Python alapok

1. óra: Python telepítése, fejlesztői környezet beállítása

Elméleti rész:

- Python bemutatása és jelentősége
- Interpreter vs. compiler
- Telepítés, konfigurálás
- Integrált fejlesztői környezetek (IDE-k) rövid bemutatása: IDLE, PyCharm, VS Code

Gyakorlati feladat:
Telepítsd a Python legújabb stabil verzióját a számítógépedre, és állíts be egy fejlesztői környezetet!

Lépések:

1. Python letöltése és telepítése (ha nincs admin jog a gépen, akkor Microsoft Store-ból)
2. Környezeti változók beállítása (ha szükséges)
3. VS Code telepítése és Python bővítmény hozzáadása
4. Első Python fájl létrehozása a VS Code-ban
5. print("Hello World!")

2. óra: Live stream beállítása

- Live stream bővítmény telepítése a gépekre
- Lehetőségek bemutatása
- Kipróbálás
- Ha nem működik megfelelően akkor más alternatív stream-elési lehetőség kipróbálása

3. óra: A print() használata

Elméleti rész:

- A print() függvény alapvető használata
  - több szöveg kiírása
  - szám kiírása
- A print() további lehetőségei

- sep, end
- escape karakter
- hármas idézőjel
- "+" operátor
- idézőjel helyett aposztróf
- f-string
- sortörés (\n)

Bemutató feladatok:

```
# Írd ki a "Hello World!" üzenetet a képernyőre!
print("Hello World!")


# Módosítsd a programot úgy, hogy a saját nevedet írja ki!
print("Hello [Saját neved]!")


# Módosítsd, hogy a neved idézőjelben legyen, használj escape karaktert
print("Hello \"[Saját neved]\"!")


# Próbáld ki hármas idézőjellel
print("""Hello "[Saját neved]"!""")


# Módosítsd úgy, hogy a név jöjjön változóból.
# A sor végi felkiáltójel kiírásához használjuk az "end=" paramétert!
nev = "[Saját neved]"
print("Hello", nev, end="!")


# Most próbáld ugyanezt az eredményt elérni a "sep=" paraméter segítségével
nev = "[Saját neved]"
print("Hello ", nev, sep="")


# Próbáld ki f-string segítségével
nev = "[Saját neved]"
print(f"Hello {nev}!")
```

```python
# Írj ki 2 szót egymás alá egy print utasítással
print("alma\nbarack")


# Írd ki 2 változóban tárolt szám összegét
szam1 = 2
szam2 = 3
print(f"{szam1} és {szam2} összege: {szam1 + szam2}")
```

4. óra: A print() gyakorlása

Téma:

- rövid összefoglaló
- gyakorlati feladatok önálló megoldása
- kiértékelés

Önálló feladatok:

1. Írd ki a nevedet és életkorodat külön sorokban, külön print utasítással.
2. Írj ki egy rövid mondatot, ami tartalmaz egy idézőjeles szót.
3. Készíts egy háromszöget csillag (*) karakterekből (3 sor).

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

4. Írd ki 4 kedvenc színedet úgy, hogy vessző és szóközzel legyen elválasztva. Használd a "sep=" paramétert!
5. Írj ki egy számot és annak négyzetét egymás mellé. Használj változót és matematikai művelettel számolj!
6. Írd ki a hét napjait, minden nap új sorban legyen. Mindezt 1 print utasításban.
7. Írd ki a hét napjait, de egy sorban, kötőjellel elválasztva, mindezt 1 print utasításban. Használd a "sep=" paramétert!
8. Hozz létre egy szöveget és egy számot tartalmazó változót, majd foglald őket mondatba f-string használatával.
9. Adj ki két print utasítást valamilyen szöveg kiírásával, de az eredmény ne külön sorban jelenjen meg, hanem egymást mellett.
10. Írj ki egy \ karaktert. Nézz utána magad az interneten a mikéntjére.

5. óra: Egyszerű hibák felismerése, javítása, VSCode hibajelzések

Elméleti rész:

- Szintaktikai vs. logikai (szemantikai) hibák
- Futásidejű vs. fordítási idejű hibák
- VSCode jelzései fejlesztés közben

Gyakorlati feladatok:

```
# Vizsgáld meg az alábbi szintaktikai hibás kódokat egyesével.
# Nézd meg milyen jelzést ad rá a VSCode és mi történik, ha futtatjuk?
# Javítsd ki őket!
print("Hello)
print(Hello)
prin("Hello")
```

```
# Vizsgáld meg az alábbi kódot.
# Nézd meg milyen jelzést ad rá a VSCode és mi történik, ha futtatjuk?
szam1 = 1
szam2 = "2"
osszeg = szam1 + szam2
```

```
# Javítsd ki a logikai hibát az alábbi kódban.
# Vedd észre, hogy ilyenkor sem a VSCode, sem a futtató környezet nem jelez hibát,
# de mégsem helyes az eredmény.
a = 5
b = 3
kerulet = a + b
print(f"A téglalap kerülete={kerulet}")
```

```
# Javítsd ki a hibát az alábbi kódban.
# Vedd észre, hogy ilyenkor sem a VSCode, sem a futtató környezet nem jelez hibát,
# de mégsem helyes az eredmény.
a = 5
b = 3
print("A téglalap kerülete=2*(a+b)")
```

Önálló gyakorlati feladatok:
Javítsd az alábbi kódokban a hibát

1. Szintaktikai hiba – hiányzó zárójel

```
# Szintaktikai hiba – hiányzó zárójel
nev = "Anna"
print(f"Üdvözöllek, {nev}"
```

2. Futásidejű hiba – nullával való osztás

```
a = 10
b = 0
eredmeny = a / b
print(f"Eredmény: {eredmeny}")
```

3. F-String hiba – változó helyett idézőjelbe tett string

```
nev = "Anna"
print(f"Az én nevem 'nev'.")
```

4. Rossz változó név – változó elírása

```
szam = 25
print(f"A kiválasztott szám: {szama}")
```

5. Felesleges zárójelek – a műveletek helytelen használata

```
a = 10
b = 5
eredmeny = (a * (b)
print(eredmeny)
```

6. óra: Python szintaxis, megjegyzések

Elméleti rész:

- Python kód szerkezete: blokkok, behúzások
- Tab használata

- Általában 4 karakteres a behúzás, de a szabály, hogy legalább 1 legyen. Vizsgáljuk meg a VSCode állapotsávján a Spaces opciót.
- A VSCode igyekszik tartani a behúzás mértékét, próbáljuk ki!
- Egysoros és többsoros megjegyzések
- Ctrl+/ (angol) vagy Ctrl+ü (magyar) billentyűkombináció használata (hotkey), alternatíva Ctrl+K+C és Ctrl+K+U
- Miért jobb ezt használni? Mert az adott programozási nyelv szerint működik!

Bemutató feladatok:

```
# Normál 4 karakteres behúzás (OK)
if 3 > 2:
    print("Három nagyobb mint kettő")


# Egy karakteres behúzás (OK)
if 3 > 2:
 print("Három nagyobb mint kettő")


# Sok karakteres behúzás (OK)
if 3 > 2:
         print("Három nagyobb mint kettő")


# Nincs behúzás (HIBA)
if 3 > 2:
print("Három nagyobb mint kettő")


# Egy sorban (működik ugyan, de a VSCode figyelmeztetést ad).
# VIGYÁZZ! Soha ne használjuk így, nagyon rosszul olvasható kód!
if 3 > 2: print("Három nagyobb mint kettő")


# Két print a feltétel teljesülése esetén azonos behúzással (OK)
if 3 > 2:
    print("Három nagyobb mint kettő")
    print("Három nagyobb mint kettő")
```

```
# Két print a feltétel teljesülése esetén eltérő behúzással (Hiba)
if 3 > 2:
    print("Három nagyobb mint kettő")
     print("Három nagyobb mint kettő")


# Több szintű behúzás
if 3 > 2:
    print("Három nagyobb mint kettő")
    if 3 < 2:
        print("Három kisebb mint kettő")
    print("Három nagyobb mint kettő")
```

Megjegyzések

```
# Ez egy egysoros megjegyzés


"""
Ez egy többsoros
megjegyzés, amely
több sort foglal el
"""


# Ez is egy többsoros
# megjegyzés más módszerrel, amely
# több sort foglal el


print("Szia") # Sor végére is írhatunk megjegyzést
```

Gyakorlati feladatok:

1. Készíts egy programot, amely elvégez három matematikai műveletet (pl. összeadás, kivonás, szorzás). Kommentáld úgy, hogy minden sor elé írsz egy magyarázó megjegyzést.
2. Készíts egy programot, amely két változóban tárolt számot összead. A program elején használj többsoros megjegyzést amely leírja, hogy mit csinál a program.
3. Javítsd ki az alábbi fordítási hibás kódot:

```
if 5 > 3:
print("Öt nagyobb, mint három")
```

4. Próbáld ki, hogy mi történik, ha egy egyszerű print utasítást különböző mértékben húzol be (pl. 1 karakter, 4 karakter, 8 karakter). Figyeld meg, milyen jelzéseket ad a VSCode.

```
print("Ez az első üzenet")
 print("Ez a második üzenet")  # 1 karakteres behúzás
     print("Ez a harmadik üzenet")  # 4 karakteres behúzás
         print("Ez a negyedik üzenet")  # 8 karakteres behúzás
```

5. Írj egy programot, amely három üzenetet ír ki egymás után, de minden üzenet között helyezz el a kódban egy vagy több üres sort a parancsok közé.
   Vizsgáld meg, hogy az üres sor mennyiben befolyásolják program működését?

7. óra: Változók és adattípusok I.

Elméleti rész:

- Változók fogalma és használata
- Elnevezési konvenció:
  - beszédes legyen
  - csak betűvel kezdődhet
  - kerüljük az ékezetes karaktereket
  - több szavas változónév esetén aláhúzás karakterrel szeparálunk (snake_case)
  - további elnevezési konvenciók, pl. SCREAMING_SNAKE_CASE, camelCase, kebab-case
  - kis/nagy betűkre érzékeny (case-sensitive)!
  - a Python-ban már használt kulcsszó nem használható, mint pl. "if". Lásd még: https://www.w3schools.com/python/python_ref_keywords.asp
- Alapvető adattípusok: int, float, str, bool
- A type() függvény
- Típuskonverzió (kasztolás, casting)

Bemutató feladatok:

```python
# int típus - egész számot tárol és negatív értéket is felvehet
szam = 1

# str típus - szöveg tárolására
szoveg = "Szia"

# float típus - tizedes szám
tizedes_szam = 3.14

# bool típus - két állapota lehet, csak igaz/hamis (True vagy False) értéket vehet fel
elinditva = True

# Kiíratás
print(szam)
print(szoveg)
print(tizedes_szam)
print(elinditva)

# Típus lekérdezése
print(type(szam))
print(type(szoveg))
print(type(tizedes_szam))
print(type(elinditva))


# A változó típusa futás közben is változhat.
# Bár ezt célszerű kerülni, mert megtévesztő lehet.
adat = 1
print(type(adat))

adat = 'Szia'
print(type(adat))


# szám tárolása str-ként
szoveg = str(1)
print(type(szoveg))

# str átalakítása int-re
szam = int("1")
print(type(szam))
```

Gyakorlati feladatok:

1. Egész számok (int):
    - Hozz létre két egész számot és végezz velük műveleteket.
    - Számold ki az összegüket, különbségüket, szorzatukat.
2. Karakterláncok (str):
    - Hozz létre két karakterláncot és fűzd őket össze.
    - Hozz létre egy egész számot és fűzd azt is hozzá.
    - Jelenítsd meg az eredmény.
3. Lebegőpontos számok (float):
    - Hozz létre két lebegőpontos számot és végezz velük műveleteket.
    - Számold ki az összegüket, különbségüket, szorzatukat.
4. Logikai értékek (bool):
    - Hozz létre igaz és hamis értékeket.
    - Ellenőrizd és írd ki típusát.
5. Típuskonverziók:
    - Alakíts át egy számot szöveggé és fordítva.
    - Konvertálj egész számot (int) lebegőpontossá (float).
    - Ellenőrizd és írd ki az eredmények típusát.
6. Az alábbi kód hibára fut, mi lehet a gond?

```
pontszam = 1
print(Pontszam)
```

8. óra: Változók és adattípusok II., egyszerűbb string műveletek

Elméleti rész:

- Konstansok (névkonvenció)
- Többszörös értékadás
- Egyszerűbb string metódusok (lower, upper, len)
- String indexálás

Bemutató feladatok:

```
# Konstans definiálás
PI = 3.14
```

```python
# Több változó beállítása egy sorban
x, y, z = "alma", "barack", "szilva"
print(x)
print(y)
print(z)

# Több változó ugyanazon értékre állítása
x = y = z = "alma"
print(x)
print(y)
print(z)


# Szöveg hosszának lekérdezése
szoveg = "Hello, World!"
print(len(szoveg))


szoveg = "Hello, World!"
# Kisbetűsre alakítás
print(szoveg.lower())
# Nagybetűsre alakítás
print(szoveg.upper())


szoveg = "123456789"
print("Teljes szöveg:", szoveg)
print("Legelső karakter:", szoveg[0])
print("Az 5. karakter:", szoveg[4])
print("Első 5 karakter:", szoveg[:5])
print("Az 5. karakter utáni rész:", szoveg[5:])
print("Utolsó karakter:", szoveg[-1])
print("Utolsó előtti karakter:", szoveg[-2])
print("A 4. karaktertől a 6.-ig:", szoveg[3:6])
print("A végéről számolva a 6. karaktertől az utolsóig:", szoveg[-6:-1])
print("Minden második karakter: ", szoveg[::2])
print("Szöveg megfordítva: ", szoveg[::-1])
```

Gyakorlati feladatok:

1. Hozz létre 2 karakterláncot (legyen bennük kis és nagybetű), fűzd össze őket és írd ki a teljes hosszát.
2. A fenti karakterláncot alakítsd csupa kisbetűssé.

3. A fenti karakterláncot alakítsd csupa nagybetűssé.
4. Hozz létre egy karakterláncot "abcdef" tartalommal és írd ki a 2. karakterét.
5. Írd ki az utolsó 2 karakterét.
6. Írd ki a középső 2 karakterét.
7. Írd ki minden második karakterét.
8. Írd ki fordítva.

9. óra: Operátorok és kifejezések I.

Elméleti rész:

- Aritmetikai operátorok (+, -, *, /, //, %, **)
- Összehasonlító operátorok (==, !=, <, >, <=, >=)
- Logikai operátorok (and, or, not)

Bemutató feladatok:

```
# Aritmetikai operátorok
a = 10
b = 3

print(a + b)  # Összeadás: 13
print(a - b)  # Kivonás: 7
print(a * b)  # Szorzás: 30
print(a / b)  # Osztás: 3.333...
print(a // b) # Egész osztás: 3
print(a % b)  # Maradékos osztás: 1
print(a ** b) # Hatványozás: 1000


# Összehasonlító operátorok
x = 5
y = 10

print(x == y)  # Egyenlő-e: False
print(x != y)  # Nem egyenlő: True
print(x < y)   # Kisebb: True
print(x > y)   # Nagyobb: False
print(x <= y)  # Kisebb vagy egyenlő: True
print(x >= y)  # Nagyobb vagy egyenlő: False
```

```
# Logikai operátorok
x = 5
y = 7
print("x nagyobb mint 3 és kisebb mint 7:", x > 3 and x < 7)
print("y nagyobb mint 3 és kisebb mint 7:", y > 3 and y < 7)
print("x kisebb mint 6 vagy nagyobb mint 8:", x < 6 or x > 8)
print("y kisebb mint 6 vagy nagyobb mint 8:", y < 6 or y > 8)
print("x nem nagyobb mint 6", not x > 6)
print("y nem nagyobb mint 6", not y > 6)

# Érdekesség, a fentiek megoldhatók logikai operátor használata nélkül is, pl:
print("x nagyobb mint 3 és kisebb mint 7:", 3 < x < 7)
print("x nagyobb mint 3 és kisebb mint 7:", 3 < y < 7)
```

Gyakorlati feladatok:

1. Készíts 2 szám változót és írd ki mennyiszer van meg a kisebbik a nagyobban.
   A maradékot is írd ki.
2. Számold ki egy szám négyzetét és köbét.
3. Készíts 2 szám változót, pl. legyen a=3 ás b=4. Készíts 4 print utasítást. Eredménytől függően írjon ki True vagy False-t ha:
   - a egyenlő b-vel
   - a eltérő b-nél
   - a kisebb mint b
   - a nagyobb mint b
4. Készíts 3 szám változót, az egyik legyen 23 a másik 18, a harmadik 29. Készíts 3 print utasítást, külön mindhárom változóra olyan operátorral ami megállapítja, hogy a szám 15 és 20 között van-e. Írjon ki True vagy False értéket az eredménytől függően.

10. óra: Operátorok és kifejezések II.

Elméleti rész:

- Értékadó operátorok (=, +=, -=, *=, /=, //=)
- Operátorok precedenciája
  https://www.w3schools.com/python/python_operators.asp

Bemutató feladatok:

```
# Értékadó operátor
szam = 10
szam = szam + 5  # 10 + 5
print(szam)  # 15

# Ugyanez rövidített szintaxissal
szam = 10
szam += 5  # 10 + 5
print(szam)  # 15

# További néhány operátor
szam -= 3  # 15 - 3
print(szam)  # 12

szam *= 2  # 12 * 2
print(szam)  # 24

szam /= 4  # 24 / 4
print(szam)  # 6.0
```

Önálló gyakorlati feladatok:

1. Mutasd be az értékadó operátorok használatát!
   Definiálj egy változót, adj hozzá értéket a += operátorral, majd vonj le belőle a -= operátorral!
   Végezz szorzást, osztást, maradékszámítást az értékadó operátorokkal!
2. Próbáld meg fejben kiszámolni mennyi lesz az alábbi kifejezések eredménye

   ```
   eredmeny1 = (10 - 2) * 3
   eredmeny2 = 10 - 2 * 3
   eredmeny3 = 8 / 2 + 3 * 2
   ```

3. Próbáld meg fejben kiszámolni mennyi lesz az alábbi kifejezés eredménye (papíron levezetheted)

   ```
   eredmeny = 5 + 3 * 2**3 / 2 - 1
   ```

Megoldások:
2. 24, 4, 10.0
3. 16.0

11. óra: A felhasználói input kezelése I.

Elméleti rész:

- Az input() függvény használata
- Típuskonverziók

Bemutató feladatok:

```
# Név bekérés
nev = input("Add meg a neved: ")
print("Szia", nev)


# Kor bekérés típuskonverzióval
kor = input("Hány éves vagy? ")
kor_szam = int(kor)
print("Jövőre ennyi idős leszel:", kor_szam + 1)


# Rövidebb, összevont változat
kor = int(input("Hány éves vagy? "))
print("Jövőre ennyi idős leszel:", kor + 1)


# Két szám bekérése és műveletvégzés
szam1 = int(input("Add meg az első számot: "))
szam2 = int(input("Add meg a második számot: "))
osszeg = szam1 + szam2
print("A két szám összege:", osszeg)


# Tizedes szám bekérése
szam = float(input("Adj meg egy tizedes számot: "))
print("A szám:", szam)


# Kérjük be a felhasználó nevét és életkorát, majd írjuk ki őket
nev = input("Add meg a neved: ")
kor = int(input("Hány éves vagy? "))
print(f"{nev} {kor} éves.")
```

Önálló gyakorlati feladatok:

1. Kérj be egy számot a felhasználótól, és írd ki annak a négyzetét!
2. Kérj be két számot a felhasználótól, és számold ki az összegüket, különbségüket és szorzatukat!
3. Kérj be egy tizedes számot a felhasználótól, és írd ki a felét!
4. Kérd be a téglalap a és b oldalát és számold ki a területét, kerületét.

12. óra: A felhasználói input kezelése II.

Elméleti rész:

- Hibakezelés

```
# Figyeljük meg mi történik, ha nem számot adunk meg.
# ValueError kivétel keletkezik.
szam = int(input("Adj meg egy számot: "))
print("A megadott szám:", szam)


# Egyszerű szám vizsgálat
# Csak egész számra működik!
szam = "1"
if szam.isdigit():
    print("Számot adtál meg")



# Alternatíva: ValueError kezelése
# Ez float-ra is működik!
try:
    szam = int(input("Adj meg egy számot: "))
    print("A megadott szám:", szam)
except ValueError:
    print("Ez nem egy érvényes szám.")
```

Összetett gyakorlati feladat:

1. A program kérje be a nevet, majd a születési évet. Ebből számolja ki hány éves. Végül írja ki az alábbi szöveget, behelyettesítve az eredménnyel:
   `Szia 'XY', jelenleg "X" éves vagy.`
   Fontos: a név legyen aposztrófok között, a kor legyen idézőjelben. A kiíratáshoz a print-ben használj f-string-et.
2. Kezeld a hibát, amennyiben nem számot ad meg.

3. Módosítsd úgy, hogy a név csupa nagy betűvel íródjon ki.
4. A kor számításnál az aktuális évet ne fixen add meg, hanem kérdezd le a rendszeridőből! Az ehhez szükséges utasítást próbáld fellelni az interneten!

Lehetséges megoldás:

```
from datetime import datetime

nev = input("Neved: ")
try:
    ev = int(input("Születési éved: "))
    kor = datetime.now().year - ev
    print(f"Szia '{nev.upper()}', jelenleg \"{kor}\" éves vagy.")
except ValueError:
    print("Nem számot adtál meg!")
```

13. óra: Feltételes utasítások I.

Elméleti rész:

- if, else
- Gyakori operátorok (==, !=, <, >, <=, >=)

Bemutató feladatok:

```
# Egyszerű feltétel egy ággal
# Megakadályozza, hogy a szám negatív legyen
szam = -1
if szam < 0:
    szam = 0
print(szam)


# Egyszerű feltétel különben ággal
szam = 10
if szam > 0:
    print("A szám pozitív.")
else:
    print("A szám nem pozitív.")
```

```python
# Egyszerű feltétel művelettel
szam = 4
if szam % 2 == 0:
    print("A szám páros.")
else:
    print("A szám páratlan.")


# Összehasonlító feltétel
a, b = 8, 12
if a > b:
    print("A nagyobb.")
else:
    print("B nagyobb vagy egyenlő.")


# Szöveg vizsgálata
valasz = input("Mi Magyarország fővárosa? ")
if valasz.lower() == "budapest":
    print("Helyes!")
else:
    print("Nem jó a válasz.")
```

Önálló gyakorlati feladatok:

1. Kérj be egy számot, és döntsd el, hogy a szám pozitív, vagy negatív!
2. Írj egy programot, amely eldönti egy számról, hogy páros-e vagy páratlan!
3. Kérj be két számot, és döntsd el, hogy melyik a nagyobb!
4. Kérdj be egy nevet és ellenőrizd, hogy a "Géza" nevet adta-e meg. Kis-nagybetűre ne legyen érzékeny!
5. Kérj be egy számot, és írd ki, hogy a szám nagyobb-e 10-nél!

14. óra: Feltételes utasítások II.

Elméleti rész:

- Többszörös elágazások elif használatával
- Logikai operátorok használata (and, or, not)

Bemutató feladatok:

```python
# Vizsgáljuk meg, hogy egy szám pozitív, negatív vagy nulla-e
szam = 0
if szam > 0:
    print("A szám pozitív.")
elif szam < 0:
    print("A szám negatív.")
else:
    print("A szám nulla.")


# Tartomány vizsgálata logikai operátorral
szam = 50
if szam >= 0 and szam <= 100:
    print("A szám 0 és 100 között van.")
else:
    print("A szám nincs 0 és 100 között.")


# Logikai operátor + művelet
szam = 15
if szam % 3 == 0 and szam % 5 == 0:
    print("A szám osztható 3-mal és 5-tel is.")
else:
    print("A szám nem osztható 3-mal és 5-tel.")


# Jelezzük ha a szám nem nulla
szam = -1

# Összehasonlító operátorral
if szam != 0:
    print("A szám nem nulla")

# Ugyanez logikai not operátorral
if not szam == 0:
    print("A szám nem nulla")
```

Önálló gyakorlati feladatok:

1. Kérj be egy számot, és ellenőrizd, hogy pozitív, negatív vagy nulla!
2. Kérj be egy életkort, és döntsd el, hogy gyerek, tinédzser vagy felnőtt-e!
3. Kérj be egy számot, és ellenőrizd, hogy osztható-e 2-vel és 3-mal is!

4. Írj egy programot, amely egy számról eldönti, hogy 100 és 200 között van-e!
5. Kérj be egy számot, és ellenőrizd, hogy a szám 10-nél nagyobb és 50-nél kisebb-e!

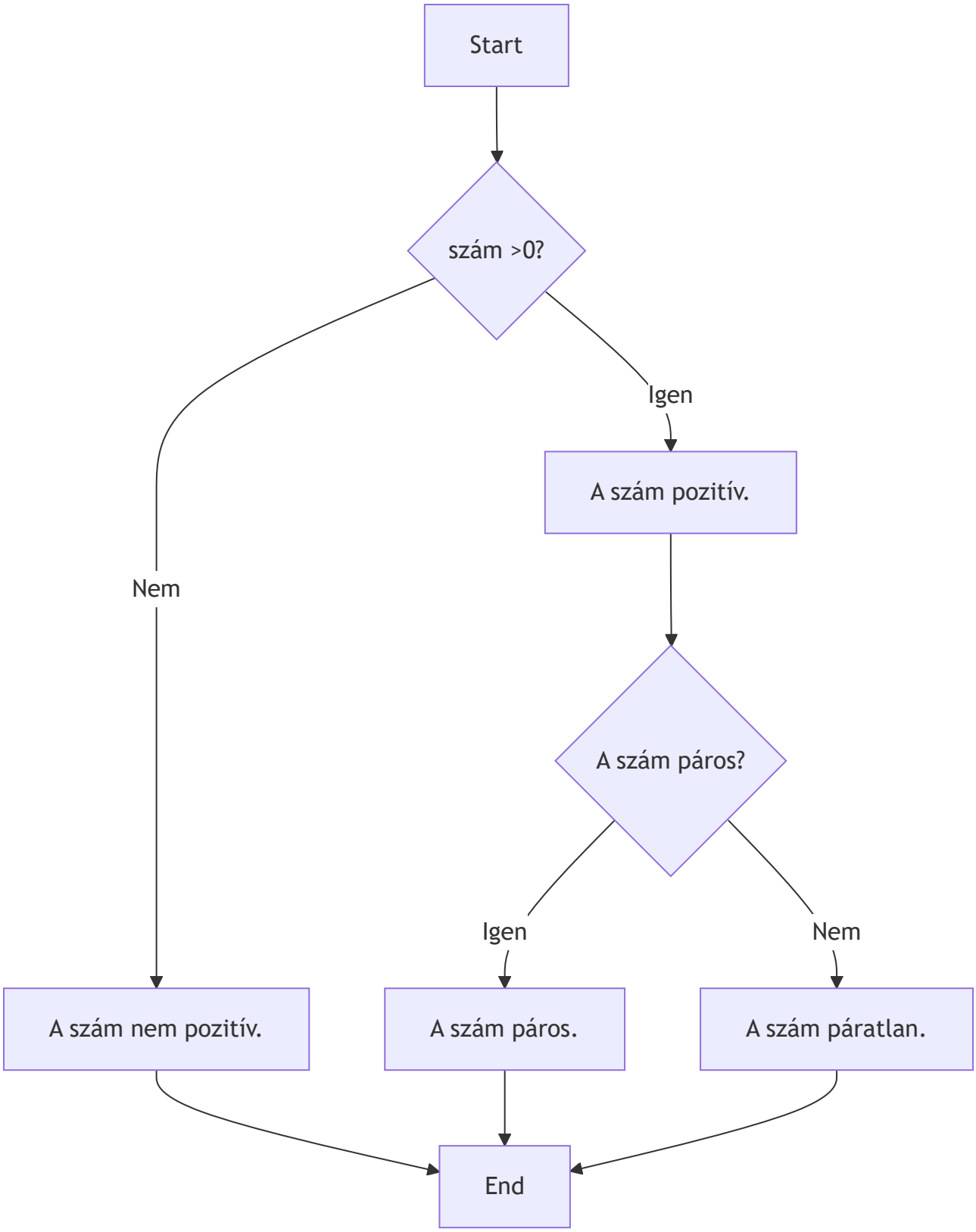
15. óra: Beágyazott feltételek

Elmélet:

- Beágyazott feltételek

Példa:

```
szam = 10
if szam > 0:
    print("A szám pozitív.")
    if szam % 2 == 0:
        print("A szám páros.")
    else:
        print("A szám páratlan.")
else:
    print("A szám nem pozitív.")
```



Önálló gyakorlati feladatok:

1. Írj egy programot, amely bekér egy teszt pontszámot 0 és 100 között, és az alábbi szabályok alapján értékeli:
   - Ha a pontszám nem 0 és 100 között van a program leáll, egyébként:
     - Ha a pontszám 80 fölött van, az értékelés "Kitűnő".

    - Ha a pontszám 60 és 79 között van, az értékelés "Jó".
    - Ha a pontszám 50 és 59 között van, az értékelés "Közepes".
    - Ha a pontszám 40 és 49 között van, az értékelés "Elégséges".
    - Ha a pontszám 40 alatt van, az értékelés "Elégtelen".
2. Írj egy programot, amely bekéri a felhasználó életkorát, és meghatározza a mozijegy árát:
    - Ha a felhasználó 12 évesnél fiatalabb, a jegy ára 1000 Ft.
    - Ha 12 és 18 év között van, kérdezze meg van-e diákkedvezménye.
        - Ha diákkedvezménye van: 1200 Ft
        - Ha nincs diákkedvezménye: 1500 Ft
    - Ha 18 év felett van, a jegy ára 2000 Ft.

Megoldási javaslat a 2. feladathoz:

```
eletkor = int(input("Hány éves vagy? "))

if eletkor < 12:
    print("A jegy ára: 1000 Ft")
elif eletkor <= 18:
    diakkedvezmeny = input("Van diákkedvezményed? (igen/nem): ").lower()
    if diakkedvezmeny == "igen":
        print("A jegy ára: 1200 Ft")
    else:
        print("A jegy ára: 1500 Ft")
else:
    print("A jegy ára: 2000 Ft")
```

16. óra: For ciklus I.

Elméleti rész:

- A for ciklus bemutatása
- range()

Bemutató feladatok:

```python
# Számok kiírása 0-tól 5-ig
for i in range(6):
    print(i)

# Számok kiírása 1-tól 5-ig
for i in range(1, 6):
    print(i)


# Minden második szám kiírása 2 és 10 között
for i in range(2, 11, 2):
    print(i)


# Számok kiírása 10-től 1-ig visszafelé
for i in range(10, 0, -1):
    print(i)
```

Önálló gyakorlati feladatok:

1. Írj egy programot, amely kiírja az 10-től 20-ig terjedő számokat.
2. Írj egy programot, amely kiírja a 20-tól 10-ig terjedő számokat.
3. Módosítsd úgy, hogy kiírja a 20-tól 10-ig terjedő számokat visszafelé, de csak minden másodikat.
4. Írj egy programot, amely 5-ször kiír egy szöveget (például "Gyakorlás").
5. Írj egy programot, amely az 1 és 10 közötti számok 4-es szorzótábláját írja ki. Például így:

```
4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36
4 * 10 = 40
```

Megoldási javaslat

```python
for i in range(1, 11):
    print(f"4 * {i} = {4 * i}")
```

17. óra: For ciklus II.

Elméleti rész:

- Beágyazott for ciklusok

Bemutató feladatok:

```
import time

for i in range(1, 4):
    for j in range(1, 6):
        print(f"Külső ciklus (i): {i}, Belső ciklus (j): {j}")
        time.sleep(2)


# Írjunk ki 3 sort, mindegyik elejére kerüljön a sor száma és egy kettőspont.
# Ezután jelenjen meg egy számsorozat 1-től 5-ig (ciklussal).
# Elvárt eredmény:
# 1. sor: 12345
# 2. sor: 12345
# 3. sor: 12345

for y in range(1, 4):  # Külső ciklus
    print(f"{y}. sor: ", end="")
    for x in range(1, 6):  # Belső ciklus
        print(x, end="")
    print()


# Teljes szorzótábla kiírása 2-től 9-ig
for i in range(2, 10):
    for j in range(1, 10):
        print(f"{i} * {j} = {i * j}")
    print()
```

Önálló gyakorlati feladatok:

1. Írj egy programot, amely két dobókocka összes lehetséges variációját megjeleníti (pl. "1,1", "1,2", stb.).
2. Írj egy programot, amely beágyazott ciklus segítségével 5 sorban 5 csillagot ír ki, így egy 5x5-ös mátrixot hoz létre.

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

3. Módosítsd úgy, hogy a mátrix közepén lévő * helyett egy szóköz legyen. Azaz üres legyen a mátrix közepe

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

4. Írj egy programot, amely egy beágyazott ciklus segítségével az alábbi ábrát rajzolja ki:

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

18. óra: For ciklus gyakorlása

Önálló gyakorlati feladatok:

1. Írj egy programot, amely 1-től 100-ig összeadja a számokat, de ha egy szám osztható 5-tel, azt kihagyja az összeadásból.
2. Írj egy programot, amely beágyazott ciklus segítségével kiírja az alábbi számmátrixot:

```
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
```

3. Írj egy programot, amely beágyazott ciklus segítségével kirajzol egy számháromszöget. A számok növekvő sorrendben jelennek meg minden sorban:

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

4. Írj egy programot, amely létrehoz egy dombormintát. A csillagok kiírását belső ciklussal oldd meg:

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

5. Írj egy programot, amely 1-től 50-ig iterál, és:
    - Ha a szám 3-mal osztható kiírja, hogy "3-mal osztható".
    - Ha a szám 5-tel osztható kiírja, hogy "5-tel osztható".
    - Ha 3-mal és 5-tel is osztható kiírja, hogy "mindkettővel osztható".

19. óra: While ciklus I.

Elméleti rész:

- While ciklus: addig ismétli a benne lévő utasításokat, amíg a feltétel igaz. A feltétel minden ciklus elején kerül ellenőrzésre. Ha a feltétel hamis lesz, a ciklus befejeződik.
- A while ciklust általában akkor használjuk, ha nem tudjuk előre, hogy pontosan hányszor kell végrehajtani (ellentétben a for ciklussal).
- Végtelen ciklus: ha a feltétel soha nem válik hamissá. Kerüljük!

Bemutató feladatok:

```python
# Számok kiírása 1-től 5-ig:
szam = 1
while szam <= 5:
    print(szam)
    szam += 1
```

```
# Minden második szám kiírása 1-től 10-ig
szam = 1
while szam <= 10:
    print(szam)
    szam += 2


# Visszaszámlálás 10-től 1-ig:
szam = 10
while szam > 0:
    print(szam)
    szam -= 1


# Ez a ciklus hibás, mert végtelen.
# Figyeljünk, hogy ne okozzunk hasonlót!
szam = 1
while szam > 0:
    print(szam)
```

Önálló gyakorlati feladatok:

1. Írj egy programot, amely while ciklus segítségével 1-től 10-ig összeadja a számokat és kiírja az eredményt.
2. Írj egy programot, amely while ciklussal kiírja az összes 5-tel osztható számot 1 és 50 között.
3. Írj egy programot, amely while ciklussal 5-től visszafelé kiírja a számokat egészen nulláig, de mindegyik után megkérdezi, hogy folytassuk-e? Ha a válasz "i" akkor folytatjuk, különben a ciklus leáll.
4. Írj egy programot, amely addig kér be számokat, amíg a felhasználó nem ad meg egy negatív számot. Ekkor a program befejezi a futását.
5. Írj egy programot, amely egy adott szám szorzótábláját írja ki 1-től 10-ig, de while ciklus segítségével.

20. óra: While ciklus II.

Elméleti rész:

- break: azonnal kilépteti a ciklust
- continue: azonnal a következő iterációra lépteti a ciklust

Bemutató feladatok:

```python
# Kilépés a ciklusból 50-nél
szam = 1
while szam <= 100:
    print(szam)
    if szam == 50:
        break
    szam += 1


# Számbekérés, mely negatív inputra leáll
szam = 0
while szam >= 0:
    szam = int(input("Adj meg egy számot: "))
    if szam < 0:
        break


# Az alábbi kód átugorja a páros számokat
szam = 1
while szam <= 10:
    if szam % 2 == 0:
        szam += 1
        continue
    print(szam)
    szam += 1
```

Önálló gyakorlati feladatok:

1. Írj egy programot, amely addig kér be számokat, amíg a bevitt számok összege 10 alatt van (break használatával).
2. Módosítsd úgy, hogy a 0 megadása esetén is álljon le.
3. Írj egy programot, amely 1-től 50-ig kiírja a számokat, de minden harmadik számot kihagy (continue használatával).
4. Írj egy programot, amely 1-től 100-ig iterál, és az alábbi feltételek alapján különböző üzeneteket ír ki:
    - Ha a szám osztható 3-mal, írja ki: "3-mal osztható"
    - Ha a szám osztható 5-tel, írja ki: "5-tel osztható"
    - Ha mindkettővel osztható, írja ki: "mindkettővel osztható", majd szakítsa meg a ciklust (break-kel).
5. Írj egy programot, amely 1-től 20-ig számol. Ha a szám 4-gyel osztható, a program átugorja az iterációt (continue-val), ha pedig a szám 15-tel osztható, a program megszakítja a ciklust (break-

kel).