Függvények

teknőc parancsok ismétlése
függvények fogalma, használata
grafikon rajzoló program
Reversi játékprogram
függvények lokális változói
rekurzió és fraktál-szerű ábrák rajzolása

Emlékeztető

- többszörös elágazás (`if-elif-else`)
- kilépés `while` ciklusból (`break`), ciklus folytatása (`continue`), és a `while` ciklus feltételéhez tartozó `else` ág
- a `for` ciklus és a `range()` függvény
- teknőc grafika (turtle modul)

Turtle modul és függvényei

- teknőc parancsok elérése:

```
from turtle import *
```

- háttérszín, tollszín, és tollméret:

```
bgcolor(”...”), pencolor(”...”), pensize(...)
```

- toll letétele és toll felemelése:

```
pendown(), penup()
```

- mozgás előre, fordulás jobbra és balra:

```
forward(...), right(...), left(...)
```

Turtle modul és függvényei

- kör és pont rajzolás:

  `circle(...), dot(...)`

- terület beszínezése:

  `fillcolor("...")`

  `begin_fill(); ....; end_fill()`

- teknőc sebessége:

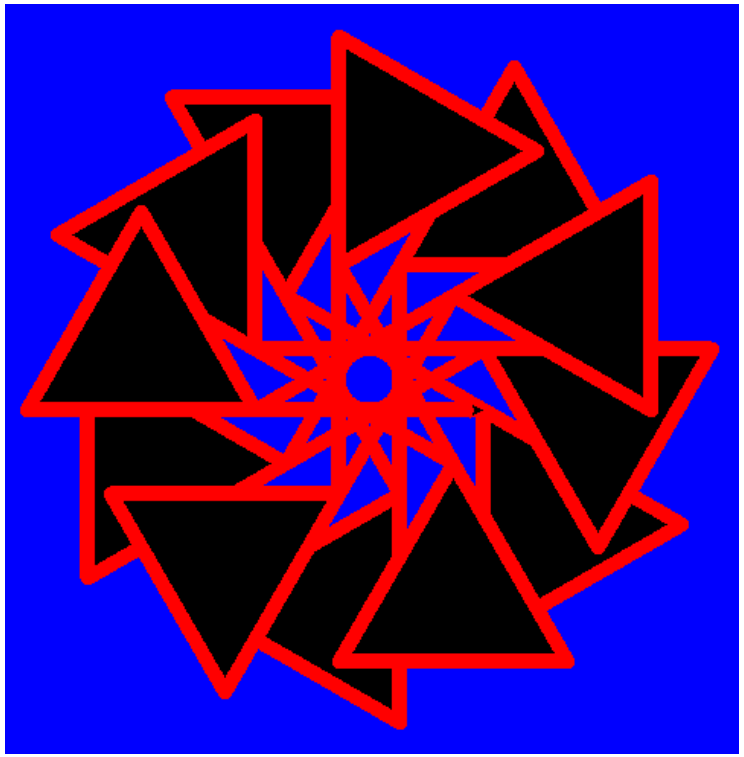
  `speed(...)` (`speed(0)` a leggyorsabb)

- rajzolás befejezése:

  `done()`

Példa

```
from turtle import *

bgcolor("blue")
pencolor("red")
pensize(10)
for i in range(12):
    begin_fill()
    right(90)
    forward(150)
    left(120)
    forward(150)
    left(120)
    forward(150)
    end_fill()
    forward(150)

done()
```



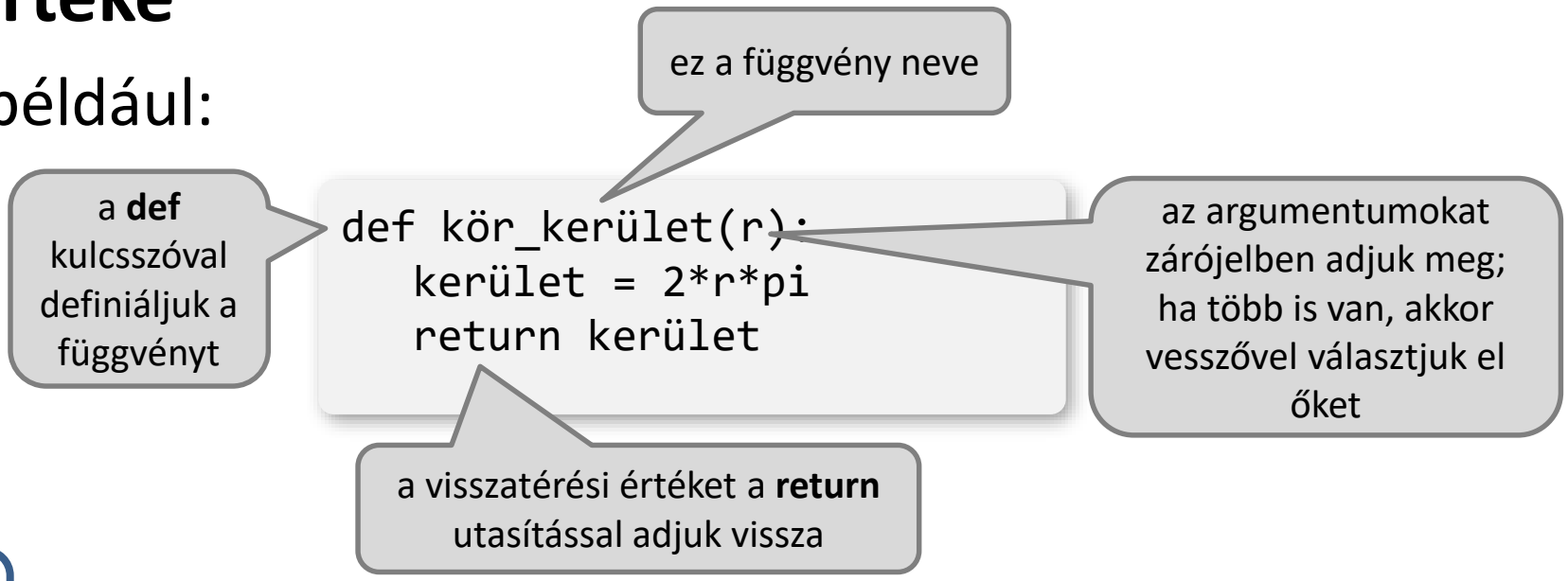
Függvények

- előfordul, hogy ugyanazt az utasítássorozatot többször is végre kell hajtani, de nem feltétlenül egymás után sokszor ismételve
  - pl. képernyő törlése a játék elején és mikor új játékot kezdünk
- az ilyen utasítássorozatot elnevezhetjük, és a nevének leírásával bármikor végrehajtathatjuk
- ezeket az elnevezett utasítássorozatokat **függvény**eknek hívjuk

Függvények

- egy függvénynek lehetnek **bemeneti paraméterei** (**argumentumai**) és **visszatérési értéke**

például:



Függvények

- Ügyelj a helyesírásra is!

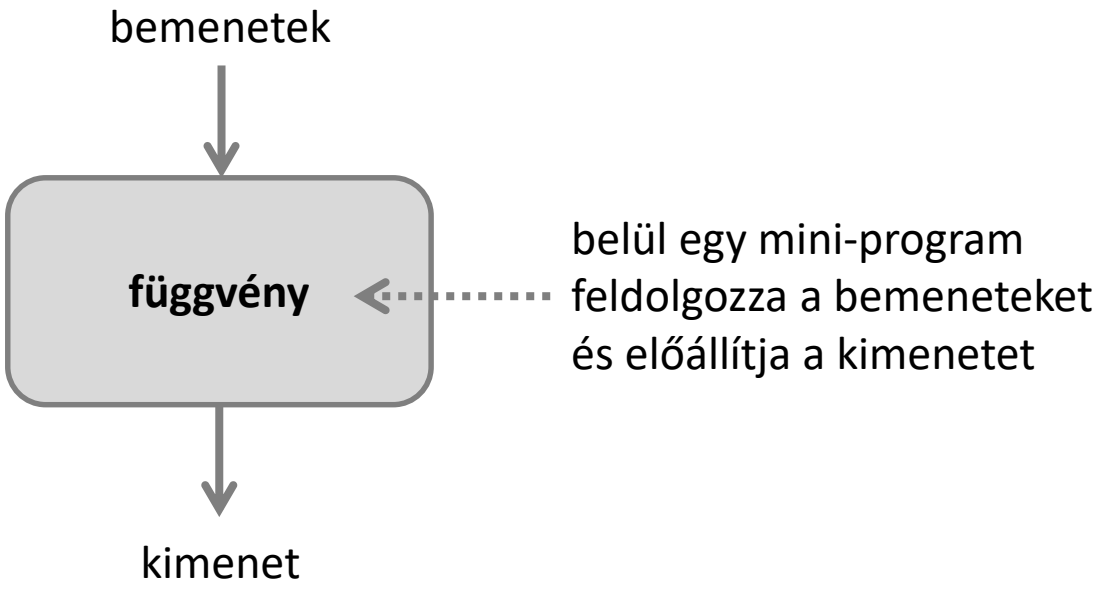
```
def kör_kerület(r):
    kerület = 2*r*pi
    return kerület
```

a függvény törzse beljebb kezdődik

Ne felejtsd el a kettőspontot!

Függvények

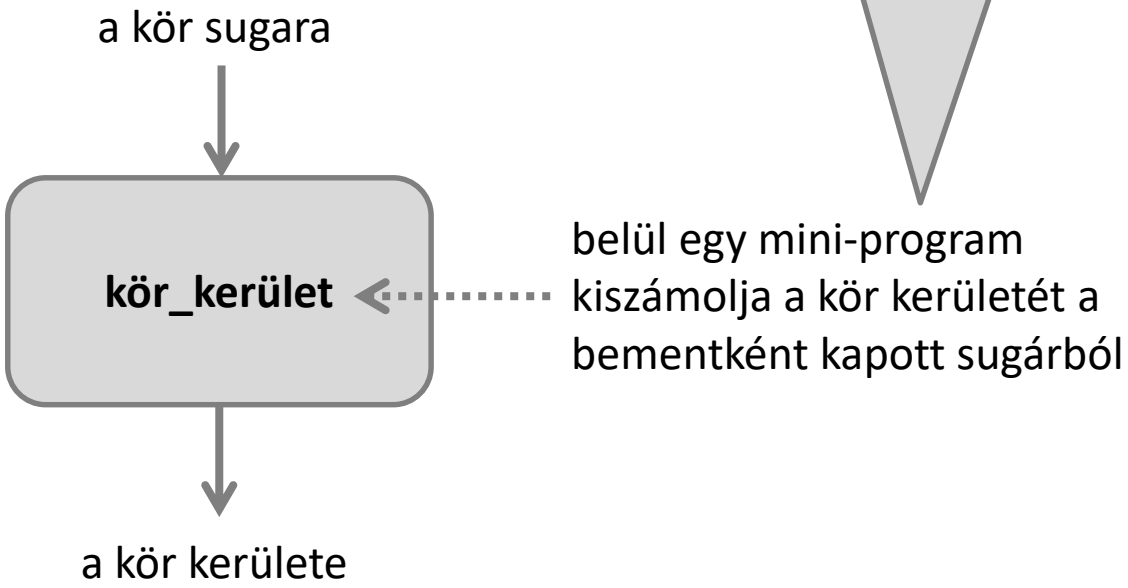
- egy függvényre gondolhatsz úgy, mint egy mini-programra:



Függvények

- például:

```
def kör_kerület(r):
    kerület = 2*r*pi
    return kerület
```



Függvények

- miután egy függvényt definiáltunk, használhatjuk (**meghívhatjuk**)
- egy függvényt a nevének leírásával és az argumentumok értékeinek megadásával lehet meghívni
- a visszatérési értékét elmenthetjük egy változóba, vagy átadhatjuk más függvényeknek, mint bemeneti paraméter

Függvények

- például:

```
from math import pi

def kör_kerület(r):
    kerület = 2*r*pi
    return kerület

k = kör_kerület(10)
print(k)

# print(kör_kerület(10))
```

így hívjuk meg a függvényt;
most 10-et adtunk meg bemeneti paraméterként, azaz egy 10 sugarú kör kerületét szeretnénk meghatározni

egy k változóban tároljuk a kiszámolt kerületet (azaz a visszatérési értéket)

így is csinálhattuk volna;
ekkor a visszatérési értéket egyből átadjuk a `print()` függvénynek bemenetként

Függvények

- függvényeket már eddig is használtunk pédául:
  - `print(”Helló!”)`
  - `str(16)`
  - `right(90)`
  - `penup()`
  - `...`
- ezeket mások definiálták
- mostmár saját függvényeket is tudunk csinálni

Gyakoroljunk!

Készíts függvényt az alábbi feladatok ellátására!

- kör sugarának kiszámítása
- téglalap kerületének kiszámítása
- téglalap átlójának kiszámítása
- egy szövegben található kis e és nagy E betűk megszámlálására

Írj programokat, melyekben meghívod a fenti függvényeket valamilyen bemeneti érték(ek)kel!

Grafikon rajzoló program

- grafikon rajzolásnál sokszor kell ismételni ugyanazokat a műveleteket:

  minden lehetséges x-re

  - ki kell számolni az ábrázolni kívánt matematikai függvény értékét
  - és ki kell rajzolni a képernyőre a kiszámolt értékhez tartozó pontot

Grafikon rajzoló program

Készíts egy parabola rajzoló programot!

A parabola képlete az $y = x^2$ egyenlet. Egy olyan programot szeretnénk készíteni, ami egy x_min és egy x_max érték között kinyomtatja a parabola pontjait a képernyőre. Ehhez az alábbi függvényekre lesz szükség:

- négyzetre emelés elvégzése
- koordináta rendszer kirajzolása
- pont kirajzolása tetszőleges helyre
- parabola kirajzolása adott x_min és x_max értékek közötti x értékekre

A programot elkezdtük megírni, de nem fejeztük be. A `parabola.py` fájlban találod ami eddig kész van. Segíts befejezni!

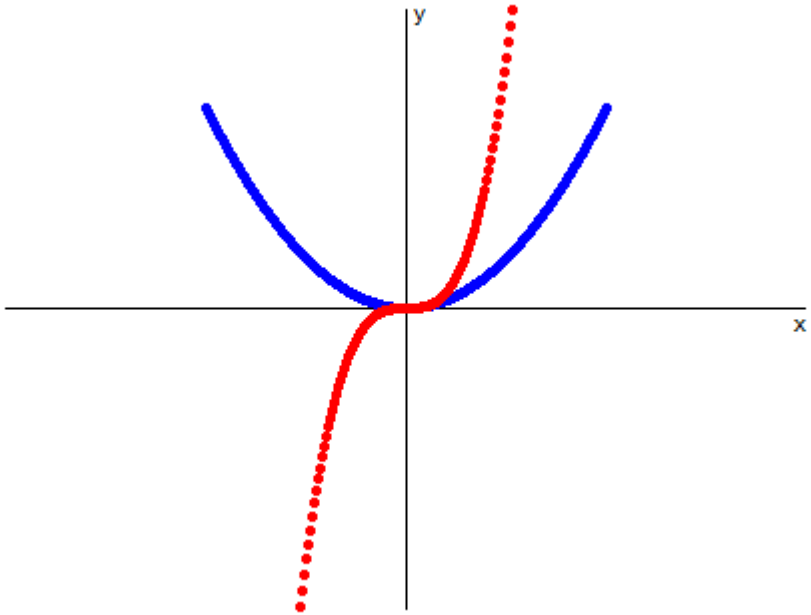
Grafikon rajzoló program (2. verzió)

Alakítsd át a parabola rajzoló programot úgy, hogy más matematikai függvényeket is ki tudjon rajzolni!

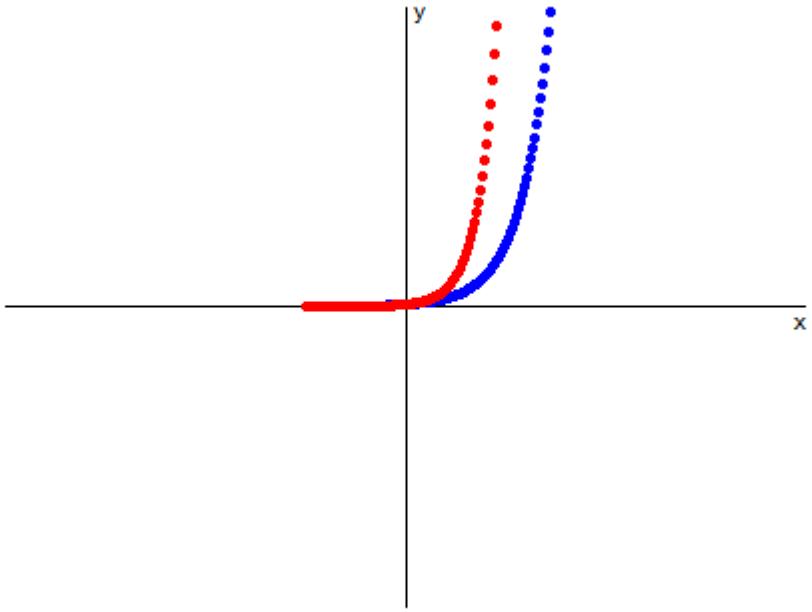
Elkezdtük megírni a programot, de nem fejeztük be. A `math_functions.py` fájlban találod, ami eddig elkészült. Segíts befejezni!

Megjegyzés: Figyeld meg, hogy a `görbe()` függvény egyik bemeneti paramétere egy függvénynév! Így egyetlen görbe rajzoló függvénnyel több matematikai függvényt is ki tudunk rajzolni, csak a megfelelő nevet kell átadni paraméterként.

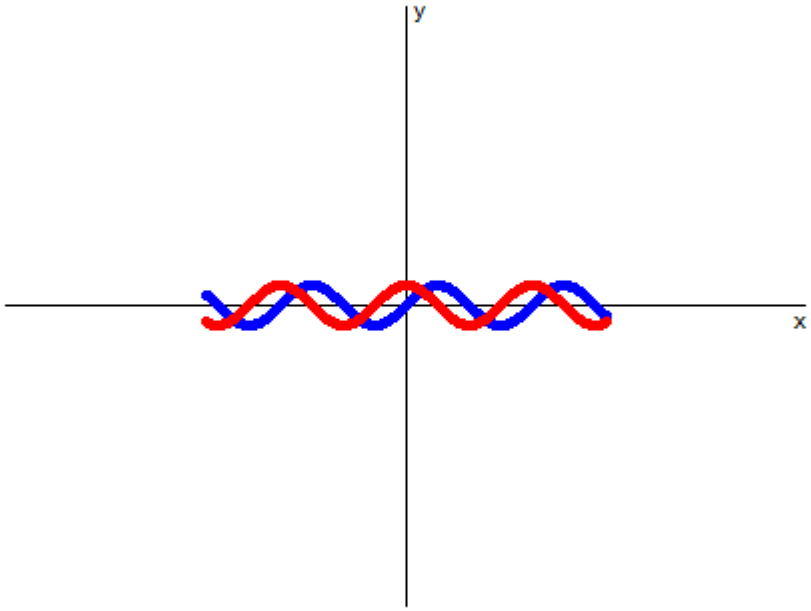
A grafikon rajzoló program rajzai



A grafikon rajzoló program rajzai



A grafikon rajzoló program rajzai

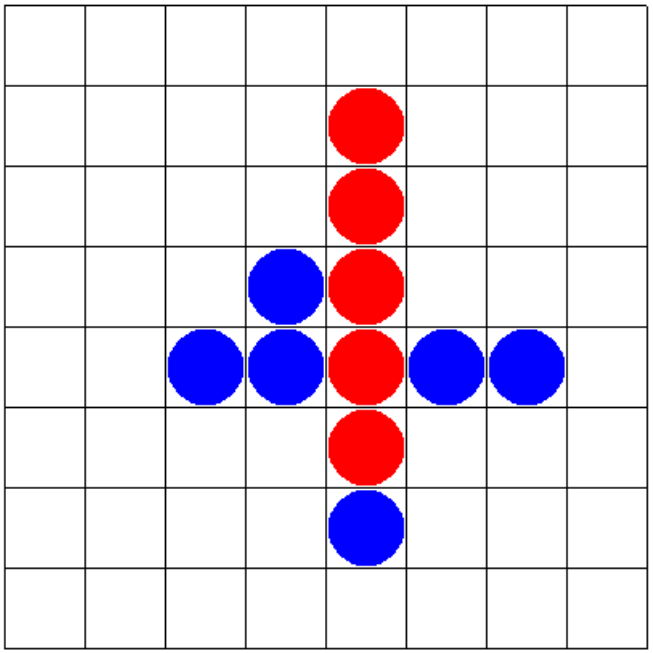


Reversi játék

Készíts egy Reversi játékprogramot!

A Reversi egy izgalmas táblajáték. A leírását itt találod meg a neten:
https://meszaros-mihaly.hu/reversi-othello/

Készítsünk együtt egy programot, amivel két játékos Reversi-t tud játszani egymással! A programot lépésről lépésre készítjük el. Kövesd a foglalkozásvezető utasításait!



Függvények lokális változói

- a függvényen belül létrehozott változók csak a függvényen belül ismertek (**lokális változók**)
- akár azonos nevű lokális változóink is lehetnek különböző függvényekben, ezek különböző változók lesznek
- a főprogramban létrehozott változók mindenhol ismertek (**globális változók**)

Lokális változók

Próbáld ki! Mit tapasztalsz?

```
def jonatán():
    alma = "jonatán függvény almája"
    print(alma)

def idared():
    alma = "idared függvény almája"
    print(alma)

jonatán()
idared()
```

Lokális változók

Próbáld ki! Mit tapasztalsz?

```
alma = "főprogram almája"

def jonatán():
    alma = "jonatán függvény almája"
    print(alma)

def idared():
    alma = "idared függvény almája"
    print(alma)

jonatán()
idared()
print(alma)
```

Lokális változók

Próbáld ki! Mit tapasztalsz?

```
alma = "főprogram almája"

def jonatán(alma):
    alma = "jonatán függvény almája"
    print(alma)

def idared(alma):
    alma = "idared függvény almája"
    print(alma)

jonatán(alma)
idared(alma)
print(alma)
```

Lokális változók

Próbáld ki! Mit tapasztalsz?

```
alma = "főprogram almája"

def jonatán(alma):
    print(alma)
    alma = "jonatán függvény almája"
    print(alma)

jonatán(alma)
print(alma)
```

Lokális változók

Próbáld ki! Mit tapasztalsz?

```
alma = "főprogram almája"

def jonatán():
    global alma
    print(alma)
    alma = "jonatán függvény almája"
    print(alma)

jonatán()
print(alma)
```

Rekurzió

- a függvény definíciója hivatkozhat magára a függvényre
  - dehát ez őrültség!
  - vagy mégsem?

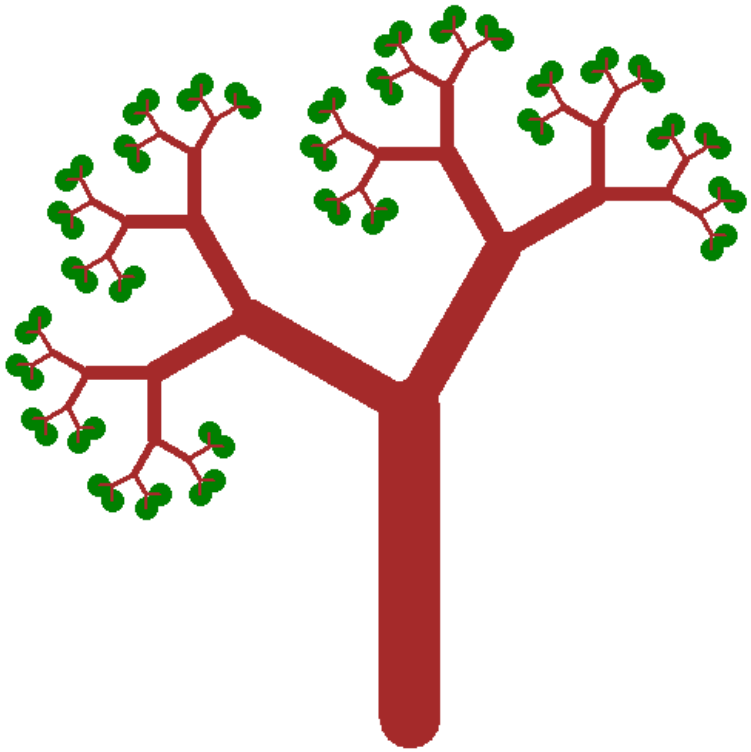
```
def oldal(hossz, mélység):
    if mélység == 0:
        forward(hossz)
    else:
        oldal(hossz/3, mélység-1)
        right(60)
        oldal(hossz/3, mélység-1)
        left(120)
        oldal(hossz/3, mélység-1)
        right(60)
        oldal(hossz/3, mélység-1)
```

Próbáld ki!

```
from turtle import *

def oldal(hossz, mélység):
    if mélység == 0:
      forward(hossz)
    else:
      oldal(hossz/3, mélység-1)
      right(60)
      oldal(hossz/3, mélység-1)
      left(120)
      oldal(hossz/3, mélység-1)
      right(60)
      oldal(hossz/3, mélység-1)
```

```
penup()
setpos(-200, 0)
pendown()
speed(0)

oldal(180, 3)
left(120)
oldal(180, 3)
left(120)
oldal(180, 3)

penup()
setpos(-400, 300)
```

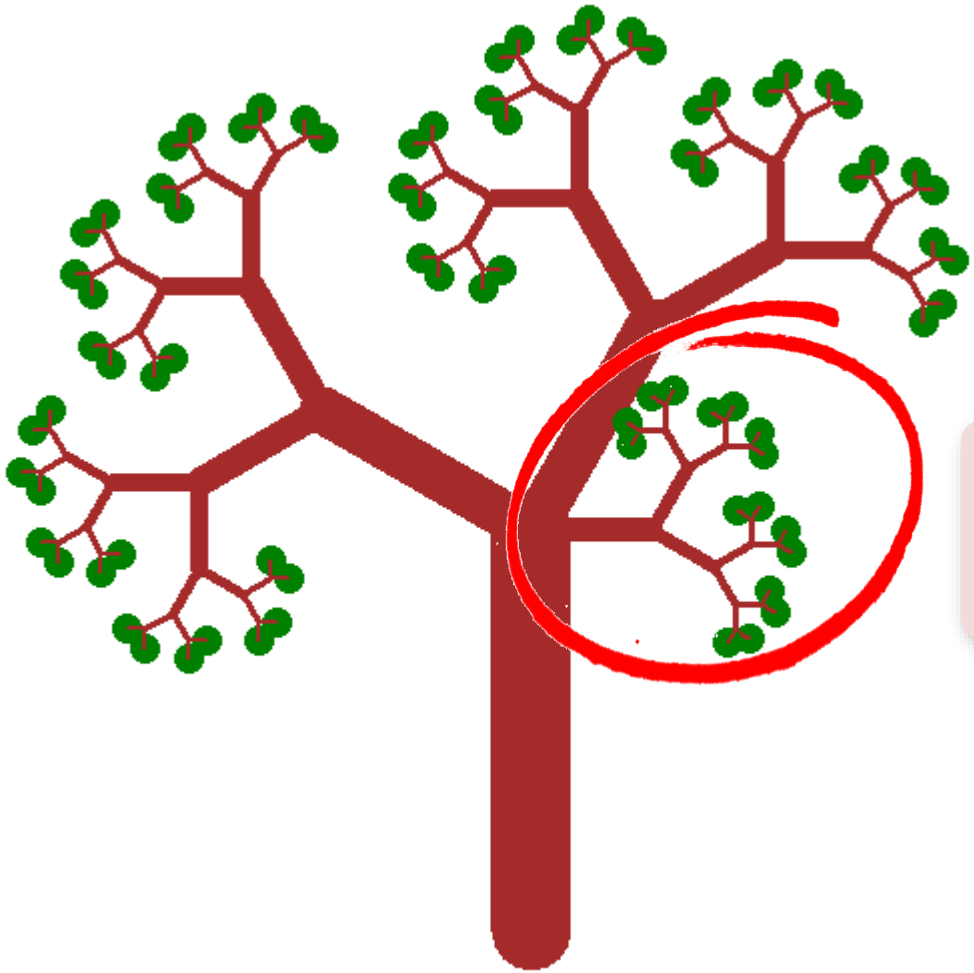
Fa



Elemzd és írd át ezt a programot, hogy olyan szép fát rajzoljon, mint az ábrán látható!

```
from turtle import *

def fa(ág_hossz):
    forward(ág_hossz)
    if ág_hossz > 10:
        left(45)
        fa(0.6*ág_hossz)
        right(90)
        fa(0.6*ág_hossz)
        left(45)
    backward(ág_hossz)

left(90)
fa(100)
```

Függvények



Hogyan varázsolnád be ennek a plusz ágnak a kirajzolását a programodba?

Házi feladat

A jobb oldalon látható programban az `összeg_for(n, N)` függvény egy `for` ciklus segítségével adja össze a számokat n-től N-ig.

Meg tudnád oldani ugyanezt a feladatot rekurzióval (`for` és `while` ciklus nélkül)?

Nevezd a függvényedet `összeg_rec(n, N)`-nek, és hasonlítsd össze a visszaadott eredményt a másik függvény eredményével!

```
def összeg_for(n, N):
    összeg = 0
    for i in range(n, N+1):
        összeg += i
    return összeg

print(összeg_for(10,20))


def összeg_rec(n, N):
    ...

print(összeg_rec(10,20))
```

Összefoglalás

- a függvények lényegében mini-programok, melyek egy nagyobb feladat egy-egy részfeladatát végezik el
- ezekből a mini-programokból bonyolult programokat építhetünk
- a függvények nagyon hasznosak ahhoz, hogy bonyolultabb programoknak áttekinthetőbb szerkezetet adjunk

Összefoglalás

- egy függvénynek
  - van neve
  - lehetnek bemeneti paraméterei (argumentumai)
  - lehet visszatérési értéke

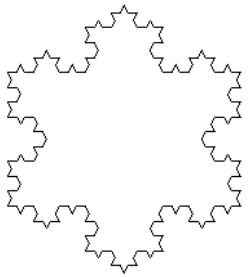
- a függvényt úgy használjuk, hogy meghívjuk
  - leírjuk a nevét
  - megadjuk a bemeneti paraméterek értékét
  - felhasználjuk a visszatérési értékét

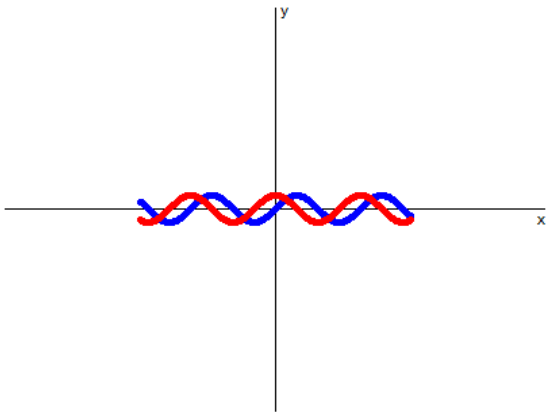
Összefoglalás

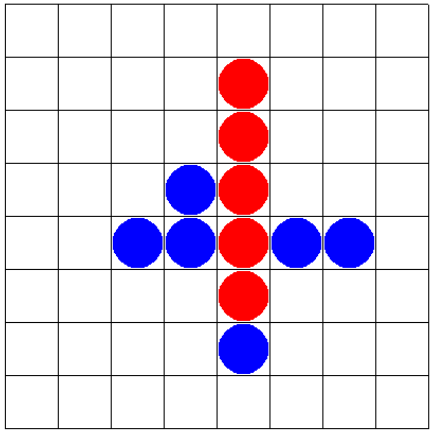
- egy függvénynek lehetnek lokális változói, amik csak a függvényen belül ismertek
- érdekességek:
  - különböző függvényeknek lehetnek azonos nevű lokális változói (ezeket a program különböző változóként kezeli)
  - függvény bemeneti paramétere lehet egy másik függvény neve
  - a függvény definíciója hivatkozhat magára a függvényre (rekurzió)

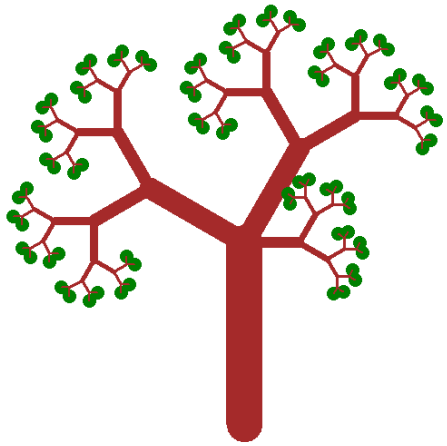
Összefoglalás

- klasz dolgokat készítettünk!









Készítette:
Buttyán Levente
Levente.Buttyan@gmail.com
CoderDojo Szentendre
2017