11. FELTÉTELES CIKLUSOK

Programozásban gyakran fordulnak elő olyan esetek, **amikor nem tervezhető előre a ciklusok lépésszáma**. Ezekben az esetekben használjuk a **feltételes ciklus**okat úgy, hogy a felhasználótól **bekérünk adatot, majd annak érvényességét vizsgáljuk és csak helyes adatok esetén megy tovább a program, különben újból bekérjük az adatot**. Tehát szükségünk van olyan ciklusra, amiben az ismétlések száma feltételhez kötött. Két fajta ciklust kell megemlítenünk: **elöltesztelő** és hátultesztelő **ciklus**. Mivel a Python csak elöltesztelő ciklust ismeri, ezért erről fogunk tanulni.

Elöltesztelő ciklus

Nézzünk meg egy példát!
A feladat két egész szám **legkisebb közös többszörös**ének megkeresése.

- Például az 5 és a 18 legkisebb közös többszörösét keressük meg!
- Ha elindulunk az 5 többszöröseivel:5, 10,15, 20, 25, 30, … 80, 85, 90,
- A 18 többszöröseivel: 18, 36, 54, 72, 90, (táblázat)
- Majd a program a legkisebb közös többszörösnél megáll. (t1=t2)
- A többszörösöket ügy képezzük, hogy hozzáadjuk az eredeti számot. (+5, +18)
- Nem fogunk egyik, vagy másik szám többszörösével előre rohanni, hanem mindig csak a kisebb többszöröst emeljük, hiszen annak van esélye utolérni a másikat.
- Tehát bekérünk két számot! (sz1, sz2).
- Majd egy t1 és t2 változóban tároljuk a számok többszörösét!
- Ezeket a többszörösöket hasonlítjuk össze egymással: t1>t2.
- Az elöltesztelő ciklusnál a **while** utasítást használjuk.
- Tehát a ciklus elejét a while kulcsszóval jelöljük, mögötte a bennmaradás feltételélt kell megadni, majd a vezérléshez tartozó szokásos kettőspontot adjuk meg.
- A ciklusmagot tagolni kell, beljebb kell kezdeni.

	t1	t2	
+5	5	18	
+5	10	18	
+5	15	18	
	20	18	+18
+5	25	36	
+5	30	36	
+5	35	36	
	40	36	+18
+5	45	54	
+5	50	54	
+5	55	54	
	60	54	+18
+5	65	72	
+5	70	72	
+5	75	72	
	80	72	+18
+5	85	90	
	90	90	

(11a.py)

Mondatszerű leírása a programnak a következő:

be: sz1, sz2,
t1:=sz1
t2:=sz2
ciklus amíg t1=t2
ha t1>t2 akkor t2:=t2+sz2
különben t1:=t1+sz1
elágazás vége
ciklus vége
ki: t1

Az elkészített **python program** a következő:

```
11a.py
1  sz1=int(input("Add meg az egyik számot: "))
2  sz2=int(input("Add meg a másik számot: "))
3  t1=sz1
4  t2=sz2
5  while t1 != t2:
6      if t1>t2:
7          t2=t2+sz2
8      else:
9          t1=t1+sz1
10 print("A legkisebb közös többszötös: ",t1)
```

```
Run:  11a
E:\00_MM\12_Python_programozas\pro
Add meg az egyik számot: 5
Add meg a másik számot: 18
A legkisebb közös többszötös:  90
```

Hátultesztelő ciklus

Írjunk olyan programot, melyben egy kockadobás eredményét kérjük be! Feltételezhetjük, hogy a felhasználó hibás adatot ad meg. A bevitt adat ellenőrzésével ezt szeretnénk elkerülni, ezért a hibás adat bekérése után újból bekérjük az adatot!
Nem tudjuk előre, hogy a felhasználó hányszor fog egymás után hibás adatot bevinni, tehát feltételes ciklus fogunk itt is használni. Az adatbekérésnek mindenképpen le kell futnia egyszer, tehát nincs értelme a ciklus elejére helyezni a feltételt, így **hátultesztelő ciklus**t fogunk alkalmazni.

- Először készítünk egy adatbekérő részt.
- Tehát a kockadobásnál 1-6-ig fogadunk el számokat.
- 1-nél kisebb és 6-nál nagyobb számok esetén újra kérünk egy számot.
- Ezt hátultesztelő ciklusba ágyazzuk.
- Megadjuk a ciklusban maradás feltételét.
- Addig kell ismételni az adatbakérést, amíg a szám nem 1, vagy 1-nél nagyobb és 6, vagy annál kisebb.
- A Python nem használ hátultesztelő ciklust, így a kódot úgy kell kialakítani, hogy elöltesztelőssé alakítjuk. Ehhez a ciklusmagban található utasításokat a ciklus elé is be kell írni a feltételt a ciklus elejére kell helyezni.

(11b.py)

Mondatszerű leírás:

ciklus
ki: „Add meg a kocka dobás eredményét: „
be: x
amíg (x<1) vagy (x>6)
ciklus vége

```
11b.py
1  a=int(input("Add meg a kockadobás eredményét: "))
2  while (a<1) or (a>6):
3      a = int(input("Add meg a kockadobás eredményét: "))
4  print("Az ellenőrzött megadott érték: ",a)
```

```
Run:  11b
C:\Users\kolmank\AppData\Local\Progra
Add meg a kockadobás eredményét: 20
Add meg a kockadobás eredményét: -3
Add meg a kockadobás eredményét: 5
Az ellenőrzött megadott érték:  5
```

Teljes lefedés elve:
Egy program megvalósítása során nagyon sokszor teszteljük a működését, különböző értékekkel.
A változók többfélék lehetnek. A programunkat úgy kell kialakítanunk, hogy az összes lehetőségre gondoljunk, így teljesen lefedjük a programunkat. Erre kiválóan megfelel a mos tanult módszer.

Végtelen ciklusok

Vannak olyan esetek, amelyekben véletlenül, vagy éppen direkt adunk meg olyan feltételt, amelyből semmi képpen nem lehet kilépni. Erre nézzünk egy példát:

(11c.py)

Ebben az egyszerű programban egyértelmű a hiba.
Ilyenkor a futtatáskor nem lép ki soha, csak folyamatosan fut.
Ekkor hiába próbálkozunk kilépni, nem tudunk.
Szoktuk „lefagyott” állapotnak is hívni. Meg kell szakitani a futást.
Egy összetett programban meg kell keresni a hibát és javítani.
A futó Python program megszakítása: Ctrl+F2 bill. kombinációval lehetséges.

```
11c.py
1  while (1<2):
2      print("Végtelen ciklus!")
```

```
Run:  11c
Végtelen ciklus!
Végtelen ciklus!
Végtelen ciklus!
```