7. hét: rekurzió

Czirkos Zoltán · 2022.10.14.

Laborfeladatok a rekurzió témakörében. Sorozatok rekurzív definíciója. Báziskritériumok és egyszerűsítési lépések. Egyszerű rekurzív ábrák elkészítése teknőcgrafikával.

Felkészülés a laborra:

- A rekurzióról szóló előadás áttekintése.
- A nyomkövetésről tanultak átismétlése.

Tartalom

1. Faktoriális

Tanultad, hogy egy n szám faktoriálisát így is lehet definiálni:

$$n! = \begin{cases} 1, & \text{ha } n = 0 \\ n\cdot(n-1)!, & \text{ha } n > 0 \end{cases}$$

Fogj egy papírlapot, és fejtsd ki n! értékét kézzel, az alábbi módon! Vagyis helyettesítsd be a faktoriális definíciója szerint megadott szabály szerint a kifejezést.

```
6! = 6 * 5!
   = 6 * 5 * 4!
   ...
```

Ha ezzel megvagy, írj Python programot, amely egy rekurzív faktoriális függvényt tartalmaz! Vagyis térjen ez vissza `1`-gyel, ha a paramétere `0`, és `n * fakt(n-1)`-gyel, ha nagyobb. Teszteld a megírt függvényed, írd vele ki a faktoriálisok értékét 0-tól 10-ig!

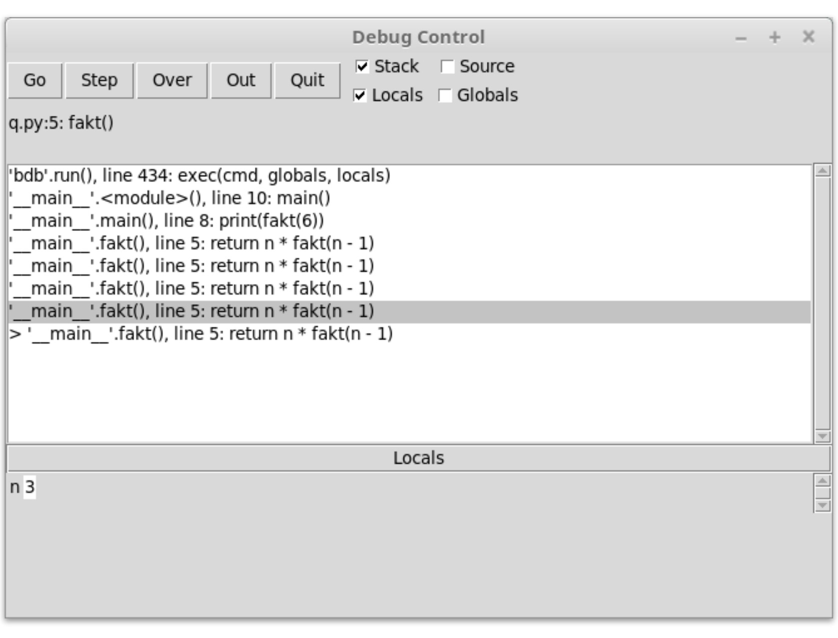
```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

> Vigyázz: rekurzív függvényt kell írnod, ebben nem kell ciklus, se `while`, se `for`. Változó sem lesz benne. A `main()`-ben viszont lehet.

▶ **Megoldás**

2. A rekurzió nyomkövetése

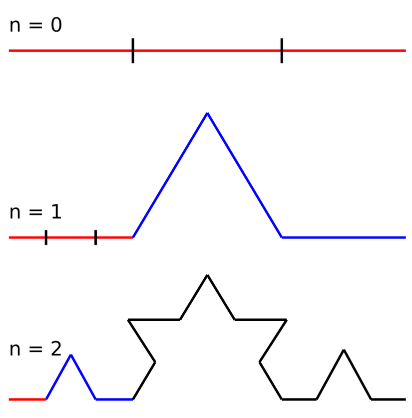
Engedélyezd most a nyomkövetést a laboron tanult módon. Figyeld meg, hogyan hívja a függvény saját magát a `fakt(6)` kifejezés kiértékelése közben. Az IDLE nyomkövető az ablak felső részében mutatja a függvényhívásokat:



Az egyes hívásokra kattintva alul látszik, hogy azokhoz milyen paraméter tartozik.

3. Fraktál

Tanulmányozd az alábbi rajzot!



A legfelső sorban egy szimpla vonalat látsz. Képzeld azt, hogy a szimpla vonalat 3 egyenlő hosszúságú darabra töröd (ahol a jelek vannak), és a középső részt egy 60 fokos háromszöggel helyettesíted. Vagyis megteszed az 1/3 hosszt, aztán balra fordulsz 60 fokot, újabb 1/3, jobbra fordulás, újabb 1/3, és végül balra fordulás, befejező 1/3. Így alakult ki a második rajz. Ha a második rajz összes szakaszát helyettesíted saját magával, akkor jutsz a harmadik rajzhoz.

Egy olyan rekurzív függvényt kell írnod, amely ezt a rajzot elkészíti teknőcgrafikával. A gondolatmenet a következő:

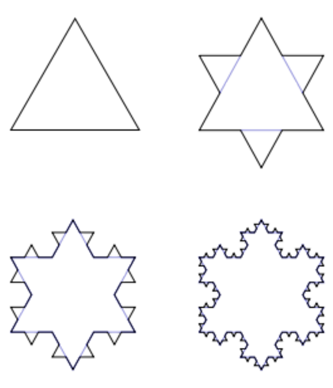
- A függvény paraméterként kap egy hosszt (h), és a rajz bonyolultságát (n).
- Ha a legegyszerűbb esetet kell megrajzolni (n = 0), akkor csak húz egy egyenes szakaszt.
- Ha egy bonyolultabb esetet, akkor pedig bejárja a szakasz-fordulás-szakasz-fordulás-szakasz-fordulás-szakasz útvonalat, megrajzolva ezt a formát: `_/\_`.
- Ahol viszont szakaszt kell rajzolnia, oda nem szakaszt rajzol, hanem az eggyel egyszerűbb ábrát: `n-1`.

Előbb készítsd el azt a változatot, amelyik csak a középső ábrát rajzolja ki, utána egyszerű a szakaszok rajzolását rekurzív hívásra cserélni!

▶ **Megoldás**

4. Hópehely

Rajzolj hópelyhet az előző feladat fraktáljával! Ehhez nincs más dolgod, mint három fraktált egymás mellé tenni, egymáshoz képest 120 fokkal elforgatva (lásd a bal felső rajzot). Minél nagyobb a bonyolultság, annál szebb lesz a hópehely.



Valósítsd meg ezt egy `hopehely(hossz, bonyolultság)` paraméterű függvényben! Használd fel ehhez az előző fraktál függvényt, változatlan formában!

▶ **Megoldás**

5. Számtani sorozat

Egy számtani sorozatot (jelöljük most `S`-sel) annak első tagjával és növekményével definiáljuk. Az első tag S_0 értéke `a`, a többi tagot pedig úgy számítjuk ki, hogy mindig az előző taghoz hozzáadjuk `d`-t, a növekményt:

$$S_i = \begin{cases} a, & \text{ha } i = 0 \\ S_{i-1} + d, & \text{ha } i > 0 \end{cases}$$

Írj rekurzív függvényt, amelyik megkapja 1) a sorozat első tagját: `a`, 2) a sorozat növekményét: `d`, és 3) hogy a sorozat hányadik elemét kell megadja: `i`, és visszatér S_i értékével! Alkalmazd a függvény megírásakor a fenti rekurzív definíciót! Egészítsd ki a megírt függvényt egy főprogrammal, amelyben kiírod egy számtani sorozat első 10 elemét!

> A főprogramban lehet ciklus, de a sorozatot megadó függvényben ne legyen se `while`, se pedig `for`!

▶ **Megoldás**

6. Gyors hatványozás

A hatványozás elvégezhető annál gyorsabban is, mintha a kitevőnek megfelelő számú szorzást csinálnánk. Pl. $a^8 = a^4\cdot a^4$, $a^4 = a^2\cdot a^2$ és $a^2 = a\cdot a$ miatt a nyolcadikra hatványozáshoz mindössze három szorzásra van szükség. A következő megfigyelést tehetjük:

$$a^k = \begin{cases} (a\cdot a)^{k/2}, & \text{ha } k \text{ páros} \\ a\cdot a^{k-1}, & \text{ha } k \text{ páratlan.} \end{cases}$$

Írj rekurzív függvényt, amely a fentiek alapján végzi el a hatványozást! Paraméterei legyenek az alap és a kitevő, visszatérési értéke pedig a hatvány. Írd ki kettő első tizenhat hatványát!

> A rekurzív függvénybe most se tegyél ciklust, dolgozz a definíció alapján! Ahhoz, hogy ez működjön, még egy báziskritériumot be kell vezetned, amit a fenti definíció nem tartalmaz. Mi lehet az?

▶ **Megoldás**

7. Számrendszer váltó

Írj függvényt, amely paraméterként kap egy pozitív egész számot valamint egy számrendszert, és kiírja a képernyőre a számot a megadott számrendszerben! Elég most, ha csak 10-es számrendszerig működik. A megoldáshoz használj rekurziót! Miért sokkal egyszerűbb ez a megoldás, mint az iteratív?

▶ **Tipp**

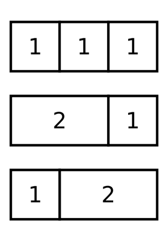
8. Három számjegyenkénti felosztás

Írj függvényt, amely a paraméterként kapott pozitív egész számot három számjegyenként csoportosított formában írja ki. Pl.: 16 077 216. Próbáld ki más számokra is: 999, 1000, 12, 0, 1000222!

▶ **Tipp**

9. Járda kövezése

Hányféleképpen lehet egy adott hosszúságú járdát kikövezni 1 és 2 méter hosszúságú járdalapokkal? Például ha 3 méteres a járda, a lehetőségek: 1+1+1, 1+2, 2+1, tehát összesen 3.



▶ **Tipp**

▶ **Megoldás**