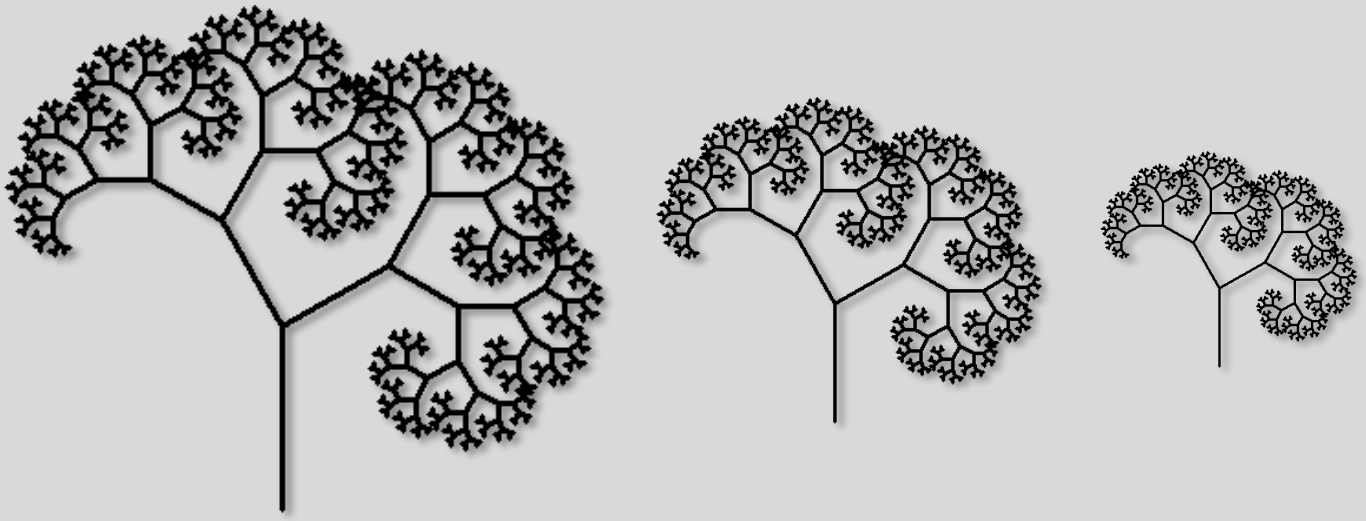
Belépő a tudás közösségébe

Szakköri segédanyagok tanároknak



Versenyfeladatok Pythonban!

H. Bakonyi Viktória

A kiadvány „A felsőoktatásba bekerülést elősegítő készségfejlesztő és kommunikációs programok megvalósítása, valamint az MTMI szakok népszerűsítése a felsőoktatásban” (EFOP-3.4.4-16-2017-006) című pályázat keretében készült 2017-ben.

Eötvös Loránd Tudományegyetem
Informatikai Kar

SZÉCHENYI 2020
MAGYARORSZÁG KORMÁNYA
Európai Unió
Európai Szociális Alap
BEFEKTETÉS A JÖVŐBE

Versenyfeladatok Pythonban!

Szerző

H. Bakonyi Viktória

Felelős kiadó

ELTE Informatikai Kar
1117 Budapest, Pázmány Péter sétány 1/C.

ISBN szám

ISBN 978-963-284-991-1

A kiadvány „A felsőoktatásba bekerülést elősegítő készségfejlesztő és kommunikációs programok megvalósítása, valamint az MTMI szakok népszerűsítése a felsőoktatásban” (EFOP-3.4.4-16-2017-006) című pályázat keretében készült 2017-ben.

Tartalomjegyzék

Bevezető

A Python ma az iparban széleskörűen használt modern, magasszintű programozási nyelv, amely többféle programozási paradigmát is támogat, például a funkcionális, a procedurális vagy akár az objektumorientált modelt. Dinamikus adattípusokat, automatikus memória kezelést használ.

A Python megtervezésénél az egyik fő szempont a könnyű olvashatóság volt, amely például a programszerkezetek egymásba ágyazott intendálásában tükröződik. Az olvashatóságon kívül a másik nagy vonzerőt az elkészített kódok egyszerű futtatása jelenti, hiszen csak egy interpreterre van szükség hozzá. A nyelv igen gazdag könyvtári szolgáltatások tekintetében is a képfeldolgozó, matematikai modulok mellett, többek között teknőc grafikai lehetőségekkel is rendelkezik. Ez teszi lehetővé, hogy a Pythont ismerő diákok bekapcsolódhassanak az Középiskolai Országos Logo Tanulmányi Versenybe. A verseny honlapja, a korábbi versenyek feladatai a http://logo.inf.elte.hu/ oldalon megtalálhatóak, ahonnan az itt olvasható feladatok is származnak.

Ez a tananyag arra szolgál, hogy az érdeklődők megismerjék a Python nyelv és annak teknőcgrafikához kapcsolódó alap funkcionalitásait olyan feladatok megoldásán keresztül, amelyek mind-mind előfordultak a Logo verseny története során. Az alapok elsajátítása után reményeim szerint az olvasók tovább haladnak a megkezdett úton és a programozás világának más területein is hasznát veszik majd az itt tanultaknak.

Python alapok

A munkakörnyezet kialakítása

A Python kódok készítéséhez, futtatásához két lehetőség áll előttünk:

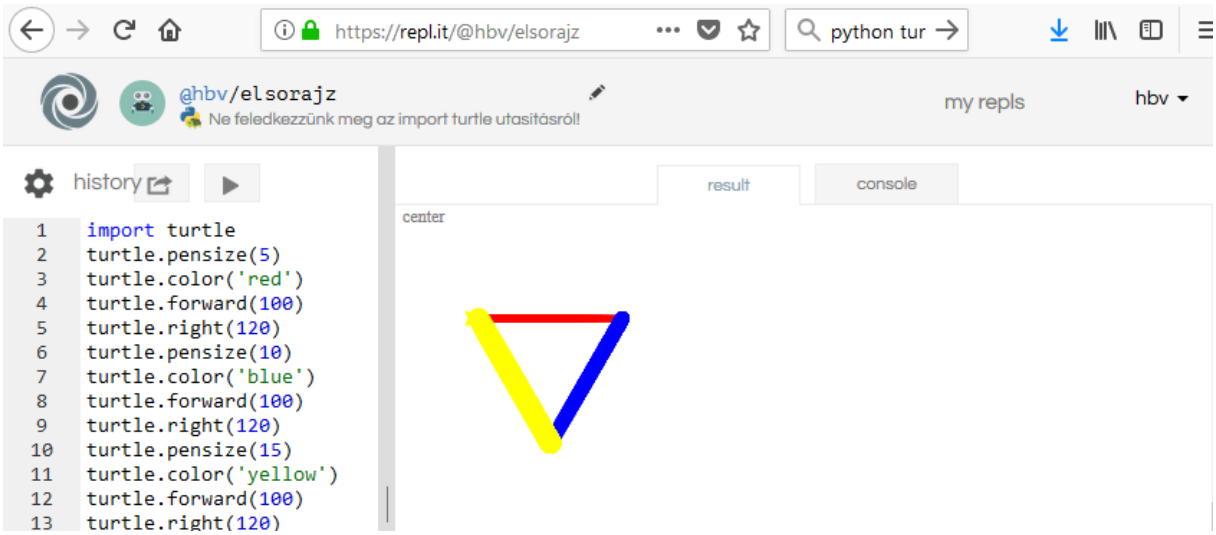
- Telepíthetjük a Pythont a számítógépünkre. Ehhez látogassunk el a https://www.python.org/downloads/ oldalra és válasszuk ki a számunkra megfelelő platformot és verziót. A tananyagban a Python 3.6. verzióját használjuk. *(Megjegyzés: A 3.0 alatti verziók jelentős szintakszisbeli eltéréseket tartalmaznak.)* Kódszerkesztőként használhatjuk az Idle-t vagy akár a professzionális Visual Studio 2017-t is.
- Használhatunk on-line Python környezetet is pl. http://repl.it, amelynek sok előnye van, hiszen regisztrált felhasználóként bármikor, bárhonnan elérhetjük az elkészített munkáinkat vagy akár meg is oszthatjuk azokat másokkal. **Figyeljünk arra, hogy a repl.it oldalon a projekt létrehozásánál a Python (with turtle) lehetőséget válasszuk!** A tananyagban olvasható kódok megtalálhatóak ezen az oldalon, a feladatok mellett megadott linkeken.

Megjegyzés: Az online rendszerben az ékezetes betűk használata nem javallott, ezért a minta kódokban kerültük ezeket!

Rajzolás

Minden teknőcgrafikát használó kódunk elején importálni kell a turtle könyvtárat, tehát írjuk be az **import turtle** sort!.Ezután egy teknőc alak segítségével, mint tollal rajzolhatunk. A teknőcöt egyszerű angol szavakkal tudjuk irányítani pl. *turtle.forward(hossz)* hossz pixelt küldi előre a teknőcöt, *turtle.right(szög)* elfordítja jobbra szög fokkal, *.turtle.color('szín')* pedig megváltoztatja a toll színét. A toll vastagságát a *turtle.pensize(méret)* méret szélességűre változtatja.

Rajzoljuk is meg az első Python ábránkat, amelyen egy szabályos háromszöget láthatunk különböző oldalszínekkel és vastagságokkal. Az 1. ábrán mind a repl.it munkakörnyezetet, mind pedig a kódot és a létrejött rajzot érdemes megfigyelni! A rajzoláshoz szükséges alapvető utasítások összefoglalóját kicsit később az anyagban megadjuk.



1. ábra repl.it munkakörnyezet (Elérhető: https://repl.it/@hbv/elsorajz)

Az 1. táblázatban összefoglaltuk a legfontosabb rajzoló utasításokat.

1. táblázat rajzoló utasítások

turtle.speed(gyorsaság)	a teknőc rajzolási sebessége 0-10 között. A 0 a leggyorsabb!
turtle.home()	a teknőc alaphelyzetébe visszatér
turtle.forward(méret)	a teknőc előre halad méret pixelt
turtle.backward(méret)	a teknőc hátrál méret pixelt
turtle.right(szög)	a teknőc elfordul az óramutató járásával megegyezően szögnyit
turtle.left(szög)	a teknőc elfordul az óramutató járásával ellentétesen szögnyit
turtle.color(szín), turtle.color(szín,szín2)	a teknőc tollának színe szín-re változik. szín lehet RGB is (r,g,b) a teknőc tollának színe szín-re,a töltőszíne pedig szín2re változik
turtle.fillcolor(szín)	a teknőc töltőszínének beállítása
turtle.pencolor(szín)	a teknőc tollszínének a beállítása
turtle.pensize(méret)	a teknőc tollvastagsága méretnyi lesz
turtle.pendown()	a teknőc leteszi a tollát, rajzolni tud.
turtle.penup()	a teknőc felemelei a tollat, nem rajzol.
turtle.circle(sugár)	a teknőc egy sugárnyi kört rajzol maga köré. Kör középpontja tőle balra sugárnyira!
turtle.circle(sugár,szög)	a teknőc egy sugárnyi körívet rajzol maga köré, szögnyi belső szöggel..
turtle.dot()	a teknőc az aktuális pozíciójába egy pontot rajzol
turtle.begin_fill()	az end_fill() hívásig a kirajzolt alakzatot megjegyzi. A kitöltéshez a toll felemelt állapotban is lehet!
turtle.end_fill()	a begin_fill() hívása után alakzatot kiszínezi.
turtle.hideturtle()	a teknőc eltűnik
turtle.showturtle()	a teknőc megjelenik

Programszerkezetek, alprogramok, adattípusok

Természetesen nincs lehetőség itt a nyelv lehetőségeinek teljes áttekintésére, csak arra szorítkozunk, ami feltétlenül szükséges az első lépések megtételéhez! Bővebb anyag található magyarul például a következő linken: *http://mek.oszk.hu/08400/08435/08435.pdf.*

Az alapvető programszerkezetek a szekvencia, az elágazás, a ciklus és a függvény, mint alprogram. Minden utasítás külön-külön sorba írandó. Az elágazás, ciklus és függvény esetében a szerkezetet bevezető utasításrészt kettősponttal zárjuk, majd a definícióját egy tabulátor pozícióval beljebb írjuk (intendálás). Nézzünk egy-egy egyszerű példát!

```
import turtle
print('Rajzoljak?')       # kiírás
valasz=input()            # szekvencia, beolvasás, szövegként
if valasz=='I':  # elágazás, egyenlőe operátor
   turtle.forward(100) # egy tabulátorral beljebb
else:               # különben ág
   print('vege')
```

Elérhető: *https://repl.it/@hbv/elagazas*

```
import turtle
def haromszog(meret):
   # függvény definiálása, egy paraméterrel, nincs típus megadás
   for i in range(3):    # ciklus, ahol a ciklusváltozó a [0,3[
                  # intervallum egész értékeit veheti fel
     turtle.forward(meret)
     turtle.right(120)
haromszog(100)    # függvény hívása aktuális értékkel
```

Elérhető: https://repl.it/@hbv/ciklusfuggv

`if feltétel:` `  utasítás(ok)`	Egyszerű elágazás. Leggyakoribb operátorok: ==, !=, >,>=,<,<=

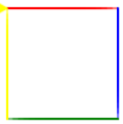
`if feltétel:`
`  utasítás(ok)`
`else:`
`  utasítás(ok)2`
	Kétirányú elágazás.
	Bővebben az operátorokról a http://bit.ly/2nhND43 linken lehet olvasni

`if feltétel:`
`  utasítás`
`elif feltétel2:`
`  utasítás(ok)2`
	Az elif kulcsszó az
	else:
	if feltétel:
	helyettesíti

`for x in lista:` `  utasítás(ok)`	A ciklusváltozó rendre felveszi a lista elemei
`for i in range(érték1,érték2):` `  utasítás(ok)`	A range függvény egy listát gyárt érték1 és érték2 között. Érték2-t már nem veszi fel.
`while feltétel:` `  utasítás(ok)`	A ciklus addig ismétlődik, amíg a feltétel igaz.
`def eljárásnév(form. paraméterek):` `  utasítások`	A formális paramétereket vesszővel elválasztva írjuk. Ha nincs egy sem, akkor is az üres zárójel kiírása kötelező. Meghívása: eljárásnév(paraméterek). NINCS címszerinti (referencia) paraméterátadás.

`def függvénynév(form. paraméterek):`
`  utasítások`
`  return visszatérési érték`
	A formális paramétereket vesszővel elválasztva írjuk. Ha nincs egy sem, akkor is az üres zárójel kiírása kötelező.
	Meghívása: függvénynév(paraméterek).

Elemi típusok: egész, valós, logikai, szöveg. Változó deklaráció azonban nincs! Léteznek konverziós függvények pl. int(szövegformában adott szám) vagy str(szám), ami szöveggé alakít. Összetett adattípus a lista és a tuple (csak ezeket használjuk a tananyagban), amely sorszámon keresztül elérhetővé teszi az elemeket. A lista konstruálása ***a [ ]*** zárójellel történik, bővítése az ***append*** vagy az ***insert*** függvénnyel lehetséges. A lista egy adott szeletét a ***lista[első:utolsó]*** megadásával olvashatjuk ki, a lista hosszát pedig a ***len*** függvénnyel. A tuple abban különbözik a listától, hogy nem módosítható a tartalma. Konstruálása ( ) zárójellel történik. Nézzünk egy-két gyors példát a fentiekre. Az elsőben rendre egy szín listából veszük a négyzet oldalának színeit!



```
import turtle
szinek=['red', 'blue','green','yellow']   # színek listája
for i in range(4):
  turtle.color(szinek[i])          #a lista i. eleme (0-tól kezdődik!)
  turtle.forward(100)
  turtle.right(90)
```

Elérhető: *https://repl.it/@hbv/szinlista*

`lista=[]`	üres lista létrehozása
`lista=elem*db`	db elemből álló listát hoz létre
`len(lista)`	a lista hossza
`lista[index]`	a lista indexedik eleme
`lista[index:]`	eredménye az eredeti lista az indexedik elemtől az utolsóig
`lista[:index]`	eredménye az eredeti lista az elsőtől az indexedik elemig
`lista.index(elem)`	eredménye az elem sorszáma a listán belül (0-tól kezdve)
`lista.append(elem)`	új elemet beszúr a listába
`szoveg.join(lista)`	a lista elemeit szöveggé összefűzi
`lista=szöveg.split()`	A szövegből a szóközök mentén darabolva létrehoz egy szavakból álló listát!

Megjegyzés: A megoldások során figyelmet fordítunk az *állapotátlátszóság* elvének betartására. (Azt jelenti, hogy egy alakzta kirajzolása után lehetőleg ugyanabba a helyzetbe és irányba kerüljön vissza a teknőc, mint amilyenben a rajzolás megkezdésekor volt. Ennek az elvnek a betartása megkönnyíti a bonyolultabb alakzatok összeépítését az elemibb részekből!

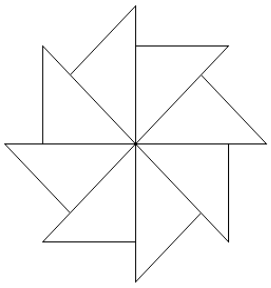
Felhasználjuk a megoldások során a *Teljes teknőc tételt is.* Záró síkbeli alakzatoknál, ha a teknőc visszatér kiindulási állapotába, akkor a megtett fordulatok összege 360o vagy annak többszöröse.

Gyakran felhasználhatóak a tükrözési tételek is a megoldások során. Különböző tükrözéseket valósíthatunk meg a negatív lépéshosszal vagy a negatív elfordulási szöggel is. Jól használható ez a tudás például a körrajzolásnál!

Sokszögek, csillagok

Ebben a leckében ízelítőül sokszög és csillag alakzatok rajzolásával oldunk meg néhány feladatot.

Feladat: Készíts derékszögű, egyenlőszárú háromszöget (`derek`), majd olyan eljárást (`dereka`) amelyek derékszögű háromszögekből a következő ábrát tudja kirakni. Az ábrát úgy kapjuk, hogy 45 fokonként elfordulunk, és kirajzoljuk a derékszögű háromszöget. A méret legyen paraméter!

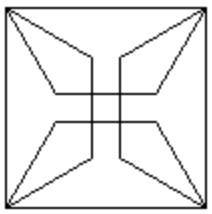


dereka 100

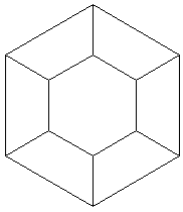
```
import turtle,math  #matematikai függvényekhez
def derek(meret):
   turtle.forward(meret)
   turtle.right(90)
   turtle.forward(meret)
   turtle.right(135)
   turtle.forward(meret*math.sqrt(2))
            #átfogó hossza befogó*gyök(2)
   turtle.right(135)
def dereka(meret):
   for i in range(8):
      derek(meret)
      turtle.right(45)
dereka(100)
```

Elérhető: *https://repl.it/@hbv/derekszogu*

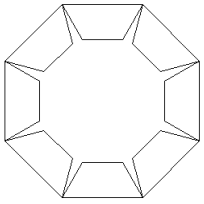
Feladat: Készíts egy `trapezok` nevű eljárást, amely különböző paraméterekkel meghívva az alábbi ábrákat tudja kirajzolni!



`trapezok(4,100)`



`trapezok(6,50)`

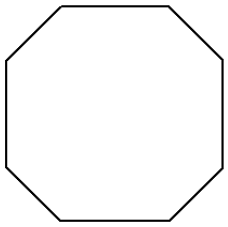


`trapezok(8,50)`

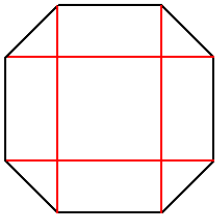
```
import turtle
turtle.speed(0)   #gyorsabban rajzol!


def trapez(h):
   turtle.forward(h)
   turtle.right( 120)
   turtle.forward(h/2)
   turtle.right( 60)
   turtle.forward(h/2)
   turtle.right( 60)
   turtle.forward(h/2)
   turtle.right( 120)

def trapezok(n,h):
   for i in range(n):
      trapez(h)
      turtle.forward(h)
      turtle.right( 360/n)


trapezok(4,100)    #különböző paraméterekkel
```

Elérhető: *https://repl.it/@hbv/trapezok*

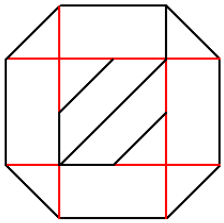
Feladat: Írj eljárásokat `nyolcszog1(oldal)`, … `nyolcszog3(oldal)` néven az alábbi nyolcszögek rajzolására, ahol `oldal` a nyolcszög oldalhossza! Figyeld meg, hogyan használjuk fel a korábbi megoldásokat az újabbak elkészítéséhez!



nyolcszog1 100



nyolcszog2 100



nyolcszog3 100

```
def atlo(oldal):
atloh=oldal*math.sqrt(2)
  turtle.left(135)
  turtle.pendown()
  turtle.forward(atloh)
  turtle.backward(atloh)
  turtle.right(135)
  turtle.penup()
```

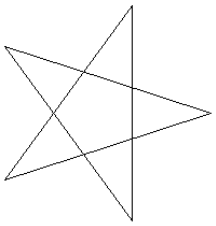
```
import turtle,math

def nyolcszog1(oldal):
  turtle.pencolor('black')
  for i in range(4):
    turtle.forward (oldal)
    turtle.right(45)
    turtle.forward((oldal/2)*math.sqrt(2))
    turtle.right(45)

 def nyolcszog2(oldal):
  nyolcszog1(oldal)
  turtle.pencolor('red')
  turtle.right(90)
  for i in range(4):
    turtle.forward(oldal*2)
    turtle.penup()
    turtle.right(135)
    turtle.forward((oldal/2)*math.sqrt(2))
    turtle.pendown()
    turtle.right(135)
  turtle.left(90)

 def nyolcszog3(oldal):
  nyolcszog2(oldal)
  turtle.penup()
  turtle.pencolor('black')
  turtle.right(90)
  turtle.forward(oldal)
  atlo(oldal/2)
  turtle.forward(oldal/2)
  atlo(oldal)
  turtle.left(90)
  turtle.forward(oldal/2)
  turtle.right(90)
  atlo(oldal/2)

turtle.speed(0)
nyolcszog1(100)
nyolcszog2(100)
nyolcszog3(100)
```

Elérhető: *https://repl.it/@hbv/nyolcszogek*

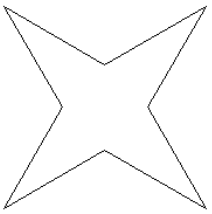
Feladat: Készíts `csillag(h,f,s)` eljárást, amely olyan sokszöget rajzol, amelynél az oldalakat töröttvonalakkal helyettesítjük és a helyettesítő szakaszpárok hossza `h`, ezek `f` fokos szöget zárnak be az eredeti sokszögoldallal, `s` a sokszög csúcsaiban `s` fokot kell fordulni! Egy oldalt az `oldal(h,f)` eljárás rajzoljon! A sokszög rajzolás akkor fejeződjön be, ha a teknőc visszatér a rajzolás előtti állapotába!

oldal(100,30)

```
import turtle
turtle.speed(0)

def oldal( h, f):
   turtle.left(f)
   turtle.forward(h)
   turtle.right(2 * f)
   turtle.forward(h)
   turtle.left(f)
```



csillag(100,0,144)



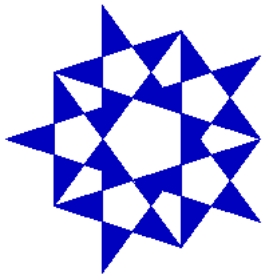
csillag(100,30,90)

```
def csillag(h,f,s):
 tortsok(h,f, s,0)

def tortsok(h,f,s,szog)
   oldal(h,f)
   turtle.left(s)
   if (s+szog) % 360 != 0 :
      tortsok(h,f,s,s+szog)

#oldal(100,30)
#csillag(100,0,144)
csillag(100,30,90)
```

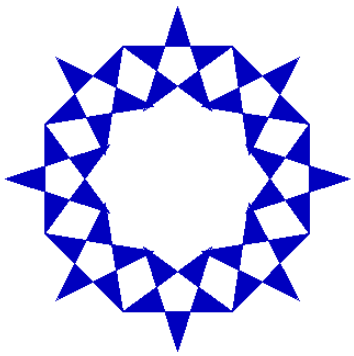
Elérhető: *https://repl.it/@hbv/csillag*

Feladat: Készíts eljárást színezett ágú ötágú csillag rajzolására `otagucsillag(h)`, ahol `h` a csillag oldalhossza! Alkoss csillagokból `n` oldalú szabályos sokszöget `csillagok(n,h)`, ahol a csillagok a csúcsaikkal érnek össze!

otagucsillag 100



csillagok 5 100



csillagok 8 100

```
import turtle,math

def kitolt(hossz):
   turtle.begin_fill()
   rovidoldal=2*hossz*math.cos(math.radians(72))
   turtle.forward(rovidoldal)
   turtle.left(108)
   turtle.forward(hossz)
   turtle.left(144)
   turtle.forward(hossz)
   turtle.end_fill()
   turtle.left(108)

def otagucsillag(hossz):
   rovidoldal=2*hossz*math.cos(math.radians(72))
   for i in range(5):
      turtle.forward(hossz)
      kitolt(hossz)
      turtle.forward(hossz+rovidoldal)
      turtle.right(144)

def csillagok(n,hossz) :
   rovidoldal=2*hossz*math.cos(math.radians(72))
   for i in range(n):
      turtle.penup()
      turtle.forward(2*hossz+rovidoldal)
      turtle.right(360/n)
      turtle.pendown()
      otagucsillag(hossz)

turtle.speed(0)
turtle.color('blue','blue')
csillagok(5,50)
#csillagok(10,50)
```

Elérhető: *https://repl.it/@hbv/csillagok*

Körök, körívek rajzolása

Feladat: Készíts megoldást a következő ábrák elkészítéséhez a `turban(n,r)` és `uszogumi(r)` eljárásokat, ahol n az elforgatott körök számát, r pedig a sugarat jelöli.!

turbán(20,100)

```
import turtle
def toltottkor(r):
  turtle.begin_fill()
  turtle.circle(r)
  turtle.end_fill()
```

```
def turban(n,m,r):
  if n%2==1: #rekurzió a váltakozó színek miatt
    turtle.color('red','red')
  else:
    turtle.color('yellow','yellow')
  toltottkor(r)
  if n>0:
    turtle.left(360/m)
    turban(n-1,m,r)

turtle.speed(0)
turban(36,36,100)
```

Elérhető: *https://repl.it/@hbv/turban*

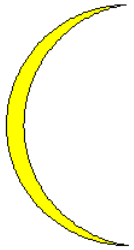
uszogumi(100)

```
import turtle

def kozepre(r):
  turtle.penup()
  turtle.right(90)
  turtle.forward(r)
  turtle.left(90)
  turtle.pendown()
```

```
def kozepretoltottkor(r):
  kozepre(r)
  toltottkor(r)   #turban feladatban megadott
  kozepre(-r)     #-r használatával hátrál

def uszogumi(r):
  turtle.color('black','red')
  turtle.pensize(2)
  kozepretoltottkor(r)
  turtle.color('black','white')
  kozepretoltottkor(r/5*3)
  turtle.pensize(r/20)
  turtle.color('white')
  kozepre(r/5*4)
  turtle.circle(r/5*4)
  vissza(r/5*4)

turtle.speed(0)
uszogumi(100)
```

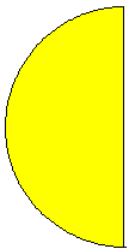
Elérhető: *https://repl.it/@hbv/uszogumi*

Feladat: Készíts holdat rajzoló eljárást `hold(r,fok)`, ahol `r` a hold külső íve sugara, `fok` pedig a belső ív dőlésszöge a körhöz képest!

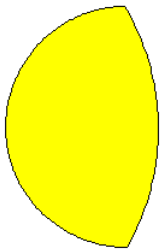


hold 100 10

```
import turtle,math

def hold(r,fok):
 turtle.color('black','yellow')
 turtle.right(180)
 turtle.begin_fill()
 turtle.circle(-r, 180)
```



hold 100 90



hold 100 120

```
if fok<90:
  turtle.right(180-fok)
  x= r/math.cos(math.radians(fok))
  turtle.circle(x, 180-2*fok)
  turtle.penup()
if fok==90:
  turtle.right(90)
  turtle.forward(2*r)
  turtle.backward(2*r)
  turtle.left(90)
if fok>90:
  turtle.right(180-fok)
  x=-r/math.cos(math.radians(180-fok)
  turtle.circle(x,2*fok-180)
  turtle.right( 180-fok)
turtle.end_fill()

turtle.speed(0)
#hold(100,150)
#hold(100,90)
hold(100,60)
```

Elérhető: *https://repl.it/@hbv/holdak*

Feladat: Készítsd el a mintának megfelelően az olimpiai ötkarikát (ötkarika(r), ahol r a színes 5 pont vastagságú körök sugara, a vízszintes körök :r/5 távolságra vannak egymástól, s az átfedéseknél 30 fokos szakadások vannak a körökön. A felső sor körei színe balról jobbra haladva: kék, fekete, piros; az alsó soré pedig: sárga, zöld.

otkarika(50)

```
def karika(r,szog1,szog2):
  turtle.pendown()
  turtle.circle(r,szog1-15)
  turtle.penup()
  turtle.circle(r,30)
  if szog2>szog1+30:
    turtle.pendown()
    turtle.circle(r,szog2-szog1-15)
    turtle.penup()
    turtle.circle(r,30)
    turtle.pendown()
    turtle.circle(r,360-szog2-30)
  else:
    turtle.pendown()
    turtle.circle(r,360-szog1-15)
```

```
def otkarika(r):
  turtle.pensize(5)
  turtle.color('blue')
  karika(r,15,15)
  turtle.penup()
  turtle.forward(r*11/5)
  turtle.color('black')
  karika(r,15,270)
  turtle.penup()
  turtle.forward(r*11/5)
  turtle.color('red')
  karika(r,280,280)
  turtle.penup()
  turtle.backward(3*r+r/5)
  turtle.left(90)
  turtle.backward(r+r/5)
  turtle.right(90)
  turtle.color('yellow')
  karika(r,100,180)
  turtle.penup()
  turtle.forward(r*11/5)
  turtle.color('light green')
  karika(r,100,180)

  turtle.speed(0)
  otkarika(50)
```

Elérhető: *https://repl.it/@hbv/otkarika*

Variációk zászlók rajzolására

A különböző országok lobogói felépítésükben, színvilágukban, de akár oldalarányaikban is rendkívül különbözőek lehetnek. Erről magunk is meggyőződhetünk, ha megtekintjük a http://bit.ly/1Hae1BH weboldalt.

Feladat: Gambia zászlaja nem azonos magasságú téglalapokból áll. Az ismétlődő csíkokat azonban érdemes egymásba rajzolt téglalapokkal megvalósítani. A zöld, fehér és piros csíkok magassága a zászló magasságának harmada. Az oldalarány: 2:3.

A színek kódjai: piros (206, 17, 38), kék (12, 28, 140), zöld (58,119, 40).

```
import turtle

def tegla(a,b):
  turtle.pendown()
  for i in range(2):
    turtle.forward(a)
    turtle.right(90)
    turtle.forward(b)
    turtle.right(90)
  turtle.penup()

def teglalap(a,b,szin):
  turtle.color(szin,szin)
  turtle.begin_fill()
  tegla(a,b)
  turtle.end_fill()
```

```
def gambia(magas,szeles):
 teglalap(magas/3,szeles,(58, 119, 40))
 turtle.forward( magas/3)
 teglalap( magas/3, szeles, "white")
 turtle.forward(magas/3/5)
 teglalap(magas/3*3/5, szeles ,(12, 28,
140))
 turtle.forward( magas/3*4/5)
 teglalap( magas/3, szeles ,(206, 17, 38))
 turtle.backward(magas/3*2)
 turtle.pencolor("black")
 turtle.pensize(3)
 tegla(magas,szeles)

turtle.speed(0)
turtle.left(90)
gambia(100,150)
```

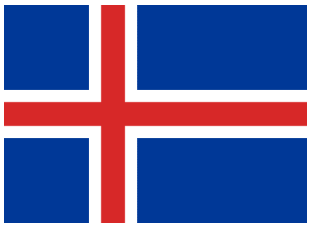
Elérhető: *https://repl.it/@hbv/gambia*

Feladat: Rajzoljuk meg Görögország zászlaját is, amelynek oldalaránya: 2:3! A kék szín kódja: (13, 94,175). A csíkok magassága a lobogó magasságának 11,11%-a. A bal felső sarokban lévő kék négyzet mérete a szélesség 37%-a. Megoldásban 5 kék csíkot, majd egy kék négyzetet rajzolunk, majd ebbe a keresztet fehérrel.

```
import turtle

def kereszt(magas,szeles):
  x=11.11/100*magas
  turtle.forward(2*x)
  teglalap(x,szeles*37/100,"white")
  turtle.backward(2*x)
  turtle.right(90)
  turtle.forward(2*x)
  turtle.left(90)
  teglalap(szeles*37/100,x,"white")
```

```
def gorogorszag(magas,szeles,szin):
 turtle.penup()
 x=11.11/100*magas
 for i in range(5):
   teglalap(x,szeles,szin)
   turtle.forward(2*x)
 turtle.backward(6*x)
 x=szeles*37/100
 teglalap(x,x,(13, 94, 175))
 kereszt(magas,szeles)

turtle.speed(0)
turtle.left(90)
gorogorszag(200,300,(13, 94, 175))
```

Elérhető: *https://repl.it/@hbv/gorogorszag*



Feladat: Izland lobogójának oldalaránya: 18:25. A színkódok: kék (0 ,56, 151), piros (215, 40, 40). A bal oldali kék téglalapok a szélesség 28%-ának felelnek meg. A fehér sáv 16% széles, a jobb oldali téglalapok szélessége így 56%. A piros csík szélessége a lobogó szélességének 8%-a. A felső téglalapok magassága a lobogó magasságának 39%-a. A fehér csík magassága 22%, a piros csíké 11%, az alsó téglalapé 39%. (A megoldás során a Gambiai zászlónál felhasznált teglalap eljárást felhasználjuk!)

```
def izland(magas,szeles):
 teglalap(magas,szeles,(0,56,151))
 kereszt(magas,szeles,"white")
 kereszt(magas,szeles,"red")
def kereszt( magas,szeles,melyik):
 turtle.penup()
 x= magas*22/100
 if melyik=="white":
   turtle.forward( magas*39/100)
   teglalap(x,szeles,melyik)
   turtle.backward(magas*39/100)
 else:
   turtle.forward( magas*89/200)
   teglalap(x/2,szeles,"red")
   turtle.backward(magas*89/200)
 turtle.right( 90)
```

```
y=szeles*16/100
if melyik=="white":
   turtle.forward( szeles*28/100)
   turtle.left(90)
   teglalap(magas,y, "white")
   turtle.right(90)
   turtle.backward(szeles*28/100)
 else:
   turtle.forward( 2*y)
   turtle.left(90)
   teglalap(magas,y/2, "red")
   turtle.right( 90)
   turtle.backward(2*y)
 turtle.left( 90)
turtle.left(90)
turtle.speed(0)
izland(180,250)
```

Elérhető: *https://repl.it/@hbv/izland*

Feladat: Törökország lobogójában a holdsarlót megvalósíthatjuk egy fehér kitöltésű körrel, amelyet részben takar egy piros kör. Az oldalarány ebben az esetben 2:3, a piros szín kódja: (227, 10, 23).A fehér kör bal oldalának széle a szélesség 18,5%-ára esik, és az 52,5%-ig tart. Vagyis az átmérő a szélesség 35%-a, így a sugár 17%. A piros kör bal oldali pontja a lobogó szélességének 54%-ra esik. Az átmérője a szélesség 27%-a. A csillag bal oldali széle a szélesség 49%-án kezdődik, oldalhossza 6%. Mind a holdsarló, mind a csillag függőlegesen középre van igazítva.

```
import turtle,math

def csillag(hossz):
  for i in range(5):
     turtle.forward(hossz)
     turtle.right(144)

def torokorszag(magas,szeles):
  turtle.penup()
  s=(227,10,23)
  teglalap(magas,szeles,s)
  holdsarlo(magas,szeles)
  csillagrajzolas(magas,szeles)

def kezdoallapot(magas,szeles):
 turtle.forward( magas/2)
 turtle.right( 90)
 turtle.forward( szeles*52.5/100)
 turtle.left( 90)
```

```
def holdsarlo(magas,szeles):
 turtle.color((227, 10, 23),'white')
 kezdoallapot(magas,szeles)
 turtle.begin_fill()
 turtle.circle(szeles*17/100)
 turtle.end_fill()
 turtle.right( 90)
 turtle.forward( szeles*1.5/100)
 turtle.left( 90)
 turtle.fillcolor(227, 10, 23)
 turtle.begin_fill()
 turtle.circle(szeles*13.5/100)
 turtle.end_fill()
 turtle.right(90)

turtle.speed(0)
turtle.left(90)
torokorszag(200,300)
turtle.color(227, 10, 23)
```

Elérhető *https://repl.it/@hbv/torokorszag*

Térkitöltés forgatással

A térkitöltést (mozaikot) klasszikus esetben sormintából kiindulva készítjük el. Egy ettől teljesen különböző eset az, amikor a területet egy körcikkbe rajzolt ábra forgatásával fedjük le. A körcikkek lehetnek diszjunktak, illetve átfedőek is.

Feladat Itt a tél! Készítsd el a `jégvirág1(hossz)` és a `jégvirág2(hossz)` eljárásokat, melyek az alábbi hópelyheket rajzolják:

`jégvirág1(10)`

`jégvirág2(10)`

A `jégvirág1` egy hatszögből és hat elforgatott elemből áll.

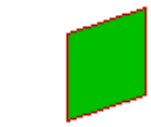
```
import turtle

def jegvirag1(hossz):
 kulso(hossz)
 turtle.forward(hossz)
 turtle.right(120)
 belso(hossz)
 turtle.left(120)
 turtle.backward(hossz)

def kulso(hossz):
 for i in range(6):
   ag(hossz)
   turtle.right(60)

def belso(hossz):
 for i in range(6):
  turtle.forward(hossz)
  turtle.right(60)

def ag(hossz):
 turtle.forward(2*hossz)
 turtle.left(60)
 vonal(hossz)
 turtle.right(120)
 vonal(hossz)
 turtle.left(60)
 turtle.backward(2*hossz)

def vonal(hossz):
  turtle.forward(hossz)
  turtle.backward(hossz)

turtle.speed(0)
turtle.color("black")
jegvirag1(20)
```

Elérhető: *https://repl.it/@hbv/jeg-virag1*

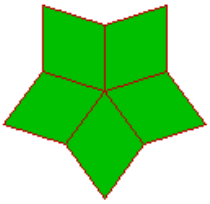
```
import turtle

def jegvirag2(hossz):
 kulso2(hossz)
 turtle.forward(hossz)
 turtle.right(60)
 belso2(hossz)
 turtle.right(120)
 turtle.backward(hossz)

def kulso2(hossz):
 for i in range(6):
   ag2(hossz)
   turtle.right(60)

def belso2(hossz):
 for i in range(6):
   for j in range(2):
    turtle.forward(hossz)
    turtle.right(120)
   turtle.right(180)

def ag2(hossz):
 turtle.forward(2*hossz)
 turtle.left(60)
 for i in range(3):
  vonal(hossz) #jegvirag1-ből
  turtle.right(60)
 turtle.left(120)
 turtle.backward(2*hossz)

turtle.speed(0)
turtle.color("black")
jegvirag2(20)
```

Elérhető: *https://repl.it/@hbv/jeg-virag2*

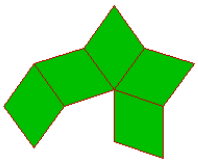
Feladat Készíts eljárásokat az alábbi ábrák rajzolására `rombusz(h)`, `belső(h)`, `külső(h)`, `rombuszok(h)`, ahol `h` a rombusz oldalhossza! A belső ábra 5 rombusz, egymáshoz képest elforgatva. A külső ábra megrajzolásában is 5 ismétlődő rész van, azt kell `alap`-ként megrajzolni:



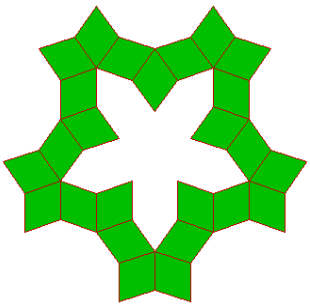
`rombusz(100)`



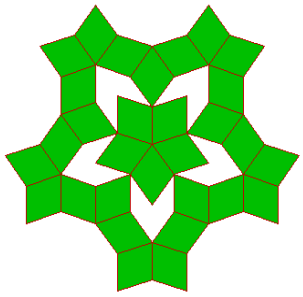
`belső(90)`



`alap(100)`



`külső(80)`



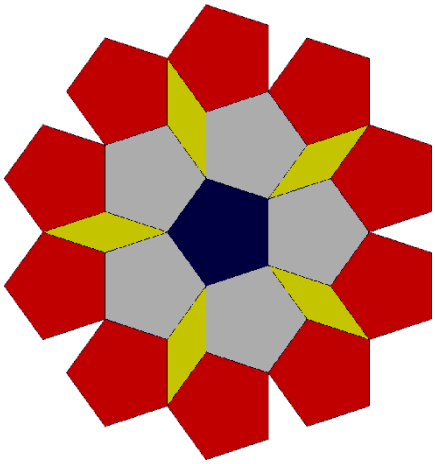
`Rombuszok(80)`

```
import turtle

def rombusz(h):
 turtle.color("black","green")
 turtle.pendown()
 turtle.begin_fill()
 for i in range(2):
  turtle.forward(h)
  turtle.right(72)
  turtle.forward( h)
  turtle.right(108)
 turtle.end_fill()
 turtle.penup()

def belso(h):
 for i in range(5):
  rombusz(h)
  turtle.right(72)

def kulso(h):
 for i in range(5):
  alap(h)
  turtle.right(72)

def alap(h):
 turtle.forward(h)
 turtle.left(36)
 rombusz(h)
 turtle.forward(h)
 turtle.right(72)
 turtle.forward(h)
 turtle.right(36)
 rombusz(h)
 turtle.forward(h)
 turtle.right(72)
 turtle.forward(h)
 turtle.left(180)
 rombusz(h)
 for i in range( 2):
  turtle.right(72)
  rombusz(h)
 turtle.right(144)
 for i in range(3):
  turtle.forward(h)
  turtle.left(36)
  turtle.right(216)

def rombuszok(h):
 belso(h)
 kulso(h)turtle.speed(0)
turtle.penup()
rombuszok(20)
```

Elérhető: *https://repl.it/@hbv/rombuszok*

Feladat Az alábbi, Penrose-mozaik színes ötszögekből készül úgy, hogy a közöttük kimaradó négyszög alakú területeket is beszínezzük.

Készíts eljárást `penrose(h)` néven, amely `h` oldalhosszúságú ötszögekből a mellékelt ábrán látható Penrose-mozaikot készíti el! Az ábra egy kék ötszög, amit szürke ötszögek vesznek körbe. A szürke ötszögek két oldalára két piros ötszöget kell rajzolni. Köztük sárga rombuszok vannak.



```
import turtle

def penrose(h):
 turtle.left(36)
 otszog(h, (0, 0, 64))
 for i in range(5):
  szurke(h)
  turtle.forward(h)
  turtle.right(72)
 turtle.left(144)
 for i in range(5):
  sarga( h)
  turtle.right(144)
  turtle.forward(h)
  turtle.left(72)
 turtle.down()

def piros(h):
 turtle.left(36)
 otszog( h, (192, 0, 0))
 turtle.right(36)
```

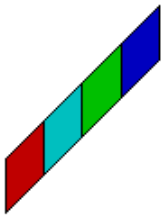
```
def szurke(h):
 turtle.left(108)
 otszog( h, (172, 172, 172))
 turtle.forward( h)
 piros( h)
 turtle.right(72)
 turtle.forward(h)
 piros( h)
 turtle.right(72)
 for i in range(3):
  turtle.forward(h)
  turtle.right(72)
 turtle.right(108)

def sarga(h):
 turtle.fillcolor(196, 196, 0)
 turtle.begin_fill()
 rombusz(h)
 turtle.end_fill()

def rombusz(h):
 for i in range(2):
  turtle.forward(h)
  turtle.right(36)
  turtle.forward(h)
  turtle.right(144)

def otszog(h, s):
 turtle.fillcolor(s)
 turtle.begin_fill()
 for i in range(5):     turtle.for-
ward(h)
  turtle.right(72)
 turtle.end_fill()
```

turtle.speed(0)
penrose(50)
Elérhető: *https://repl.it/@hbv/penrose*

Feladat Egy mozaikot színes rombuszokból építünk össze `rombusz(h,s)`, amelynek oldalhossza `h`, kisebbik szöge pedig `s` fokos. A `sor(m,h,s)` eljárás `m` darab, különböző színű rombuszból álló sort rajzol, a `mozaik(n,m,h,s)` eljárás pedig `n` darab sort helyez egymás mellé úgy, hogy a rombuszok színe átlósan egyforma legyen! Legvégül pedig az `fmozaik(n,m,h,s)` eljárás a mozaikot elforgatja `s` fokonként, amíg körbe nem érünk. A színek tetszőleges sorrendben követhetik egymást!

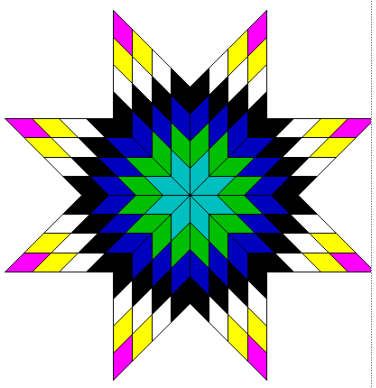
Rombusz(10, 45)

```
import turtle
turtle.speed(0)
turtle.pencolor("black")
```



Sor(4, 5, 45)

Mozaik(4 ,4, 5, 45)



Fmozaik(4, 4, 5, 45)

```
szinek=("blue","lightblue","green","white","yel-
low","red","black","lightgreen","gray","orange",
"purple","pink","brown","darkg-
reen","darkblue","silver")

def rombusz(h,s):
 turtle.begin_fill()
 for i in range(2):
   turtle.forward(h)
   turtle.right(s)
   turtle.forward(h)
   turtle.right(180-s)
 turtle.end_fill()

def sor(n,m,h,s):
 turtle.fillcolor(szinek[(m+n) % 15])
 rombusz(h,s)
 if m>1:
   turtle.right(s)
   turtle.forward(h)
   turtle.left(s)
   sor(n,m-1,h,s)
   turtle.right(s)
   turtle.backward(h)
   turtle.left(s)

def mozaik(n,m,h,s):
 sor(n,m,h,s)
 if n>1:
   turtle.forward(h)
   mozaik(n-1,m,h,s)
   turtle.backward(h)

def fmozaik(n,m,h,s):
 for i in range(360/s):
   mozaik(n,m,h,s)
   turtle.right(s)

fmozaik(4,4,20,45)
```

Elérhető: *https://repl.it/@hbv/forgo*

Sorminták

Feladat Gyöngyökből láncot fűzünk. Ehhez készítsd el a `gyöngy` eljárást! A gyöngyöket fesd színesre! Írd meg a `lánc(n)` eljárást!

Megjegyzés:Mivel más nyelvekkel ellentétben a színezéshez nem egy zárt alakzat belsejébe kell eljutni és onnan indítani a színkitöltést, hanem körbe kell azt rajzolni, problémás lehet, ha az alakzatot nem egyetlen vonallal lehet megrajzolni. A gyöngy kitöltésénél azt a trükköt alkalmaztuk, hogy először a külső négyzetet pirosra, majd a belsőt fehérre színeztük!

gyöngy lánc(3) lánc(6)

```
import turtle,math

def gyongy():
 turtle.pendown()
 negyzet(30,"red")
 turtle.penup()
 turtle.right(45)
 turtle.forward(30/4*math.sqrt(2))
 turtle.left(45)
 turtle.pendown()
 negyzet(15,"white")
 turtle.penup()
 turtle.right(45)
 turtle.backward(30/4*math.sqrt(2))
 turtle.left(45)
 turtle.pendown()

 def negyzet(hossz,szin):
   turtle.fillcolor(szin)
   turtle.begin_fill()
   for i in range(4):
     turtle.forward(hossz)
     turtle.right(90)
   turtle.end_fill()

lanc(5)
```

```
def lanc(db):
 turtle.right(90)
 for i in range(db-1):
    turtle.left(45)
    gyongy()
    for i in range(2):
        turtle.forward(30)
        turtle.right(90)
    turtle.left(135)
    turtle.forward(30)
 turtle.left(45)
 gyongy()
 turtle.right(45)
 turtle.penup()
 turtle.backward((db-
1)*30*(1+math.sqrt(2)))
 turtle.pendown()
 turtle.left(90)

turtle.speed(0)
turtle.color("black")
turtle.left(90)
#gyongy()
```

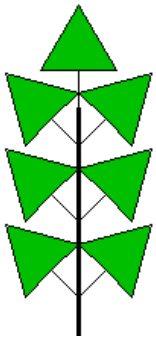
Elérhető: *https://repl.it/@hbv/lanc*

Feladat Rajzold le a mesebeli égig érő paszulyt! Készítsd el először az ábra szerinti levél eljárást `levél(méret)`, ahol a levelet egy szabályos háromszögből állíthatod elő, ahol a `méret` a háromszög oldalhossza! Színezd is ki! Írd meg a paszuly rajzoló eljárást is `paszuly(db,méret)`, ahol a `db` paraméter a levélpárok számát adja meg, a `méret` paraméter pedig a levél nagyságát és a levélpárok távolságát határozza meg! A paszuly szára legyen háromszor olyan vastag, mint a levél szára!

Level(50)

```
import turtle

def level(r):
 turtle.forward(r/2)
 turtle.left(90)
```

Paszuly(1, 50)



Paszuly(3, 50)

```
 turtle.forward(r/2)
 turtle.right(120)
 turtle.begin_fill()
 for i in range(2):
  turtle.forward(r)
  turtle.right(120)
 turtle.end_fill()
 turtle.forward(r/2)
 turtle.right(90)
 turtle.backward(r/2)

 def paszuly(n,r):
 turtle.pensize(3)
 turtle.forward(n*r)
 turtle.backward(n*r)
 turtle.pensize(1)
 for i in range(n):
   turtle.forward(r/2)
   turtle.right(45)
   level(r )
   turtle.left(90)
   level(r)
   turtle.right(45)
   turtle.forward(r/2)
 level(r)
 turtle.backward(n*r)

turtle.speed(0)
turtle.left(90)
turtle.pencolor("black")
turtle.fillcolor("green")
paszuly(3,50)
```

Elérhető: *https://repl.it/@hbv/paszuly*

Feladat Egy zongorán fekete és fehér billentyűk vannak. A fekete billentyűk 2-es és 3-as csoportokban helyezkednek el. Készíts `zongora(db)` eljárást, amely `db` 2-es és 3-as csoportot tartalmazó zongorabillentyűzetet rajzol!

Megjegyzés: A zongora kirajzolásához készítsük el a különböző típusú billentyűket!

```
import turtle

def zongora(db):
 turtle.pendown()
 turtle.right(90)
 for i in range(db):
     harmas()
     negyes()
 turtle.penup()
 pozic_jobbra(-db*5*14 )
 turtle.pendown()
```

```
def jobbfeher():
 elore_jobbra([50, 7])
 turtle.forward(30)
 turtle.left(90)
 elore_jobbra( [3, 20, 10])

def kettosfeher():
 turtle.forward(20)
 pozic_jobbra(3)
 turtle.forward(30)
 turtle.right(90)
 turtle.forward(4)
 pozic_jobbra(30)
 elore_jobbra([3, 20, 10])
```

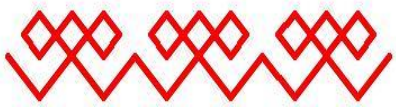
```
def harmas():
 jobbfeher()
 fekete()
 kettosfeher()
 fekete()
 balfeher()
 pozic_jobbra(10)

def negyes():
 jobbfeher()
 fekete()
 kettosfeher()
 fekete()
 kettosfeher()
 fekete()
 balfeher()
 pozic_jobbra(10)

def fekete():
 pozic_jobbra(10)
 turtle.forward(20)
 turtle.left(90)
 turtle.forward(3)
 turtle.right(90)
 turtle.begin_fill()
 elore_jobbra( [30, 6, 30, 6])
 turtle.end_fill()
 pozic_jobbra(3)
 turtle.backward(20)
```

```
def balfeher():
 turtle.forward(20)
 pozic_jobbra(3)
 elore_jobbra([30,7,50,10])

# kód rövidítése, gyakran előfor-
duló esetek

def pozic_jobbra(hossz)
 turtle.right(90)
 turtle.forward(hossz)
 turtle.left(90)

def elore_jobbra(hosszak):
 for h in hosszak:
     turtle.forward(h)
     turtle.right(90)

turtle.speed(0)
turtle.color("black","black")
zongora(6)
```

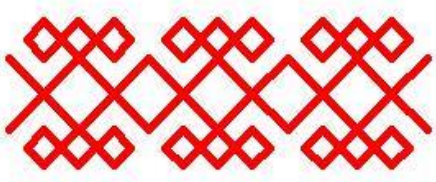
Elérhető: *https://repl.it/@hbv/zon-gorabill*

Feladat Készíts eljárást az ábrákon látható írásos hímzés sorminta kirajzolására `sorminta(db,méret)` néven! Vedd észre, hogy a sorminta egy egyszerűbb minta `sormintaA(db,méret)` tükrözésével keletkezik (negatív értékkel használjuk a `méret` paramétert)! Írd meg az `alapminta :méret` kirajzolását is!

`alapminta(50)`



`sormintaA(3,50)`



`sorminta(3,50)`

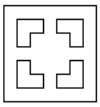
```
turtle.left(90)
 turtle.forward(meret)
 turtle.left(90)
 turtle.forward(meret)
 turtle.left(90)
 turtle.forward(meret*4)
 turtle.right(90)
 turtle.forward(2*meret)
 turtle.left(90)

def sormintaA(db,meret):
 for i in range(db):
    alapminta(meret)

def sorminta(db,meret):
 sormintaA(db,meret)
 turtle.penup()
 turtle.right(90)
 turtle.backward(2*meret)
 turtle.right(90)
 turtle.backward(2*meret)
 turtle.left(180)
 turtle.pendown()
```

```
import turtle, math

def alapminta(meret):
 turtle.forward(2*meret)
 turtle.right(90)
 turtle.forward(4*meret)
 turtle.left(90)
 turtle.forward(meret)
 turtle.left(90)
 turtle.forward(meret)
 turtle.left(90)
 turtle.forward(meret*2)
 turtle.right(90)
 turtle.forward(2*meret)
 sormintaA(db,-1*meret)
 turtle.penup()
 turtle.backward(2*meret)
 turtle.right(90)
 turtle.forward(2*meret)
 turtle.left(90)
 turtle.pendown()

turtle.speed(0)
turtle.pencolor("red")
turtle.pensize(4)
turtle.left(45)
sorminta(5,10)
```

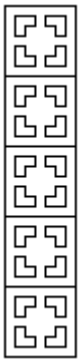
Elérhető: *https://repl.it/@hbv/irasos*

Mozaik – sorminták egymás fölé

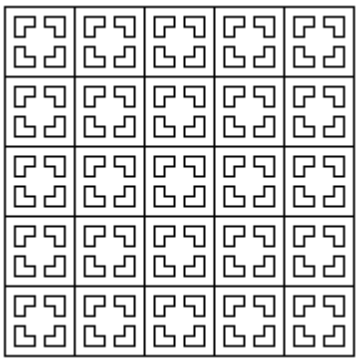
Feladat Az ábrán látható csempével szeretnénk egy falat kicsempézni. Készíts eljárást a csempe rajzolására `csempe(méret)`, ahol `méret` a négyzet alakú csempe oldalhossza), valamint hosszú és négyzetes falak csempézésére `hfal(db,méret)`, `nfal(db,méret)`! A `csempe` egy nagyobb négyzet és egy kisebb négyzet sarkain levő négy szabályos alakzat:



`csempe(50)`



`hfal(5,35)`



`nfal(5,35)`

```
def nfal(db,meret):
 for i in range(db):
     hfal(db, meret)
     turtle.right(90)
     turtle.forward(meret*7)
     turtle.left(90)
 turtle.right(90)
 turtle.backward(db*meret*7 )
 turtle.left(90)
```

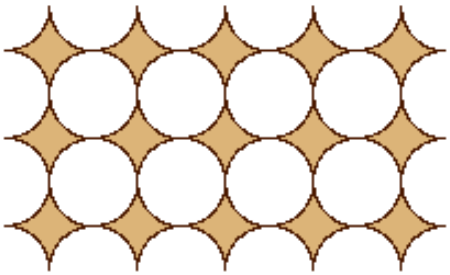
```
import turtle

def csempe(meret):
 for i in range(4):
      turtle.forward(meret*7)
      turtle.right(90)
      turtle.penup()
      turtle.forward(meret)
      turtle.right(90)
      turtle.forward(meret)
      turtle.left(90)
      turtle.pendown()
      turtle.forward(meret*2)
      turtle.right(90)
      turtle.forward(meret)
      turtle.right(90)
      turtle.forward(meret)
      turtle.left(90)
      turtle.forward(meret)
      turtle.right(90)
      turtle.forward(meret)
      turtle.right(90)
      turtle.forward(meret*2)
      turtle.penup()
      turtle.forward(meret)
      turtle.right(90)
      turtle.backward(meret)
      turtle.pendown()

def hfal(db, meret):
 for i in range(db):
     csempe(meret)
     turtle.forward(meret*7
 turtle.backward(db*meret*7)

turtle.speed(0)
turtle.pencolor("black")
#csempe(35)
#hfal(5,35)
nfal(5,35)
```

Elérhető:
https://repl.it/@hbv/csempe

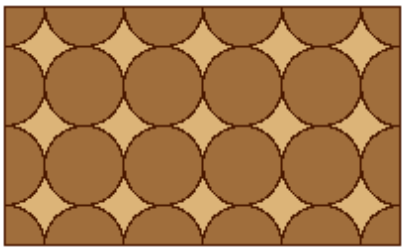
Feladat Egy régi épület padlóját mozaikminta díszíti. Készíts eljárást `padló(m,n,h)` néven, amely ilyen mintát rajzol!

Megjegyzés: Egyszerűbb a megoldás, ha először megrajzoljuk a háttért adó színezett téglalapot, majd rá a mintákat!

alap(50)



mozaik(3,5,20)



padló(3,5,20)

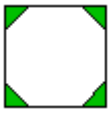
```
def padlo(m,n,h):
 turtle.color("black","brown")
 turtle.penup()
 turtle.begin_fill()
 for i in range(2):
  turtle.forward(n*h)
  turtle.right(90 )
  turtle.forward(m*h)
  turtle.right(90)
 turtle.end_fill()
 turtle.left(90)
 turtle.backward(h/2)
 turtle.right(90)
 turtle.pendown()
 mozaik(m,n,h)
```

```
import turtle

def alap(h):
 turtle.fillcolor("orange")
 turtle.begin_fill()
 for i in range(4):
     turtle.circle(h/2,90)
     turtle.right(180)
 turtle.end_fill()

def mozaik(m,n,h):
 for i in range(n):
  sor(m,h)
  turtle.penup()
  turtle.forward(h)
  turtle.pendown()
 turtle.penup()
 turtle.backward(n*h)
 turtle.pendown()

def sor(m,h):
 for i in range(m):
     alap(h)
     turtle.penup()
     turtle.right(90)
     turtle.forward(h)
     turtle.left(90)
     turtle.pendown()
 turtle.penup()
 turtle.right(90)
 turtle.backward(m*h)
 turtle.left(90)
 turtle.pendown()

turtle.speed(0)
padlo(3,5,40)
```

Elérhető:

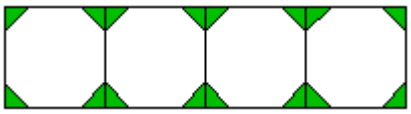
https://repl.it/@hbv/padlo

Feladat Egy fürdőszoba falat a mellékelt mintázatú csempével szeretnénk befedni. Az alsó és a felső csempesort mindkét oldaláról egy-egy zöld szegély szegély csempesor határolja.Készíts eljárásokat a fal és az egyes alapelemek rajzolására! A `csempe(hossz)` eljárás egyetlen csempét rajzoljon, amely egy `hossz` oldalhosszúságú négyzet, melynek sarkaiban zöld derékszögű háromszögek vannak (befogójuk az oldalhossz negyede)! A `csempesor(db,hossz)` eljárás egy sort rajzol, `db` darab `hossz` méretű csempéből. A `szegély(hossz)` eljárás egy `hossz` szélességű, negyedannyi magasságú zöld téglalapot rajzol, a `szegélysor(db,hossz)` pedig egy `db` darab `hossz` méretű vonalból álló sort. A `fal(n,m,hossz)` eljárás `n` csempesort rajzol, amely `m` darab `hossz` méretű csempéből áll.



```
import turtle,math
```

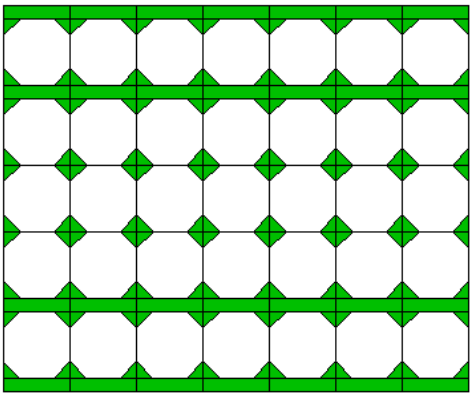
csempe(50)



csempesor(4,50)

szegély(50)

szegélysor(4,50)



fal(5,7,40)

```
def szelso(m,hossz):
 szegelysor(m,hossz)
 turtle.forward(hossz/5)
 csempesor(m,hossz)
 turtle.forward(hossz)
 szegelysor(m,hossz)
 turtle.forward(hossz/5)

def fal(n,m,hossz):
 szelso(m,hossz)
 for i in range(n-2):
    csempesor(m,hossz)
    turtle.forward(hossz)
 szelso(m,hossz)
 turtle.backward(n*hossz+4*hossz/5)

turtle.speed(0)
turtle.left(90)
turtle.color("black"," green")
fal(5,7,40)
```

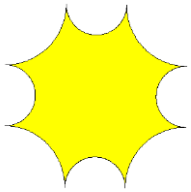
```
def csempe(h):
 for i in range(4):
     alap(h/4)
     turtle.forward(h)
     turtle.right(90)

def alap(h):
 turtle.begin_fill()
 turtle.forward(h)
 turtle.right(135)
 turtle.forward(h*math.sqrt(2))
 turtle.right(135)
 turtle.forward(h)
 turtle.right(90)
 turtle.end_fill()

def csempesor(csdb,hossz):
 for i in range(csdb):
    csempe(hossz)
    turtle.right(90)
    turtle.forward(hossz)
    turtle.left(90)
 turtle.right(90)
 turtle.backward(csdb*hossz)
 turtle.left(90)

def szegely(h):
 turtle.begin_fill()
 for i in range(2):
    turtle.forward(h/5)
    turtle.right(90)
    turtle.forward(h)
    turtle.right(90)
 turtle.end_fill()

def szegelysor(csdb,hossz):
 for i in range(csdb):
    szegely(hossz)
    turtle.right(90)
    turtle.forward(hossz)
    turtle.left(90)
 turtle.right(90)
 turtle.backward(csdb*hossz)
 turtle.left(90)
```

Elérhető: *https://repl.it/@hbv/fur-docsempe*

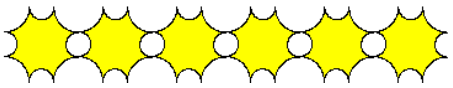
Feladat A középkorban díszes ablakokat csak nagyon kicsi üvegtáblákból tudtak kirakni. Egy lehetséges üvegtábla például nyolcszögletű, negyed- és félkörökkel határolt alakzat. Ebből egymás mellé helyezhetünk `M` darabot, így kialakul egy üvegtábla sor. A sorokból egymás fölé helyezhetünk `N` darabot, így kialakul az üvegablak. Készíts eljárásokat `alap`, `sor`, `üveg` néven, amelyek az alábbi ábrákat rajzolják!

Megjegyzés: Az alapelemünk negyed- és félkörívekből áll, amit a megrajzolás után a paraméterként kapott színnel befestünk. Az alapelemünk negyed- és félkörívekből áll, amit a megrajzolás után a paraméterként kapott színnel befestünk. A feladathoz látszólag kétféle sor kell. De igazából egy is

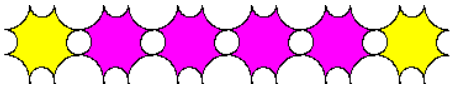
elég, ha a sor egy alapelemből, utána egy m-2 hosszú sorból és még egy alapelemből áll. A két szélső elem színe legyen sárga, a többié pedig a paraméterként kapott szín!



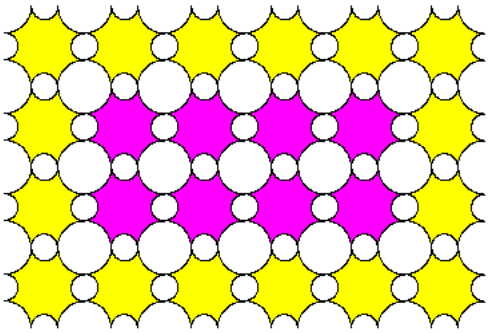
alap(100,"yellow")



sor(6,20,"yellow","yellow")



sor(6,20,"yellow","magenta")



üveg(6,4,20)

```
def uveg( n, m, h):
 sor(m,h,"yellow","yellow")
 turtle.penup()
 turtle.forward( 3*h)
 turtle.pendown()
 for i in range( n-2):
  sor(m,h,"yellow","magenta")
  turtle.penup()
  turtle.forward( 3*h)
  turtle.pendown()
 sor(m,h,"yellow","yellow")
 turtle.penup()
 turtle.forward( -3*h*n)
 turtle.pendown()
```

```
import turtle

def alap( h, szin):
 turtle.fillcolor(szin)
 turtle.begin_fill()
 for i in range( 4):
  turtle.right( 90)
  turtle.circle(h/2,180)
  turtle.right( 180)
  turtle.circle( h,90)
  turtle.right( 90)
 turtle.end_fill()

def sor(m,h,szin1,szin2):
 alap(h,szin1)
 eltol(h)
 for i in range( m-2):
   alap(h, szin2)
   eltol(h)
 alap(h,szin1)
 eltol(h)
 turtle.penup()
 turtle.left( 180)
 eltol( m*h)
 turtle.right( 180)
 turtle.pendown()

def eltol( h):
 turtle.penup()
 turtle.right( 90
 turtle.forward( 3*h)
 turtle.left( 90)
 turtle.pendown()

turtle.speed(0)
uveg(6,4,20)
```

Elérhető:

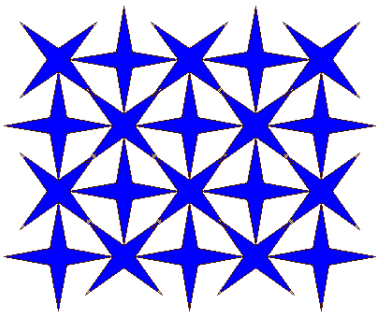
https://repl.it/@hbv/sargaszelu

Feladat Írj eljárásokat a `h` oldalhosszúságú, `20` fokos szögű 4-ágú festett csillag, valamint azokból készült alakzatok megrajzolására `csillag(h,melyik)`, `csillagsor(o,h,melyik)` és `minta(s,o,h)`, ahol `s` a minta sorai, `o` pedig az oszlopai száma!

Megjegyzés: Ez is négyzet alapú mozaik lesz, de a négyzetek egymáshoz képest 45 fokkal el vannak forgatva, emiatt át is fedik egymást. De ezzel nem kell törődnünk, a négyzet helyére kerülő csillagok jól fognak érintkezni. Kétféle elemet tartalmazó mozaik esetén az egyik lehetséges hozzáállás kétféle sort definiálunk és felváltva hívjuk őket. A soron belül kétféle elemet – most ugyanazt az elemet forgatva – definiálunk és ezt felváltva meghívjuk. A csillagot a közepéből kiindulva rajzoljuk!

csillag(20,0)

csillagsor(4,20,0)



minta(4,5,20,0)

```
def minta(n,m,h):
 turtle.penup()
 for i in range( n):
     csillagsor( m,h,i % 2)
     turtle.for-
ward(h*math.sqrt(3))
 turtle.pendown()
```

```
import turtle,math

def csillag(h,melyik):
 x=math.sqrt(3)*math.sqrt(2)/2
 if melyik%2==1:
     turtle.right(45)
 turtle.backward(h*x)
 turtle.pendown()
 turtle.left( 15)
 turtle.begin_fill()
 for i in range( 4 ):
     turtle.forward( h)
     turtle.left( 60)
     turtle.forward( h)
     turtle.right( 150)
 turtle.end_fill()
 turtle.right(15)
 turtle.penup()
 turtle.forward(h*x)
 if melyik%2==1:
      turtle.left(45)

def csillagsor( o, h, melyik):
 for i in range(o):
    csillag( h,i+melyik)
    turtle.right(90)
    turtle.forward(h*math.sqrt(3))
    turtle.left(90)
 turtle.right(90)
 turtle.backward(h*math.sqrt(3)*o)
 turtle.left(90)

turtle.speed(0)
turtle.left(90)
turtle.fillcolor("blue")
#csillag(20,0)
#csillagsor(4,20,0)
minta(4,5,20)
```

Elérhető: *https://repl.it/@hbv/kekcsillagok*

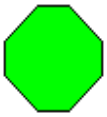
Feladat Egy mozaikot háromféle alapelemből építünk fel `alapelem1(h)`, `alapelem2(h)`, `alapelem3(h)`, ahol `h` az alapelemek köré írható négyzet oldalának hossza. Az alapelemek sorokba rendezhetők `sor1(m,h)`, `sor2(m,h)`, `sor3(m,h)`, ahol a sorok `m*2+3` darab alapelemet tartalmaznak. A sorok felépítése az ábrán látható. Sorok alkalmas egymás mellé helyezésével készíts mozaikot `mozaik(m,h)`, amelynek belsejében `m*2+3` sorban soronként `m` piros négyzet található!

Megjegyzés: Ebben a mozaikban háromféle sort kell rajzolnunk! Ha a kék alaptéglalapot előre kirajzoljuk, akkor a festése sokkal egyszerűbb.

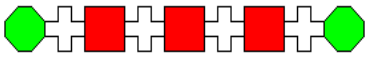
alapelem1(100)

```
import turtle,math

def alapelem1(a):
 turtle.fillcolor("red")
 turtle.begin_fill()
 for i in range(4):
     turtle.forward(a)
     turtle.right(90)
 turtle.end_fill()
```

alapelem2(100)

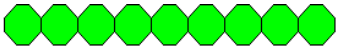


alapelem3(100)

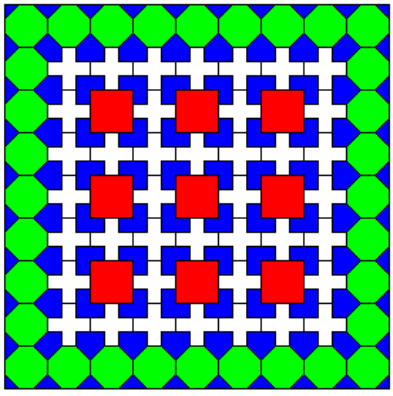


sor1(3,30)

sor2(3,30)



sor3(3,30)



```
def mozaik(n,a):
 kekalap((n*2+3)*a)
 sor3(n,a)
 elore(a)
 sor2(n,a)
 elore(a)
 for i in range(n):
     sor1(n,a)
     elore(a)
     sor2(n,a)
     elore(a)
```

```
def alapelem2(a):
 turtle.fillcolor("white")
 elore(a/3)
 turtle.begin_fill()
 for i in range(4):
    turtle.forward(a/3)
    turtle.right(90)
    turtle.forward(a/3)
    turtle.left(90)
    turtle.forward(a/3)
    turtle.right(90)
 turtle.end_fill()
 elore(-a/3)
```

```
def alapelem3(a):
 elore(a/3)
 turtle.fillcolor("green")
 turtle.begin_fill()
 for i in range(4):
    turtle.forward(a/3)
    turtle.right(45)
    turtle.forward(a/3*math.sqrt(2))
    turtle.right(45)
 turtle.end_fill()
 elore(-a/3)
```

```
def sor1(n,a):
 alapelem3(a)
 eltol(a)
 for i in range(n):
     alapelem2(a)
     eltol(a)
     alapelem1(a)
     eltol(a)
 alapelem2(a)
 eltol(a)
 alapelem3(a)
 eltol(-(n*2+2)*a)
```

```
def sor2(n,a):
 alapelem3(a)
 eltol(a)
 for i in range(n*2+1):
     alapelem2(a)
     eltol(a)
 alapelem3(a)
 eltol(-(n*2+2)*a)
```

```
def sor3(n,a):
 for i in range(n*2+3):
     alapelem3(a)
     eltol(a)
 eltol(-(n*2+3)*a)
```

```
def eltol(a):
 turtle.penup()
 turtle.right(90)
 turtle.forward(a)
 turtle.left(90)
 turtle.pendown()
```

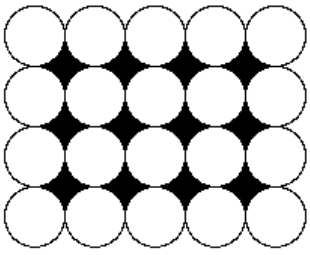
```
def elore(a):
 #felemelt tollal előre
 turtle.penup()
```

```
  sor3(n,a)

def kekalap(h):
 turtle.fillcolor("blue")
 turtle.begin_fill()
 for i in range(4):
     turtle.forward(h)
     turtle.right(90)
 turtle.end_fill()
 turtle.forward(a)
 turtle.pendown()

turtle.speed(0)
turtle.left(90)
mozaik(3,30)
```

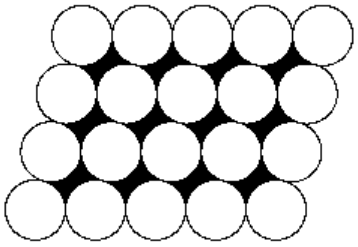
Elérhető:

https://repl.it/@hbv/vegyesmozaik

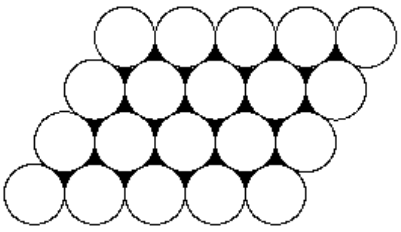
Feladat Egy mozaikot `r` sugarú körökből építünk fel. Megadjuk a sorok és oszlopok számát, valamint azt, hogy egy sor első köréhez képest hány fokkal van eltolva a következő (0 és 30 közötti szám). Az eltolásnál a sorokat, amennyire lehet, összecsúsztatjuk. A körök közötti részeket befestjük. Készítsd el a `mozaik(m,n,r,fok)` eljárást, ami az alábbi ábrákat rajzolja! A `sor(n,r)` `eljárás` pedig egy sort rajzol `n` darab `r` sugarú körből.



mozaik 4 5 20 0



mozaik 4 5 20 15



mozaik 4 5 20 30

```
def hatter(m,n,r,szog):
 turtle.fillcolor("black")
 turtle.right(szog)
 turtle.begin_fill()
 for i in range(2):
    turtle.forward((m-1)*2*r)
    turtle.right(90-szog)
    turtle.forward((n-1)*2*r)
    turtle.right(90+szog)
 turtle.end_fill()
 turtle.left(szog)
```

```
import turtle

def kozeppontkor(r):
    turtle.backward(r)
    turtle.right(90)
    turtle.pendown()
    turtle.begin_fill()
    turtle.circle(r)
    turtle.end_fill()
    turtle.penup()
    turtle.left(90)
    turtle.forward(r)

def korsor(n, r):
 for i in range(n):
     kozeppontkor(r)
     turtle.right(90)
     turtle.forward(2*r)
     turtle.left(90)
 turtle.right(90)
 turtle.backward(2*n*r)
 turtle.left(90)

def kormozaik(m,n,r,szog):
 hatter(m,n,r,szog)
 turtle.color("black","white")
 for i in range(m):
    korsor(n,r)
    turtle.right(szog)
    turtle.forward(2*r)
    turtle.left(szog)
 turtle.right(szog)
 turtle.backward(m*r*2)
 turtle.left(szog)

turtle.speed(0)
turtle.penup()
turtle.left(90)
kormozaik(3,4,20,30)
```

Elérhető:

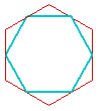
https://repl.it/@hbv/korosmozaik

Ásványok – molekulák – kristályok

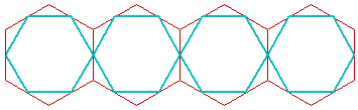
Egyes ásványokban (pl. a szilikátok ilyenek) hatszög alapú rácsba rendeződnek el az atomok. Ha a térbeli rácsot síkban ábrázoljuk, akkor két egybefonódó hatszögrácsot látunk, ahol a belső hatszög oldalhossza a külső hatszög oldalhosszának `gyök(2)/3`-szorosa.

Feladat Készíts `szilikát(n,m,h)` eljárást a szilikát kirajzolására, ahol a szilikátnak `2*n` sora és `m` oszlopa van, a nagyobb hatszög oldalhossza pedig `h`. Részfeladatként írj egy `sor(m,h)` eljárást, amely egysoros, `m` oszlopos szilikátot rajzol! A nagyobb hatszögek piros, a kisebbek pedig kétszeres vonalvastagságú, kék színűek legyenek!

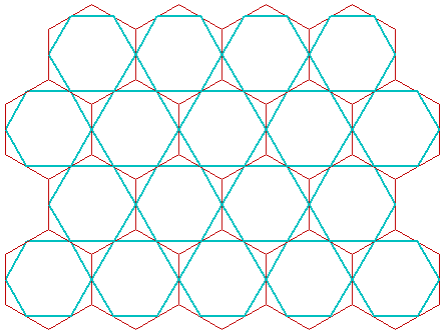
Megjegyzés: A sormintában balról jobbra(megrajzoljuk a piros, majd jobbról balra(a kék hatszögeket. Mivel a szilikátnak páros számú sora van és minden második sor eggyel kevesebb alapelemből áll, érdemes a mozaikot rajzoló eljárás dupla sorokra megírni. Itt célszerű (legalábbis a `dupla` eljárásnál) eltekinteni az állapotátlátszóságtól.



`sor(1,50)`



`sor(4,50)`



`szilikát(2,5,50)`

```
def szilikat(n,m,h):
 for i in range(n):
     dupla(m, h)
 turtle.penup()
 turtle.backward(3*n)

def dupla(m, h):
 turtle.left(60)
 turtle.forward(h)
 turtle.right(60)
 sor(m, h)
 turtle.right(60)
 turtle.forward(h)
 turtle.left(60)
 sor(m-1,h)
```

```
import turtle,math

def sor(m, h):
 turtle.pensize(1)
 turtle.color("red")
 for i in range(m):
     hatszog(h)
     for i in range(4):
         turtle.forward(h)
         turtle.right(60)
     turtle.left(240)
 turtle.left(180)
 turtle.backward(h/2)
 turtle.right(30)
 turtle.pensize(2)
 turtle.color("blue")
 for i in range(m):
     hatszog(h/2*math.sqrt(3))
     for i in range(3):
         turtle.forward(h/2*math.sqrt(3))
         turtle.right(60)
     turtle.left(180)
 turtle.pensize(1)
 turtle.right(150)
 turtle.forward(h/2)

def hatszog(h):
 turtle.pendown()
 for i in range(6):
    turtle.forward(h)
    turtle.right(60)
 turtle.penup()

turtle.speed(0)
turtle.left(90)
szilikat(2,5,30)
```

Elérhető:

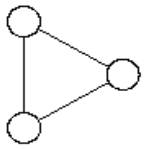
https://repl.it/@hbv/szilikat

A kristályok struktúráját sok esetben gráffal ábrázolják. A gráf csomópontokból és azokat valamilyen szabályszerűséggel összekötő élekből áll. A legtöbb esetben a gráf többféle csomópontot tartalmaz, azaz legalább kétféle alapelem rajzolására lesz szükségünk.

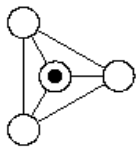
A legegyszerűbb esetekben a gráf egy sokszög, illetve elemek sokszögek csúcsaiba vagy oldalaira elhelyezve.

Feladat Egy szilikát ásvány (Si_2O_7) háromszög alakban elhelyezkedő 3 oxigénatomból (10 sugarú kör), valamint térben egymás fölött elhelyezkedő egy szilícium (10 méretű fekete pötty) és egy oxigénatomból álló pár összekapcsolódásából (van egy közös oxigén) áll. Készíts eljárásokat az alábbi ábrák megrajzolására, ahol a hosszabb vonal (kör középpontjától a másik kör középpontjáig) `h` hosszúságú, a rövidebb pedig ennek `(gyök 3)/3`-szorosa!

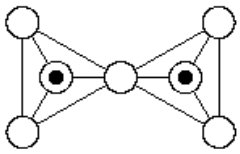
Megjegyzés: Először rajzoljuk meg az oxigént (`o`) és a szilícium-oxigén párt (`sio`)! Az élen a körökön belül felemelt tollal, a körök között leengedett tollal (rajzolva) megyünk végig. Az egyik def az él végpontjában marad (`él`), a másik def állapotátlátszó (`élaa`).



három(100)



alap(100)



szilikat(100)

```
def kor(r):
    turtle.backward(r)
    turtle.right(90)
    turtle.pendown()
    turtle.circle(r)
    turtle.penup()
    turtle.left(90)
    turtle.forward(r)

def szilikat(h):
 turtle.left(120)
 alap(h)
 turtle.right(180)
 alap(h)
 turtle.left(60)

turtle.speed(0)
turtle.penup()
#sio()
#alap(100)
szilikat(100)
```

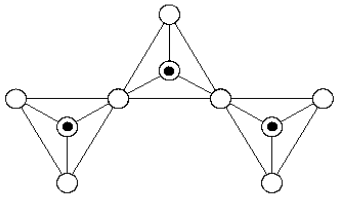
```
import turtle,math

def o():
 kor(10)

def sio():
 turtle.pendown()
 turtle.dot(5)
 kor(10)
 turtle.penup()

def harom(h):
 for i in range(3):
     o()
     el(h)
     turtle.right(120)

def el(h):
 turtle.forward(10)
 turtle.pendown()
 turtle.forward(h-20)
 turtle.penup()
 turtle.forward(10)

def elaa(h):
 el(h)
 turtle.backward(h)

def alap(h):
 harom(h)
 turtle.right(30)
 el(h/3*math.sqrt(3))
 sio()
 turtle.left(60)
 for i in range(2):
  elaa(h/3*math.sqrt(3))
  turtle.right(120)
 turtle.forward(h/3*math.sqrt(3) )
 turtle.right(150)
```

Elérhető:

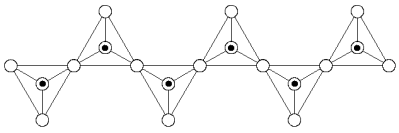
https://repl.it/@hbv/grafszilikat

Feladat Polimer szilikátok szerkezetét mutatják az alábbi rajzok. Egy alapelem 4 molekulából áll az ábra szerinti elrendezésben. (lsd. az előzőfeladatot!) Készítsd el az `alap(h)`, a `piroxén(n,h)` és az `amfiból(n,h)` `eljárásokat`, ahol `n` az alapelemek száma. (Az amfiból mindkét sorában (`n` biztosan páratlan)!

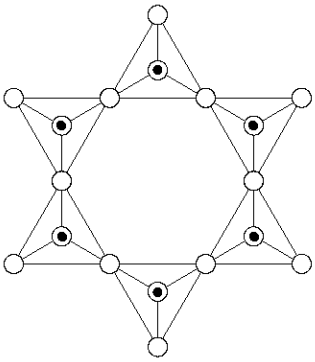
Megjegyzés az amfiból két sor piroxénből áll, azaz ha a piroxén rajzolás nem lenne állapotátlátszó, akkor az első piroxén után 180 fokos elfordulással rajzolhatnánk a másodikat és az amfiból így elkészülne.



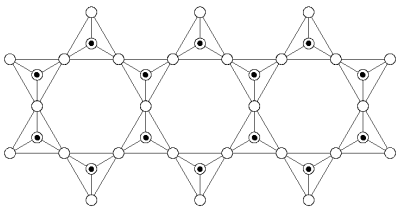
`piroxén(3,50)`



`piroxén(6,50)`



`amfiból(3,50)`



`amfiból(7,50)`

```
def piroxen(n,h):
 turtle.left(30)
 for i in range(n//2):
  turtle.right(60)
  turtle.forward(2*h)
  turtle.right(120)
  alap(h) #előző feladatból
  turtle.forward(2*h)
  turtle.right(180)
  alap(h)
 turtle.right(60)
 turtle.forward(2*h)
 turtle.right(120)
 if n%2==1:
     alap(h)
     turtle.left(60)
     turtle.forward((n-1)*h)
 else:
   turtle.left(60)
   turtle.forward(n*h)
 turtle.right(60)
def amfibol(n,h):
 turtle.right(30)
 for i in range(n//2):
  alap(h)
  turtle.right(60)
  turtle.forward(h)
  alap(h)
  turtle.forward(h)
  turtle.left(60)
 alap(h)
 turtle.forward(2*h)
 turtle.right(180)
 for i in range(n//2):
   alap(h)
   turtle.right(60)
   turtle.forward(h)
   alap(h)
   turtle.forward(h)
   turtle.left(60)
 alap(h)
 turtle.right(60)
turtle.speed(0)
turtle.penup()
turtle.left(90)
#piroxen(7,50)
amfibol(7,50)
```

Elérhető:

https://repl.it/@hbv/polimerek

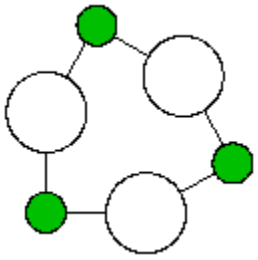
Feladat A Béta-kvarc rácsstruktúrája síkra vetíthető. Ebben az ásványban egy szilíciumatom négy oxigénatomhoz, illetve minden oxigénatom két szilíciumatomhoz kapcsolódik.

Készíts `bétakvarc1(r)`, `bétakvarc2(r)` eljárásokat a Béta-kvarc megrajzolására! Ehhez a következő eljárásokat használd!

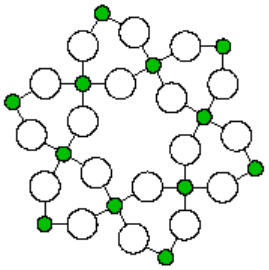
`szilícium(r)`	`r` sugarú, zöld színű kör a szilíciumatom képének;
`oxigén(r)`	`r` sugarú, üres kör az oxigénatom képének;
`alap(r)`	3 oxigén- és 3 szilíciumatomból álló struktúra, a szilíciumatom `r`, az oxigénatom pedig `2*r` sugarú, a köztük levő kötések pedig `2*r` hosszúak;
`hatszög(r)`	6 alapelemből felépülő struktúra, hatszög alakban;
`bétakvarc1(r)`	a hatszög szélein levő szilíciumatomokból egy-egy újabb alapelem nő ki;
`bétakvarc2(r)`	a hatszög szélein levő szilíciumatomokból egy-egy újabb hatszög nő ki;

`szilícium(10)`

`oxigén(20)`

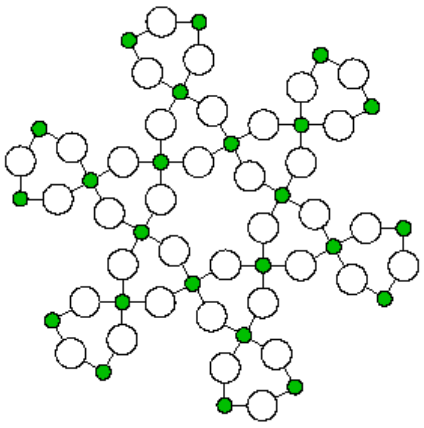


`alap(10)`

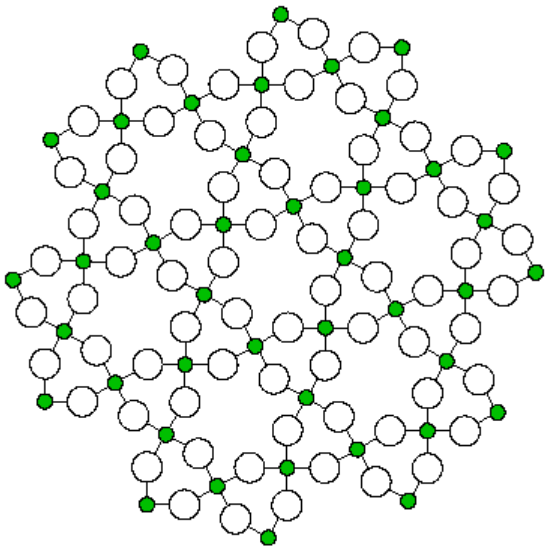


`hatszög(4)`

```
import turtle

def el(h,x,y):
 turtle.forward(x)
 turtle.pendown()
 turtle.forward(h)
 turtle.penup()
 turtle.forward(y)

def szilicium(r):
    turtle.fillcolor("green")
    turtle.begin_fill()
    kor(r) #szilikátos feladatból
    turtle.end_fill()

def oxigen(r):
    kor(r)

def alap(r):
 for i in range( 3):
     szilicium( r)
     el(2*r,r,2*r)
     oxigen( 2*r)
     turtle.right( 30)
     el(2*r,2*r,r)
     turtle.right( 90)

def hatszog( r):
 for i in range( 6):
     alap( r)
     turtle.forward(5*r)
     turtle.right(30)
     turtle.forward(5*r)
     turtle.left(180)
     turtle.right(90)
```



bétakvarc1(4)



bétakvarc2(4)

```
def betakvarc1( r):
 hatszog( r)
 turtle.right(90)
 for i in range( 6):
     turtle.forward(5*r)
     turtle.left(30)
     turtle.forward(5*r)
     alap(r)
     turtle.left(90)
     turtle.forward(5*r)
     turtle.left(30)
     turtle.forward(5*r)
     turtle.right(90)

def betakvarc2( r):
 for i in range(6):
     hatszog(r)
     turtle.right(180)
     turtle.forward(5*r)
     turtle.right(30)
     turtle.forward(5*r)
     turtle.left(90)
     hatszog(r)
     turtle.right(180)
     turtle.forward(5*r)
     turtle.right(30)
     turtle.forward(5*r)
     turtle.right(90)

turtle.speed(0)
turtle.penup()
turtle.left(90)
#hatszog(10)
#betakvarc1(10)
betakvarc2(5)
```

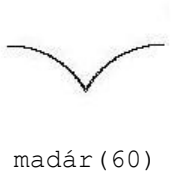
Elérhető:

https://repl.it/@hbv/betakvarc

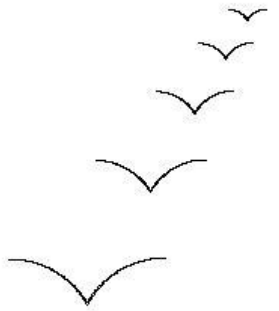
Rekurzió

Elsőre talán bonyolultnak tűnik a rekurzió használata, de a későbbiekben megtapasztaljuk, hogy sok olyan feladat van, amelyet sokkal egyszerűbben oldhatunk meg rekurzióval, mint iterációval!

Feladat Rajzold le az égen egymás után szálló madarakat `madarak(db,méret,arány)`. Az első a legnagyobb, majd a következő arányosan mindig kisebb. A `db` paraméter a madarak számát, a `méret` az első madár nagyságát, az `arány` pedig a kicsinyítés mértékét adja meg.



`madár(60)`



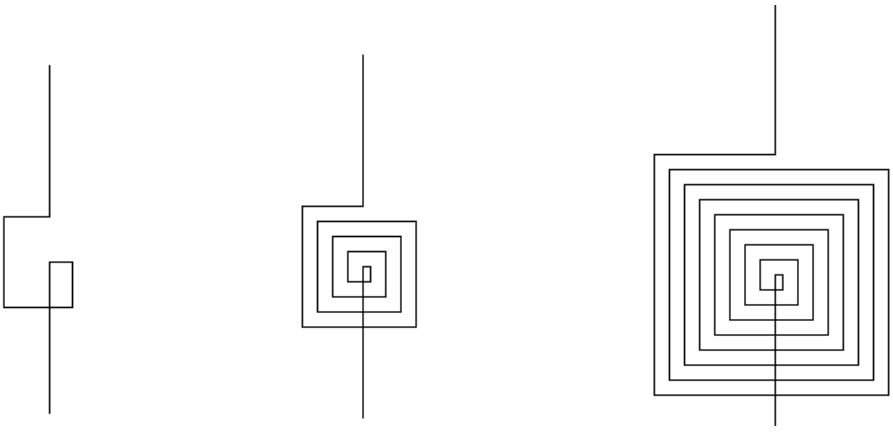
`madarak(5,60,0.9)`

```
import turtle

def madar(meret):
 turtle.left(30)
 turtle.circle(meret,60)
 turtle.circle(meret,-60)
 turtle.right(60)
 turtle.circle(-meret,60)
 turtle.circle(-meret,-60)
 turtle.left(30)

def madarak(db,meret,arany):
 if db>0:
     madar( meret)
     turtle.penup()
     turtle.right( 30)
     turtle.forward( meret*1.5)
     turtle.left( 30)
     turtle.pendown()
     madarak( db-1, meret*arany, arany)

turtle.left(90)
#madar(50)
madarak(4,100,0.8)
```

Elérhető: *https://repl.it/@hbv/madarak*

Feladat Egy drótszálat spirál alakban tekertek fel az ábrának megfelelő módon. Készíts eljárást `tekergő(db,h)` néven, amely egy `db`-szer feltekert drótot rajzol, amelynél a legrövidebb egyenes drótdarab hossza `h`, a drót két szélső darabja hossza pedig `10*h`!

Megjegyzés: A drót egyszeri körbetekerése 4–szer hívja meg a spirált, tehát minden teljes fordulat 4-szeres hívást jelent. A plusz 1 hívás azért kell, mert nem teljes fordulatot teszünk meg, hanem többet.



`tekergő(1,10)` `tekergő(3,10)` `tekergő(8,10)`

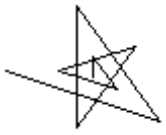
```
import turtle
def tekergo(db,h):
 turtle.forward(10*h)
 spiral(1+4*db,h,h)
 turtle.forward(10*h)

turtle.speed(0)
turtle.left(90)
tekergo(8,10)
```

```
def spiral(db,h,nov):
 if db>0:
     turtle.forward(h/2)
     turtle.right(90)
     spiral(db-1,h+nov,nov)
 else:
     turtle.forward(h/4)
     turtle.left(90)
```

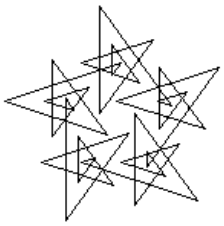
Elérhető: *https://repl.it/@hbv/tekergo*

Feladat Úgynevezett spiráloldalú sokszöget úgy rajzolhatunk, hogy a sokszög rajzolásában szereplő elmozdulást és elfordulást egyetlen spirálrajzoló def hívásával helyettesítjük. Készíts `spirálisábra(hossz,szög,db)`, amely spirálisoldalú sokszöget rajzol! Egy spirált a `spirál(hossz,szög,db)` eljárás rajzoljon! A spirál legrövidebb oldala és annak növekménye `hossz`, elfordulás-szöge `szög`, oldalai száma pedig db legyen!

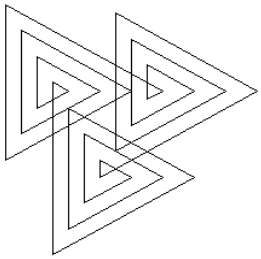
Megjegyzés: Annyiszor kell ismételni a spirális ábrát, amíg vissza nem térünk a kiinduló állapotba, azaz a fordulatok összege 360 fok többszöröse nem lesz.



`spirál(10,144,8)`



`spirálisábra(10,144,8)`

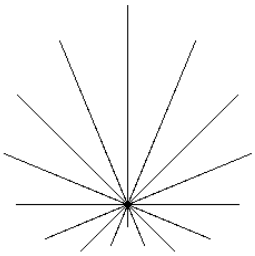


`spirálisábra(10,120,10)`

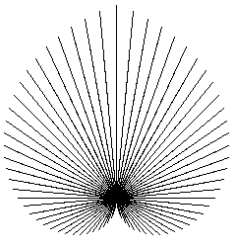
```
import turtle
def spiral(h,f,db):
 spiralis(h,f,db,1)

def spiralis(h,f,max,db):
 turtle.forward( h*db)
 turtle.right( f)
 if db<max:
     spiralis(h,f,max,db+1)

def spiralisabra(h,f,max):
 for i in range(int(keres(max*f % 360,1))):
     spiral(h,f,max)

def keres(f,n):
 if  360*n% f==0:
     return n*360/f
 else:
     return keres(f,n+1)

turtle.speed(0)
#spiralisabra(10,120,10)
spiralisabra(10,144,8)
```

Elérhető: *https://repl.it/@hbv/spiralisok*

Feladat Egy játékban adott hosszúságú pálcikákat kell egymáshoz illeszteni a mellékelt ábrának megfelelően. Készíts eljárást `pálcika(méret,db)` néven, amely ilyen ábrát tud készíteni! Az első paramétere a pálcikák hossza, a második pedig a pálcikák száma legyen! Az egyes pálcikák egymáshoz képest azonos szöggel legyenek elforgatva, az elforgatás helye pedig a pálcika végéhez képest egyenletesen változzon! (Figyeljünk rá, hogy a db osztója legyen 180.nak!)



pálcika(100,8)



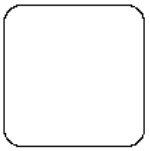
pálcika(100,18)

```
import turtle

def palcika(meret,db):
 palcikak(db,meret,meret/(db+2),180/db,me-
ret/(db+2))

def palcikak(db,meret,hol,szog,mennyivel):
 turtle.backward( hol)
 turtle.forward( meret)
 turtle.backward( meret-hol)
 turtle.left( szog)
 if db>=1:
     palcikak(db-1,meret,hol+mennyi-
vel,szog,mennyivel)

turtle.speed(0)
turtle.left(90)
palcika(100,18)
```

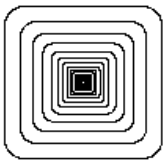
Elérhető: *https://repl.it/@hbv/palcikak*

Feladat Készítsd el az `íveselem(oldal)` és az `íves(oldal,q,bal,le)` eljárásokat, az ábrának megfelelően! A `q` paraméter legyen a méretcsökkenés arány, a `bal` és a `le` pedig a két irányú eltolás aránya a következő íves elem rajzolásához!

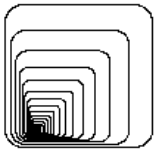
Megjegyzés: Az íves elem lekerekített sarkú négyzet, a kerekítés egy-egy negyedkör.



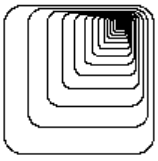
íveselem(100)



íves(100,0.8,0,0)



íves(100,0.8,0.8,0.8)



íves(100,0.8,-0.8,-0,8)

```
import turtle

def íves(r,q,bal,le):
 if r>1:
     íveselem(r,bal,le)
     íves(r*q,q,bal,le)

def íveselem(a,bal,le):
 turtle.penup()
 turtle.forward( a*(1+le)/2)
 turtle.right( 90)
 turtle.forward( a*3 *(1+bal)/8)
 turtle.pendown()
 for i in range( 4):
     turtle.circle(-a/8,90)
     turtle.forward( a/8*6)
 turtle.penup()
 turtle.backward( a*3*(1+bal)/8)
 turtle.left( 90)
 turtle.backward( a*(1+le)/2)
 turtle.pendown()

turtle.speed(0)
turtle.left(90)
#íves(100,0.8,0.8,0.8)
íves(100,0.8,-0.8,-0.8)
```

Elérhető: *https://repl.it/@hbv/iveselem*

Mozaik – rekurzívan

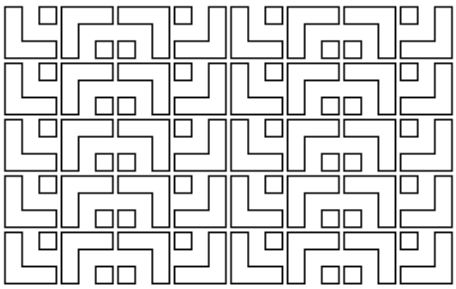
A feladatokat akár iterációval is megoldhatnánk!

Feladat Egy mozaik alapeleme az alábbi ábrán látható lemez, amit az `alap(hossz)` eljárással rajzolhatunk meg. A `sor(darab,hossz)` eljárás egymás mellé tesz `darab` ilyen lemezt, a szomszédjukhoz képest 90 fokkal elforgatva. A `mozaik(sordb,oszlopdb,hossz)` eljárás úgy tesz egymás alá `sordb` darab sort, hogy az új sorok elején levő lemez az előző sor végén levő után következő (azaz 90 fokkal elforgatott) legyen. Az egyes lemezek egymástól 3 egység távolságra vannak. Készítsd el az `alap`, a `sor` és a `mozaik` eljárásokat!

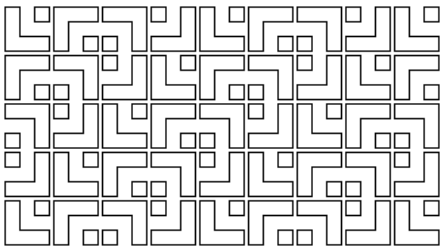
Megjegyzés: Itt ugyan négyféle elem van, de ezek az alapelem 0, 90, 180 és 270 fokos elforgatottjai. Rekurzív eljárást írunk, amelynek egyik paramétere ez a 4 érték lesz, ciklikusan léptetve. Alapelem rajzolásból azonban elég egyet megírnunk, mert az elforgatások csak annyit jelentenek, hogy a négyzet más-más sarkából kell elindulni a rajzolással.

alap(20)

sor(9,20,0)



mozaik(5,8,20,0)



mozaik(5,9,20,0)

```
def alap(r):
 turtle.forward(r)
 turtle.right(90)
 turtle.forward(r/3)
 turtle.right(90)
 turtle.forward(2*r/3)
 turtle.left(90)
 turtle.forward(2*r/3)
 turtle.left(90)
 turtle.penup()
 turtle.forward(r/3)
 turtle.pendown()
 for i in range(4):
```

```
import turtle

def forgat(r,szog):
 turtle.penup()
 for i in range(int(szog/90)):
     turtle.forward(r)
     turtle.right(90)
 turtle.pendown()
 alap(r)
 turtle.penup()
 for i in range(4-int(szog/90)):
     turtle.forward(r)
     turtle.right(90)
 turtle.pendown()

def sor(m,r,szog):
 if m>0:
     forgat(r,szog)
     turtle.penup()
     turtle.right(90)
     turtle.forward(r+3)
     turtle.left(90)
     turtle.pendown()
     sor(m-1, r, (szog+90) %360)
     turtle.penup()
     turtle.right(90)
     turtle.backward(r+3)
     turtle.left(90)
     turtle.pendown()

def mozaik(n,m,r,szog):
 sor(m, r, szog)
 if n>1:
     turtle.penup()
     turtle.backward(r+3)
     turtle.pendown()
     mozaik(n-1,m,r, (szog+90*(m % 4)) %
360)
     turtle.penup()
     turtle.forward(r+3)
     turtle.penup()

turtle.speed(0)
turtle.left(90)
#mozaik(5,9,20,0)
```

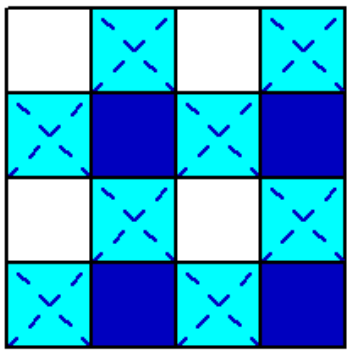
```
        turtle.forward(r/3)
        turtle.left(90)
    turtle.penup()
    turtle.backward(r/3)
    turtle.pendown()
    turtle.right(180)
    turtle.forward(r/3)
    turtle.right(90)
    turtle.forward(r)
    turtle.right(90)
```

```
mozaik(5,8,20,0)
```

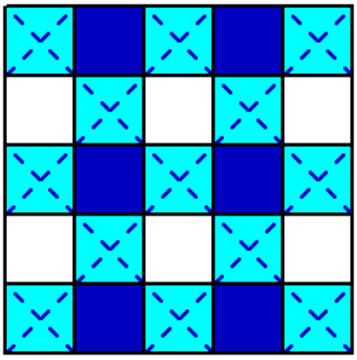
Elérhető: *https://repl.it/@hbv/forgatva*

Feladat Készítsd el a kockás abroszt `kockásabrosz(sordb,oszlopdb,méret)` eljárással, amely három különböző színű négyzetekből áll és a kockákon keresztül szaggatott vonalak is díszítik az ábrának megfelelően. A `sordb` a sorok, az `oszlopdb` az oszlopok számát, a `méret` pedig a négyzetek méretét jelöli.

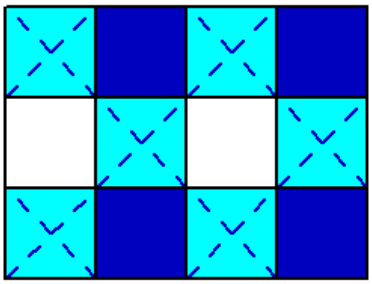
Megjegyzés: Egy logikai értékű paraméter használunk arra, hogy szaggatott vonalas vagy vonal nélküli alapelemet kell rajzolni. A terítő sorai is kétfélék, azaz két sor rajzoló eljárást kell írni, szintén logikai paraméterrel. (A feladat kiírásban a logikai paraméter nem szerepel, ezért egy ún. vezérlő eljárást írunk (`kockásabrosz`), ami meghívja eggyel több paraméterrel a mozaik rajzolót.)



`kockásabrosz(4,4,50)`



`kockásabrosz(5,5,50)`



`kockásabrosz(3,4,60)`

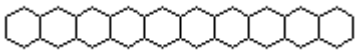
```
import turtle,math
def kockasabrosz(sordb,oszlopdb,me-
ret):
 mozaik(sordb,oszlopdb,meret,False)

def mozaik(sordb,oszlopdb,meret,log):
 if sordb>0:
     if log:
         sor13(oszlopdb,meret,log)
     else:
         sor23(oszlopdb,meret,log)
     turtle.forward(meret)
     mozaik(sordb-1,oszlopdb,meret,not
log)
     turtle.backward(meret)

def sor13(db,meret,log):
 if db>0:
    if log:
        kocka(meret,"white",False)
    else:
        kocka(meret, "aquamarine",
True)
    turtle.right(90)
    turtle.forward(meret)
    turtle.left(90)
    sor13(db-1,meret,not log)
    turtle.right(90)
    turtle.backward(meret)
    turtle.left(90)

def sor23(db,meret,log):
 if db>0:
    if log:
        kocka(meret,"blue", False)
    else:
        kocka(meret, "aquamarine",
True)
    turtle.right(90)
    turtle.forward(meret)
    turtle.left(90)
```

```
def kocka(meret,szin,csikos):
 gyk= math.sqrt(2)
 turtle.fillcolor(szin)
 turtle.begin_fill()
 for i in range(4):
    turtle.forward(meret)
    turtle.right(90)
 turtle.end_fill()
 if csikos:
    turtle.right(45)
    csik(meret*gyk)
    turtle.penup()
    turtle.backward(meret*gyk)
    turtle.right(45)
    turtle.forward(meret)
    turtle.left(135)
    turtle.pendown()
    csik(meret*gyk)
    turtle.penup()
    turtle.backward(meret*gyk)
    turtle.right(135)
    turtle.backward(meret)
    turtle.left(90)
    turtle.pendown()
    sor23(db-1,meret,not log)
    turtle.right(90)
    turtle.backward(meret)
    turtle.left(90)

def csik(meret):
 turtle.pendown()
 for i in range(4):
    turtle.forward(meret/8)
    turtle.penup()
    turtle.forward(meret/8)
    turtle.pendown()

turtle.speed(0)
turtle.left(90)
kockasabrosz(5,5,20)
```

Elérhető: *https://repl.it/@hbv/terito*

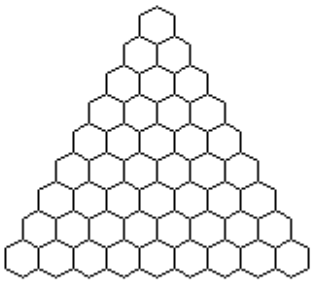
Feladat Szabályos hatszöget is vehetünk a mozaik alapjául. Hatszögekkel a sík sokféle tartománya fedhető le, ezek közül nézünk meg néhányat a továbbiakban.

A mozaikok nemcsak téglalap alakúak lehetnek! Készíts eljárást, amely méhsejtből (szabályos hatszögekből) különböző alakzatokat tud építeni! A `hatszög(méret)` eljárás egyetlen méhsejtet rajzoljon, ahol `méret` a hatszög oldalhossza! A `sor n méret` def n darab méhsejtet rajzoljon egymás mellé



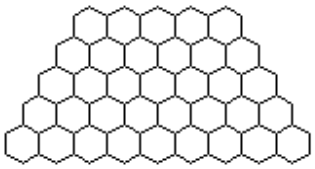
A `három(n,méret)`, a `trapéz(n,m,méret)` és a `hat(n,méret)` eljárások pedig az alábbi ábrákat rajzolják, ahol `n` az alsó sorban levő hatszögek száma, `méret` pedig a hatszögek oldalhossza. A trapéznál `m` a felső sorban levő hatszögek száma legyen!

Megjegyzés: Ha a hatszög nem állapotátlátszó, hanem a baloldali alsó csúcsából kezdjük rajzolni és a jobboldali alsó csúcsában fejezzük be, akkor a sor rajzolása sokkal egyszerűbb lesz. A háromszög lefedése egyszerű, ugyanannyi sort kell rajzolni, ahány elem az alsó sorban van. A trapéz csak kicsit bonyolultabb, a rekurzió feltételét kell módosítani!

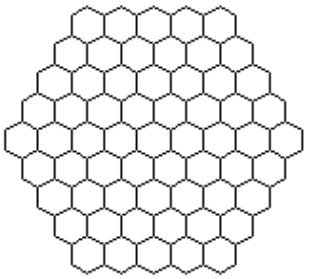


három(9,20)

```
import turtle,math

def hatszog( hossz):
 for i in range( 10):
     turtle.forward( hossz)
     turtle.right( 60)
 turtle.right( 120)
```



trapéz(9,5,20)



hat(5,20)

```
def hat(n,hossz):
 trapez(n*2-1 ,n,hossz)
 for i in range( n*2-1):
     for i in range( 2):
         turtle.left( 60)
         turtle.backward(
hossz)
     turtle.right( 120)
 turtle.forward( hossz)
 trapez(n*2-1,n,-hossz)

turtle.speed(0)
turtle.left(90)
#harom(9,20)
#trapez(9,5,20)
hat(5,20)
```

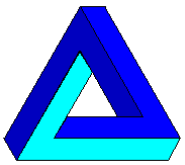
```
def sor(n,hossz):
 for i in range( n):
     hatszog(hossz)
 for i in range( n):
     turtle.left( 120)
     for i in range( 2):
         turtle.forward( hossz)
         turtle.right( 60)

def harom(n,hossz):
 sor(n,hossz)
 if n>1:
     turtle.forward( hossz)
     turtle.right( 60)
     turtle.forward( hossz)
     turtle.left( 60)
     harom(n-1,hossz)
     turtle.right( 60)
     turtle.backward( hossz)
     turtle.left( 60)
     turtle.backward( hossz)

def trapez(n,m,hossz):
 sor(n,hossz)
 if n>=m:
     turtle.forward( hossz)
     turtle.right( 60)
     turtle.forward( hossz)
     turtle.left( 60)
     trapez(n-1,m,hossz)
     turtle.right( 60)
     turtle.backward( hossz)
     turtle.left( 60)
     turtle.backward( hossz)
```

Elérhető:https://repl.it/@hbv/hatszoglefedes

Optikai csalódások

Egy optikai csalódást (más néven vizuális illúziót) vizuálisan észlelt képek jellemeznek, melyek eltérnek az objektív valóságtól.

Feladat A paradox illúziókat olyan tárgyak váltják ki, melyek paradoxak vagy lehetetlenek, mint például a következő Penrose-háromszög. Rajzold meg háromféle kék színnel a mellékelt háromszöget `háromszög`! (Segítség egy 80 oldalhosszúságú háromszögnek levágtunk a három sarkából egy-egy 10 oldalhosszúságú háromszöget, majd 10 egység széles sávokat színeztük.)



```
import turtle,math

def oldal(h, szin):
 turtle.fillcolor(szin)
 turtle.penup()
 turtle.forward( h/8)
 turtle.pendown()
 turtle.begin_fill()
 turtle.forward( 6*h/8)
 turtle.right(120)
 turtle.forward( 5*h/8)
 turtle.right(120)
 turtle.forward( h/8)
 turtle.right( 60)
 turtle.forward( 3*h/8)
 turtle.left( 120)
 turtle.forward( 5*h/8)
 turtle.right(120)
 turtle.forward( h/8)
 turtle.penup()
 turtle.end_fill()
 turtle.right(90)
 turtle.forward( h/16)
 turtle.backward( h/16)
 turtle.left( 90)
 turtle.right( 60)
 turtle.backward( h/8)
 turtle.pendown()
```

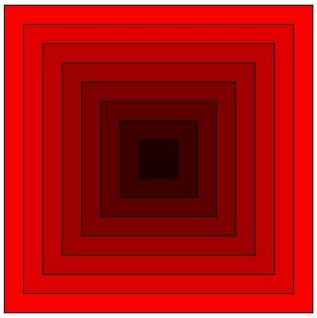
```
def haromszog( h):
 turtle.right(30)
 oldal(h,"dark blue")
 turtle.penup()
 turtle.forward(h)
 turtle.right(120)
 turtle.pendown()
 oldal(h,"blue")
 turtle.penup()
 turtle.forward(h)
 turtle.right(120)
 turtle.pendown()
 oldal(h,"aquamarine")
 turtle.penup()
 turtle.forward(h)
 turtle.right(120)
 turtle.pendown()
 turtle.left(30)

turtle.speed(0)
turtle.left(90)
haromszog(100)
```

Elérhető: *https://repl.it/@hbv/penroseharomszog*

Feladat Josef Albert német festőművész művei inspirálta feladat a színárnyalatok térbeli képzetet adnak az ábrának Készítsd el a minta szerinti ábrát `rács(db,hossz)` eljárással! A belső négyzet oldala `hossz*2` egység hosszú, a sávok `hossz` vastagságúak. Az egyes sávokat kifelé haladva egyre világosabb pirosra színezd!

Megjegyzés: Egy piros árnyalatú színt az alábbi utasítással hozhatsz létre (`piroserték,0,0`). A rekurzív hívás miatt szükség lesz egy plusz paraméterre a növekvő négyzetek oldalának változásának mértékére is `d`. Indulásnál ez fele akkora legyen, mint az `oldal` értéke! A `db` értéke csak akkora lehet, hogy a szín piros árnyalata 0 és 255 közé essen!

A rajzolásnál kívülről befelé rajzoljuk a színezett négyzeteket, hiszen különben felülszínezné a már elkészülteket!



racs(10,200,10)

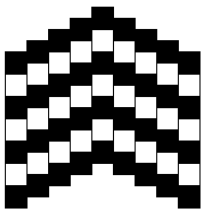
```
import turtle,math

def racs(db,oldal,d):
 if db>0:
     negyzet(oldal,(db*15,0,0))
     turtle.penup()
     turtle.right(45)
     turtle.backward(d/2*math.sqrt(2))
     turtle.forward(d*3/2*math.sqrt(2))
     turtle.left(45)
     turtle.pendown()
     racs(db-1,oldal-2*d,d)

def negyzet(oldal,szin):
 turtle.fillcolor(szin)
 turtle.begin_fill()
 for i in range(4):
      turtle.forward(oldal)
      turtle.right(90)
 turtle.end_fill()

turtle.speed(0)
turtle.left(90)
#turtle.colormode(255)
racs(10,200,10)
```

Elérhető: *https://repl.it/@hbv/szinesracs*

Feladat Sok esetben válunk optikai csalódás áldozatává, amikor egy ábrát másnak látunk, mint amilyen a valóságban, mint amilyennel a következő feladatban találkozunk. Készíts mozaik-rajzoló eljárást `oszlop`, `hullám`, `hullámsor`, amely egymástól négyzetoldalnyi egységnyire lévő négyzetekből épül fel! A szomszédos négyzetek fél négyzetoldallal vannak eltolva, a hullámsorok egymástól egy négyzetoldalnyi távolságban vannak!

Az `oszlop(n,h)` eljárás `n` darab fekete és fehér, `h` oldalhosszúságú négyzetekből állítson össze egy oszlopot (feltehetjük, hogy `n` páratlan)! A `hullám(m,n,h)` eljárás `m` oszlopot tegyen egymás mellé az ábrának megfelelően (feltehető, hogy `m` is páratlan)! A `hullámsor(db,m,n,h)` eljárás `db` darab hullámot tegyen egymás mellé az ábrának megfelelően!

oszlop(7,10)



hullám(9,7,10)

hullámsor(4,9,7,10)

```
import turtle,math
def oszlop(n,h):
 for i in range(int(n//2)):
     telielem(h)
     turtle.forward(h)
     ureselem(h)
     turtle.forward(h)
 telielem(h)
 turtle.backward((n-1)*h)

def telielem(h):
 turtle.begin_fill()
 ureselem(h)
 turtle.end_fill()

def ureselem(h):
 for i in range(4):
    turtle.forward(h)
    turtle.right(90)
```

```
def hullam(m,n,h):
 for i in range(int(m//2)):
     oszlop(n,h)
     turtle.right(90)
     turtle.forward(h)
     turtle.left(90)
     turtle.forward(h/2)
 for i in range(int(m//2)):
     oszlop(n,h)
     turtle.right(90)
     turtle.forward(h)
     turtle.left(90)
     turtle.backward(h/2)
 oszlop(n,h)
```

```
def hullamsor(db,m,n,h):
 for i in range(db):
     hullam(m,n,h)

turtle.speed(0)
turtle.left(90)
#oszlop(7,10)
#hullam(9,7,10)
hullamsor(4,9,7,10)
```

Elérhető: *https://repl.it/@hbv/hullamsor*

Feladat A következő feladatban két, ártatlannak tűnő színminta egymásra rajzolásával érünk el meglepő eredményt. Készíts eljárásokat az alábbi, érzékcsalódást okozó ábrák megrajzolására!

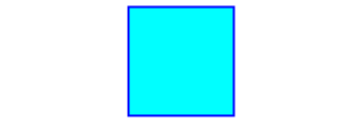
A `négyzet(h,külső,belső)` eljárás a `h` oldalhosszúságú négyzet oldalát `külső`, a belsejét pedig `belső` színnel rajzolja ki!

Az `alapsor(m,h)` eljárás `m` darab zöld belsejű és határvonalú négyzetet rajzoljon egy sorba, az `alapmozaik(n,m,h)` eljárás `n` sort egymás fölé! A négyzetek közötti távolság legyen egyenlő a négyzet oldalhosszával!

A `felsősor(m,h,s)` eljárás az ábrának megfelelő mintázatban `m` darab fehér belsejű és piros határvonalú, valamint piros belsejű és fehér határvonalú négyzetet rajzoljon egy sorba, az `felsőmozaik(n,m,h,s)` eljárás `n` sort egymás fölé! A négyzetek közötti távolság legyen egyenlő a négyzet oldalhossza felével! A `s` az alsó sor első kitöltött négyzetének a sorszáma legyen!

A `mozaik(n,m,h)` eljárás a mintának megfelelően helyezze el az alapmozaikra a felsőmozaikot! Az alapmozaik négyzetei mérete `3*h`, a felsőmozaiké pedig `2*h` legyen!

Megjegyzés: Érdemes ezt a trükköt máshol is alkalmazni. Ha nagyon összetett egy mozaik, jó ötlet lehet több könnyebben implementálható részből összeállítani!

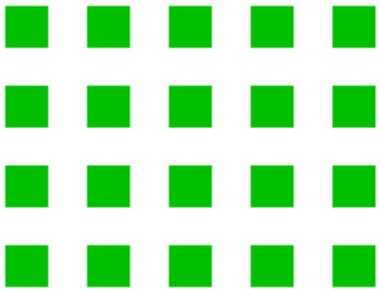


`négyzet(60,1,10)`

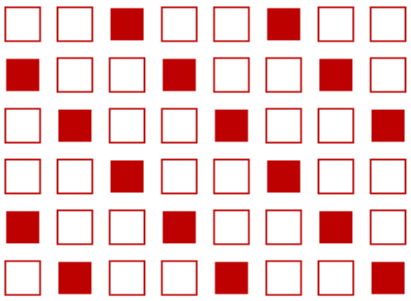
`alapsor(5,30)`

`felsősor(8,30,2)`

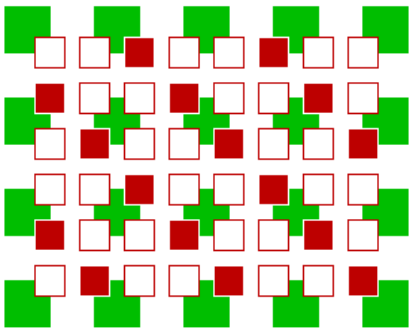
```
import turtle

def mozaik(n,m,h):
 alapmozaik(n,m,3*h)
 turtle.penup()
 turtle.forward( 2*h)
 turtle.right(90)
 turtle.forward( 2*h)
 turtle.left(90)
 turtle.pendown()
 felsomozaik(2*(n-1),2*(m-1),2*h,h)
 turtle.penup()
 turtle.right(90)
 turtle.backward( 2*h)
```



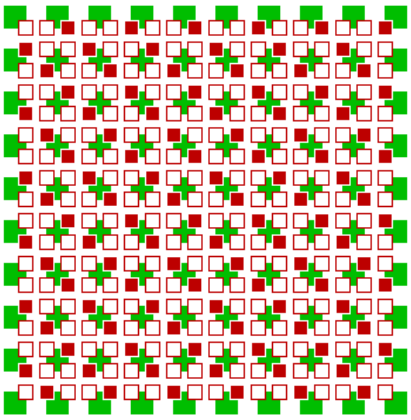
alapmozaik(4,5,30)



felsőmozaik(6,8,30,2)



mozaik(4,5,60)



mozaik(10,10,10)

```
 turtle.left(90)
 turtle.backward( 2*h)
 turtle.pendown()

def alapmozaik(n,m,h):
 for i in range( n):
     alapsor(m,h)
     turtle.penup()
     turtle.forward( 2*h)
     turtle.pendown()
 turtle.penup()
 turtle.backward( 2*n*h)
 turtle.pendown()

def alapsor(m,h):
 for i in range( m):
     negyzet(h,"green","green")
     turtle.penup()
     turtle.right(90)
     turtle.forward( 2*h)
     turtle.left(90)
     turtle.pendown()
 turtle.penup()
 turtle.right(90)
 turtle.backward( 2*m*h)
 turtle.left(90)
 turtle.pendown()

def felsomozaik(n,m,h,i):
 felsosor(m,h,i)
 if n>1:
    turtle.penup()
    turtle.forward(1.5*h)
    turtle.pendown()
    felsomozaik(n-1,m,h, (i+1)%3)
    turtle.penup()
    turtle.backward(1.5*h)
    turtle.pendown()

def negyzet(h,s,bs):
 turtle.pensize(1)
 turtle.fillcolor(bs)
 turtle.begin_fill()
 for i in range( 4):
    turtle.forward( h)
    turtle.right(90)
 turtle.end_fill()
 turtle.pencolor(s)
 for i in range( 4):
     turtle.forward( h)
     turtle.right(90)
 turtle.pencolor("black")

turtle.speed(0)
turtle.left(90)
mozaik(5,5,10)
```

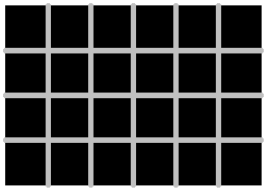
Elérhető: *https://repl.it/@hbv/optikai*

Feladat Készítsd el az alábbi, optikai csalódást mutató ábrákat rajzoló eljárásokat `tégla(a,b)`, `rács(dba,dbb,t)`, `pöttyök(dba,dbb,t)`! A `tégla` eljárás `a*b` méretű fekete téglalapot rajzoljon! A `rács` eljárás `dba*dbb` méretű szürke rácsot rajzoljon 5 egység vastagságú

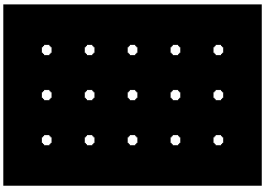
vonalakból, ahol a rácsvonalak `t` távolságra vannak egymástól! A `pöttyök` eljárás `dba*dbb` méretű fehér pöttymintát rajzoljon 10 egység nagyságú pontokból, ahol a pöttyök `t` távolságra vannak egymástól!

Megjegyzés: Az alábbi képeket egy vagy több def egymás utáni hívásával kaptuk- A feladat megoldása egy fekete téglalap, továbbá két speciális (szürke, illetve fehér) mozaik kirajzolása.

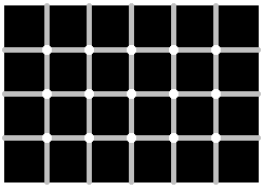
`tégla(60, 90)`



`tégla(60, 90)`
`rács(3,5,15)`



`tégla(60, 90)`
`pöttyök(3,5,15)`

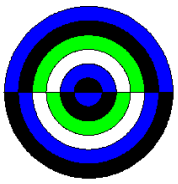


`tégla(60, 90)`
rács(3,5,15)
pöttyök(3,5,15)

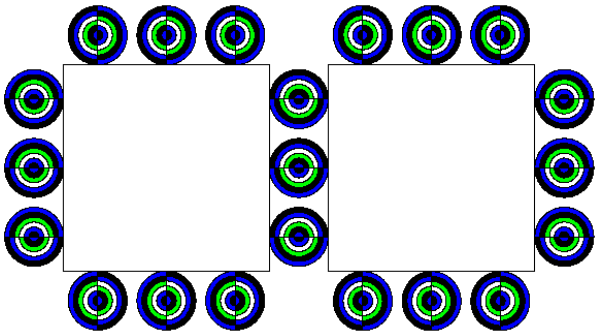
```
def tegla(x, y):
 turtle.pensize(1)
 turtle.fillcolor("black")
 turtle.begin_fill()
 for i in range(2):
     turtle.forward(x)
     turtle.right(90)
     turtle.forward(y)
     turtle.right(90)
 turtle.end_fill()

turtle.speed(0)
turtle.left(90)
tegla(120,180)
racs(3,5,30)
pottyok(3,5,30)
```

```
import turtle

def pottyok(n,m,t):
 turtle.penup()
 turtle.pencolor("white")
 for i in range(n):
     turtle.forward(t)
     turtle.right(90)
     for i in range(m):
         turtle.forward(t)
         turtle.pendown()
         turtle.dot(5)
         turtle.penup()
     turtle.backward(m*t)
     turtle.left(90)
 turtle.backward(n*t)
 turtle.pendown()

def racs(n,m,t):
 turtle.pensize(5)
 turtle.pencolor("gray")
 for i in range(m):
     turtle.penup()
     turtle.right(90)
     turtle.forward(t)
     turtle.left(90)
     turtle.pendown()
     turtle.forward((n+1)*t)
     turtle.backward((n+1)*t)
 turtle.penup()
 turtle.right(90)
 turtle.backward(t*m)
 turtle.left(90)
 turtle.pendown()
 for i in range(n):
     turtle.penup()
     turtle.forward(t)
     turtle.pendown()
     turtle.right(90)
     turtle.forward((m+1)*t)
     turtle.backward((m+1)*t)
     turtle.left(90)
 turtle.penup()
 turtle.backward(t*n)
 turtle.pendown()

Elérhető:
https://repl.it/@hbv/pottyos
```

Feladat Ha sokáig nézed a jobb oldali mellékelt ábrát, akkor úgy látszik, mintha forognának a körök. Készíts `köröm(r)` és `forgó(hossz)` eljárást, ahol a `r` a legnagyobb kör sugara, a `hossz` pedig a négyzetek oldala!



Köröm(100)



Forgó(200)

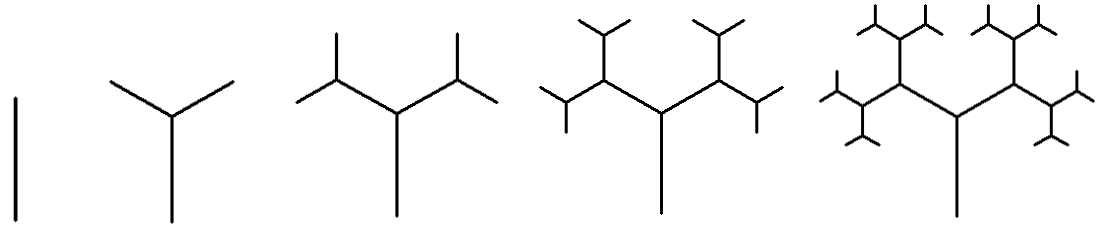
```
import turtle

def korom(r):
 kor2(r, "blue",
"black")
 kor2(r/6*5, "black",
"blue")
 kor2(r/3*2, "white",
"green")
 kor2(r/2, "green",
"white")
 kor2(r/3, "blue",
"black")
 kor2(r/6, "black",
"blue")

def kor2(r,a,b):
 turtle.right(90)
 turtle.forward(r)
 turtle.left(90)
 turtle.fillcolor(a)
 turtle.begin_fill()
 turtle.circle(r,180)
 turtle.left(90)
 turtle.forward(r*2)
 turtle.end_fill()
 turtle.fillcolor(b)
 turtle.begin_fill()
 turtle.backward(r*2)
 turtle.left(90)
 turtle.circle(-r,-180)
 turtle.end_fill()
 turtle.left(90)
 turtle.backward(r)
 turtle.left(90)

turtle.speed(0)
turtle.left(90)
#korom(100)
forgo(200)
```

```
def forgo(oldal):
 negyzet(oldal)
 turtle.penup()
 turtle.right(90)
 turtle.forward(oldal*9/7)
 turtle.left(90)
 turtle.pendown()
 turtle.forward(oldal)
 turtle.right(90)
 negyzet2(oldal)
 turtle.penup()
 turtle.right(90)
 turtle.backward(oldal*9/7)
 turtle.left(90)

def negyzet(oldal):
 for i in range(4):
    for i in range(3):
        turtle.forward(oldal/6)
        turtle.left(90)
        turtle.forward(oldal/7)
        turtle.right(90)
        korom(oldal/7)
        turtle.left(90)
        turtle.backward(oldal/7)
        turtle.right(90)
        turtle.forward(oldal/6)
    turtle.right(90)

def negyzet2(oldal):
 for i in range(3):
    for i in range(3):
        turtle.forward(oldal/6)
        turtle.left(90)
        turtle.forward(oldal/7)
        turtle.right(90)
        korom(oldal/7)
        turtle.left(90)
        turtle.backward(oldal/7)
        turtle.right(90)
        turtle.forward(oldal/6)
    turtle.right(90)
```

Elérhető: *https://repl.it/@hbv/forgomozak*

Variációk fa rajzolásra

Feladat Egy fa egy H hosszúságú törzsből áll, amelynek végén szimmetrikusan, egymással 120 fokos szöget bezárva egy-egy újabb fa nő ki. A fa gyökerétől az ágak végéig N lépést lehet megtenni, a törzsből kinövő újabb fák törzshossza az eredeti fa törzshosszának kétharmada.

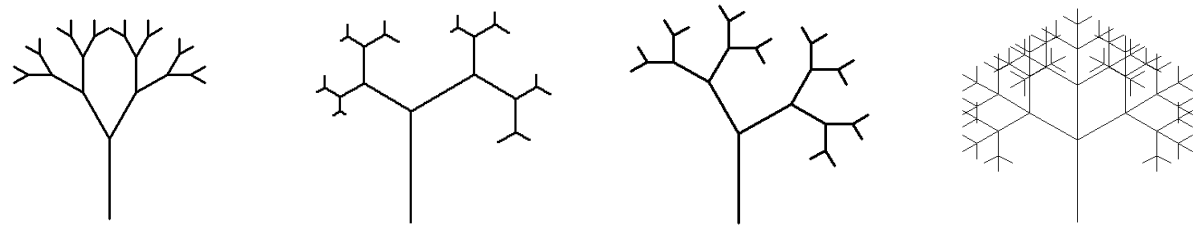


1 2 3 4 5

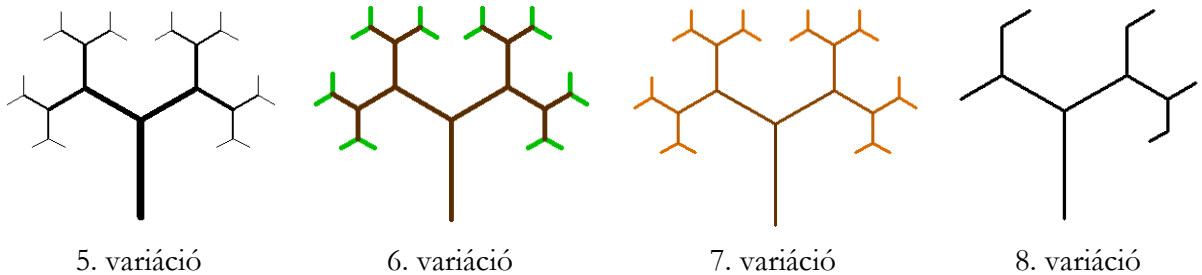
```
import turtle
def fa(n,h):
 turtle.forward(h)
 if n>1:
     turtle.left(60)
     fa(n-1,h*2/3)
     turtle.right(120)
     fa(n-1,h*2/3)
     turtle.left(60)
 turtle.backward(h)

turtle.speed(0)
turtle.left(90)
fa(5,100)
```

Elérhető: *https://repl.it/@hbv/fa*

Megjegyzés: A kód módosításával számtalan fa kirajzolható: ágak száma, szöge, vastagsága stb. változhat!

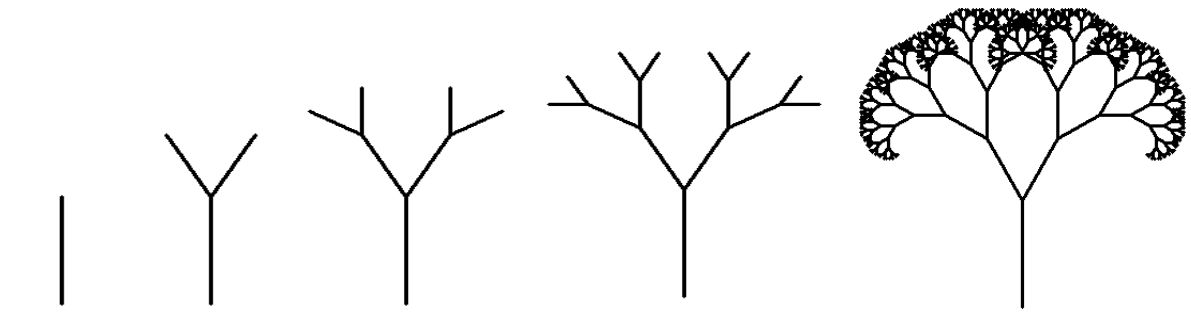


1. variáció 2. variáció 3. variáció 4. variáció



5. variáció 6. variáció 7. variáció 8. variáció

Feladat Mások legyenek az ágak közötti szögek!



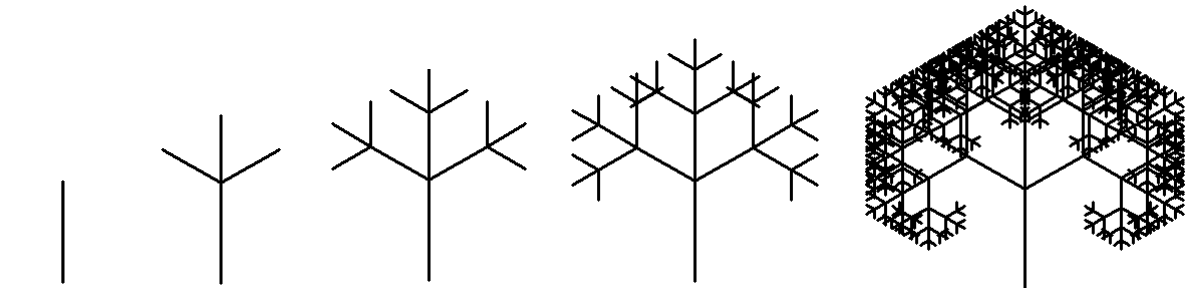
fa1(1,50) fa1(2,50) fa1(3,50) fa1(4,50) fa1(10,50)

```
import turtle

def fa1(n,h):
 turtle.forward(h)
 if n>1:
     turtle.left(30)
     fa1(n-1,h*2/3)
     turtle.right(60)
     fa1(n-1,h*2/3)
     turtle.left(30)
 turtle.backward(h)

turtle.speed(0)
turtle.left(90)
fa1(5,100)
```

Elérhető: *https://repl.it/@hbv/fa1*

Feladat Az ágak számát változtassuk meg!

Megjegyzés: A megoldásban háromszor hívjuk rekurzívan a rajzolást, a három ágnak megfelelően.



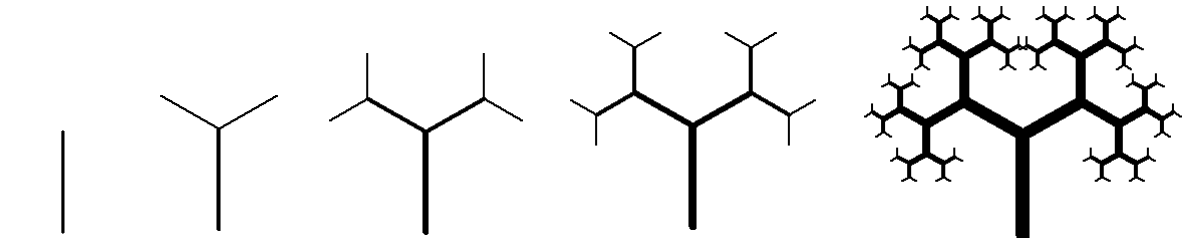
fa2(1,50) fa2(2,50) fa2(3,50) fa2(4,50) fa2(7,50)

```
import turtle

def fa2(n,h):
 turtle.forward(h)
 if n>1:
     turtle.left(60)
     fa2(n-1,h*2/3)
     turtle.right(60)
     fa2(n-1,h*2/3)
     turtle.right(60)
     fa2(n-1,h*2/3)
     turtle.left(60)
 turtle.backward(h)

turtle.speed(0)
turtle.left(90)
fa2(5,100)
```

Elérhető: *https://repl.it/@hbv/fa2*

Feladat Az ágak vastagságát változtassuk meg!

Megjegyzés: A megoldásban a tollvastagságot változtatjuk a szinteknek megfelelően!

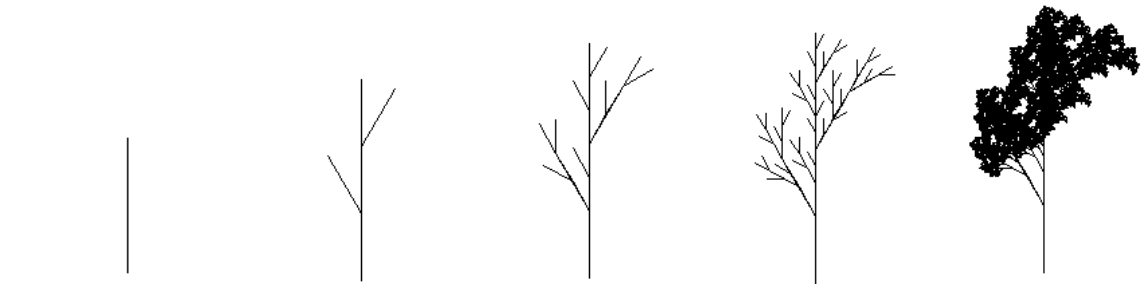


fa3(1,50) fa3(2,50) fa3(3,50) fa3(4,50) fa3(7,50)

```
import turtle

def fa3(n,h):
 turtle.pensize(n)
 turtle.forward(h)
 if n>1:
     turtle.left(60)
     fa3(n-1,h*2/3)
     turtle.right(120)
     fa3(n-1,h*2/3)
     turtle.left(60)
 turtle.backward(h)

turtle.speed(0)
turtle.left(90)
fa3(5,100)
```

Elérhető: *https://repl.it/@hbv/fa3*

Feladat A törzs egyik fele legyen állandó, a másik fele pedig ág! Az ága aljából balra, az ág végéből pedig felfelé, illetve jobbra is nő újabb ág.



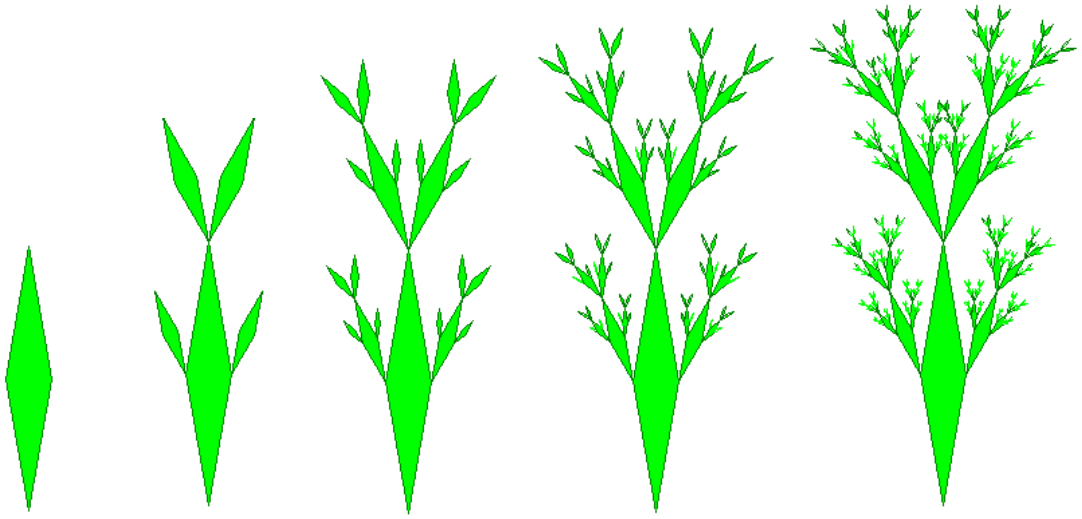
fa4(1,50) fa4(2,50) fa4(3,50) fa4(4,50) fa4(8,50)

```
import turtle

def fa4(n,h):
 if n==1:
    turtle.forward(h)
    turtle.backward(h)
 else:
    turtle.forward(h/2)
    turtle.left(30)
    fa4(n-1,h/2)
    turtle.right(30)
    fa4(n-1,h/2)
    turtle.forward(h/2)
    turtle.right(30)
    fa4(n-1,h/2)
    turtle.left(30)
    fa4(n-1,h/2)
    turtle.backward(h)

turtle.speed(0)
turtle.left(90)
fa4(3,100)
```

Elérhető: *https://repl.it/@hbv/fa4*

Feladat A fa törzse rombusz. A törzs két oldalsó sarkából egy-egy (harmad hosszúságú), a tetejéből pedig 2 ág nőhet ki (fele akkora).

Megjegyzés: A kaktusznál – a belseje festésén túl – az az újdonság, hogy az oldalából is nőnek ki újabb ágak.

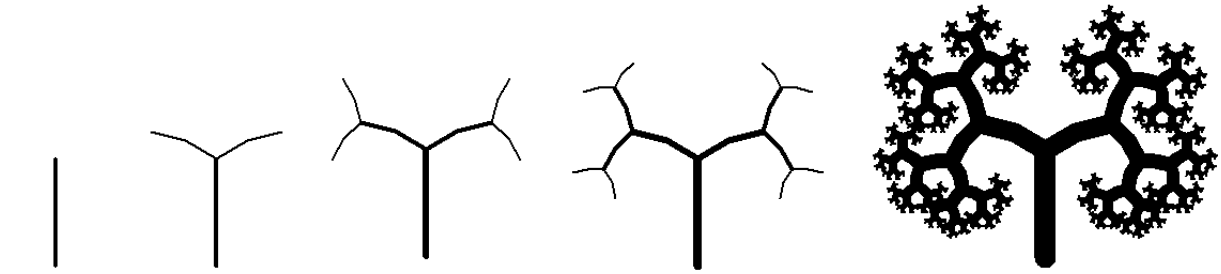


fa5(1,50) fa5(2,50) fa5(3,50) fa5(4,50) fa5(8,50)

```
import turtle

def rombusz(h):
    turtle.begin_fill()
    for szog in (20,160,20,160):
        turtle.forward(h)
        turtle.right(szog)
    turtle.end_fill()

def fa5(n,h):
 if n>=1:
    rombusz(h)
    turtle.forward(h)
    turtle.left(20)
    fa5(n-1,h/3)
    turtle.right(40)
    turtle.forward(h)
    turtle.left(40)
    fa5(n-1,h/2)
    turtle.right(40)
    fa5(n-1,h/2)
    turtle.right(160)
    turtle.forward(h)
    turtle.left(160)
    fa5(n-1,h/3)
    turtle.right(180)
    turtle.forward(h)

turtle.right(160)turtle.speed(0)
turtle.left(100)
turtle.fillcolor("green")
fa5(3,100)


Elérhető:
https://repl.it/@hbv/fa5
```

Feladat A következő variációban a fa „külső hatásoktól függően növekszik. A fa függőlegestől balra hajló ágai közepükön 15 fokkal balra, a jobbra hajlók pedig 15 fokkal jobbra hajlanak!

Megjegyzés: Ebben a feladatban alkalmaztunk először közvetett rekurziót! Fa6 hívja fa6a-t, fa6a hívja fa6-ot! Másik tudnivaló a teknőc alapállapota az óramutató 3-as jelzésénél van, így ha függőlegesen áll, 90 fok az értéke!



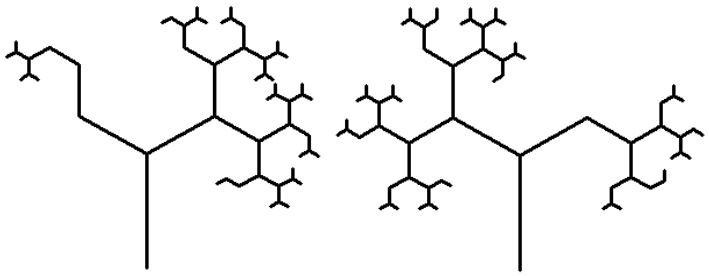
fa6(1,50) fa6(2,50) fa6(3,50) fa6(4,50) fa6(10,50)

```
import turtle

def fa6(n,h):
 turtle.pensize(n*2)
 turtle.forward( h/2)
 if turtle.heading()==90.0 or
turtle.heading==270.0:
    fa6a(n,h)
 else:
     if turtle.heading()<90.0:
        turtle.right(15)
        fa6a(n,h)
        turtle.left(15)
     else:
        turtle.left(15)
        fa6a(n,h)
        turtle.right(15)
 turtle.backward(h/2)
```

```
def fa6a(n,h):
    turtle.forward( h/2)
    if n>1:
       turtle.left(60)
       fa6(n-1,h/3*2)
       turtle.right(120)
       fa6(n-1,h/3*2)
       turtle.left(60)
    turtle.backward( h/2)
turtle.speed(0)
turtle.left(90)
fa6(4,100)


Elérhető: https://repl.it/@hbv/fa6
```

Feladat Ebben a feladatban a fák véletlenszerűen növekednek. Egy ág véletlenül nő vagy nem nő egy adott helyen. Az ág növekedés valószínűségét a `veletlenfa` eljárás `s` paraméterével befolyásoljuk!

Megjegyzés: Többször végrehajtva ugyanazt mégis mindig más eredményt kapunk, köszönhetően a véletlenszerű növekedésnek! Véletlenszámokat a random modul importálásával érhetünk el!



fa7(7,100)

```
import turtle, random

def fa7(n,h):
 turtle.forward( h)
 if n>1:
    turtle.left(60)
    veletlenfa(90,n-1,h/3*2)
    turtle.right(120)
    veletlenfa(90,n-1,h/3*2)
    turtle.left(60)
 turtle.backward(h)
```

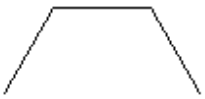
```
def veletlenfa(s,n,h):
    if random.randint(0,100)<s:
         fa7(n,h)
turtle.speed(0)
turtle.left(90)
turtle.fillcolor("green")
fa7(7,100)


Elérhető: https://repl.it/@hbv/fa7
```

Fraktálok

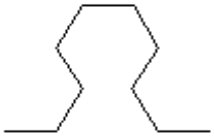
A fraktálok végtelenül komplex geometriai alakzatok, amelyek határoló vonalai vagy felületei végtelenül „gyűröttek” vagy „érdesek”, illetve „szakadásosak” (szakkifejezéssel, nem differenciálhatóak)” Innen nyerték nevüket is a latin fractus melléknév, a frangere, „törni” ige származéka, ugyanis elsősorban töröttet, darabosat (vö. „mindenütt tüskésség” vagy „mindenütt szakadásosság”), másodsorban szabálytalant, kivételest jelent. Benoit B. Mandelbrot, a fogalom névadója ebből a latin szóból alkotta meg a fraktál kifejezést. (Wikipedia)

További olvasnivalót lehet találni a http//fraktál.lap.hu címen.

Feladat Sierpinski nyílhegy görbéje úgy keletkezik, hogy egy adott hosszúságú szakaszt helyettesítünk három feleakkora hosszúságúval, az ábrának megfelelően. A második görbénél ugyanezt a módszert alkalmazzuk az első görbe szakaszaira, a harmadiknál pedig a második szakaszaira. Készíts `nyíl(n,h)` eljárást a `h` hosszúságú szakaszból kiinduló `n`-edik nyílhegygörbe rajzolására!

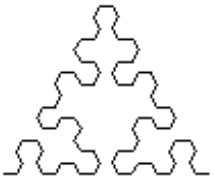


`nyíl(2,100)`

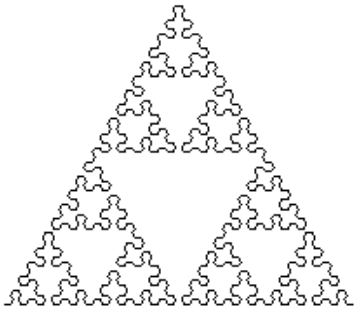


`nyíl(3,100)`

`nyíl(4,100)`



`nyíl(5,100)`



`nyíl(7,100)`

```
import turtle

def nyil(n,h):
 if n==1:
    turtle.forward(h)
 else:
    turtle.left(60)
    nyilj(n-1,h/2)
    turtle.right(60)
    nyil(n-1,h/2)
    turtle.right(60)
    nyilj(n-1,h/2)
    turtle.left(60)
```

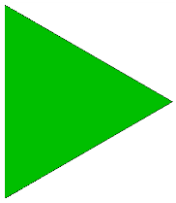
```
def nyilj(n,h):
 if n==1:
     turtle.forward(h)
 else:
     turtle.right(60)
     nyil(n-1,h/2)
     turtle.left(60)
     nyilj(n-1,h/2)
     turtle.left(60)
     nyil(n-1,h/2)
     turtle.right(60)

turtle.speed(0)
turtle.left(90)
nyil(5,100)
```

Elérhető: *https://repl.it/@hbv/nyil*

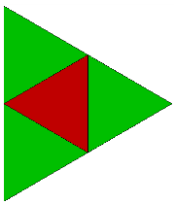
Feladat A Sierpinski háromszög úgy keletkezik, hogy egy `h` oldalhosszú háromszög alakú lapból kivágjuk a középső negyedrészt, majd a megmaradt három `h/2` oldalhosszú háromszögre ugyanezt alkalmazzuk. A kivágott részeket pirossal, a megmaradtakat pedig zölddel rajzoljuk.

Készíts `sier(db,h)` eljárást a `db`-edik, `h` oldalhosszú Sierpinski háromszög rajzolására!

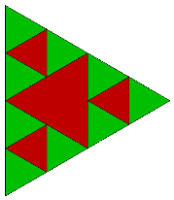
Megjegyzés: Gyakran alkalmazott trükk, hogy egy vezérlő eljárást (itt a külső háromszöget) hívjuk meg először, majd ez hívja meg rekurzív kivágandó háromszöget megvalósító eljárást!



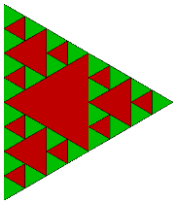
Sier(0,100)



Sier(1,100)



Sier(2,100)

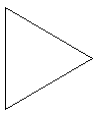


Sier(3,100)

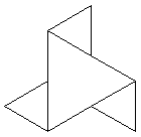
```
import turtle

def sier(db,h):
 haromszog(h,"green")
 sierpinszki(db,h)

def sierpinszki(db,h):
  if db>0:
    turtle.forward(h/2)
    turtle.right(60)
    haromszog(h/2,"red")
    turtle.left(60)
    turtle.backward(h/2)
    for i in range(3):
        sierpinszki(db-1,h/2)
        turtle.forward(h)
        turtle.right(120)

def haromszog(h,szin):
 turtle.fillcolor(szin)
 turtle.begin_fill()
 for i in range(3):
     turtle.forward(h)
     turtle.right(120)
 turtle.end_fill()

turtle.speed(0)
turtle.left(90)
sier(3,150)
```

Elérhető: *https://repl.it/@hbv/sier*

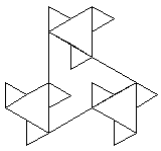
Feladat Az előállítás elve legyen az, hogy minden háromszögoldal egyik felén egy újabb háromszög jelenjen meg! Készíts `hszög(h,n)` eljárást, amely egy `h` oldalhosszúságú háromszögből kiindulva `n`-szer alkalmazza az oldalakra újabb ábrák elhelyezését!



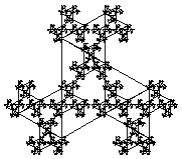
hszög(50,0)



hszög850,1)



hszög(50,2)



hszög(50,6)

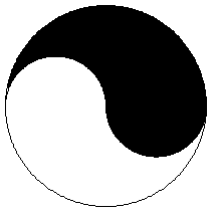
```
import turtle

def hszog(h,n):
 for i in range(3):
   harom(h,n)
   turtle.right(120)
```

```
def harom(h,n):
 if n==0:
   turtle.forward(h)
 else:
   turtle.forward(h/2)
   turtle.left(180)
   harom(h/2,n-1)
   turtle.right(120)
   harom(h/2,n-1)
   turtle.right(120)
   harom(h/2,n-1)
   turtle.left(60)
   turtle.forward(h/2)
```

```
turtle.speed(0)
hszog(50,2)
```

Elérhető:
https://repl.it/@hbv/hszog

Feladat A kínai *jinjang* szimbólum egy kör, amit egy két félkörvonal oszt két részre, az egyik fekete, a másik része pedig fehér színű. Készíts `jinjang(r,db)` eljárást, amely rekurzívan, a jinjang szimbólumba írt feleakkora kör alakú területekre az ábrának megfelelően db mélységig újra alkalmazza az eljárástt!



jinjang(100,1)

jinjang(100,2)

jinjang(100,3)

```
import turtle

def jingjang(r,db):
 alap(r,db%2)
 turtle.penup()
 turtle.right( 90)
 turtle.forward( r/4)
 turtle.right( 90)
 turtle.pendown()
 if db>1:
   jingjang(r/4,db-1)
 turtle.penup()
 turtle.left( 90)
 turtle.forward( r/4*3)
 turtle.right( 90)
 turtle.pendown()
 if db>1:
   jingjang(r/4,db-1)
```

```
def alap(r,melyik):
 if melyik==1:
  turtle.fillcolor("white")
 else:
  turtle.fillcolor("black")
 turtle.begin_fill()
 turtle.circle(r)
 turtle.end_fill()
 if melyik==1:
  turtle.fillcolor("black")
 else:
  turtle.fillcolor("white")
 turtle.begin_fill()
 turtle.circle(r,180)
 turtle.left(180)
 turtle.circle(-r/2,180)
 turtle.circle(r/2,180)
 turtle.end_fill()
 turtle.left(180)

turtle.speed(0)
turtle.left(90)
jingjang(200,3)
turtle.hideturtle()
```

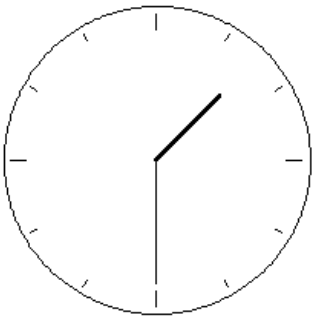
Elérhető:
https://repl.it/@hbv/jingjang

Számításokkal vezérelt rajzolás

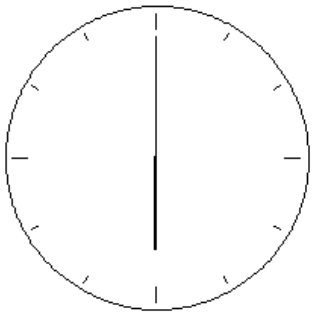
Időnként szükség lehet arra, hogy a rajz elkészítéséhez különböző számításokat is elvégezzünk. Néhány példán keresztül szeretnénk megvilágítani, hogy miről is van szó. Ha az a feladatunk, hogy rajzoljuk ki egy analóg óra számlapját a helyes mutató állásokkal akkor, ha az éjfél óta eltelt időt adjuk meg percekben, akkor ki kell számolni, hogy hányadik órában járunk és az utolsó egész órától hány perc telt el. Nézzük meg a néhány további példát is megvalósításukkal együtt!

Feladat Készíts `órakép(o,p)` `eljárást`, amely az alábbi óralapot képes rajzolni, ahol az `o` paraméter az óra értékét adja meg, a `p` paraméter pedig a perceket.

Megjegyzés: A mutató körbefordulása közben 360 fokot tesz meg. Mivel egy órában 60 perc van, a nagymutató egy perc alatt 6 fokot fordul el. A kismutató 12 óra alatt fordul körbe, azaz fordul 360 fokot. Egy óra alatt a kismutató így 30 fokot, egy perc alatt 0,5 fokot fordul. Ezek alapján már kiszámítható egy adott időben az elfordulás szöge, ha meghatározzuk, hogy hány perc telt el éjfél óta (összesen hány fokot kellett elfordulnia).



Órakép(13,30)

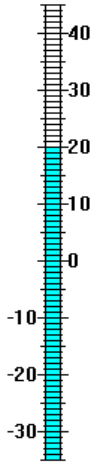


órakép(6,0)

```
import turtle

def orakep(o,p):
 beosztas(100)
 turtle.penup()
 turtle.right(90)
 turtle.backward(100)
 turtle.left(90)
 turtle.pendown()
 mutato(80,1,6*p)
 mutato(60,2,(60*o+p)/2)

def beosztas(h):
 for i in range(12):
   if i%3==0:
     turtle.pensize(2)
     vonal(10)
   else:
     turtle.pensize(1)
     vonal(5)
   turtle.circle(100,30)

def vonal(h):
  turtle.right(90)
  turtle.backward(h)
  turtle.forward(2*h)
  turtle.backward(h)
  turtle.left(90)

def mutato(h,v,f):
 turtle.pensize(v)
 turtle.right(f)
 turtle.forward( h)
 turtle.backward( h)
 turtle.left( f)

turtle.speed(0)
turtle.left(90)
turtle.hideturtle()
orakep(24,0)
```

Elérhető: *https://repl.it/@hbv/ora*

Feladat Egy hőmérő –35 és +45 fok közötti hőmérséklet mérésére alkalmas.



Készíts eljárást `hőmérő(fok)` néven, amely kirajzolja a hőmérőt, fokonként beosztást rajzol rá, 5 fokonként hosszabb, a 0-ra végződő fokoknál pedig még hosszabb vonallal! A 0-ra végződő fokokat számmal is kiírja a hőmérő mellé, a negatívakat turtle.left(, a pozitívokat pedig turtle.right(. A hőmérő aljától a `fok` magasságig a higanyszálat is belerajzolja.

Megjegyzés: A higanyszál aktuális magasságának kiszámításánál figyelembe kell venni, hogy negatív értékeket is mérünk, így egy eltolást kell alkalmaznunk. Mivel -35 foknál lesz a szál magassága nulla, a transzformációban az aktuális fok értékhez hozzá kell adnunk 35-öt. A feliratozásnál pedig arra kell figyelni, hogy negatív értékeknél a hőmérő bal oldalán, pozitívaknál pedig a jobb oldalon kell elhelyezni a skála értékeket.

```
import turtle

def homero(fok):
 teglalap(80*4,10)
 higany(fok)
 vonal(3)
 for i in range(4):
  vonal(1)
 vonal(5)
 for i in range(7):
   for i in range(4):
     vonal(1)
   vonal(3)
   for i in range(4):
     vonal(1)
   vonal(5)
 for i in range(4):
   vonal(1)
 turtle.penup()
 vonal(3)
 turtle.backward( 80*4)
 feliratoz(-30,40)

def higany(fok):
 turtle.fillcolor("aquamarine")
 turtle.begin_fill()
 teglalap((35+fok)*4,9)
 turtle.end_fill()
 turtle.fillcolor("black")

def teglalap(x,y):
 for i in range(2):
   turtle.forward(x)
   turtle.right(90)
   turtle.forward(y)
   turtle.right(90)
```

```
def feliratoz(tol,ig):
 turtle.forward(12)
 while tol<=ig:
   if tol<0:
     cimke(tol,-25)
   else:
     cimke(tol,15)
   if tol<=ig:
     tol+=10
     turtle.forward(2*5*4)

def vonal(h):
 turtle.right(90)
 turtle.backward(h)
 turtle.forward(10+2*h)
 turtle.backward(10+h)
 turtle.left(90)
 turtle.forward(4)

def cimke(hom,mennyit):
 turtle.penup()
 turtle.right(90)
 turtle.forward(mennyit)
 turtle.left(90)
 turtle.pendown()
 turtle.write(hom)
 turtle.penup()
 turtle.right(90)
 turtle.backward(mennyit)
 turtle.left(90)
 turtle.pendown()

turtle.speed(0)
turtle.left(90)
turtle.hideturtle()
homero(20)
```

Elérhető:

https://repl.it/@hbv/homero

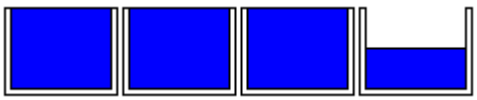
Feladat Egy négyzet alapú, adott magasságú kádba vizet töltünk. A kád fala három egység vastag. Ha a víz több, mint amennyi egyetlen kádba belefér, akkor újabb kádakba kerül a víz. A kádak az előzőtől turtle.right(három egység távolságra helyezkednek el.

Írd meg a `kád(szél,mag,víz)` `eljárást`, amely megrajzolja a kád(ak)at! A kád szélességét és magasságát cm-ben, a vízmennyiséget literben adjuk meg!

Megjegyzés: A kád térfogatát a **V=szél*szél*mag** képlettel számítjuk.

kád 100 80 800

kád 100 80 500



kád 50 40 350

```
def kad(szel,mag,viz):
 ureskad(szel,mag)
 if viz>mag*szel*szel/1000:
    ont(szel,mag)
    turtle.penup()
    turtle.right( 90)
    turtle.forward( szel+9)
    turtle.left( 90)
    turtle.pendown()
    kad(szel,mag,viz-
mag*szel*szel/1000)
    turtle.penup()
    turtle.right( 90)
    turtle.backward( szel+9)
    turtle.left( 90)
    turtle.pendown()
 else:
    ont( szel,
viz*1000/(szel*szel))
```

```
import turtle

def ureskad(szel,mag):
 turtle.right( 90)
 turtle.forward( szel)
 turtle.left( 90)
 turtle.forward( mag)
 turtle.right( 90)
 turtle.forward( 3)
 turtle.right( 90)
 turtle.forward( mag+3)
 turtle.right( 90)
 turtle.forward( szel+6)
 turtle.right( 90)
 turtle.forward( mag+3)
 turtle.right( 90)
 turtle.forward( 3)
 turtle.right( 90)
 turtle.forward( mag )
 turtle.left( 180)

def ont(szel,mag):
  turtle.fillcolor("blue")
  turtle.begin_fill()
  for i in range( 2):
    turtle.forward( mag)
    turtle.right( 90)
    turtle.forward( szel)
    turtle.right( 90)
  turtle.end_fill()

turtle.speed(0)
turtle.left(90)
kad(100,80,500)
#kad(50,40,350)


Elérhető: https://repl.it/@hbv/kad
```

Feladat A pénzkifizetéseknél érdemes a lehető legkevesebb darabszámú pénzzel kifizetni a kívánt összeget. A pénzeket (100, 200, 500, 1000, 2000, 5000, 10000, 20000 Ft) különböző színű téglalapokkal jelöljük (tetszőleges), melyekre az összeget is felírjuk. *Vegyük észre, hogy itt egy vegyes alapú számrendszerről van szó, ahol a számrendszer alapszámait a pénzcímletek jelentik!*

Írd meg a `címlet(ft)` eljárástt, amely megrajzolja az `ft` összeg kifizetéséhez szükséges pénzeket! Az azonos címleteket egymás mellé rajzold!

Megjegyzés. A nyolc pénzféle egyetlen eljárásban is elkészíthető. A címletezésnél mindig a lehető legnagyobb címletű pénzekből kell összeállítani az összeget! A megoldás menete is ezt a gondolatot követi, nagyság szerint csökkenő sorrendben megpróbálja az adott címletet felhasználni a kifizetéshez, majd a maradék pénzzel ugyanezt a folyamatot végigcsinálni, amíg a címletezendő pénz el nem fogy.

címlet 500

címlet 1400

címlet 38800

```
import turtle

szinek=("red","aquama-
rine","green","orange","brown","pin
k","yellow","silver","gold")

def penz(szin, ft):
 turtle.fillcolor(szinek[szin])
 turtle.begin_fill()
 for i in range( 2):
   turtle.forward( 20 )
   turtle.right( 90)
   turtle.forward( 50)
   turtle.right( 90)
 turtle.end_fill()
 turtle.fillcolor("black")
 turtle.penup()
 turtle.forward(5)
 turtle.right(90)
 turtle.forward(10)
 turtle.left(90)
 turtle.write(str(ft))
 turtle.right(90)
 turtle.backward(10)
 turtle.left(90)
 turtle.backward(5)
 turtle.pendown()

def leptet():
 turtle.penup()
 turtle.right( 90)
 turtle.forward( 60)
 turtle.left( 90)
 turtle.pendown()
```

```
def cimlet(ft):
 if ft>=20000:
   penz(8,20000)
   leptet()
   cimlet(ft-20000)
 elif ft>=10000:
    penz(7,10000)
    leptet()
    cimlet(ft-10000)
 elif ft>=5000:
    penz(6,5000)
    leptet()
    cimlet(ft-5000)
 elif ft>=2000:
    penz(5,2000)
    leptet()
    cimlet(ft-2000)
 elif ft>=1000:
    penz(4,1000)
    leptet()
    cimlet(ft-1000)
 elif ft>=500:
    penz(3,500)
    leptet()
    cimlet(ft-500)
 elif ft>=200:
    penz(2,200)
    leptet()
    cimlet(ft-200)
 elif ft>=100:
    penz(1,100)
    leptet()
    cimlet(ft-100)

turtle.left(90)
turtle.speed(0)
cimlet(33700)
```

Elérhető: *https://repl.it/@hbv/penz*

Szöveggel, listákkal vezérelt rajzolás

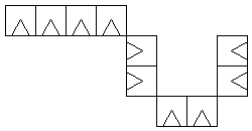
Eddigi feladatainkban a rajzaink paraméterszáma ismert volt már a fejlesztési idő során – tudtuk előre, hogy pontosan milyen alakú ábrát kívánunk rajzolni, milyen színnel. Amennyiben ezeket az adatokat nem tudjuk előre, akkor valamilyen összetett adatszerkezetre, sorozatra van szükségünk a rajz elkészítéséhez. A Pythonban a sorozatokat listaként lehet ábrázolni, amelyek első eleméhez illetve első nélküli elemeihez könnyen hozzá lehet férni. Egy szót, mondatot, szöveget magát is listaként lehet felfogni – karakterek, szavak vagy mondatok listájaként.

A szövegben, szövegekben különböző parancsokat írhatunk le, amelyeket akár futás közben, interaktívan is kiadhatunk. Így készíthetünk például egyszerű rajzoló programot vagy akár rovásíró alkalmazást is. A szövegekben megadhatjuk egy sor vagy mozaik kirajzolandó elemeit, ilyenkor természetesen nem kell ezeknek valamilyen szabályszerűségnek megfelelően következni egymás után. Lássunk néhány példát szöveggel irányított rajzolásra.

Feladat Meandernek nevezik az olyan sormintákat, amelyek valamilyen szabályszerűség szerint kanyarognak. Készíts `alap(h)` eljárást, amely egy, az alábbi ábrának megfelelő alapelemet rajzol! A meander kanyargását paraméterekkel szeretnénk vezérelni. Készíts `meander(h,sz)` eljárást, amely az alapelemet az `sz` szöveg karakterei szerinti sorrendben ismétli. A meander először jobbra indul. Ha az **X** betű következik, akkor a haladási irányt megtartja; a **J** betű hatására az irány jobbra változik 90 fokkal, a **B** hatására pedig balra 90 fokkal.

alap(30)

```
import turtle

def meander(h,sz):
 if  sz!=[]:
   alap(h)
   mozdit(h,sz[0])
   meander(h,sz[1:])
 else:
   alap(h)

def alap(h):
 for i in range( 4):
   turtle.forward( h)
   turtle.right( 90)
 turtle.right( 90)
 turtle.forward( h/5)
 turtle.left( 60)
 for i in range( 2):
   turtle.forward( 3*h/5)
   turtle.right( 120)
 turtle.backward( h/5)
 turtle.forward( h)
 turtle.right( 90)
```



meander(15,"XXXJXBXBX")

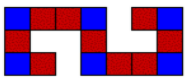
```
def mozdit(h,b):
 if b=="X":
   turtle.right( 90)
   turtle.forward( h)
   turtle.left( 90)
 if b=="J":
  turtle.right( 90)
  turtle.forward( h)
 if b=="B":
   turtle.right( 90)
   turtle.forward( h)
   turtle.left( 90)
   turtle.forward( h)
   turtle.left( 90)
   turtle.backward( h)

turtle.speed(0)
turtle.hideturtle()
turtle.left(90)
meander(15,"XXJXBXBX")
```

Elérhető:

https://repl.it/@hbv/meander

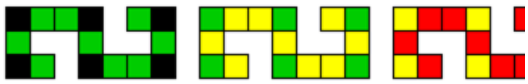
Feladat Készítsd el az alábbi sormintát lerajzoló `görögsor(oldal,leírás)` nevű eljárást, amelynek váltakozó színeit egy szöveges paraméterrel (`leírás`) adhatod meg, az alapelem hosszát pedig az `oldal` paraméterrel. A sor egy eleme négyzetekből álló minta, amely két különböző színnel van kiszínezve `görögös(oldal,szín1,szín2)`. Mindig a sarokszínek azonosak egy sorelemben. Figyeld meg, hogy két egymást követő elemben a második elem sarokszíne, az első

elem kitöltő színe volt és az új kitöltőszínt a leíró paraméterből olvassuk ki. A `leírás`-ban a színek rendre p=piros, f=fekete, z=zöld, s=sárga, k=kék.



`görögös(20, "red","blue")`

```
import turtle

def gorogos(oldal,szin1,szin2):
 for i in range(2):
   negyzet(oldal,szin2)
   leptet(-oldal)
   negyzet(oldal,szin1)
   turtle.forward( 2*oldal)
   turtle.right(90)
 for i in range(2):
   negyzet(oldal,szin2)
   leptet(-oldal)
 negyzet(oldal,szin1)
 turtle.forward(2* oldal)
 turtle.right(90)
 negyzet(oldal,szin2)
 leptet(-oldal)
 negyzet(oldal,szin1)
 turtle.right(90)
 for i in range(2):
   negyzet(oldal,szin2)
   leptet(oldal)
 negyzet(oldal,szin1)
 turtle.forward(oldal)
 negyzet(oldal,szin2)
 turtle.forward(oldal)
 negyzet(oldal,szin1)
 turtle.left(90)
 negyzet(oldal,szin2)
 turtle.penup()
 turtle.forward(4*oldal)
 turtle.right(90)
 turtle.backward(2*oldal)

def negyzet(oldal,szin):
  turtle.fillcolor(szin)
  turtle.begin_fill()
  for i in range( 4):
    turtle.forward( oldal)
    turtle.right( 90)
  turtle.end_fill()
```



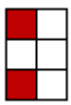
`görögsor(20,"fzspk")`

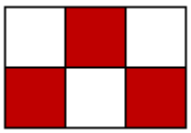
```
def leptet(oldal):
  turtle.right( 90)
  turtle.forward( oldal)
  turtle.left( 90)

def gorogsor(oldal,leiras):
 if len(leiras[1:])!=0:
   gorogos(oldal,szin(lei-
ras[0]),szin(leiras[1]))
   turtle.penup()
   turtle.right( 90)
   turtle.forward( 7*oldal)
   turtle.left(90 )
   turtle.pendown()
   gorogsor(oldal,leiras[1:])

def szin(sz):
 if sz=="f":
   return "black"
 if sz=="z":
   return "green"
 if sz=="k":
  return "blue"
 if sz=="s":
   return "yellow"
 if sz=="p":
  return "red"

turtle.speed(0)
turtle.left(90)
gorogsor(10,"fzspk")
```

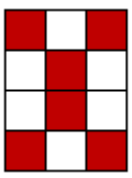
Elérhető:

https://repl.it/@hbv/gorogos

Feladat A `kép(sor,oszlop,oldal,lista)` eljárás egy `sor*oszlop`–os négyzetrács bizonyos elemeit színezi. A listában „`k`" jelenti a pirosra kiszínezendő mezőket, „`f`" a fehéren maradókat. A lista első eleme a bal alsó sarokban lévő négyzetet jelenti.







```
kép( 3,2,15,
[„k”,”f”, „f”,
„f”, „k”,”f”]
```

```
kép(2,3,30,[„k”, „f”,
„k”,„f”, „k”,”f”]
```

```
kép(4,3,20, ["k", "f",
"k","f","k","f", "f",
"k","f","k","f","k"])
```

```
import turtle

def kep(sor,oszlop,oldal,lista):
 for i in range( sor):
   for j in range( oszlop):
     if lista[i*oszlop+j]=="k":
       negyzet(oldal,True)
     else:
        negyzet(oldal,False)
     turtle.right(90)
     turtle.forward(oldal)
     turtle.left(90)
   turtle.right(90)
   turtle.backward(oszlop*oldal)
   turtle.left(90)
   turtle.forward(oldal)
 turtle.backward(oldal*sor)
```

```
def negyzet(oldal,fest):
  if fest:
    turtle.fillcolor("red")
    turtle.begin_fill()
  for i in range(4):
    turtle.forward(oldal)
    turtle.right(90)
  if fest:
    turtle.end_fill()

turtle.speed(0)
turtle.left(90)
kep(4,3,20,["k","f","k","f","k","f","
f","k","f","k","f","k"])
```

Elérhető: *https://repl.it/@hbv/festes*

Feladat A Morze ábécében az egyes betűket hosszú és rövid vonalakkal jelöljük. Az egyes karaktereknek megfelelő pontok és vonalak kirajzolását vezéreljük majd a parancsunkkal. A következő feladatban az alábbi betűket használjuk

a • – o – – – p • – – • r • – • t –

Készíts egy `szórajzol(szó)` eljárást, amely egy szó morzejeleit rajzolja úgy, hogy az egyes betűk közé két üres helyet tesz!

`szórajzol("por”)` -> • – – • – – – • – •

`szórajzol("tar”)` -> – • – • – •

Megjegyzés: Karakterenként dolgozzuk fel a szót – karakterenként rajzoljuk ki a morzejeleket.

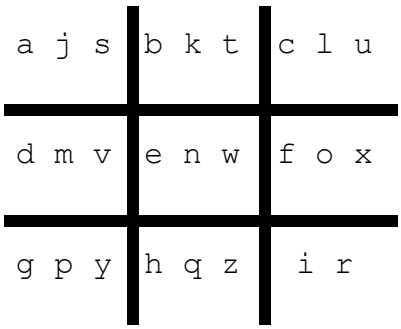
```
import turtle

def szorajzol(szo):
  for b in szo:
    betu(b)
    leptet()

def betu(b):
 if b=="a" :
   rovid()
   hosszu()
 if b=="o" :
   hosszu()
   hosszu()
   hosszu()
 if b=="p" :
   rovid()
   hosszu()
   hosszu()
   rovid()
 if b=="r" :
   rovid()
   hosszu()
   rovid()
 if b=="t" :
   hosszu()
```

```
def rovid():
 turtle.dot(5)
 leptet()

def hosszu():
  turtle.forward(15)
  leptet()

def leptet():
  turtle.penup()
  turtle.forward(15)
  turtle.pendown()

turtle.speed(0)
turtle.hideturtle()
szorajzol("por")
```

Elérhető:

https://repl.it/@hbv/morze

Feladat Achiram úgy készített titkosírást, hogy a betűket egy táblázatba helyezte és a betűnek megfelelő mező szegélyvonalát rajzolta le.



Készíts `achiram(szó)` eljárást, amely titkosírást készít egy ilyen kulcstábla alapján! Az azonos részben lévő betűket (pl. a j s) úgy különböztetjük meg, hogy az első betű fekete, a második piros, a harmadik pedig kékszínű legyen! (Így például a cica szó csupa fekete jellel rajzolható.) A program a beírt szöveget rajzolja meg a szabályok szerint!

`achiram "cica`

`archiram "kata`

Megjegyzés: A betűk kódolva vannak. A kódok egymás után következő egész számok. Az a betű kódja 97! Ennek megfelelően a 9-cel való osztással lehet a betű helyét a táblázatban meghatározni. A mezőn belüli betűsorszám alapján választunk színt! A betűsorszámból az is kiszámolható, hogy a táblázat melyik mezőjébe kell elhelyezni, azaz most milyen keretet kell kirajzolni.

```
import turtle

def vonal(a,sz):
 szinez(sz)
 if a<6:
   turtle.right( 90)
   turtle.forward( 20)
   turtle.backward( 20)
   turtle.left( 90)
 if a>2:   turtle.penup()
   turtle.forward( 20)
   turtle.pendown()
   turtle.right( 90)
   turtle.forward( 20)
   turtle.backward( 20)
   turtle.left( 90)
   turtle.penup()
   turtle.backward( 20)
   turtle.pendown()

 if a % 3!=0 :
   turtle.forward( 20)
   turtle.backward( 20)
```

```
def szinez(sz):
 if sz==0:
   turtle.pencolor("black")
 elif sz==1:
    turtle.pencolor("red")
 elif sz==2:
    turtle.pencolor("blue")
 turtle.pensize(5)

def achiram(szo):
 for b in szo:
   rajzol(ord(b))

def rajzol(kod):
 vonal((kod-97) %9 , (kod-97)// 9)

def leptet():
 turtle.penup()
 turtle.right( 90)
 turtle.forward( 30)
 turtle.left( 90)
 turtle.pendown()


turtle.speed(0)
turtle.left(90)
#achiram("cica")
achiram("kata")
```

```
if a %3!=2:
  turtle.penup()
  turtle.right( 90)
  turtle.forward( 20)
  turtle.pendown()
  turtle.left( 90)
  turtle.forward( 20)
  turtle.backward( 20)
  turtle.right( 90)
  turtle.penup()
  turtle.backward( 20)
  turtle.left( 90)
  turtle.pendown()
```

Elérhető:

https://repl.it/@hbv/titkos

Szövegmanipuláció

Feladat A Bumm nevezetű társasági játékban a játékosok egy kiinduló számtól kezdve egyesével mondják a következő számot, DE van egy tiltott szám, s ha olyan szám következne, amelyiknek valamelyik számjegye éppen a tiltott szám, vagy a kimondandó szám osztható a tiltott számmal, akkor a szám helyett azt kell mondani, hogy „Bumm”. Aki eltéveszti, zálogot ad.

Írd meg a `bumm(ettől,eddig,tiltott)` eljárást, amely kiírja a játékban elhangzottakat! A program a „Bumm” helyett két felkiáltójelet írjon!

`bumm(7,14,3) eredménye 7 8 !! 10 11 !! !! 14`

Végighalad a megadott számintervallumon és az eredménylistába írja az aktuális számot vagy a bummot.

```
def bumm(honnan,meddig,tiltott):
  valasz=""
  for szam in range(honnan,meddig):
    valasz+=kiir(szam,tiltott)+" "
  print(valasz)

def kiir(szam,tiltott):
  if (szam % tiltott==0) or
(str(tiltott) in str(szam)):
    return "!!"
  else:
   return str(szam)

bumm(7,34,3)
```

Elérhető:

https://repl.it/@hbv/bumm

Feladat Egy titkosírást úgy készítenek, hogy egy táblázat alapján az ábécé betűit más betűkre cserélik. Készíts `kódol_dekódol(szó,kódole` függvényt, amelyek az alábbi 2 konstans szöveg (ábécé, kódok)segítségével megadja a paraméterként megadott szó kódját, illetve a kódolt szó eredeti alakját! A megoldásban csak kisbetűkkel kell foglalkozni!

`kódol_dekódol("alma”,True) eredménye „öőúö”`

`kódol_dekódol("whwk,False) eredménye eper`

Megjegyzés: kódok="öüóűqwertzuiopőúasdfghjkléáíyxcvbnm”,
ábécé="aábcdeéfghiíjklmnoóöőpqrstuúüűvwxyz”

```
def kodol_dekodol(szo,kodoloe):
 valasz=""
 for b in szo:
   if kodoloe:
      valasz+=helyette(b,abece,kodok)
   else:
      valasz+=helyette(b,kodok,abece)
 return valasz
```

```
def helyette(b,abc,kod):
  return kod[abc.index(b)]

kodok="öüóűqwertzuiopőúasd-
fghjkléáíyxcvbnm"
abece="aábcdeéfghiíjklmnoóöőpqrs-
tuúüűvwxyz"
print(kodol_dekodol("alma",True))
print("Dekódol: "+ko-
dol_dekodol("whwk",False))
```

Elérhető:

https://repl.it/@hbv/kodolo

Feladat A 21-es kártyajátékot magyar kártyával játsszák. Először mindenki kap két lapot, majd felváltva húznak, amíg kérnek. A lapok értéke 2,3,4,7,8,9,10,11 lehet.

Készíts `kinyert(pakli,húzások)` függvényt, amely egy 21-est játszó párosnál megadja, hogy ki nyert! A `pakli` listában a kártyák értékeit találjuk. A `húzások` listában az egymás után

következő húzásokhoz tartozó személyek azonosítója áll. [„A”,”A”,”B”, „B”] azt jelenti, hogy 2 alkalommal A kapott lapot a pakliból, majd kettőt B is kap. Ha valaki lapjainak az összege 21-nél több, az vesztett („befuccsolt”), egyébként az nyer, aki lapjainak nagyobb az értéke.

```
Kinyert([2,3,7,2,10,3,4 ],[„A”,”A”,”B”,”B”,”A”,”B”])
eredménye A lapjai 15, B lapjai 12, tehát A nyert.
```

Megjegyzés: Összeszámoljuk a két játékos által kapott lapok pontjait és meghívjuk a kiértékelést végző nyerés eljárást.

```
def kinyert(pakli,huzasok):
 nyeres(osszeszamol("A",pakli, huzasok), osszeszamol("B",pakli, huzasok))

def osszeszamol(ki,pakli,huzasok):
  ertek=0
  for i in range(len(huzasok)):
    if huzasok[i]==ki:
      ertek+=pakli[i]
  return ertek
```

```
def nyeres(A,B):
 valasz=""
 if (A>21) and (B>21):
   valasz="befuccsoltak mind a ketten"
 if  (A>21) and (B<=21):
   valasz="A fuccs, nyert a B"
 if  (A<=21) and (B>21):
   valasz="B fuccs, nyert az A"
 if  (A<=21) and (B<=21) and (A<B):
   valasz="nyert a B"
 if  (A<=21) and (B<=21) and (A>B):
   valasz="nyert az A"
 if (A<=21) and (B<=21) and (A==B):
   valasz="döntetlen"
 print(valasz)

kinyert([2,3,7,2,10,3,4],["A","A","B","B","A","B"])
```

Elérhető:

https://repl.it/@hbv/kartya

Feladat Készíts függvényt `szűr(mondat)` néven, amely egy magyar mondatból előállít egy olyan mondatot, amelyben az eredeti mondat ékezetes betűket tartalmazó szavait annyi * karakterrel helyettesíti, ahány betűből álltak! A megoldás részeként készíts egy függvényt `ékezetes?(szó)` néven, amely egy tetszőleges szóról megmondja, hogy tartalmaz-e ékezetes betűket!

```
szűr(„ez is jó hír lesz”)     ⇒  „ez is ** *** lesz”
ékezetes?( "ez”)              ⇒  hamis
ékezetes?("jó”)               ⇒  igaz
```

Megjegyzés: Vizsgáljuk végig szavanként, hogy van-e az adott szóban ékezetes betű! Például a „*”*5 5 darab *-ot fűz egymás mellé!

```
def szur(mondat):
  eredmeny=[]
  for szo in mondat.split():
    if ekezetese(szo):
      eredmeny.append(  eredmeny.append("*"*len(szo))
    else:
      eredmeny.append(szo)
  return  " ".join(eredmeny)
```

```
def ekezetese(szo):
 ekezet=False
 if len(szo)==0:
   return False
 i=0
 while i<len(szo) and not ekezet:
   if szo[i] in ekezetesbetuk:
     ekezet=True
   i+=1
 return ekezet

ekezetesbetuk="áéíóöőüűúÁÉÍÖŐÜŰÚ"
print(szur("ez is jó hír lesz"))
```

Elérhető: *https://repl.it/@hbv/szur*

Feladat Egy mondatban szavak és pozitív egész számok is szerepelnek. Készíts `átalakít(mondat)` eljárást, amely a mondatban a 100-nál kisebb számokat lecseréli betűkkel írtakra, a nagyobbakat pedig változatlanul hagyja!

```
átalakít(„A kertben 26 kutya és 4 macska játszott 100 egérrel.”) ⇒
„A kertben huszonhat kutya és négy macska játszott 100 egérrel.”
```

Megjegyzés: Szavanként dolgozzuk fel a mondatot, ha számot találunk, helyette a neki megfelelő szöveges formát írunk! Az átalakítás során figyeljünk a kivételes esetekre (10,20)!

```
def atalakit(mondat):
 atalakitott=""
 for szo in mondat.split():
   if szo.isdigit():
     atalakitott+=szovegge(szo)+" "
   else:
     atalakitott+=szo+" "
 return atalakitott

def szovegge(szo):
 if len(szo)==1:
   return egyes[int(szo)]
 if 2<len(szo):
   return szo
 if szo=="10":
   return "tíz"
 if szo=="20":
   return "húsz"
 if 0==szo[-1]: #a -1 jelenti az
utolsót!
   return tizes[int(szo[0])]
 return ti-
zes[int(szo[0])]+egyes[int(szo[1])]
```

```
tizes=("tizen","huszon", "har-
minc","negyven","ötven","hat-
van","hetven", "nyolcvan", "ki-
lencven")
egyes=("nulla","egy","kettő","há-
rom","négy","öt",
"hat","hét","nyolc","kilenc")
print(atalakit("  "))
print(atalakit("A kertben 26 kutya
és 4 macska játszott 100 egér-
rel.") )
```

Elérhető:

https://repl.it/@hbv/atalakit

Feladat Egy magyar mondat végén pont, felkiáltójel vagy kérdőjel van.

Készíts függvényt `darab(szoveg)` néven, amely megadja, hogy a paraméterében hány magyar mondat van! Ezután add meg a kijelentő, a kérdő és a felkiáltó (felszólító, óhajtó) mondatokat!

```
def darab(szoveg):
  db=0 #mondatok száma
  dbf=0 #felkiáltó
  dbk=0 #kérdő
  dba=0 #állító
  for betu in szoveg:
    if betu in irasjelek:
      db+=1
      if betu=="?":
        dbk+=1
      elif betu=="!":
        dbf+=1
      else:
        dba+=1
  return (db,dba,dbk,dbf)
```

```
irasjelek=".?!"
print(darab("Egy. Kettő? Három? Ez a
negyedik!"))
```

Elérhető:

https://repl.it/@hbv/mondatok