Elágazások

*Egy programban az elágazások leginkább útválasztókhoz hasonlíthatóak. Képzeljük el, ahogy a program olvassa a kódunkat, több sornyi utasítást, majd egy elágazáshoz ér. Itt egy adott **feltételnek** megfelelően a program eldönti, hogy melyik irányban folytatja az útját, majd a program tovább olvassa a kódot, bejáratlanul hagyva a másik utat.*

Logikai érték

A program elágazáshoz érve **feltétel** alapján dönti el a következő lépést. A feltétel kiértékelésének eredménye mindig egy logikai érték: Igaz (**True**) vagy Hamis (**False**).

A programozásban GEORGE BOOLE után a logikai értékeket Boolean-értékeknek hívjuk, melyek True vagy False értéket tudnak felvenni. Típusa röviden: **bool**. Változók bool-á konvertálhatóak a **bool()** függvény segítségével. Python-ban nagy T-vel és nagy F-fel kell írni a változók értékeit (True, False).

```
bool_igaz = True
bool_hamis = False

print(type(bool_igaz))
print(type(bool_hamis))
```

Például a fenti kód hatására a képernyőn egymás alatt megjelenik „<class 'bool'>” szöveg.

Ahogy említettem, más típusú változók is bool-á konvertálhatóak. Van, amely True, míg más másmilyenek False értéket adnak vissza. Néhány példa ezekhez alább látható. Ellenőrzni a változók kiíratásával lehet **print** segítségével. A változónév utal, hogy milyen bool értéket adna vissza.

```
bool_hamis1 = bool(0)
bool_hamis2 = bool(0.0)
bool_hamis3 = bool("")

bool_igaz1 = bool(1)
bool_igaz2 = bool(4.2)
bool_igaz3 = bool("False")
```

(Egyirányú) elágazás

Miután megismerkedtünk a bool fogalmával, lássuk, hogyan is néz ki a kódban egy elágazás.

Szintaktika

```
#elágazás előtt

if feltetel_teljesul:
    print("Történjen ez")
    #elágazás közben

#elágazás után
```

Elágazást kisbetűkkel mindig „**if**” kulcsszóval kezdünk. Ezt követi a **feltétel**, majd rögtön egy **kettőspont**. A kettőspont utáni sorban **behúzást** kell tennünk – ez általában 1db tabulátor vagy 4db szóköz – és tart az elágazásba tartozó kódig. Ha nincs több kód, ami az elágazáshoz tartozik, akkor befejezzük a kód behúzását.

Feltétel

A jól említett feltétel dönti el, hogy a program végrehajtja-e az elágazásban található kódot vagy sem. Ha a feltétel teljesül – tehát Igaz (True) – akkor **végrehajtja**, viszont, ha nem teljesül – tehát Hamis (False), akkor nem.

Egy feltétel megírásához az alábbi lehetőségeink vannak.

Összehasonlító operátorok

Írhatunk feltételt, mely két változót/értéket összehasonlít egymással. Ezt a következő operátorokkal tehetjük meg:

- **egyik egyenlő-e a másikkal:** *egyik* == *másik* (kettő egyenlőségjel)
- **egyik nem egyenlő-e a másikkal:** *egyik* != *másik* (felkiáltójel és egy egyenlőség jel)
- **egyik kisebb, mint a másik:** *egyik* < *másik*
- **egyik kisebb vagy egyenlő, mint a másik:** *egyik* <= *másik*
- **egyik nagyobb, mint a másik:** *egyik* > *másik*
- **egyik nagyobb vagy egyenlő, mint a másik:** *egyik* >= *másik*

```
szam_kicsi = 42
szam_nagy = 1000

if szam_kicsi < szam_nagy:
    print(szam_kicsi, "a kisebb szám!")
```

Például a fenti kód hatására a képernyőn megjelenik a „42 a kisebb szám!” felirat.

Logikai operátorok

Írhatunk feltételt, melyeket logikai operátorokkal választunk el. Ilyenek a: „nem”; „és”; „vagy”, ahol ez a sorrend számít **precedencia szabálynak** több logikai operátor együttes használata során.

- **not**: Akkor és csak akkor igaz, ha az **állítás hamis**.
- **and**: Akkor és csak akkor igaz, ha **mindegyik** állítás külön-külön is igaz.
- **or**: Akkor és csak akkor igaz, ha az állítások közül **legalább egy** igaz.

Az egyszerűség kedvéért igazságtáblát készítünk, és így tanuljuk meg a logikai operátorok kiértékelését (igaz vagy hamis értékkel térnek vissza).

Az **IGAZSÁGTÁBLA** készítése kettő állítás – x és y – mentén történik.

x	y	not x	x and y	x or y
False	**False**	True	False	False
False	**True**	True	False	True
True	**False**	False	False	True
True	**True**	False	True	True

```
int_also = 1
int_felso = 10

random_szam = 5

if int_also < random_szam and random_szam < int_felso:
    print(random_szam, " az intervallumba esik.")
```

Például a fenti kód ötvözi az összehasonlító és logikai operátorok használatát, így segítségükkel elágazás feltételében ellenőrízhetjük, hogy a random_szam (amit kézzel megadtam) az intervallumba – 1 és 10 közé - esik-e vagy sem. Amennyiben a feltétel teljesül, az elágazásoban található kód lefut. Magyarul a feltétel pedig így hangozna: „Ha random_szam nagyobb, mint az int_also és kisebb, mint az int_felso, akkor a feltétel igaz.”

Többirányú elágazás

Többirányú elágazás során azt az „útvonalat” is definiáljuk, amikor a feltétel kiértékelése Hamis (False) eredményt hozna. Ehhez az **else** parancsot kell használnunk, hasonló szintaktikával, mint egyirányú elágazás esetén.

```
szam = 42

if szam <= 5:
    print(szam, " kisebb ötnél.")
else:
    print(szam, " nagyob ötnél.")
```

Például a fenti kód során a képernyőre a következő íródik ki: „42 nagyobb ötnél.”

Lehetőségünk van további elágazásokat is létrehozni. Ezeket további else paranccsal tehetjük meg. Amennyiben szeretnénk mindegyik iránynál („útvonalnál”) feltételt létrehozni az „if” kulcsszó után is, akkor ezt az **elif** paranccsal tehetjük meg.

```
szam = 42

if szam <= 41:
    print(szam, " kisebb 41-nél.")
elif szam >= 43:
    print(szam, " nagyob 43-nál.")
else:
    print(szam, " pontosan 42.")
```

Például a fenti kód során a képernyőre a következő íródik ki: „42 pontosan 42.”