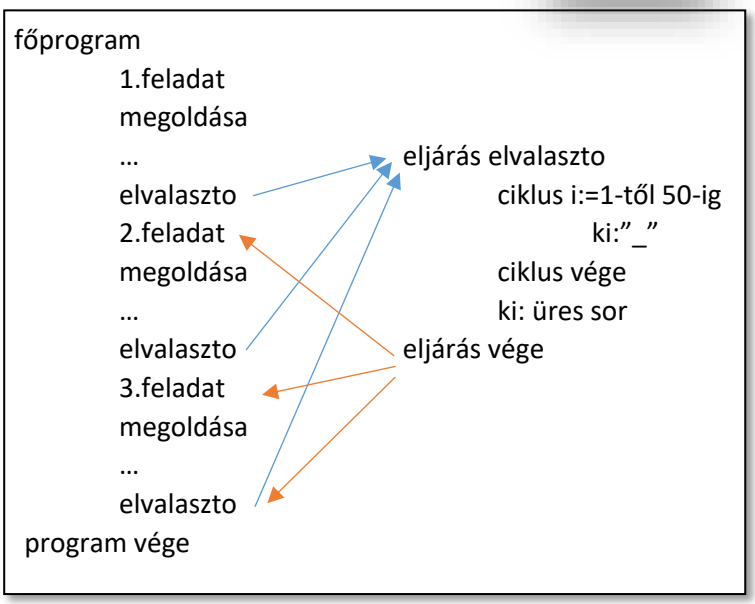
13. ELJÁRÁSOK (PROCEDURE)

Az eljárások (és függvények) tulajdonképpen **alprogramok, vagy más néven részprogramok**. **Az alprogramokat névvel látjuk el, majd a főprogramban a nevével hivatkozunk rá és végrehajtódik.**

- Az eljárások és függvények előnye a kódismétlés elkerülése.
- A programunk rövidebb, áttekinthetőbb lesz.
- Logikusabban építhetjük fel. Többször felhasználhatunk alprogramokat.
- Elég az alprogramon változtatni, ha kód hibás, vagy éppen másképp szeretnénk működtetni.
- Nagyobb program csoportmunkában fejleszthető.
- Használatakor megváltozik az alapvető sorrendi végrehajtás.



Nézzük a jobb oldali mondatszerű leírást az **eljárások meghívásá**ra:

- A főprogram elindulása után számozott feladatokat hajt végre.
- A baloldalon a főprogramot, a jobb oldalon az alprogramot látjuk.
- A képernyőre- való kiírásnál a feladatokat ötven darab „_" karakter vonallal szeretnénk elválasztani.
- Azért, hogy ne kelljen ugyanazt, többször begépelni, bemásolni, akkor eljárást használunk.
- Készítünk egy eljárást, amit meghívhatunk, ahányszor szeretnénk, egyszerűen csak a nevével.
- Aztán mindig visszatér a főprogramba.

```
eljárás alprogram2
…
eljárás vége

eljárás alprogram1
        alprogram2
        …
eljárás vége

főprogram
        alprogram1
```

Az eljárások **lefutásának sorrendje**:

- Arra mindig figyelni kell, hogy amely eljárást felhasználunk, azt mindenképpen előbbre definiálni kell!
- A jobb oldali példában látni, hogy a főprogram lefutásakor az alprogram1-et hívja meg, amelyben az alprogram2 van meghívva. Tehát az alprogram2-nek előbbre kell lennie mint az alprogram1-nek.

Eljárás paraméter nélkül

(13a.py)

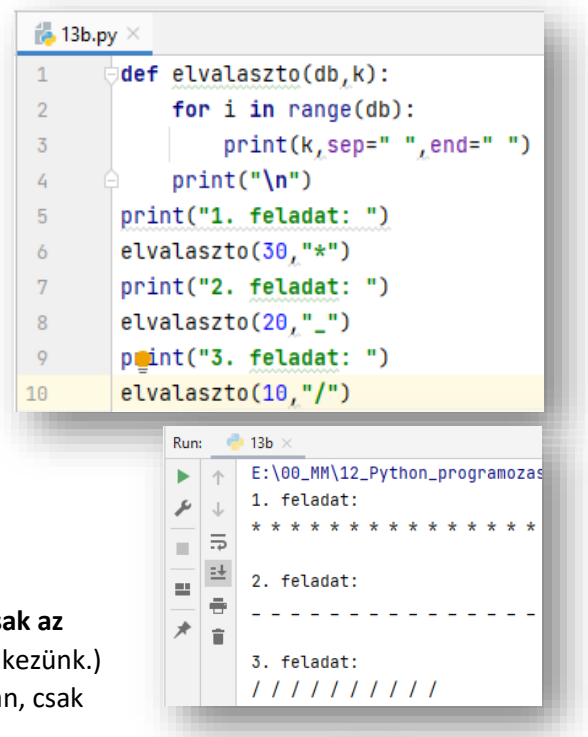
- Az eljárás létrehozását a **def kulcsszóval kezdünk**
- **Ezt követi az eljárás neve, majd egy üres zárójelpár** (ha nincsenek átadandó paraméterek) és **végül kettőspont**tal zárul a sor.
- Az eljárás **tartalmi rész**ét egy bekezdéssel beljebb kell kezdeni.
- A **tagolás vége jelzi az eljárás végét**.
- Az eljárást a főprogramból a **nevével és az üres zárójelpárral hívjuk** meg.
- A főprogramban ahányszor szeretnénk, annyiszor hívhatjuk meg az eljárást.

13a.py

```
1  def elvalaszto():
2      for i in range(60):
3          print("_",sep=" ",end=" ")
4      print("\n")
5  print("1. feladat: ")
6  elvalaszto()
7  print("2. feladat: ")
8  elvalaszto()
9  print("3. feladat: ")
10 elvalaszto()
```

Eljárás paraméterátadással

(13b.py)



A jobb oldali példán látjuk a paraméterek megadását:

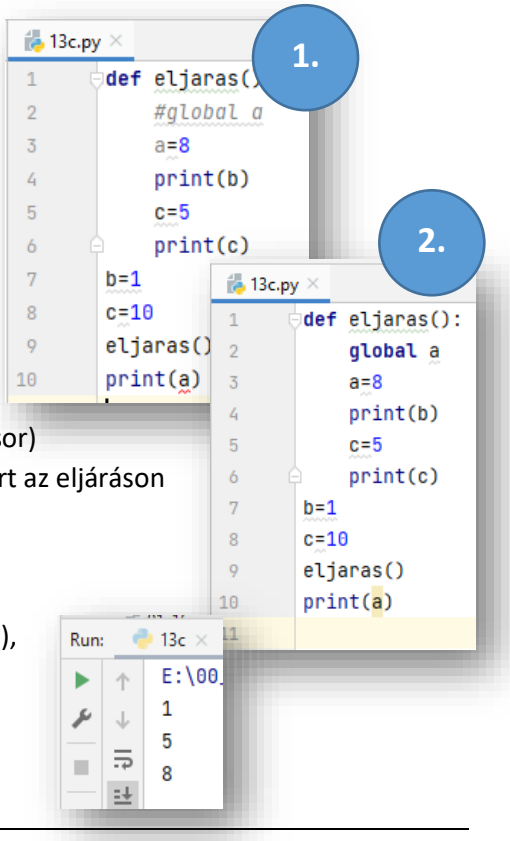
- Az eljárások működését paraméterek segítségével befolyásolhatjuk.
- Az előző példa folytatásaként / kibővítéseként megadhatjuk, hogy milyen széles legyen a vonal, és azt is, hogy milyen karakterből legyen megrajzolva.
- A paramétereket a zárójelek között adjuk meg vesszővel elválasztva egymástól.
- Ezeket a paramétereket felhasználjuk az eljárás megfelelő részében.
- Értelmezzük a mintaprogramot!

Változók hatásköre

Fontos tudni, hogy **az alprogramokban létrehozott változók, csak az eljáráson belül használhatók**. (Ha erről másképpen nem rendelkezünk.) Hiába hivatkozunk rá a főprogramban, vagy egy másik eljárásban, csak hibaüzenetet kapunk.
A **modulokban létrehozott változókat lokálisnak nevezzük**. Csak a modulban van érvényességi körük.
A **főprogramban létrehozott változókat globálisnak nevezzük**.
Érvényességi körük a teljes program, ha nincs azonos nevű lokális változó!
A Pythonban van lehetőség, hogy modulban is létrehozhatunk globális változót, ha global kulcsszót használjuk.

(13c.py)

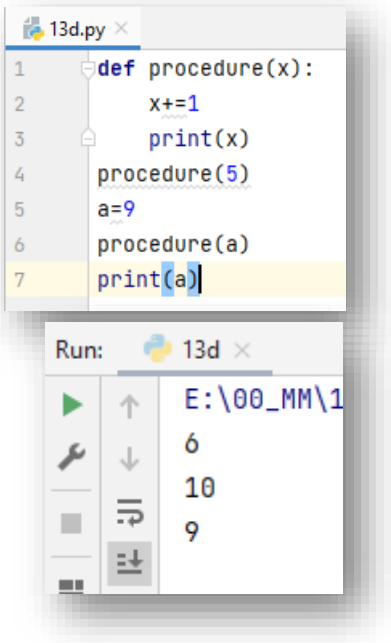
Nézzük a **változók használat**át alaposabban:



- Az „a” változót az 1. képen (3. sor) az eljáráson belül hozzuk létre, csak ott használjuk. Ez egy lokákis változó. Ezért a program lefutásakor, amikor ki akarjuk íratni (10. sor) hibát fog jelezni.
- A főprogramban létrehozott változók alapvetően a teljes programban használhatók.
- A programban létrehozunk egy „b” változót 1 értékkel (7. sor) és egy „c” változót 10 értékkel (8.sor).
- Egy „c” nevű változót az eljárásban is létrehozunk, 5 értékkel. (5. sor)
- Ebben az esetben a lokális változóértékét fogja megjeleníteni, mert az eljáráson belüli megjelenítés következik utána (6.sor).
- Ha az 5. sort kivesszük az eljárásból például # karakter eléírásával, akkor a c értékként a 10 fog megjelenni.
- Ha kivesszük a 2. sorban a # karaktert a global utasítás elől (2. kép), akkor az „a” kiíratásánál meg fog jelenni a 8 érték

Érték szerinti paraméterátadás

(13d.py)



Az előző oldalon lévő felső feladatban (13b.py) megismert módszerben, a főprogramból csak a változó értéke kerül át az alprogramba.

Ebben az esetben pedig nézzük a program lefutását részletesen:

- A 4. sorban indul a program lefutása a „procedure” alprogram meghívásával. Ahol 5-ös értéket ad át.
- A program végrehajtása az első sorban folytatódik, ahol az 5-ös az x változóba kerül, majd a második sorban eggyel növekszik, tehát 6 lesz. Ezt a 6-os értéket fogja a következő, harmadik sor miatt kiírni a képernyőre.
- Aztán az alprogramból kilépve, visszatér a főprogram ötödik sorába, ahol egy „a” nevű változó kap egy 9-es értéket.
- A hatodik sorba lépve meghívjuk újból az alprogramot, most az „a” változót beküldve.
- Így újra az első sorba ugorva, az „a” értéket használva kerül az „x” helyére, így eggyel növelve az érték már 10 és ez kerül kiíratásra a következő sorban.
- Visszaugrunk a főprogramba, ahol az eredetileg megadott „a” értéket kiíratjuk a képernyőre.

Minden esetben a logikáját kell megérteni a feladatnak, hogy mi, miért, és hogyan történik. Hogyan adunk át értéket az alprogramnak, az mit kezd vele, mi történik, amikor visszatér a főprogramba.

(13e.py)

A feladatban grafikus módban meg kell rajzolni egy magyar zászlót! Mivel a zászló három darab vízszintes téglalapból áll, ezért alprogramként definiáljunk egy „teglalap” nevű eljárást, amely rajzol egy 100*400 képpontos téglalapot! Majd a kitöltőszínek változtatásával és a kiindulópont áthelyezésével rajzoljuk meg a zászlót!

Nézzük a lépéseket:

- Meghívjuk a rajzoláshoz szükséges „turtle” eszközt.
- Definiálunk egy olyan eljárást, amely megrajzol egy téglalapot. Felhasználjuk az eljárás elején a téglalapok kitöltéséhez szükséges utasítást!
- A főprogramban először letöröljük a képernyőt a „reset()” utasítással.
- Majd megadjuk a rajzolószínt a „color()” utasítással.
- Aztán az első kitöltőszínt adjuk meg a „fillcolor()” utasítással.
- Meghívjuk az eljárásunkat.
- Felemeljük a tollat. up()
- Elpozícionálunk a 100 kp-al arrébb, ahova rajzolni kell a második téglalapot, és irányba állunk. Aztán letesszük a tollat. down()
- Majd a következőkben megrajzoljuk a másik két téglalapot a megváltoztatott színekkel és át pozícionált kiinduló pontokkal, a minta szerint!
- Teszteljük a programot közben és javítsuk az esetleges hibákat!!