Ciklus

2025

1 Feltételes ciklus (while-loop)

Ha egy utasítást vagy utasítás sorozatott többször végre kell hajtani, akkor sose másoljuk le az adott kódrészletet többször. Helyette használjunk ciklust.
A feltételes ciklus egy elől tesztelő ciklus, amely ha igaz feltételt tartalmaz, akkor lefuttatja a ciklusmagjában található utasításokat.

```
while <feltétel>:
    #ciklusmag
```

[ ]:
```python
# kiírjuk a számokat 1-től 10-ig
x = 1  # ciklusváltozó kezdőértéke
while x <= 10:  # ciklusfeltétel (ciklusváltozó vizsgálata)
    print(f"x értéke: {x}")  # ciklusmag
    x = x + 1  # ciklusváltozó léptetése

print("Már nem vagyok a while ciklusban")
```

A ciklusmag annyiszor fog lefutni, ameddig a ciklusfeltétel igaz értéket ad.
Ha a ciklusfeltétel az első vizsgálat során `False` értékű, akkor a ciklusmag egyszer sem fog lefutni.
Ha a ciklusfeltétel sohasem lesz hamis, akkor egy végtelen cikust készítettünk. A végtelen ciklust a `Ctrl + C` billentyű kombinácival vagy a `Stop` gombbal lehet megszakítani.

[ ]:
```python
while False:
    print("Soha sem fut le")

print("Most jön egy végtelen ciklus, ilyet NE csináljunk")

# példa végtelen ciklusra
while True:
    print("Soha sem áll le - végtelen ciklus")

print("Már nem vagyok a while ciklusban.")
```

1.1 Break utasítás

Ez egy speciális, ciklusok esetén használható utasítás, amely azonnal megszakítja a ciklus futását és a ciklus után következő utasítás végrehajtásával folytatja.
A `break` és a ciklus vége közötti utasítások már nem lesznek végrehajtva.

[ ]:
```
x = 1
while x <= 10:
    print(f"x értéke: {x}")
    # ha x értéke 4, akkor kilépünk a ciklusból
    if x == 4:
        break
    x = x + 1

print("Már nem vagyok a while ciklusban")
```

1.2 Continue utasítás

A `continue` használata esetén a ciklus futása nem szakad meg, de a ciklusmag maradék utasításai már nem hajtódnak végre, és újra a ciklusfeltétel kiértékelésére ugrik.
Figyeljünk arra, hogy a ciklusváltozó léptetése soha ne kerüljön átugrásra, mert akkor végtelen ciklust kapunk.

[ ]:
```
x = 0
while x <= 10:
    x = x + 1
    # ha x értéke 4, akkor visszaugrunk a ciklusfeltételhez
    if x == 4:
        continue
    print(f"x értéke: {x}")


print("Már nem vagyok a while ciklusban")
```

1.3 Else szerkezet

A Python `while` szerkezetnek van egy opcionális `else` ága. Az `else` ág csak akkor fut le, ha a `while` ciklus nem szakad meg `break` utasítással, hanem teljesen lefutott, azaz a ciklusfeltétel hamis lesz.

```
while <feltétel>:
    #ciklusmag
else:
    #normál leállláskor
```

[ ]:
```
x = 1
while x <= 10:
    print(f"x értéke: {x}")
    # ha megszakítjuk a ciklusmagot, akkor nem lép be az else ágba
    #if (x == 2): break
    x = x + 1
else:
    print("A ciklus lefutott, mert a feltétel hamis lett.")
```

```
print("Már nem vagyok a while ciklusban")
```

2 Véletlen számok készítése

A `random` modul segítségével lehet véletlenszámokat generálni. A használatához be kell importálnunk a program elején.
A program minden egyes futtatásakor más-más számokat fog adni. A `random` működéséhez a kezdő értéket az éppen aktuális idő értéke fogja adni.
Ha mi magunk szeretnénk ezt a kezdő értéket beállítani, akkor a `random.seed()` függvénynek kell azt megadni (ilyenkor mindig ugyanaz a számsor fog generálódni)
Lehetőség van egész (`randint`) és tört számok (`random`) véletlen generálására.

[ ]:
```
import random

# random.seed(42) # Saját kezdő értékkel történő inicializálás, nem szükséges

# Egész szám a megadott tartományban (1 és 100 között)
veletlenszam_egesz = random.randint(1, 100)
print(f"Egész véletlenszám: {veletlenszam_egesz}")

# Lebegőpontos szám 0 és 1 között
veletlenszam_lebegopontos = random.random()
print(f"Lebegőpontos véletlenszám: {veletlenszam_lebegopontos}")
```

Egész számot generálhatunk egy tartomány elemeinek definiálásával (start, stop, step) is a `randrange()` függvény segítségével.

Lebegőpontos számot elő lehet állítani egy tartomány megadásával úgy: - hogy a tartományból egyenletes eloszlás szerint veszi figyelembe az elemeket (`uniform`) - hogy a tartományban kijelölünk egy pontot, amelyik a legnagyobb valószínűséget adja (`triangular`)

Normál (Gauss) eloszlást is használhatunk egy véletlen szám generálásához (`gauss`)

[ ]:
```
import random

# Egész szám 10 és 100 közötti tartományon 5-ös lépésekkel, 100-már nem része a␣
 ↪tartománynak
veletlenszam_egesz = random.randrange(10, 100, 5)
print(f"Egész véletlenszám: {veletlenszam_egesz}")

# Lebegőpontos szám 3.0 és 5.0 között (egyenletes eloszlás)
veletlenszam_lebegopontos = random.uniform(3.0, 5.0)
print(f"Egyenletes eloszlás: {veletlenszam_lebegopontos}")

# Lebegőpontos szám 3.0 és 5.0 között (4.0-nél lesz a legnagyobb a␣
 ↪valószínűsége)
# A generált számok valószínűsége a háromszög alakú eloszlás szerint változik.
veletlenszam_lebegopontos = random.triangular(3.0, 5.0, 4.0)
```

```
print(f"Háromszög alapú eloszlás: {veletlenszam_lebegopontos}")

# Normál (Gauss) eloszlást használó véletlenszerű szám generálására
mu = 0  # Középérték
sigma = 1  # Szórás
veletlenszam_lebegopontos = random.gauss(mu, sigma)
print(f"Normál eloszlás: {veletlenszam_lebegopontos}")
```

Listák esetén lehetőség van véletlen elem kiválasztására, összekeverésére, kisorsolására is. (listák ismertetésénél kerül bemutatásra)

3 Gyakorló feladatok

1. Feladat

   Egy 18 méter mély kiszáradt kútba beleesett egy csiga. Megpróbál felmászni, egy nap alatt 5 métert tud megtenni, de éjjel mindig visszacsúszik 4 métert. Hány nap alatt ér a felszínre?

[ ]:
```
# 1. nap
nap = 1
magassag = 5

while magassag < 18:
    magassag -= 4  # éjszaka visszacsúszik
    print(f"{nap}. nap {magassag} méter")
    magassag += 5
    nap += 1

print(f"{nap}. nap {magassag} méter")
```

2. Feladat

   Adjon meg egy pozítiv egész számot (n) és adja meg az első n db szám összegét.

[ ]:
```
n = 10

osszeg = 0
szamlalo = 1

while szamlalo <= n:
    osszeg += szamlalo
    szamlalo += 1

# Eredmény kiírása
print(f"Az első {n} pozitív egész szám összege: {osszeg}")
```

3. Feladat

   Írja ki 1-100 tartományban azokat a számokat, amelyek 3-mal és 7-tel osztható számok.

[ ]:
```
szamlalo = 1

while szamlalo <= 100:
    if szamlalo % 3 == 0 and szamlalo % 7 == 0:
        print(szamlalo)
    szamlalo += 1
```

4. Feladat

   Írja ki a Fibonacci sorozat elemeit 100-ig.

[ ]:
```
a = 0
b = 1

while b < 100:
    print(b)
    a, b = b, a + b
```

5. Feladat

   Határozza meg a sin(x) függvény gyökét x = [3,4] tartományon intervallum felezés módszerével. leírás

[ ]:
```
import math

kezd = 3
veg = 4

kozep = (kezd + veg) / 2
lepes = 1

hiba = 1e-6

while math.fabs(math.sin(kozep)) > hiba and lepes < 100:
    print(f"{lepes}. {kozep} {math.sin(kozep)}")
    if math.sin(kozep) * math.sin(kezd) > 0:
        kezd = kozep
    else:
        veg = kozep
    lepes += 1
    kozep = (kezd + veg) / 2

print(f"{lepes}. {kozep} {math.sin(kozep)}")
```

6. Feladat

   Faktoriális számítása while ciklussal
   Meddig működik jól?

[4]:
```
szam = 5
factorial = 1
while szam > 1:
    print(szam, end="*")
    factorial *= szam
    szam -= 1

print("1 =", factorial)
```

```
5*4*3*2*1 = 120
```

7. Feladat

   Készítse el a következő számháromszöget (Használjon két egymásba ágyazott while ciklust):
   1
   22
   333
   4444
   55555

[ ]:
```
n = 1
while n <= 5:
    s = 1
    while s <= n:
        print(n, end="")
        s += 1
    print()
    n += 1
```

8. Feladat

   Készítse a 10x10-es szorzótáblát. (Használjon tabulátort a kiíratásnál)

[ ]:
```
a = 1
while a <= 10:
    b = 1
    while b <= 10:
        print(a * b, end="\t")
        b += 1
    print()
    a += 1
```

9. Feladat

   Készítsen egy 3x3-as (vagy nxm-es) egységmátrixot.

[ ]:
```
a = 1
while a <= 3:
    b = 1
    while b <= 3:
```

```
        if a == b:
            print(1, end="\t")
        else:
            print(0, end="\t")
        # print(1 if a==b else 0, end="\t")
        b += 1
    print()
    a += 1
```

10. Feladat

    Határozza meg egy számról, hogy prím-e. (A prímszámok azok az egynél nagyobb természetes számok, amiknek pontosan két osztójuk van.)

[ ]:
```
szam = 31

prim = True
oszto = 2
while oszto <= szam**0.5:
    if szam % oszto == 0:
        prim = False
        break
    oszto = oszto + 1

if prim and szam > 1:
    print(f"{szam} prímszám")
else:
    print(f"{szam} nem prímszám")
```

[ ]:
```
szam = 31

oszto = 2
while oszto <= szam**0.5:
    if szam % oszto == 0:
        print(f"{szam} nem prímszám")
        break
    oszto = oszto + 1
else:
    if szam > 1:
        print(f"{szam} prímszám")
    else:
        print(f"{szam} nem prímszám")
```

11. Feladat

    Írjon egy programot, ami a bankban elhelyezett 4,3 % os kamatozású tőke 20 év alatt felhalmozódott évi kamatait számolja ki.

[ ]:
```python
# Kezdeti tőke és kamatláb beállítása
kezdeti_toke = 15e6
kamatlab_szazalek = 6.5

# Évek számának beállítása
evek = 20

# Kamatok kiszámítása és kiírása évente
ev = 1
while ev <= evek:
    kamat = (kezdeti_toke * kamatlab_szazalek) / 100
    kezdeti_toke += kamat
    print(f"{ev}. év: {kamat:.0f} Ft kamat, Töke: {kezdeti_toke:.0f} Ft")
    ev += 1

# Végösszeg kiírása
print(f"\nVégösszeg {evek} év után: {kezdeti_toke:.0f} Ft")
```

12. Feladat

    Készítsen egy egyszerű számkitaláló játékot. A gép egy véletlenszerű számot választ 1 és 100 között. A játékosnak lehetősége van tippelni, és a program visszajelzi, hogy a gondolt szám nagyobb vagy kisebb-e. A játék addig folytatódik, amíg a játékos el nem találja a számot.

[ ]:
```python
import random

print("Gondoltam egy számra 1 és 100 között. Próbáld kitalálni!")

gondolt_szam = random.randint(1, 100)
probalkozasok = 0

while True:
    tipp = int(input("Tippelj egy számot: "))
    probalkozasok += 1

    if probalkozasok >= 10:
        print(f"Sajnálom {probalkozasok} próbálkozásból nem találtad ki.")
        break

    if tipp < gondolt_szam:
        print("A gondolt szám nagyobb.")
    elif tipp > gondolt_szam:
        print("A gondolt szám kisebb.")
    else:
        print(f"Gratulálok! Eltaláltad a számot {probalkozasok} próbálkozásból.
 ↪")
        break
```

13. Feladat

Készítse el a következő számháromszöget: (az üres helyekre tegyen space karaktert)

```
0123456789
 012345678
  01234567
   0123456
    012345
     01234
      0123
       012
        01
         0
```

[ ]:
```
a = 0
while a <= 9:
    b = 0
    while b < a:
        print(" ", end="")
        b += 1

    c = 0
    while c <= 9 - a:
        print(c, end="")
        c += 1

    print()
    a += 1
```

14. Feladat

Adja meg egy derékszögű háromszög egyik befogóját és az átfogót. Határozza meg a harmadik oldalhosszt.
Ellenőrizze le, hogy jó adatokat adott-e meg a felhasználó. Ha nem, akkor kérje be újra.

[ ]:
```
import math

a = float(input("Kérem, adja meg az egyik befogót: "))
c = float(input("Kérem, adja meg az átfogót: "))
while c < a or a <= 0 or c <= 0:
    print("Nem megfejelelő adatok")
    a = float(input("Kérem, adja meg az egyik befogót: "))
    c = float(input("Kérem, adja meg az átfogót: "))
b = math.sqrt(c**2 - a**2)
print(f"A derékszögű háromszög harmadik oldala: {b:.2f}")
```

15. Feladat

Numerikusan integrálja a sin(x) függvényt [0,pi/2] tartományon. leírás

[ ]:
```python
import math

kezd = 0
veg = math.pi / 2

dx = 0.01
integral = 0

while kezd < veg:
    integral += (math.sin(kezd) + math.sin(kezd + dx)) / 2 * dx
    kezd += dx

print(f"Integrál értéke: {integral}")
```

16. Feladat

    Mennyi a kétjegyű páros számok összege?

[ ]:
```python
osszeg = 0
szam = 10

while szam < 100:
    osszeg += szam
    szam += 2

print(f"A kétjegyű páros számok összege: {osszeg}")
```