05. VÉLETLEN SZÁMOK GENERÁLÁSA/HASZNÁLATA

A programozás feladatok nagyon sok hányadában van szükségünk **véletlen számok generálására**! Ha példát kell mondani, akkor általában a lottó sorsolás vagy a kockadobást szokták említeni.
Először az első sor(ok)ban meg kell adni a „math" függvénykönyvtár meghívásához hasonlóan, be kell gépelni a `from random import *` parancsot!
Python programozásban véletlen egész számokat a **randrange** utasítással készíthetünk. Az **utasítás után meg kell adni egy (n) számot**, mely akkor egy **0 és n-1 közötti véletlen számot generál**!
Egy 0 és 1 közötti véletlen számot a random utasítással generálhatunk! (A szám soha nem lehet 1!)
Ezekkel az utasításokkal generált számokkal műveleteket végeznetünk. Hozzá adhatunk számokat, kivonhatunk belőlük, szorozhatjuk, stb…
Tehát ezek alapján, ha **kockadobást** modelleznénk, akkor a randrange(6)-al csak egy 0-5 közötti számot hoznánk létre, de ha **randrange(6)+1**-et írnánk, akkor 1-6-ig generálna véletlen számot!

A **véletlen számok készítése nem tartozik a Python alaputasításai közé**, ezért itt is az import utasítást kell használni a program elején. Mégpedig a „**from random import** *" - ot kell begépelni! (A * azt jelenti, hogy minden hozzá tartozó függvényt elérhetővé tesz.)
Egyébként a randrange utasításnál meg lehet adni egy kezdő értéket, egy felső határt és hogy milyen lépésközzel ugorjon! `randrange(start, stop, step)`

(05a.py)
Hozzunk létre egy új python fájlt, a neve legyen 05a.py!

Készítsünk programot, melyben kettő darab (az előbb említett) kockadobást modellezed, majd utána két 0 és 1 közötti véletlen számot generálj!

- A program elején írjuk be az első sorba a `from random import *` utasítást!
- Generálj a feladatban leírt módon véletlen számokat a változókba!
- A kiírt számok formátumát a következő képen állítsd be: az első kettő egész, a második kettő valós, négy tized pontossággal legyen ábrázolva!
- A számok egymás mellett jelenjenek meg a minta szerint!

```
05a.py
1  from random import *
2  a = randrange(6)+1
3  b = randrange(6)+1
4  c = random()
5  d = random()
6  print( "%d, %d, %.4f, %.4f" % (a,b,c,d))

Run:  05a
   C:\Users\kolmank\AppData\Local\Prog
   5, 6, 0.3135, 0.9907

   Process finished with exit code 0
```

(05b.py)
Hozzunk létre egy új python fájlt, a neve legyen 05b.py!

Készítsünk programot, melyben generálunk öt olyan számot, ami 100 és 200 között van!
A számok 10-esével legyenek léptetve!

- A program elején írjuk be az első sorba a `from random import *` utasítást!
- Generálj a feladatban leírt módon véletlen számokat a változókba!
- Végül Írassuk ki a minta szerint egyszerűen egymás mellé a számokat!

```
05b.py
1  from random import *
2  sz1=randrange(100,200,10)
3  sz2=randrange(100,200,10)
4  sz3=randrange(100,200,10)
5  sz4=randrange(100,200,10)
6  sz5=randrange(100,200,10)
7  print(sz1,sz2,sz3,sz4,sz5)

Run:  05b
   C:\Users\kolmank\AppDa
   140 170 170 110 190
```

06. LOGIKAI VÁLTOZÓK HASZNÁLATA

A logikai változók értéke igaz (**True**) vagy hamis (**False**) lehet. Kötelezően nagy betűkkel használjuk! Egy logikai változót, egy eldöntendő kérdésre adott válasz tárolására használhatjuk.
Az Excel táblázatkezelő programnál tanult logikai függvények, az és, vagy, nem itt is használhatók.
A logikai értékeket adhatunk, közvetlen megadással: A=True; vagy kifejezés eredményeként: B=(C<D)
Nézzük át a függvények igazságtábláit!

A	B	A ^ B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A v B
0	0	0
0	1	1
1	0	1
1	1	1

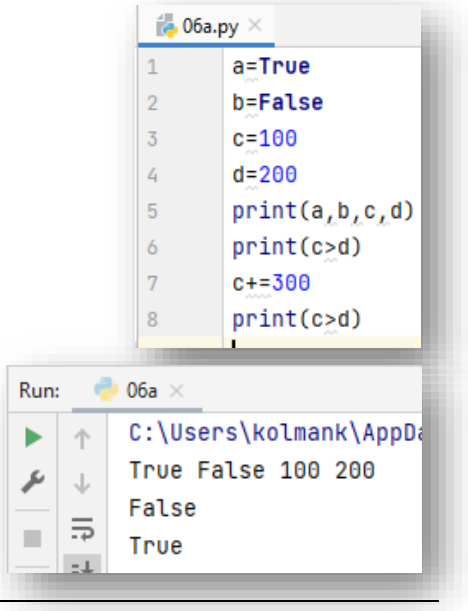
A	NEM(A)
0	1
1	0

- **Az ÉS függvény: Akkor igaz, ha minden állítás igaz.**
- **A VAGY függvény: Akkor igaz, ha legalább az egyik állítás igaz.**
- **A NEM függvény: Az ellenkezőjére változtatja az értéket!**

(06a.py)
Hozzunk létre egy új python fájlt, a neve legyen 06a.py!

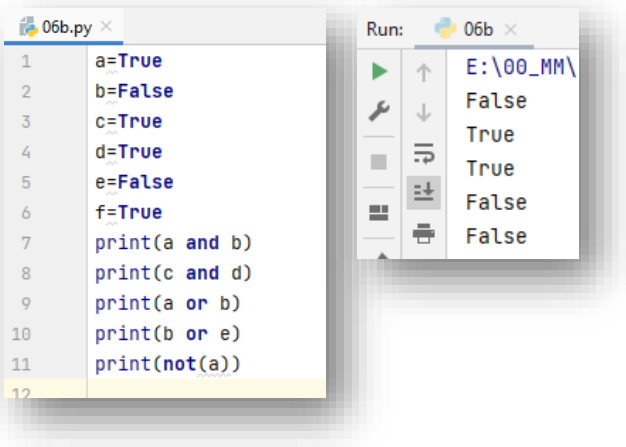
Próbáljuk ki a logikai utasításokat!

- Adjuk értéket a,b,c,d változóknak a minta alapján!
- Írassuk ki az eredeti értékeket!
- Nézzük meg, hogy a c értéke nagyobb-e mint a d?
- Növeljük meg a c értékét 200-al!
- Majd újból nézzük meg, hogy a c értéke nagyobb-e a d-nél?



(06b.py)
Hozzunk létre egy új python fájlt, a neve legyen 06b.py!

Készítsünk programot, melyben True vagy False értéket adunk a,b,c,d,e,f változóknak!
Majd képen látható módon nézzük meg milyen értéket kapnak, ha futtatjuk a kódunkat!



07. GRAFIKUS MEGJELENÍTÉS

Általában a programozási ismereteket az úgynevezett „teknős” útjának rajzolásával szokták kezdeni. Mert látványosabb mint karakteres felületen angol nyelven programozni.
A Python programban is van grafikus felület, ahol sok olyan dolgot ki tudunk próbálni, ami kell az alapok megismeréséhez.

Ahhoz, hogy grafikus módban dolgozzunk, szükség van a program elején begépelni a „**from turtle import ***” parancsot! (Mint a „math” és a „random” függvénykönyvtárnál.)
Az első utasítások melyekkel meg kell ismerkedni:

- **előre menni/haladni: forward(képpont_száma)**
- **fordulni jobbra: right(fok)**
- **fordulni balra: left(fok)**
- **képernyőt törölni: reset()**
- **rajzoló szín megadása: color(„alap_szín_angolul”)**
- **kitöltő szín megadása: fillcolor(„alap_szín_angolul”)**
- **a kitöltendő alakzat előtt és után: begin_fill() … end_fill()**
- **a rajzoló „ceruza” felemelésére az up(), letétele a down() utasítást használjuk**

(07a.py)
Hozzunk létre egy új python fájlt, a neve legyen 07a.py!

Az első grafikus feladatunkban, rajzoljunk egy téglalapot!

- Az első sorban hívjuk meg a rajzoláshoz szükséges eszközöket!
- Menjünk előre 300 képpontot!
- Forduljunk el balra 90 fokot!
- Majd menjünk előre 100 képpontot!
- Aztán gépeljünk be utasításokat, hogy visszaérjünk a kiindulópontba!
- Futtassuk le a programunkat!

07a.py

```
1  from turtle import *
2  forward(300)
3  left(90)
4  forward(100)
5  left(90)
6  forward(300)
7  left(90)
8  forward(100)
```

(07b.py)
Az előző programunkat mentsük el másként 07b.py néven!

Az előző programunkat bővítsük ki, hogy egy zöld színnel rajzolt, sárga kitöltésű téglalapot kapjunk!

- Maradjon az első sorban meghívott rajzoláshoz szükséges eszköztár!
- Töröljük a képernyőt!
- Adjuk meg a zöld rajzolószínt!
- A kitöltőszint állítsuk be sárgára!
- Ahhoz, hogy a kitöltés működjön, meg kell adni az alakzat rajzolásának kezdetekor a begin_fill() utasítást!
- Majd a megrajzolt alakzat végén a lezáró end_fill() utasítást!
- Futtassuk le a programunkat!
- Javítsuk az esetleges hibákat!

07b.py

```
1  from turtle import *
2  reset()
3  color("green")
4  fillcolor("yellow")
5  begin_fill()
6  forward(300)
7  left(90)
8  forward(100)
9  left(90)
10 forward(300)
11 left(90)
12 forward(100)
13 end_fill()
```

Bővítsük az utasítások ismeretét!

(07c.py)

Hozzunk létre egy új python fájlt, a neve legyen 07c.py!

Ebben a programban csak egy egyszerű téglalapot rajzolunk ki, de közben megtanulunk pár hasznos utasítást!

- A program elején **meghívjuk a grafikus módot**! Az első sor megmondja a Pythonnak, hogy töltse be a **turle nevű modult**.
- Az előző modul két új típust hoz be a látótérbe, amelyeket ezután használhatunk: a **Turtle**, azaz **teknőc típust** és a **Screen**, azaz **képernyő típust**. A turtle.Turtle szövegben a pont jelölés azt jelenti, hogy „a Turtle típus, ami a turtle modulban van definiálva”. (Megjegyzés: a Python érzékeny a kis és nagy betűkre, így a modul neve t-vel írva különbözik a Turtle típus nevétől.)
- A 3. sorban létrehozunk, nevet adunk az ablaknak, amiben rajzolunk!
- A 4. sorban nevet adunk annak a „teknőcnek” amivel rajzolunk.
- Majd utasításokkal irányítjuk a teknőcünket. Rajzolunk egy téglalapot.
- A 15. sorban olyan utasítást adunk ki, amely vár a felhasználóra, hogy bezárja az ablakot!
- Tehát az utolsó sor is fontos szerepet játszik: az ablak változó hivatkozik az ablakra, ahogy fentebb bemutattuk. Ha meghívjuk a **mainloop** nevű **metódus**át, belép egy állapotba, ahol egy **eseményre vár** (mint például a billentyűleütés vagy egérmozgatás és kattintás).

07c.py

```
1  import turtle                    # Lehetővé teszi a teknőc használatát
2
3  window = turtle.Screen()         # Hozzunk létre egy "játszóteret" a teknőcnek!
4  teki = turtle.Turtle()           # Hozzunk létre egy teknőcöt "teki" néven!
5
6  teki.forward(100)                # A teki menjen 100 egységet előre!
7  teki.left(90)                    # A teki forduljon 90 fokot!
8  teki.forward(50)
9  teki.left(90)
10 teki.forward(100)
11 teki.left(90)
12 teki.forward(50)
13 teki.left(90)
14
15 window.mainloop()                # Várj, amíg a felhasználó bezárja az ablakot!
```

(07d.py)

Hozzunk létre egy új python fájlt, a neve legyen 07d.py!

Ebben a programban rajzolunk kék színnel, és vastagabb „ceruzával” egy háromszöget!

Parancsok:

- color(„red”)
- pensize(3)

07d.py

```
1  import turtle
2
3  ablak = turtle.Screen()
4
5  teknoc = turtle.Turtle()
6  teknoc.color("red")          # Mond meg a teknoc-nek, hogy változtasson színt!
7  teknoc.pensize(3)            # Mond meg a teknoc-nek, hogy változtassa meg a tolla vastagságát!
8
9  teknoc.forward(200)
10 teknoc.left(120)
11 teknoc.forward(200)
12 teknoc.left(120)
13 teknoc.forward(200)
14
15 ablak.mainloop()
```