Nézzük részletesen a programunkat:

```
16d.py
1  import datetime
2  auto=["AAA-111, szűrke, Citroen, C4, 2012, 3500000",
3        "BBB-222, fehér, Audi, A4, 2015, 5500000",
4        "CCC-333, kék, BNW, Q6, 2020, 25000000",
5        "DDD-444, zöld, Skoda, Octávia, 2008, 1300000"]
6  ma = datetime.date.today()
7  #print(ma.year)
8  for i in range(4):
9          auto[i]=auto[i].replace(",","")
10 print(auto)
11 print("----------------------")
12 evek=0
13 atlag=0
14 max=0
15 for i in range(4):
16       auto_adatok=auto[i].split()
17       print(auto_adatok)
18       if max<int(auto_adatok[5]):
19             max=int(auto_adatok[5])
20       evek=evek+int(auto_adatok[4])
21       atlag=evek/4
22 print("----------------------")
23 print("Az autók átlagéletkora: ",ma.year-atlag," év")
24 print("A legdrágább autó ára: ",max," Ft.")
```

- A programunk első sorában meghívjuk a „datetime” modult, mert a későbbiekben szükségünk lesz az aktuális dátumra.
- A második sortól létrehozzuk az auto nevű tömböt, melybe begépeljük a négy autó adatait vesszőkkel elválasztva, a minta szerint!

AAA-111, szűrke, Citroen, C4, 2012, 3500000
BBB-222, fehér, Audi, A4, 2015, 5500000
CCC-333, kék, BNW, Q6, 2020, 25000000
DDD-444, zöld, Skoda, Octávia, 2008, 1300000

- A hatodik sorban egy ma nevű változóba beletesszük az aktuális dátumot!
  ma=datetime.date.tuday()
  Majd a következő sorba kiíratjuk a képernyőre! Később ezt # karakterrel megjegyzésbe tesszük! Sőt mivel az autók átlagéletkorának kiszámításához csak az évre lesz szükség, a dátumról az évet jelenítjük csak meg! (*ma.year()*)
- A nyolcadik, kilencedik sorban kivesszük a vesszőket a szövegünkből úgy, hogy a négy tömbelemen végig megyünk for utasítással a replace() parancs felhasználásával!
- Ellenőrzésképpen kiíratjuk az auto tömb tartalmát és beszúrunk egy választóvonalat a következő sorba!
- A 12-14 sorokban létrehozunk változókat, melyekre szükség lesz a számolásokban. Lenullázzuk őket!
- A 15. sortól indul a program lényege.
  Ugye szükségünk van az autók átlag életkorára és a legdrágább autó árára!
  Ehhez az adatink közül a rendszámra, színre, márkára, típusra nincs szükség.
  Viszont a tömbünk elemeinek ötödik „elemére” az évszámra igen, és a legvégén szereplő milliós összegre is!
  Arra kell figyelni, hogy ezek az értékek stringek, és arra is, hogy a belső tömbünk ötödik és hatodik (a 0 kezdősorszám miatt 4. és 5.) helyen állnak!
- Először for ciklust indítok, hogy végig menjen a 4 autó adatán!
- Aztán a split() utasítással a szóközök mentén újabb tömböket alakítok ki, auto_adatok néven! (14. sor)
- Kiíratom a tömb elemeit, soronként! (15. sor)
- Aztán megvizsgálom, hogy a max aktuális értéke nagyobb-e mint az auto_adatok utolsó eleme (az autó ára)! Mert ha igen akkor cseréljük, különben megy tovább a program!
- Majd az autók átlagéletkorának kiszámításához összeadom az autók gyártási éveit, majd elosztjuk néggyel. Erre kapunk egy 2013.75-ös értéket!
- Végül kiíratjuk az eredményeket a minta szerint!
- Az átlagéletkort úgy kell kiíratni, hogy mindig az aktuális évhez képest mondja meg a program, hogy mennyi idősek az autók. Ezért akkor fog pontosan működni a programunk, ha a „ma” változó évéből kivonjuk az autók gyártási évének átlagát!
- Futtassuk a programunkat, és javítsuk az esetleges hibákat! Mindig úgy használjuk a különböző utasításokat, hogy végiggondoljuk, hogy mi miért történik, és tudatosan haladjuk a sorokban! Ne csak másolás legyen!

Listák leképezése (list comrehension)

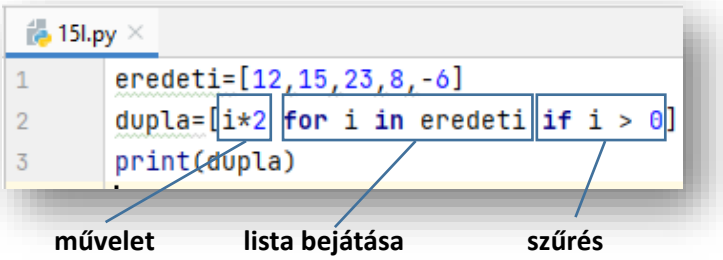
Ez a kifejezés egy kicsit idegenül hangozhat számunkra, de ha egyszerűbben akarunk fogalmazni, akkor mondhatjuk hogy a **listákat feldolgozzuk azokkal műveleteket végzünk**. **Tulajdonképpen lerövidíthetjük a kódunkat**. **Összevonhatjuk a műveleteket egy sorba!**
Nézzünk egy egyszerű példát:

- Vegyünk egy „szamok” nevű öt elemű listát **-> szamok=[12,15,23,8,-6]**
- A feladat az, hogy a lista elemeinek értékét duplázzuk meg, akkor, ha a szám pozitív és az eredményül kapott értékeket tároljuk egy „dupla” nevű listában. **-> dupla =[24,30,26,16]**
- Ezt meg tudjuk valósítani a mostani tudásunk alapján is a következők szerint:

```
1    szamok = [12,15,23,8,-6]
2    dupla = []
3    for i in szamok:
4        if i > 0:
5            dupla.append(i * 2)
6    print(dupla)
```

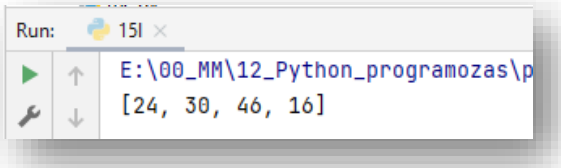
Nézzük meg ennek a feladatnak az egyszerűbb megoldását a leképezést, amikor összevonjuk a műveleteket egy sorba:

(15l.py)



Tehát az „eredeti” nevű listában szereplő számok közül duplázzuk meg a pozitív tagok értékét és tegyük bele egy „dupla” nevű új listába, majd írassuk ki a képernyőre a „dupla” lista tartalmát!

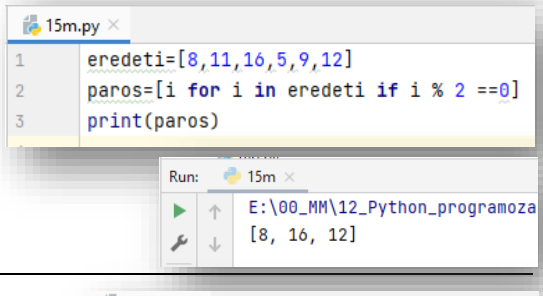
- Először töltsük fel az „eredeti” listát a minta szerinti számokkal.
- Majd a második **sor elején megadjuk az új lista nevét, és az egyenlőség jel után a két szögletes zárójelet!**
- **A zárójelen belül először mindig a lista bejárását adjuk meg az eddig tanult módon.** (for i in eredeti)
- **Majd megadjuk az elejére, hogy milyen műveletet szeretnénk elvégezi.** (i*2)
- **Végül az esetleges feltételt is megadjuk a művelet bejárása után.** (if i>0)
- Aztán a harmadik sorban egyszerűen kiíratjuk a „dupla” nevű lista tartalmát.



(15m.py)

A feladatban gyűjtsük ki a páros számokat egy új listába!

- A lista bejárása és a szűrés egyértelmű!
- A művelet megadásánál csak egyszerűen beletesszük a páros nevű listába. Tehát csak egy „i”-t adunk meg.



(15n.py)

A feladatban az „eredeti” listában lévő számokat kell négyzetre emelni és egy új listába beletenni.

- Ha nincs feltétel egyszerűen kihagyjuk!