (if-else) a Pythonban?

Helyes válasz: c) Az else ág nem kötelező, elhagyható ☑

Magyarázat:

- Pythonban az `if` önmagában is működik.

```
python
if feltétel:
    utasítás
```

- Az `else` vagy `elif` csak **opcionális**.

Miért nem jók?

- a) Egy `if`-hez **csak egy `else`** tartozhat, nem több.
- b) Pythonban az **„if”** kulcsszót kell használni, nem „ha”.
- d) Az `if` feltétele **lehet hamis is**, akkor nem hajtódik végre.

---

7. Melyik műveleti jel NEM található meg a Pythonban?

Helyes válasz: c) <> ☑

Magyarázat:

- A `<>` **régi Python 2 szintaxis** volt a „nem egyenlő” jelölésére, de **Python 3-ban már nem támogatott**.
- Helyette: `!=`.

A többiek mind léteznek:

- `+` – összeadás
- `%` – maradékos osztás (modulo)
- `//` – egész osztás

---

8. Melyik modult kell importálni a véletlenszám-generáláshoz?

Helyes válasz: b) random ☑

Magyarázat:

- A `random` modul tartalmaz véletlenszám-generáláshoz szükséges függvényeket, pl. `randint()`, `random()`, `choice()`.

Miért nem jók?

- a) `math`: matematikai függvények (pl. sqrt, pow), **nem véletlenszámhoz való**.
- c) `statistics`: statisztikai számításokhoz.
- d) `sys`: rendszerfunkciókhoz, mint pl. `sys.exit()`.

---

9. Hogyan lehet Pythonban egész számot szöveggé alakítani?

Helyes válasz: b) str() ☑

Magyarázat:

- A `str()` függvény segítségével **bármilyen értéket szöveggé alakíthatunk**.

```
python
szam = 5
szoveg = str(szam)  # '5'
```

Miért nem jók?

- a) `toString()` – Java-szintaxis, Pythonban nem létezik.
- c) `convertToString()` – nem létező függvény.
- d) `string()` – nincs ilyen beépített Python függvény.

---

10. Mi történik, ha egy ciklus feltétele mindig igaz?

Helyes válasz: b) Végtelen ciklus alakul ki ☑

Magyarázat:

- Ha a feltétel sosem válik hamissá, a ciklus **folyamatosan fut**, ezt nevezzük **végtelen ciklusnak**.

```
python
while True:
    print("Ez soha nem áll le")
```

Miért nem jók?

- a) „Azonnal kilép” – nem igaz, csak ha van `break`.
- c) „Egyszer sem fut le” – az csak akkor igaz, ha a feltétel eleve hamis.
- d) A Python **nem állítja le automatikusan** – a programozónak kell kezelni.

---

11. Melyik programozási nyelvek tekinthetők magas szintű nyelveknek?

Helyes válaszok:

- ☑ a) Python
- ☑ b) C++
- ❌ c) Assembly
- ☑ d) Java

Magyarázat:

- A **magas szintű nyelvek** az emberi nyelvhez közel álló utasításokat használnak, könnyebben olvashatók és hordozhatók.
    - **Python**: magas szintű, könnyen olvasható, interpretált nyelv.
    - **C++**: magas szintű, de közelebb van a gépi szinthez.
    - **Java**: szintén magas szintű, platformfüggetlen.

Miért nem jó a c)?

- Az **Assembly** alacsony szintű nyelv: egy adott processzor utasításkészletéhez közeli, nehezen olvasható.

---

12. Melyek az algoritmusok fő tulajdonságai?

Helyes válaszok:

- ☑ a) Véges számú lépésből állnak
- ☑ b) Determinisztikusak
- ❌ c) Csak számítógéppel végrehajthatók
- ❌ d) Mindig ciklusokat tartalmaznak

Magyarázat:

- **Végesség**: az algoritmusnak **véget kell érnie**.
- **Determináltság**: minden lépés **egyértelműen meghatározott**.

Miért nem jók a többi?

- c) Az algoritmus **nemcsak számítógéppel** hajtható végre (pl. főzési recept).
- d) Az algoritmus **nem feltétlenül tartalmaz ciklust** (pl. egyszerű döntésfa).

---

13. Melyik adattípus létezik Pythonban?

Helyes válaszok:

- ☑ a) int
- ☑ b) bool
- ❌ c) character
- ☑ d) float

Magyarázat:

- **int** – egész szám (pl. 3, -7)
- **bool** – logikai érték: `True`, `False`
- **float** – lebegőpontos szám (pl. 3.14)

Miért nem jó a c)?

- Pythonban **nincs külön "character" típus**, a karakter is **egyelemű stringként** szerepel: `'a'` → `str`

---

14. Mely műveletek érvényesek Pythonban?

Helyes válaszok:

- ☑ a) Összeadás
- ☑ b) Kivonás
- ☑ c) Szorzás
- ☑ d) Hatványozás

Magyarázat:
Pythonban mindegyik alapművelet használható:

- `+` – összeadás
- `-` – kivonás
- `*` – szorzás
- `**` – hatványozás (pl. `2**3 = 8`)

☑ Minden válasz **helyes**.

---

15. Melyek lehetnek a Python elágazásai?

Helyes válaszok:

- ☑ a) if
- ❌ b) elseif
- ☑ c) elif
- ☑ d) else

Magyarázat:

- **if** – alap feltételvizsgálat.
- **elif** – „else if” rövidítése, csak Pythonban így írjuk.
- **else** – minden más esetben.

Miért nem jó a b)?

- Pythonban **nincs „elseif” kulcsszó**, a helyes: `elif`

---

16. Mely változónevek érvényesek Pythonban?

Helyes válaszok:

- ☑ a) `valtozo_1`
- ❌ b) `1valtozo`
- ☑ c) `_ertek`
- ☑ d) `osszeg`

Magyarázat:
Pythonban a változónév:

- csak betűvel vagy aláhúzással kezdődhet,
- tartalmazhat számokat, de **nem kezdődhet számmal**,
- kis- és nagybetűérzékeny.

Helyes nevek:

- `valtozo_1` – betű+aláhúzás+sám kombinációja, jó.
- `_ertek` – aláhúzás is lehet kezdő karakter.
- `osszeg` – sima betűs név, érvényes.

Hibás:

- `1valtozo` – **számmal kezdődik**, ez szintaktikai hiba.

---

17. Mely programozási környezetek támogatják a Python nyelvet?

Helyes válaszok:

- ☑ a) PyCharm
- ☑ b) Visual Studio Code
- ☑ c) IntelliJ IDEA
- ❌ d) NetBeans

Magyarázat:

- **PyCharm** – kifejezetten Pythonhoz készült.
- **VS Code** – bővítményekkel remek Python támogatás.
- **IntelliJ IDEA** – Java központú, de Python plugin elérhető.
- **NetBeans** – főként Java és C/C++, Python támogatása **nagyon korlátozott** vagy elavult.

---

18. Mely parancsokkal íratható ki szöveg Pythonban?

Helyes válaszok:

- ❌ a) echo
- ☑ b) print()
- ❌ c) display
- ❌ d) write()

Magyarázat:

- **`print()`** a Python **alapvető kiíró függvénye**:

```
python
print("Helló, világ!")
```

Miért nem jók?

- `echo` – **Bash/parancssori utasítás**, nem Python.
- `display` – Pythonban csak bizonyos csomagokkal érhető el (pl. Jupyterben).
- `write()` – fájlba írásra használjuk, nem közvetlen szövegkiírásra konzolra.

---

19. Mely állítások igazak a Python ciklusaira?

Helyes válaszok:

- ☑ a) `while` ciklust használ feltételes ismétlésre
- ☑ b) `for` ciklust használhat iterálásra
- ❌ c) A ciklusok feltételei nem módosíthatók futás közben
- ❌ d) A `while` ciklus csak számokat kezel

Magyarázat:

- `while` ciklus: addig fut, amíg a feltétel igaz → **feltételes ciklus**
- `for` ciklus: **iterálható objektumokon** (lista, string stb.) fut végig
- A ciklus feltételei **futás közben is változhatnak** (pl. változók módosítása a ciklusban).
- A `while` **bármilyen logikai feltételt** vizsgálhat, nemcsak számokat.

---

20. Mely függvények tartoznak a `random` modulhoz?

Helyes válaszok:

- ☑ a) `randint()`
- ☑ b) `choice()`
- ❌ c) `rand()`
- ☑ d) `shuffle()`

Magyarázat:

- `randint(a, b)` – egész számot ad `a` és `b` között.
- `choice(lista)` – véletlenszerűen választ elemet egy listából.
- `shuffle(lista)` – megkeveri egy lista elemeit.

Miért nem jó a c)?

- `rand()` – ilyen **nincs** a `random` modulban. Ez pl. a C nyelv `stdlib.h` függvénye.

---

21. A Python egy fordított nyelv.

Válasz: ❌ Hamis

Magyarázat:

- A Python **értelmezett nyelv (interpreted)**, nem fordított (compiled).
- Nincs szükség előzetes fordításra `.exe` fájlba; soronként hajtódik végre.

---

22. Az „else” ág nem kötelező az if-else szerkezetben.

Válasz: ☑ Igaz

Magyarázat:

- Egy `if` elágazás működik `else` nélkül is.

```
python
MásolásSzerkesztés
if feltetel:
    print("Ez igaz")
```

- Az `else` csak az **opcionális „más esetben”** kezelést szolgálja.

---

23. A Pythonban a változókat előzetesen deklarálni kell.

Válasz: ❌ Hamis

Magyarázat:

- Python **dinamikusan típusos nyelv**, nem szükséges változót előre deklarálni.

```
python
a = 5  # automatikusan létrejön az "a" változó
```

- Nem kell megadni a típusát vagy előre bejelenteni, mint pl. C-ben.

---

24. A Python csak objektumorientált programozásra alkalmas.

Válasz: ❌ Hamis

Magyarázat:

- Python **több programozási paradigmát támogat**:
  - Objektumorientált (OOP)
  - Procedurális (utasítások egymás után)
  - Funkcionális (lambda, map, filter)

---

25. Az „and” és az „or” logikai operátorok Pythonban is használhatók.

Válasz: ☑ Igaz

Magyarázat:

- A logikai operátorok a Pythonban:
  - `and` → és
  - `or` → vagy
  - `not` → nem

```
python
if x > 0 and y > 0:
    print("Mindkettő pozitív")
```

---

26. Egy Python függvény mindig visszaad egy értéket.

Válasz: ❌ Hamis

Magyarázat:

- Ha nem használunk `return` utasítást, a függvény **automatikusan `None`-t ad vissza**.

```
python
def f():
    pass
print(f())  # None
```

---

27. A „while” ciklus addig fut, amíg a feltétele igaz.

Válasz: ☑ Igaz

Magyarázat:

- Ez a **while ciklus lényege**:

```
python
while feltetel:
    utasitas
```

- Amint a feltétel hamissá válik, kilép a ciklusból.

---

28. A Python azonos módon kezeli a kis- és nagybetűs változókat.

Válasz: ✗ Hamis

Magyarázat:

- Python **kis- és nagybetű érzékeny (case-sensitive)**:
  - `szam`, `Szam`, `SZAM` → három különböző változó

---

29. A Python beépített „math” modulja tartalmaz matematikai függvényeket.

Válasz: ☑ Igaz

Magyarázat:

- A `math` modul tartalmaz:
  - `sqrt()` – négyzetgyök
  - `pow()` – hatványozás
  - `floor()`, `ceil()` – kerekítések
  - `pi`, `e` – konstansok

---

30. A Python egy dinamikusan típusos nyelv.

Válasz: ☑ Igaz

Magyarázat:

- Pythonban **nem kell előre megadni a típusát** egy változónak:

```
python
x = 5       # int
x = "alma"  # str – típus változhat
```
