Gyakorló feladatok

előző foglalkozás összefoglalása,
gyakorlató feladatok
a feltételes elágazásra, a while ciklusra, és
sokminden másra amit eddig tanultunk

Változók elnevezése

- a változók nevét a programozó választja
- egy változónév nem lehet foglalt Python kulcsszó, mint pl. `if`, `else`, `while`, `def`, `and`, ...
- a változónevek betűket és számokat is tartalmazhatnak, de...
  - nem kezdődhetnek számmal
  - nem lehet bennük szóköz vagy olyan speciális karakter, mint #, &, $, ...
- a változó nevek tartalmazhatnak _ (aláhúzás) karaktert

Változók elnevezése

- válasszunk rövid, de kifejező neveket!
  - pl. egy háromszög magassága lehet `magassag` vagy néha egyszerűen csak `m`
  - pl. egy bekért név és életkor lehet `nev` és `kor`
- ha mégis hosszabb nevet kell választani, akkor kövessünk egységes elnevezési mintát
  - bármilyen mintát használhatnánk, de a Python programozók az alábbi mintát követik:

```
haromszog_magassag, kor_sugar, ...
```

Értékadás

- az értékadás az = jellel történik
- az első értékadással hozzuk létre a változót
- további értékadással változtathatjuk a változóban tárolt értéket
- az értékadás meghatározza a változó típusát is
- létezik többszörös értékeadás, amivel egyszerre több változó értékét lehet állítani

Műveletek egész és tört számokkal

- alapműveletek: `+`, `-`, `*`, `/`
- további hasznos műveletek:

  `**` (hatványozás), `//` (egész osztás), `%` (maradék)
- műveletek precedenciája:

  1. `**`
  2. `*`, `/`, `//`, `%`
  3. `+`, `-`
- beépített függvények:

  `min()`, `max()`, `round()`

Karakterláncok (string-ek)

- a karakterláncokat jelölhetjük a "..." vagy a '...' jelekkel

- string műveletek (s stringen):
  - szöveg hosszának megállapítása: `len(s)`
  - átalakítás kis- vagy nagybetűssé: `s.lower(), s.upper()`
  - keresés szövegben: `s.find('...')`
  - előfordulás számlálása: `s.count('...')`

Típusátalakítás

- sokszor szükség van egy változóban tárolt érték típusának átalkítására
- **`str()`**: egész és tört számokból csinál string-et
- **`int()`** és **`float()`**: egészet és törtet csinál string-ből
  - ha a `float()` bemenete egy egész → törtté alakítja
  - ha az `int()` bemenete egy tört → egész részét veszi

Logikai műveletek

- Boolean típusú (logikai) változók csak `True` (igaz) vagy `False` (hamis) értéket vehetnek fel
- ilyen változókkal logikai műveleteket végezhetünk:
  1. `not` (negáció)
  2. `and` (és)
  3. `or` (vagy)

Logikai műveletek

A	B	A and B
F	F	F
F	T	F
T	F	F
T	T	T

A	B	A or B
F	F	F
F	T	T
T	F	T
T	T	T

A	not A
F	T
T	F

Feltételes elágazás

- egy programrész végrehajtását feltételhez kötheted az `if` utasítással (az *if* jelentése magyarul *ha*)
- az `if` párja az `else` (jelentése *egyébként*); az `else` utáni programrész akkor hajtódik végre, ha az `if` feltétele nem teljesül

```
szam = int(input("Írj be egy számot! "))
if (szam > 100):
    print("Ez egy nagy szám.")
else:
    print("Ez egy kicsi szám.")
```

Figyeld meg a behúzásokat!

Ügyelj a kettőspontra!

Feltételes elágazás

A <feltétel> egy kifejezés, melynek értéke **igaz** (`True`) vagy **hamis** (`False`) lehet

```
if <feltétel>:
```

Az azonos méretű behúzással rendelkező, egymás utáni sorok egy összetartozó blokkot alkotnak

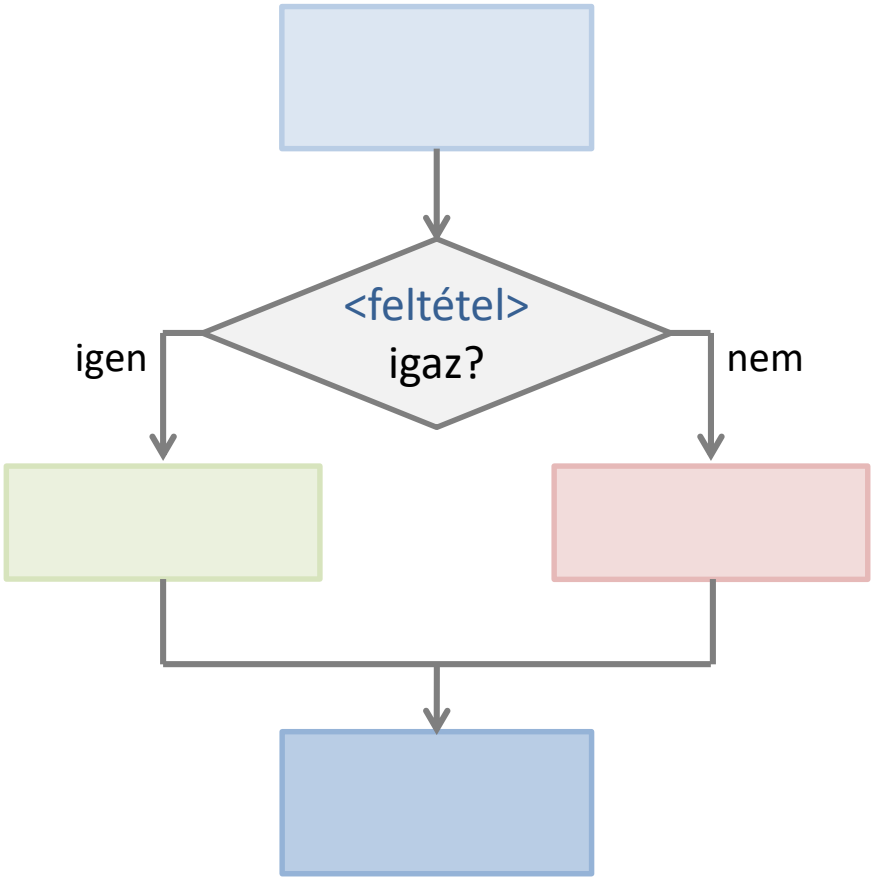
Ez az utasításblokk hajtódik végre ha **a <feltétel> igaz**

```
else:
```

Ez az utasításblokk hajtódik végre ha **a <feltétel> hamis**

Feltételes elágazás

```
if <feltétel>:
    
else:
    
```



Ismétlés `while` ciklussal

- sokszor előfordul, hogy egy programrész végrehajtását addig kell ismételni, amíg valamilyen feltétel fennáll
- ezt a célt szolgálja a `while` ciklus (a *while* jelentése *amíg*):

```
szam = 0
while (szam <= 100):
    szam = int(input("Írj be egy 100-nál nagyobb számot! "))
print("Köszönöm!")
```

Ismétlés `while` ciklussal

A <feltétel> egy kifejezés, melynek értéke **igaz** (`True`) vagy **hamis** (`False`) lehet

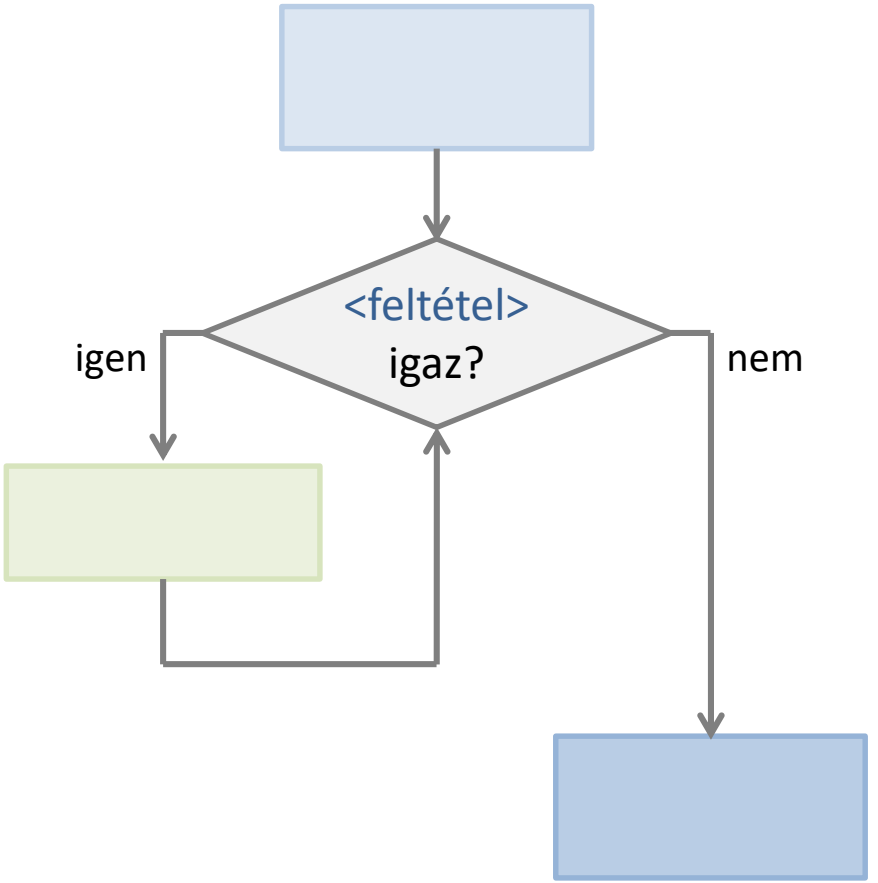
```
while <feltétel>:
```

Itt is a behúzás jelzi a while-hoz tartozó, ismételendő utasításblokkot

Ez az utasításblokk mindaddig újra és újra végrehajtódik amíg **a <feltétel> igaz**

Ismétlés `while` ciklussal

```
while <feltétel>:
```



Egy szám osztói

Feladat

Írj programot, mely bekér egy pozitív egész számot, majd kiszámolja és kiírja annak osztóit! Az osztók egy sorban, pontosvesszővel elválasztva jelenjenek meg!

A program várt működése pl. a következő:

```
Írj be egy pozitív egész számot: 15
15 osztói:
1; 3; 5; 15
```

Egy szám osztói

Megoldási terv

- Kérd be a számot, és azonnal alakítsd át egész számmá!
- Hozz létre egy változót, amiben a kiírandó osztókat gyűjtöd majd! Mivel az 1 minden számnak osztója, ezért a változó kezdő értéke lehet ”1”. Ehhez fűzöd majd a további osztókat.
- Minden *n* számra 2-től a megadott szám feléig, vizsgáld meg, hogy n osztója-e a számnak! Ha igen, akkor *n*-et fűzd hozzá az osztókat tároló változó értékéhez!
- Végül fűzd hozzá az osztókhoz magát a számot, és írd ki az osztókat tároló változó értékét!

Egy szám osztói

Ötletek

- *n* osztója egy *X* számnak, ha *X*-et *n*-nel elosztva 0 maradékot kapunk. Ezt így tesztelheted a programodban:

```
if (X%n == 0):
```

- Meg kell vizsgálnod minden *n* számra 2-től a megadott szám feléig, hogy *n* osztója-e a számnak. Ezt egy `while` ciklussal lehet megoldani. A ciklus belsejében ne felejtsd el *n* aktuális értékének vizsgálata után *n* értékét megnövelni eggyel. Különben a programod végtelen ciklusba kerül, és soha nem áll le.

Egy szám osztói

```
szam = int(input("Írj be egy pozitív egész számot: "))

osztok = "1"
n = 2
while (n <= szam//2):
    if (szam%n == 0):
        osztok = osztok + "; " + str(n)
    n = n+1
osztok = osztok + "; " + str(szam)

print(str(szam) + " osztói:")
print(osztok)
```

Tökéletes számok

Feladat

Egy pozitív egész számot tökéletesnek nevezünk, ha a szám megegyezik az önmagánál kisebb osztóinak összegével. Pl. a 6 tökéletes szám, mert az önmagánál kisebb osztói az 1, 2, 3, és ezek összege pont 6.

Az előző feladat mintájára, írj programot, mely bekér egy pozitív egész számot, majd megállapítja, hogy az tökéletes szám-e vagy nem! A program várt működése pl. a következő:

```
Írj be egy pozitív egész számot: 6
Tökéletes szám!

Írj be egy pozitív egész számot: 12
Nem tökéletes szám.
```

Tökéletes számok

Megoldási terv

- Kérd be a számot, és azonnal alakítsd át egész számmá!
- Hozz létre egy változót, amiben az osztók összegét tárolod majd! Mivel az 1 minden számnak osztója, ezért a változó kezdő értéke lehet 1. Ehhez adod majd hozzá a további osztókat.
- Minden *n* számra 2-től a megadott szám feléig, vizsgáld meg, hogy n osztója-e a számnak! Ha igen, akkor *n*-et add hozzá az osztók összegét tároló változó értékéhez!
- Végül ellenőrizd, hogy az osztók összege megegyezik-e a számmal, és az ellenőrzés eredményétől függően írd ki a megfelelő üzenetet a képernyőre!

Tökéletes számok

```
szam = int(input("Írj be egy pozitív egész számot: "))

osztok_osszege = 1
n = 2
while (n <= szam//2):
    if (szam%n == 0):
        osztok_osszege = osztok_osszege + n
    n = n+1

if (osztok_osszege == szam):
    print("Tökéletes szám!")
else:
    print("Nem tökéletes szám.")
```

Tökéletes számok keresése

Feladat

Az előző feladatban készített programmal megpróbálhatsz tökéletes számokat keresni, de hamar rájössz majd, hogy ez kézzel elég hosszadalmas.

Írj programot, mely egy megadott felső határig az összes számot megvizsgálja, és kiírja a tökéletes számokat! A program várt működése pl. a következő:

```
Tökéletes szám keresésének felső határa: 100
Ezeket a tökéletes számokat találtam:
6
28
```

Tökéletes számok keresése

Megoldási ötlet

- Lényegében az előző programodat kell beletenni egy `while` ciklus belsejébe, mely az összes számon végigmegy 6-tól (tudjuk, hogy 6-nál nincs kisebb tökéletes szám) a bekért felső határig.

Tökéletes számok keresése

```
H = int(input("Tökéletes szám keresésének felső határa: "))
print("Ezeket a tökéletes számokat találtam:")

N = 6
while (N <= H):
    osztok_osszege = 1
    n = 2
    while (n <= N//2):
        if (N%n == 0):
            osztok_osszege = osztok_osszege + n
        n = n+1
    if (osztok_osszege == N):
        print(N)
    N = N+1
```

Tökéletes számok keresése

- a tökéletes számok nagyon ritkák, eddig 49-et ismerünk
  - 6, 28, 496, 8128, 33550336, ...
  - https://en.wikipedia.org/wiki/List_of_perfect_numbers
- a páros tökéletes számok $2^{p-1}(2^p - 1)$ alakúak, ahol $2^p - 1$ prím (és p-nek is annak kell lennie)
  - $6 = 2^{2-1}(2^2 - 1) = 2 \cdot 3$
  - $28 = 2^{3-1}(2^3 - 1) = 4 \cdot 7$
- nem tudjuk, hogy
  - végtelen sok tökéletes szám van-e vagy csak véges sok?
  - van-e páratlan tökéletes szám?

Prímszámok

Feladat

Egy pozitív egész számot prímszámnak nevezünk, ha a számnak csak 1 és önmaga az osztója. Az 1 definíció szerint nem prím. A legkisebb prímszám így a 2, hiszen csak 1-gyel és önmagával osztható.

Írj programot, mely bekér egy pozitív egész számot, majd megállapítja, hogy az prímszám-e vagy sem! A program várt működése pl. a következő:

```
Írj be egy pozitív egész számot: 6
Nem prím.

Írj be egy pozitív egész számot: 13
Prím.
```

Prímszámok

Megoldási terv

- Kérd be a számot, és azonnal alakítsd át egész számmá!
- Ha a szám 1, akkor írd ki, hogy nem prím, egyébként (és csak akkor) csináld a következőket...
- Hozz létre egy Boolean típusú változót, amiben azt tárolod majd, hogy a szám prím-e vagy sem! Ennek kezdeti értéke legyen `True`!
- Írj egy `while` ciklust, amiben ellenőrzöd minden számra 2 és a vizsgált szám gyöke között, hogy osztja-e a vizsgált számot! Ha találsz osztót, akkor a Boolean változót állítsd `False`-ra (hiszen a szám ekkor nem lehet prím)!
- A `while` ciklus után, a Boolean változó értékétől függően írd ki, hogy a szám prím vagy nem prím!
- Teszteld a programod! Helyesen működik?

Prímszámok

Ötletek

- Egy szám gyökét a `math` modulban található `sqrt()` függvénnyel lehet kiszámolni. Ehhez a program elején importálni kell ezt a függvényt:

```
from math import sqrt
```

- Ezután egy *X* szám gyökét az `sqrt(X)` leírásával kapod meg, pl:

```
while (n <= sqrt(szam)):
   ...
```

Prímszámok

```
from math import sqrt

szam = int(input("Írj be egy pozitív egész számot: "))
if (szam == 1):
    print("Definíció szerint nem prím.")
else:
    prim = True
    n = 2
    while (n <= sqrt(szam)):
        if (szam%n == 0):   # a szám osztható n-nel
            prim = False
        n = n + 1
    if (prim):
        print("Prím.")
    else:
        print("Nem prím.")
```

Prímszámok keresése

Feladat

Az előző feladatban készített programmal megpróbálhatsz prímszámokat keresni, de hamar rájössz majd, hogy ez így elég kényelmetlen.

Írj programot, mely egy megadott felső határig az összes számot megvizsgálja, és kiírja a prímszámokat! A program várt működése pl. a következő:

```
Prímszám keresésének felső határa: 100
Ezeket a prímszámokat találtam:
2; 3; 5; 7; 11; 13; 17; 19; 23; 29; 31; 37; 41; 43; 47; 53;
59; 61; 67; 71; 73; 79; 83; 89; 97;
```

Prímszámok keresése

Megoldási ötlet

- Lényegében az előző programodat kell beletenni egy `while` ciklus belsejébe, mely az összes számon végigmegy 2-tól (tudjuk, hogy 2-nél nincs kisebb prímszám) a bekért felső határig.

Prímszámok keresése

```
from math import sqrt

M = int(input(" Prímszám keresésének felső határa: "))
szam = 2
while (szam <= M):
    prim = True
    n = 2
    while (n <= sqrt(szam)):
        if (szam%n == 0):   # a szám osztható n-nel
            prim = False
        n += 1
    if (prim):
        print(szam, end="; ")
    szam += 1
```

Pitagoraszi számhármasok

Feladat

A Pitagorasz-tétel azt mondja ki, hogy bármely derékszögű háromszög leghosszabb oldalának (átfogójának) négyzete megegyezik a másik két oldal (a befogók) négyzetének összegével. Röviden: $a^2 + b^2 = c^2$

Írj programot, mely bekéri egy háromszög három oldalának hosszát, majd megállapítja, hogy a háromszög derékszögű-e! A program várt működése pl. a következő:

```
1. oldal: 5
2. oldal: 4
3. oldal: 3

Derékszögű háromszög.
```

```
1. oldal: 3
2. oldal: 6
3. oldal: 7

Nem derékszögű háromszög.
```

Pitagoraszi számhármasok

Megoldási terv

- Kérd be a három oldal hosszát, és azonnal alakítsd át őket tört számmá! Nevezd a három változót amiben az oldalhosszakat tárolod pl. a-nak, b-nek, és c-nek!
- Lehet, hogy nem c-ben van a leghosszabb oldal hossza. Ha valóban így van, akkor a, b, és c értékeinek megfelelő cseréivel, próbáld meg elérni, hogy c-ben legyen a legnagyobb szám!
- Ha ez megvan, akkor ellenőrizd, hogy a**2 + b**2 megegyezik-e c**2-tel!
- Az ellenőrzés eredményétől függően, írd ki a megfelelő választ!
- Ellenőrizd a programod! Jól működik akkor is, ha nem az utoljára megadott oldal a legnagyobb?

Pitagoraszi számhármasok

```
a = float(input("1. oldal: "))
b = float(input("2. oldal: "))
c = float(input("3. oldal: "))

M = max(a, b, c)
if (c != M):
    c, a = a, c
    if (c != M):
        c, b = b, c

if (a*a + b*b == c*c):
    print("Derékszögű háromszög.")
else:
    print("Nem derékszögű háromszög.")
```

Pitagorszi hármasok keresése

Feladat

Az előző feladatban készített programmal megpróbálhatsz pitagoraszi számhármasokat keresni, de hamar rájössz majd, hogy ez így elég lassú és kényelmetlen.

Írj programot, mely egy megadott felső határig az összes pozitív egész számot behelyettesíti a és b helyére, kiszámolja a hozzájuk tartozó megfelelő c értéket, leellenőrzi, hogy a, b, és c kielégítik-e Pitagorasz tételét, és ha igen, kiírja őket a képernyőre! A program várt működése pl. a következő:

```
Befogó max hossza: 10
3 4 5; 4 3 5; 6 8 10; 8 6 10;
```

Pitagorszi hármasok keresése

Megoldási ötlet

- Két egymásba ágyazott `while` ciklust kell csinálnod, ahol az egyikben az a értékét, a másikban pedig b értékét futtatod végig 1-től a megadott max értékig.
- A belső `while` cikluson belül, ki kell számolnod c értékét. Ehhez először $c^2 = a^2 + b^2$-et kell kiszámolni, majd c^2-ből gyököt kell vonni, ennek az eredménye lesz c.
- Ha c megvan, akkor azt kell ellenőrizni, hogy egész számot kaptál-e. Hogyan csinálnád ezt?
- Ha c egész, akkor (és csak akkor) a három számot írd ki!
- Ne felejtsd el a `while` ciklusokon belül (a végén) a és b értékét eggyel növelni! Különben végtelen ciklusba kerülsz...

Pitagoraszi hármasok keresése

```
from math import sqrt

M = int(input("Befogó max hossza: "))
a = 1
while (a <= M):
    b = 1
    while (b <= M):
        c2 = a**2 + b**2
        c = sqrt(c2)
        if (c%1 == 0.0):
            print(a, b, int(c), end="; ")
        b += 1
    a += 1
```

Gyakorló feladatok

Készítette:
Buttyán Levente
Levente.Buttyan@gmail.com
CoderDojo Szentendre
2017