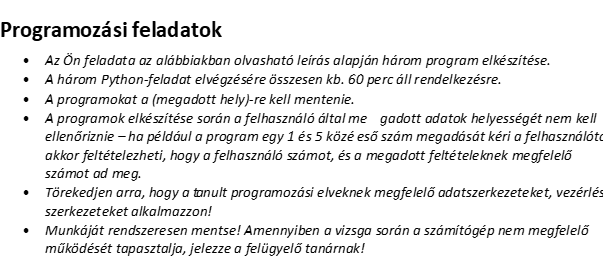
# Dolgozat



A projektet mentse: neve\_csoportja helyre. A modulokat a főprogramból (main.py) hívja meg eljáráshívásokkal.

## Első feladatt -8 pont

Írjon modult **szin.py** néven! A program kérjen be egy szín keverési módszert szövegként. pl.: RGB, HEX, HSL, RGBA, HSLA …stb. Majd olvassa be a hozzá tartozó színkódot szintén szövegként. Ha a felhasználó HEX kulcsszót adott meg akkor 6 hosszúnak kel lennie a kódnak. Ha a felhasználó RGB akkor 5-11 karakterű lehet a kód, ha vagy HSL kódot adott meg akkor 7-13 karakterű lehet a kód. Írd ki a felhasználónak a mintában szereplő kódokat, hogy megfelelnek-e hossz alapján a feltételeknek. A többi módszernél írja ki a program, hogy bonyolult kiszámolni.

A program üzeneteinek megfogalmazásában kövesse az alábbi példát! Azokat a részeket, amiket a felhasználó gépel be, a mintában vastagított és döntött betűkkel emeltük ki.

Minta futások:

Első feladat:

Kérek egy színkeverési módszert: ***RGB***

Kérem a kódot: **60,179,113**

Megfelelő hossz.

Első feladat:

Kérek egy színkeverési módszert: ***HSL***

Kérem a kódot: **0,100%,50%**

Megfelelő hossz.

Első feladat:

Kérek egy színkeverési módszert: ***HSL***

Kérem a kódot: **20,1000%,5000%**

Nem megfelelő hossz.

Első feladat:

Kérek egy színkeverési módszert: ***RGBA***

Kérem a kódot: **60,179,113,1**

Bonyolult kiszámolni.

### Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz modult szin.py néven, a program hibaüzenet nélkül lefut.
2. Bekéri a felhasználótól a színt, és tárolja.
3. Bekéri a felhasználótól a kódot, és tárolja
4. Ellenőrzi a bekért kódok hosszát, tárolja.
5. Elágazást használ a különböző esetek kezelésére.
6. Helyesen állapítja meg és írja ki, hogy megfelelő hosszú vagy nem.
7. Helyesen állapítja meg, és írja ki, hogy bonyolult kiszámolni.
8. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök, írásjelek).

## Második feladat -14 pont

A program egy listát vizsgál. Egy függvény segítségével megszámolja, hogy a listában hány darab héttel osztható szám található.

Írjon modult **oszthato.py** néven!

Írja meg azt a **hettelOszthato** nevű függvényt, amely megszámolja egy adott listában hány darab héttel osztható szám van! A függvény paramétere lista legyen, a visszatérési értéke a listában szereplő héttel osztható számok darabszáma legyen!

Hozzon létre egy **szamok** listát, melyet töltsön fel 50 darab 1 és 100 közé eső véletlen számokkal! Írja ki a lista elemeit!

Hívja meg a programban a megírt függvényt és írja ki, hogy hány darab héttel osztható szám van a listában!

Minta:

A lista elemei:

72, 40, 8, 36, 29, 51, 38, 34, 89, 70, 77, 3, 37, 5, 47, 96, 41, 83, 31, 62, 75, 66, 9, 70, 61, 54, 57, 47, 55, 37, 9, 42, 5, 20, 49, 73, 70, 71, 27, 31, 82, 50, 70, 8, 81, 84, 33, 10, 37, 16

A listában  8  darab héttel osztható szám van.

### Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot **oszthato.py** néven, a program hibaüzenet nélkül lefut.
2. Helyesen írja meg a **hettelOszthato()** függvényt.
3. Helyes paraméterrel írja meg a függvényt.
4. Helyesen adja meg a visszatérési értékeket.
5. Helyesen írja meg a függvény törzsét.
6. Létrehozza a **szamok** listát.
7. Helyesen állít elő egy véletlen számot.
8. Hozzáfűz egy véletlen számot a listához.
9. Ciklust szervez a véletlen számok listába írására.
10. Minden véletlen számot hozzáfűz a listához.
11. Helyesen írja ki a lista elemeit.
12. Helyesen hívja a **hettelOszthato** () függvényt.
13. A **hettelOszthato** () függvény hívásával helyesen írja ki a héttel osztható számok számát.
14. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök, mondatvégi írásjelek).

Harmadik feladat – 18 pont

## Harmadik feladat 18 pont

Készítsen programot **autom.py** néven. Olvassa be **auto.txt** fájlból az auto adatait, tárolja **Auto** osztály típusú **kocsi** nevű listában a példányokat.

Készítsen metódust:

* **ko**r néven, amely a paraméterként kapott autó márkáját és korát adja vissza a konzolra.
* **kiir** metódust, amely kiírja **ki.txt** fájlba az autók márkáját és korát az alábbi formában.

Készíts függvényt:

* **flotta** néven, amely kiírja hány autónk van.
* **ertekes** néven, ami kiírja a legöregebb autó nevét, és gyártási dátumát.

Minta:

III/Kor:

mercedes – 7 éves

III/Flotta:

Autók száma: 7.

III/Értékes

A legöregebb autó: Suzuki Swift, 1999.

ki.txt

mercedes – 7 éves

### Pontozás – minden teljesülő feltétel egy-egy pontot ér

1. Létrehoz programot autom.py néven, a program hibaüzenet nélkül lefut.
2. Listát hoz létre kocsi néven az autók Auto osztályú objektum tárolására.
3. A fejlécet nem olvassa be.
4. Beolvas egy adatsort.
5. Beolvas egy sornyi adatot a fájlból.
6. A beolvasott adatokból Auto osztályú objektumot hoz létre.
7. Egy Auto osztályú objektumot elhelyez a létrehozott kocsi adatszerkezetben.
8. Minden beolvasott alapján három objektumot hoz létre.
9. Minden objektumot elhelyezi az adatszerkezetben.
10. Az adatszerkezetben tárolt objektumok alapján megjelenít az objektumot a Név - kor formában, a konzolon.
11. A kor() metódus megfelelően számolja ki az autó korát.
12. A kor() metódusnak van visszatérési értéke, és megfelelő.
13. A kiir() metódus létezik. A ki.txt kiir() metódus minden adatsort kiír megfelelően.
14. Létezik flotta() függvény és kiírja az autók számát.
15. Létezik ertekes() néven függvény.
16. Megkeresi a legöregebb autót nevét.
17. Megkeresi a legöregebb autót gyártási dátumát.
18. A kiírt üzenetek helyesek (pl.: nincs benne elgépelés, helyesen jelennek meg a szóközök, jók az írásjelek).