

Diszkrét matematika

2. előadás

MÁRTON Gyöngyvér
<https://ms.sapientia.ro/~mgyongyi/>
mgyongyi@ms.sapientia.ro

Sapientia EMTE,
Matematika-Informatika Tanszék
Marosvásárhely, Románia

2024, őszi félév



Miről volt szó az elmúlt előadáson?

- áttekintő
- követelmények, osztályozás, könyvészet
- Python:
 - telepítés, használat,
 - alapfogalmak: változók, operátorok, osztályok, típusok, függvények, metódusok,
 - típusok közötti átalakítások (`int()`, `float()`, `str()`, `list()`),
 - a programírás lépései.

Miről lesz szó?

- Python:
 - operátorok,
 - vezérlő szerkezetek (`if`, `for`, `while`)
 - a `range` függvény, az `in` operátor,
 - logikai kifejezések
- Algoritmusok:
 - a faktoriális függvény - iteratív, rekurzív változatok,
 - tesztelés: egy szám négyzetszám-e,
 - osztók száma,
 - nullások száma a faktoriális végén,

Python vezérlő szerkezetek

Az **if** elágazásutasítás

- definiálása:

```
if <kif> : <elágazástörzs>
(elif < kif > : <elágazástörzs>)
[else : <elágazástörzs>]
```

- jelentése: ha igaz a megfelelő kif kifejezés, akkor a megfelelő elágazástörzs hajtódik végre. Az elif, else ágak nem kötelezőek.
- mentjük el eload1.py néven azt az állományt, amelybe a következő kódsorokat tesszük:

```
x = 10
y = 13
if x > y: print (x, ' nagyobb, mint ', y)
else: print (y, ' nagyobb vagy egyenlo, mint ', x)
```

- **sikeres fordítás majd futtatás** után az eredmény:
13 nagyobb vagy egyenlo, mint 10
- megfelelő tördeléssel kell jelezni, hogy melyek azok az utasítások amelyek egy elágazástörzshöz tartoznak

Python programok futtatása

Windows parancssorból (Command Prompt, PowerShell) való futtatás:

- be kell állítani a PATH környezetváltozót (environment variable), úgy hogy megadjuk a python.exe állomány elérési útvonalát (System/Advanced system settings)
- hogy megtudjuk hol van a python.exe, a Python shellbe írjuk be a következőket:

```
>>> import os
>>> import sys
>>> os.path.dirname(sys.executable)
```
- a beállítás után az op rendszer fel fogja ismerni a python parancsot, ez később is hasznos lesz amikor Python csomagot akarunk telepíteni
- a parancssorban a CD rendszerparanccsal kiválasztjuk azt a mappát, ahol az eload1.py állomány van
- futtatás: ...> python eload1.py
- futtatás: kétszer klikkelve az eload1.py-on

Python, megjegyzések

Megjegyzések, "kommentek" használata:

- egysoros megjegyzés: `#ez egy egysoros megjegyzés`
- többsoros megjegyzés:

```
"""ez egy  
többsoros megjegyzés"""
```
- használatuk nagyon fontos, gyakori
- alkalmazásuk:
 - rövid leírás/útbaigazítás helyezhető el a programsorban,
 - ha nincs szükség vagy helytelen egy kódrészlet, akkor *ki lehet kommentálni*

Python, vezérlőszervezetek

a **range** függvény:

- egy számsorozatot generál, a megadott határértékkel/határértékekkel,
- a `list` átalakítja lista típusú értéké a generált számsorozatot
- a `tuple` átalakítja lista típusú értéké a generált számsorozatot

```
>>> range(0, 10)
range(0, 10)
```

```
>>> list(range(0, 10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
>>> tuple(range(15, 100, 10))
(15, 25, 35, 45, 55, 65, 75, 85, 95)
```

```
>>> list(range(5, -10, -2))
[5, 3, 1, -1, -3, -5, -7, -9]
```

Python, vezérlőszerkezetek

A **for** ciklusutasítás

```
for <elem> in <halmaz> :  
    <ciklustörzs>
```

- a **for** a halmaz minden elem elemére végrehajtja a ciklustörzs-ben megadott utasításokat.

Példák:

```
>>> for i in range(10):  
    print (i) #ugyeljunk a tordelesre!!  
    #ket ENTER-t kell nyomni
```


Python, lekérdezések

Írjuk be az eload1.py állományba a következő kódsorokat is majd futassuk az állományt, a korábban beírt kódsorokat kommenteljük ki. **Figyeljünk a tördelésre!**

```
for i in range(0, 10):  
    print (i, end = " ")  
print()  
  
for i in range(10):  
    print (i, end = "")  
print()  
  
for i in range(10):  
    print (2 ** i, end = " ")
```

Az állomány futtatásakor a megfelelő számsorozatok kerülnek kiírtásra:

```
0 1 2 3 4 5 6 7 8 9  
0123456789  
1 2 4 8 16 32 64 128 256 512
```

Python, vezérlőszervezetek

A **while** ciklusutasítás

- definiálása:

```
while <kifejezes>:  
    <ciklustörzs>
```

- a ciklustörzs addig kerül ismételtlen végrehajtásra, ameddig a kifejezes logikai értéke igaz
- bármely kifejezes logikai értéke igaznak minősül, ha az nullától különbözik
- ha a kifejezes értéke indulásból hamis, akkor egyszer sem hajtódik végre a ciklustörzs

1. feladat

Határozzuk meg $n!$ értékét, azaz az összes n -nél kisebb pozitív szám szorzatát.

A matematikai definíció szerint:

$$\begin{aligned}n! &= n \cdot (n-1) \cdot \dots \cdot 2 \cdot 1 \\ 0! &= 1\end{aligned}$$

```
n, res = 10, 1  
while n >= 1:  
    res = res * n  
    n = n - 1  
print(res)
```

Python függvények

Könyvtárfüggvénnyel:

```
>>> from math import factorial
>>> factorial(10)
3628800
```

Iteratív függvénnyel:

for ciklusutasítással:

```
def factorialFor(n):
    res = 1
    for i in range(1, n+1):
        res *= i
    return res
```

while ciklusutasítással, 2. módszer:

```
def factorialWhile(n):
    i, res = 1, 1
    while i <= n:
        res *= i
        i += 1
    return res
```

Python függvények

Rekurzív függvényekkel: A matematikai definíció szerint:

$$\begin{aligned}n! &= n \cdot (n-1)! \\ 0! &= 1\end{aligned}$$

```
def factorialRek1(n):  
    if n == 0: return 1  
    return n * factorialRek1(n-1)  
  
def factorialRek2(n):  
    return factorialAux(n, 1)  
  
def factorialAux(n, res):  
    if n == 0: return res  
    return factorialAux(n - 1, res * n)  
  
def factorialRek3(n, res = 1):  
    if n == 0: return res  
    return factorialRek3(n - 1, res * n)
```

Python függvények

- a `factorialRek1` a következő sorrendbe szorozza össze a számokat:
 - $1 \cdot 2, 1 \cdot 2 \cdot 3, 1 \cdot 2 \cdot 3 \cdot 4, \text{stb.}$,
 - akkor számol mikor jön vissza a rekurzióból
- a `factorialRek2` feltételezve, hogy $n = 10$, a következő sorrendbe szorozza össze a számokat:
 - $1 \cdot 10, 1 \cdot 10 \cdot 9, 1 \cdot 10 \cdot 9 \cdot 8, \text{stb.}$,
 - azaz akkor számol mikor megy be a rekurzióba,
 - a res inicializálását a `factorialAux` függvény meghívásával oldja meg
- a `factorialRek3`:
 - ugyanolyan sorrendbe szorozza össze a számokat, mint a `factorialRek2`,
 - nem használ segédfüggvényt, a res inicializálását a **a függvény fejlécének a megadásakor** végzi

Python, az `in` operátor

Az `in` operátor segítségével eldönthető egy adott értékről, hogy hozzátartozik egy adathalmazhoz vagy sem:

```
>>> 4 in range(10)
True
>>> "A" in "sapientia"
False
```

Ha azt szeretnénk megtudni, hogy nem tartozik hozzá egy adott érték az adathalmazhoz, akkor a `not in` operátort kell használni:

```
>>> -1 not in range(10)
True
>>> "int" not in ["float", "bool", "int", "str"]
False
```

Python logikai kifejezések

Egy logikai kifejezés logikai értéke **False**, ha:

- egyenlő a **False** vagy a **None** konstanssal,
- egyenlő üres **list**, **tuple**, vagy **str**-el,
- a kifejezés numerikus és értéke egyenlő **0**-val,
- minden más esetben **True** a kifejezés logikai értéke.

Logikai kifejezések **összehasonlító operátorok** segítségével adhatók meg:

==, **!=**, **>**, **>=**, **<**, **<=**, logikai értékük pedig a következő:

a == b, ha a **egyenlő** b-vel, akkor **True**

a != b, ha a **nem egyenlő** b-vel, akkor **True**

a > b, ha a **nagyobb**, mint b, akkor **True**

a >= b, ha a **nagyobb vagy egyenlő**, mint b, akkor **True**

a < b, ha a **kisebb**, mint b, akkor **True**

a <= b, ha a **kisebb vagy egyenlő**, mint b, akkor **True**

Python függvények

- a függvények a bemeneti értéken/értékalmazon végeznek műveleteket, adott esetben meghatározva egy kimeneti értéket/értékalmazt.
- Python függvénydefiniálás:

```
def <fvnév>( <paramlista> ) :  
    <fügtörzs>
```
- a függvtörzs-ben **return** utasítások adhatók meg, amelyek lehetővé teszik, hogy a függvényben kilépési pontokat adjunk meg, illetve a **return** alkalmazásával határozható meg a függvény kimeneti értéke,
- a **return** alkalmazása maga után vonja, hogy az utána következő műveletsorok ne kerüljenek elvégzésre,
- a függvtörzs-ben megadott **return** műveleteken keresztül **különböző típusú** kimeneti értékeket is meghatározhatunk.

Python függvények

2. feladat

Írjunk egy Python függvényt, amely megvizsgálja, hogy a függvény bemenete kisbetű-e vagy sem.

```
def tesztBetu1(x):  
    if 'a' <= x and x <= 'z':  
        print('kisbetu')  
    else: print('nem kisbetu')
```

```
>>> tesztBetu1('a')  
kisbetu
```

```
>>> tesztBetu2('/')  
False
```

```
def tesztBetu2(x):  
    if 'a' <= x <= 'z':  
        return True  
    return False
```

```
>>> 'a'.islower()  
True  
>>> 'ab'.islower()  
True  
>>> 'aB'.islower()  
False
```

Python függvények

3. feladat

Határozzuk meg hány nullás számjegy van $n!$ végén, anélkül, hogy meghatároznánk $n!$ értékét. Pl. hány nullás van $1000!$ végén.

- Megszámoljuk hogy az hány 5 -el, 5^2 -el, 5^3 -al, stb. osztható szám van n -ig.
- Ha $n = 91$, akkor összesen $18 + 3 = 21$ nullás számjegy van $91!$ végén, mert
 - 5 -el osztható számok száma: $91 // 5 = 18$
 - 5^2 -el osztható számok száma: $18 // 5 = 3$
 - 5^3 -al osztható számok száma: $3 // 5 = 0$

```
def factorialNullasok(n):  
    res = 0  
    while n > 0:  
        temp = n // 5  
        res += temp  
        n = temp  
    return res
```

```
>>> factorialNullasok(1000)  
249
```

```
def factorialNullasok_(n):  
    res = 0  
    while n > 0:  
        res += n // 5  
        n = n // 5  
    return res
```

```
>>> factorialNullasok_(1000)  
249
```

Python függvények

4. feladat

Teszteljük, hogy egy szám négyzetszám-e vagy sem.

```
def negyzetTeszt(x):  
    y = int(x ** 0.5)  
    if float(y * y) == x: return True  
    return False
```

5. feladat

Számoljuk meg, és írjuk ki a négyzetszámokat.

```
def negyzetTeszt2(x):  
    y = x ** 0.5  
    if y.is_integer(): return True  
    return False
```

Python függvények

6. feladat

Írjunk egy Python függvényt, amely meghatározza egy pozitív természetes szám osztóinak számát.

```
def osztokSzama(n):  
    if not isinstance(n, int):  
        return "helytelen bemenet"  
    if n <= 0:  
        return "helytelen bemenet"  
    db = 0  
    i = 2  
    while i <= n // 2:  
        if n % i == 0: db += 1  
        i += 1  
    return db
```