Elágazások

Eddig csupa olyan programot írtunk, ami elkezdődött az elején, sorban egymás után végrehajtott minden utasítást, aztán kilépett. Ebben a leckében ezen változtatni fogunk.

Gondolunk egy számra

A témakör első leckéjében már láttunk egy olyan programot, amelyikben elágazás van. Az a program mást csinál, ha nem vagyunk még tizennégy évesek, és mást, ha már betöltöttük ezt az életkort. Hasonló működésű az ugyanott megismert „Mi a neve Mátyás királynak?” program.

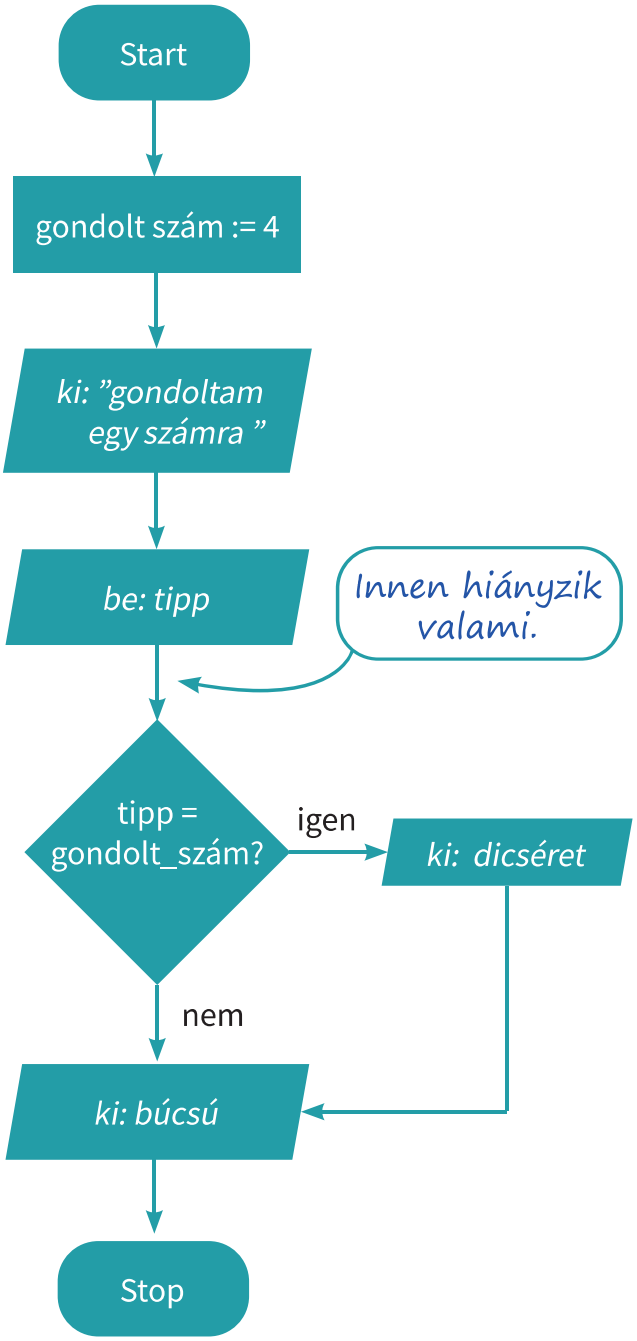
Az alábbi **folyamatábra** is egy hasonló programot ír le. Nincsenek benne konkrét utasítások, mert fontosabb, hogy gondold először át, hogy mit csinál a programod, és ráérsz utána azon elmélkedni, hogy miként **kódolod** a programodat.

1. Mit csinál a program?
2. Mi az a lépés, amit nem tüntettünk fel? (Ha most nem jövünk rá, nem probléma: hamarosan úgyis megírjuk a programkódot, akkor szólni fog a Python.)
3. Hogyan olvassuk ki a := jelet? Mi a megfelelője Pythonban?

Miközben a programunkat folyamatábrával megfogalmazzuk, **algoritmust** (receptet) adunk a bennünket érdeklő probléma megoldására. A folyamatábrák elég látványosak, de hamar lelógnának a könyvlapról, így aztán a gyakorlatban gyakrabban használunk egy másik algoritmusleíró eszközt, a **mondatszerű leírást.**

A leírás szabályaira rá fogunk érezni.

4. Vessük össze ezt a leírást a folyamatábrával!
5. Melyik az a művelet, ami a mondatszerű leírásban már megvan, de a folyamatábrában még hiányzik?




```
gondolt_szám := 4
ki: „gondoltam egy számra”
be: tipp
tipp átalakítása egésszé
elágazás
ha tipp = gondolt_szám:
      ki: dicséret
elágazás vége
ki: búcsú
```

Készítsük el a fenti két algoritmusleíró eszközzel megadott programkódot!

```
1. gondolt_szám = 4
2. tipp = input('Gondoltam egy számra. Tippeld meg! ')
3. tipp = int(tipp)
4. if tipp == gondolt_szám:
5.     print('Ügyes!')
6. print('Pápá.')
```

A kettőspont hatására a következő sor bentebb kezdődik.

Ez tényleg 2 egyenlőségjel.

Egy TAB vagy 4 szóköz KELL!!!

Teszteljük a programunkat: adjuk meg a helyes megoldást, de próbáljuk ki helytelennel is!

A program következő változatában, más szóval verziójában kicsit bővebben dicsérünk, illetve a hibásan tippelő felhasználókat kicsit ugratjuk. A mondatszerű leírás a következő:

6. Módosítsuk ez alapján a folyamatábrát (segítség: a rombuszból lefelé nem lesz nyíl, de balra igen, és a két nyíl a rombusz alatt összetalálkozik)!

A dicséret második sora újabb print utasítást jelent az előző alatt. Írjuk be az új sort:

```
print('Gratulálok.')
```

behúzva és behúzás nélkül! Mindkét esetben teszteljük a programunkat, és vessük össze a viselkedését. Fogalmazzuk meg a behúzás szerepét!

```
gondolt_szám := 4
ki: „gondoltam egy számra”
be: tipp
tipp átalakítása egésszé
elágazás
ha tipp = gondolt_szám:
        ki: kétsoros dicséret
különben:
        ki: ugratás
elágazás vége
ki: búcsú
```

Ha megvagyunk, írjuk meg az ugratós részt is. A „különben” szó angolul „else” – ezt kell használnunk kódoláskor. A kész program:

```
1. gondolt_szám = 4
2. tipp = input('Gondoltam egy számra. Tippeld meg! ')
3. tipp = int(tipp)
4. if tipp == gondolt_szám:
5.     print('Ügyes!')
6.     print('Gratulálok.')
7. else:
8.     print('Hosszan gondolkodtál rajta?:)')
9.     print('Nem érte meg.;)')
10. print('Pápá.'))
```

Ez itt az elágazás FELTÉTELE.

Ez a két utasítás a „ha”-ág.

Ez a két utasítás pedig a „különben”-ág.

A programunk tanulságai:

1. Ami az `if` után következik, az az elágazás feltétele.
2. A feltételvizsgálatban két egyenlőségjel kell. A programozásban az egy egyenlőségjel egy felszólítás (ismerjük már, értékadáskor használjuk), a két egyenlőségjel kérdés: A tipp egyenlő a gondolt számmal?
3. Ami az elágazás ágaiban van (a fenti program 5–6. és 8–9. sora), az bentebb kezdődik. A Python onnan tudja, hogy mikor kezdődik egy ág, hogy a programsor bentebb kezdődik, és onnan tudja, hogy hol van vége az ágnak, hogy az újabb programsor már nem kezdődik bentebb.

Összetett feltétel

Szeretnénk úgy bővíteni a programunkat, hogy ha csak eggyel tippelt mellé a felhasználó, akkor ezt eláruljuk neki. Elsőként az algoritmus mondatszerű leírását módosítjuk. Az új, „különbenha” ágat megvalósító Python-utasítás az `elif`.

Akár neki is foghatnánk a kódolásnak, de hogy programozandó a „csak egyet tévedett”? Ilyen utasítás nincs, ezért az algoritmusunkat egy-két lépéssel tovább kell finomítanunk. Észrevesszük, hogy kétféleképp lehet egyet tévedni: vagy eggyel nagyobbat, vagy eggyel kisebbet tippelve. Az „eggyel nagyobbat tippelt” így írható le:

```
gondolt_szám := 4
ki: „gondoltam egy számra”
be: tipp
tipp átalakítása egésszé
elágazás
ha tipp = gondolt_szám:
      ki: kétsoros dicséret
különbenha csak egyet
tévedett:
      ki: csak egyet tévedtél
különben:
      ki: ugratás
elágazás vége
ki: búcsú
```

```
tipp = gondolt_szám+1.
```

A másik feltétel megfogalmazása nem okozhat gondot, de ezek szerint két feltétel lett az egyből. Megírhatjuk úgy az algoritmust (és a programot), hogy két „különbenha” ágat adunk meg, ugyanazzal a kiírandó üzenettel, de ez nem szerencsés – például azért, mert ha módosítani kell az üzenetet, két helyen is módosítanunk kell, és az egyiket előbb-utóbb elfelejtjük megcsinálni.

Alakítsuk inkább a két feltételünket egy összetett feltétellé:

```
különbenha tipp = gondolt_szám+1 vagy tipp = gondolt_szám-1:
```

A két feltételből így lett egy. Lévén a „vagy” szó kapcsolja őket össze, elég, ha az egyik teljesül. Ha „és” kapcsolná őket össze, mindkettőnek teljesülnie kellene, hogy a teljes összetett feltétel igaz legyen. A kód most így néz ki:

```
1. gondolt_szám = 4
2. tipp = input('Gondoltam egy számra. Tippeld meg! ')
3. tipp = int(tipp)
4. if tipp == gondolt_szám:
5.     print('Ügyes!')
6.     print('Gratulálok.')
7. elif tipp == gondolt_szám + 1 or tipp == gondolt_szám - 1:
8.     print('Ó, csak eggyel tévedtél.')
9. else:
10.    print('Hosszan gondolkodtál rajta?:)')
11.    print('Nem érte meg.;)')
12. print('Pápá.')
```

Kérdések

1. Hány `elif`-ág és hány `else`-ág szerepelhet egy elágazásban?
2. Melyiket kell utolsóként megadni?
3. Melyiknek nincs feltétele?
4. Létezhet olyan elágazás, amelyikben nincs egyik sem?
5. Kihívást jelentő feladat: Az alábbi sor nem jó (bár a program nem jelez hibát): `elif tipp == gondolt_szám + 1 or gondolt_szám - 1:`
   Miért nem jó?

Véletlenszám-előállítás

Meglehetősen unalmas lehet, hogy a programunk mindig a négyre gondol. A legtöbb programozási nyelvben van valamilyen módszer véletlen számok előállítására. A Pythonban több is van, ezek közül a `random.randint` (randint: random integer, azaz véletlen egész) az, amelyik véletlenszerű egész számot állít elő, más szóval generál. Ha ezt az utasítást használni akarjuk, akkor előbb be kell töltetni a programmal a `random` nevű modult. A programunk első sorai a következőképp alakulnak:

```
1. import random
2.
3. gondolt_szám = random.randint(1,6)
4. print('Súgok:', gondolt_szám)
5. tipp = input('Gondoltam egy számra. Tippeld meg! ')
```

Itt töltjük be a „random” modult. Az importálásokat követően szokás egy sort kihagyni.

1 és 6 közötti egész számot állítunk elő.

A súgás a program tesztelésekor hasznos, a végső változatból vegyük ki!