3. Előadás

Python kurzus

Tárgyfelelős:
Dr. Tejfel Máté

Előadó:
Dr. Király Roland

3. Előadás tematikája

Feltételes utasítások és ciklusok

1. UV
2. Pytest
3. Logikai és bitenkénti műveletek
4. If - else és elif használata
5. While és for ciklusok
6. Függvények bevezetése, paraméterek, visszatérési értékek

1. UV és pytest

- UV letöltése
- UV init
- UV env futtatása és használata
- Pytest
- Pytest és UV – alkalmazásfejlesztés
- VS Code

2. Logikai és bitenkénti műveletek

- Logikai értékek: **True, False**
- Logikai műveletek: **not, and, or, ^ (xor)**
- Logikai kifejezések:

  Példák: not (A and B) = (not A) or (not B)
  not (A or B) = (not A) and (not B)

- Bitenkénti műveletek (~, &, |, ^) ➔

a	b	~b	a&b	a|b	a^b
1	1	0	1	1	0
1	0	1	0	1	1
0	1	0	0	1	1
0	0	1	0	0	0

- Bináris balra és jobbra eltolás (<<, >>)

```
szám = 15
sz_jobbra = szám >> 1          # Osztunk 2-vel ( // )
sz_balra = szám << 2           # Szorzunk 4-gyel
print(szám, sz_jobbra, sz_balra)
```

```
15 7 60
```

- Logikai bit műveletek:

Példa:

```
x =14    00000000000000000000000000001110
y =23    00000000000000000000000000010111
x & y    00000000000000000000000000000110
```

- Shortcut:

```
x = x & y      x &= y
x = x | y      x |= y
x = x ^ y      x ^= y
```

- Műveleti azonosságok, ha b egy bit:

```
b = ~(~b)
b & 1 = b      b & 0 =0
b | 1 =1       b | 0 = b
b ^ 1 = ~b     b ^ 0 = b
```

- Példa: egyes bitek lekérdezése &-del:

```
If flag_register & the_mask:
    # A vizsgált bit 1 (set)
else:
    # A vizsgált bit 0 (reset)
```

```
flag_register = 45   #0b00101101
the_mask = 4         #0b00000100
if flag_register & the_mask:
   print('A vizsgált bit 1')
else:
   print('A vizsgált bit 0')
```

A vizsgált bit 1

3. Az if - else és az elif használata

Feltételes végrehajtás: a logikai kifejezés kiértékelése szerint

- Az if - else utasítás

```
if feltétel:
    utasítások, ha igaz a feltétel
else:
    utasítások, ha hamis a feltétel
```

```
if kánikula_van:
    menj_strandra()
else:
    menj_sétálni()
```

- Az elif utasítás

```
if jó_az_idő:
    menj_sétálni()
elif van_jegy:
    menj_moziba()
elif van_szabad_asztal:
    menj_étterembe()
else:
    tévézz_otthon()
```

- Beágyazott if utasítás

```
if jó_az_idő:
    if van_jó_étterem:
        menj_ebédelni()
    else:
        menj_sétálni()
else:
    if van_jegy:
        menj_moziba()
    else:
        tévézz_otthon()
```

4. A while és a for ciklus

- A while ciklus:

```
while feltétel:
    utasítás_1
    utasítás_2
    .
    .
    utasítás_n
```

```
i = 0                                   #kezdőérték megadása
while i < 5:                            #ciklusfeltétel megadása
    print(i)                            #ciklusmag
    i += 1                                  ##léptetés
print("léptetés után", i)
```

```
0
1
2
3
4
léptetés után 5
```

- Végtelen ciklus:

```
while True:
    print(„Benn ragadtam egy ciklusban.")
```

Interrupt a billentyűzetről: Ctrl + c

- A for ciklus:

```
for i in range(10):
    print(„Az i aktuális értéke", i)
```

```
Az i aktuális értéke 0
Az i aktuális értéke 1
Az i aktuális értéke 2
Az i aktuális értéke 3
Az i aktuális értéke 4
Az i aktuális értéke 5
Az i aktuális értéke 6
Az i aktuális értéke 7
Az i aktuális értéke 8
Az i aktuális értéke 9
```

```
for i in range(2,10,3):
    print(i)
```

```
2
5
8
```

```
for i in range(10,2,-3):
    print(i)
```

```
10
7
4
```

```
for i in range(5):                  #ciklusfeltétel megadása
    print(i, end='')                #ciklusmag
    print(i*'#')
print("léptetés után", i)
```

```
0
1#
2##
3###
4####
léptetés után 4
```

- A break és continue utasítások:

break:
azonnal kilép a ciklusból és a ciklus után a következő utasításra ugrik
continue:
a ciklusmag végére ugrik és megkezdődik a következő forduló, a feltétel kiértékelése

```
#break példa
print("break alkalmazása:")
for i in range(1, 6):
    if i == 3:
        break
    print("A ciklusban:", i)
print("A cikluson kívül.")

#continue példa
print("continue alkalmazása:")
for i in range(1, 6):
    if i == 3:
        continue
    print("A ciklusban:", ", i)
print("A cikluson kívül.")
```

```
break alkalmazása:
A ciklusban: 1
A ciklusban: 2
A cikluson kívül.
continue alkalmazása:
A ciklusban: 1
A ciklusban: 2
A ciklusban: 4
A ciklusban: 5
A cikluson kívül.
```

A **while** / **for** ciklusok és az **else** ág:

```
i = 1
while i < 5:
    print(i)
    i += 1
else:
    print("else:", i)
    print(i*'# ')
```

```
1
2
3
4
else: 5
# # # # #
```

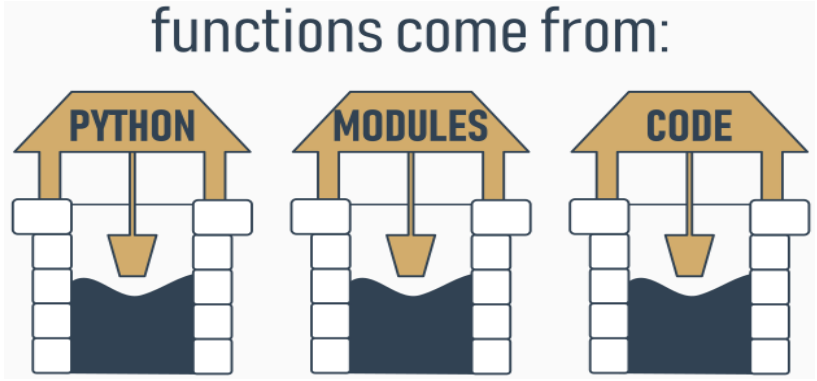
```
for i in range(5):
    print(i)
else:
    print("else:",i)

i = 100
for i in range(4, 0):
    print(i)
else:
    print("else:", i)
```

```
0
1
2
3
4
else: 4
else: 100
```

5. Függvények bevezetése, paraméterek, visszatérési értékek

- Miért van szükségünk függvényekre?
  - Pl. ha egy adott kódrészlet sokszor ismétlődik a programban
  - Részfeladatokra bontás - decomposition
- Honnan jönnek a függvények?
- Szintaxis:

```
def függvény_neve():
    #függvény_utasításai
```

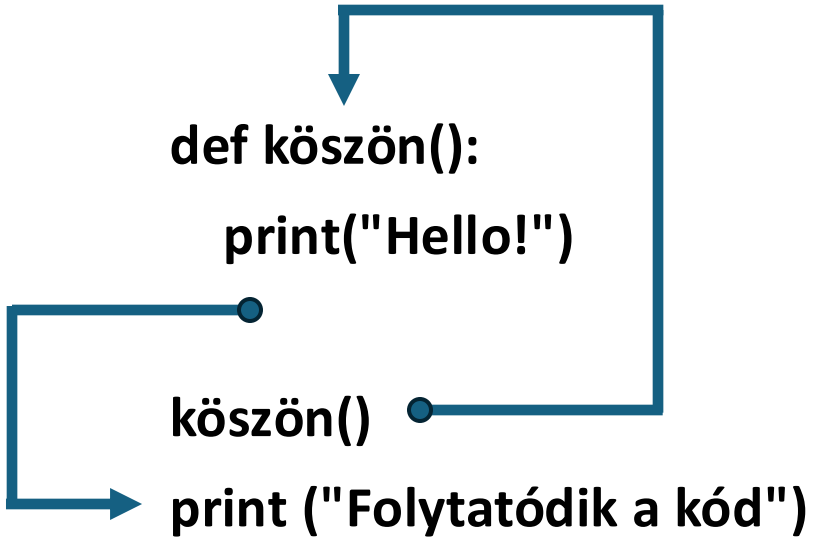


- Az első függvény:

```
def köszön():
    print("Hello!")

köszön()
```

Hello!

- Hogyan működnek a függvények:



- Paraméterezett függvények:

```
def függvény(paraméter):
    ###
```

```
def kiír(szám):
    print("A kapott érték:", szám)
kiír(12)
```

- Pozicionális és paraméter neve szerinti paraméterátadás:

```
def összead(a, b, c):
    print(a + b + c)
összead(1, 2, 3)
összead(1, 2, c =3)
összead(c = 3, a = 1, b =2)
```

```
összead(3, a = 1, b =2)
```

```
TypeError: összead() got multiple values for argument 'a'
```

Példák paraméter átadásra:

```
def bemutatkozás(vezeték_név, kereszt_név):
    print("Jó napot, a nevem: ", vezeték_név, kereszt_név)

bemutatkozás(vezeték_név = "Nagy", kereszt_név = "Virág")
bemutatkozás(kereszt_név = "Matyi", vezeték_név = "Ludas")
```

```
def bemutatkozás(vezeték_név, kereszt_név = "Lala"):
    print("Jó napot, a nevem: ", vezeték_név, kereszt_név)

bemutatkozás("Tündér", "Ilona")
bemutatkozás("Tündér")
```

```
bemutatkozás(kereszt_név = "Ilona")
```

TypeError: bemutatkozás() missing 1 required positional argument: 'vezeték_név'

Hatások és eredmények : a return utasítás

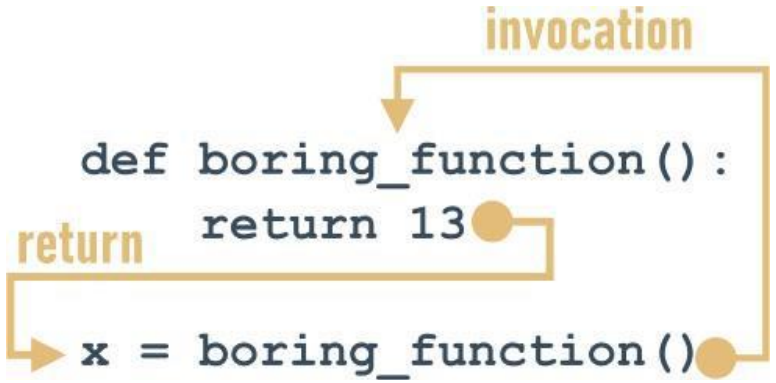
Visszatérés érték nélkül

```
def visszaszámlál(ez = True):
    print("Három...")
    print("Kettő..")
    print("Egy...")
    if not ez:
        return
    print("Indulás!")
visszaszámlál()
visszaszámlál(False)
```

Három...
Kettő...
Egy...
Indulás!

Három...
Kettő...
Egy...

Visszatérés értékkel




```
def boring_function():
    print("'Boredom Mode' ON.")
    return 13
print("This lesson is interesting!")
boring_function()
print("This lesson is boring ...")
```

- Láthatóság:

```
def példa_függvény():
    print("Ismered ezt a változót?", x)
x = 5
példa_függvény()
print(x)

def másik_függvény():
    z = 10
másik_függvény()
print("Ismered ezt a változót?", z)
```

```
Ismered ezt a változót? 5
5
NameError: name 'z' is not defined
```

```
def páros(n):
    if n % 2 == 0:
        return True
print(páros(4))
print(páros(3))
```

```
True
None
```

- A None kulcsszó:

```
érték = None
if érték is None:
    print("Sajnálom, de nem adtál meg értéket.")

print(None+1)
```

```
Sajnálom, de nem adtál meg értéket.
TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

- Listák ciklussal:

```
def lista(n):
    li = []
    for i in range(0, n):
        li.append(i)
    return li
print(lista(6))
```

[0, 1, 2, 3, 4, 5]

```
def lista_új(n):
    li_új = []
    for i in range(0, n):
        li_új.insert(0,i)
    return li_új
print(lista_új(6))
```

[5, 4, 3, 2, 1, 0]

```
def fibonacci(n):
    a, b = 1, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a + b
fibonacci(10)
```

1 1 2 3 5 8

```
def fib2(n):
    a, b = 1, 1
    fibo = []
    for i in range(n):
        fibo.append(a)
        a, b = b, a + b
    return fibo
print(fib2(10))
```

[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]

Példa 1.

```
#1.példa: Kitalálós játék logaritmikus kereséssel
def gép_kitalálós_játék():
    a_h = 1
    f_h = 100
    próba = 0
    print(f"Gondolj egy számra {a_h} és {f_h} között. A gép megpróbálja kitalálni!")
    while True:
        tipp = (a_h + f_h) // 2
        próba += 1
        print(f"A gép tippje: {tipp}")
        válasz = input("A tipp alacsony (a), magas (m), vagy helyes (h)? ").lower()
        if válasz == "h":
            print(f"A gép eltalálta a számot {próba} próbálkozás után!")
            break
        elif válasz == "a":
            a_h = tipp + 1
        elif válasz == "m":
            f_h = tipp - 1
        else:
            print("Érvénytelen válasz. Kérlek írd be, hogy 'a', 'm' vagy 'h'.")
gép_kitalálós_játék()
```

Példa 2.

```
# 2. példa: Szorzótábla kiírása, egymásba ágyazott ciklusokkal
def szorzótábla_kiírása(n):
    for i in range(1, n + 1):        # sorok
        for j in range(1, n + 1):    # oszlopok
            print(f"{i * j:3}", end=" ")    # Szorzat formázva
        print()  # Új sor

n = int(input("Add meg a szorzótábla méretét: "))
szorzótábla_kiírása(n)
```

Példa 3.

```
# 3. példa: Négyzetszámok szemléltetése: *-okkal kirajzolt négyzet
import math
def négyzet(szám):
    méret = int(math.sqrt(szám))
    if méret * méret != szám:
        print("A megadott szám nem négyzetszám.")
        return
    for i in range(méret):
        print("*  " * méret)

négyzetszám = int(input("Add meg a négyzetszámot: "))
négyzet(négyzetszám)
```

Példa 4.

```
#4.példa: A turtle modullal csigavonal rajzolása
import turtle
def csigavonal(t, hossz, lépés):
   for i in range(lépés):
      t.forward(hossz)   # szakasz
      t.right(90)       # jobbra 90 fok
      hossz += 10       # a következő szakasz hossza
# Turtle beállítások
ablak = turtle.Screen()    # Ablak létrehozása
ablak.bgcolor("white")   # Háttérszín beállítása
csiga = turtle.Turtle()      # Turtle létrehozása
csiga.speed(4)     # Rajzolás sebessége

csigavonal(csiga, 10, 40)      # Kezdő hossz, lépések száma
ablak.mainloop()      # Program befejezése
```

Összegzés

- Végeztünk logikai és bitenkénti műveleteket
- Bemutattuk az if - else és elif használatát
- Készítettünk while és for ciklusokat
- Megbeszéltük a függvény, a paraméterek és a visszatérési értékek fogalmát
- Bemutattunk néhány egyszerű példa programot

Konzultáció

Minden héten csütörtökön 18:00 – 20:00

Köszönöm a figyelmet!