2. LISTA BEJÁRÁSA (FOR)

Gyakori feladat, hogy a lista minden elemével valamilyen műveletet kell végrehajtanunk. Legegyszerűbb esetben formázottan kiíratni, vagy feltölteni véletlenszerű értékekkel. Ilyenkor azt mondjuk, bejárjuk a listát.

A lista bejárásához készítették a Pythonban a **for** ciklust eredeti alakjában.

Ha csak a lista elemeire van szükség, és az indexük nem fontos, alkalmazhatjuk a 3–4. sorban látható módon. Az **in** mögött megadjuk a lista nevét (honap). A ciklus minden egyes lefutásakor a lista egy még sorra nem került eleme (azaz a példában az egyik hónap neve, ami még nem volt az előző lefutások alkalmával) az i változóban lesz. Az ismétlés addig tart, míg a lista minden eleme sorra nem került. Így a példában a 3–4. sor kiírja a képernyőre a listában tárolt hónapok nevét.

Kicsit bonyolultabb a helyzet, ha szükség van az elem indexére is.

Ehhez először ismerkedjünk meg a **len** utasítással, ami megadja a lista elemeinek számát. Az 5. sor eredménye az 5 lesz, hiszen a listában 5 hónap neve található.

Tegyük fel, hogy a hónapok neve előtt szeretnénk megjeleníteni a sorszámukat is. Ehhez szükségünk lesz az elemek indexére is. Ilyen esetekben az **in** mögött az *indexek tartományát* adjuk meg (6. sor). A **len** utasítás megadja a lista elemszámát (a példában 5), így a **range** a 0-tól kezdődően egy egész számokból álló sorozatot hoz létre, amik pont a lista indexei (a példában: 0, 1, 2, 3, 4). A j változóba a ciklus minden egyes lefutásakor egy index kerül. Ezt használjuk fel a 7. sorban előbb a hónap sorszámának (j+1, mert 0. hónapról nem szokás beszélni), majd nevének (honap[j]) kiírására.

```
1 honap = ["január", "február", "március", \
2          "április", "május"]
3 for i in honap:
4     print(i, end=" ")
5 print("\nA tömb mérete: ", len(honap))
6 for j in range(len(honap)):
7     print("%d. hónap: %s" % (j+1, honap[j]))
```

49. mintafeladat – listaelemek bejárása

Készítsünk egy listát az első öt hónap nevével. Jelenítsük meg először a hónapok nevét, egymástól szóközzel elválasztva. Határozzuk meg a lista elemeinek számát. Írassuk ki a hónapok nevét a sorszámukkal együtt. Minden hónap külön sorba kerüljön, ilyen formában: 2. hónap: február.

```
============== RESTART: C:/python/l
istabejaras.py ==============
január február március április május
A tömb mérete:  5
1. hónap: január
2. hónap: február
3. hónap: március
4. hónap: április
5. hónap: május
```

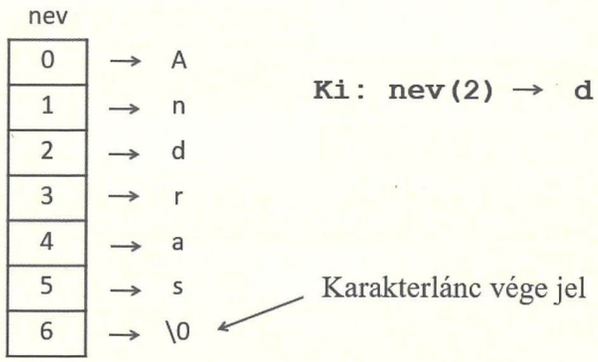
1. Gépeljük be az elméletnél látható Python-kódot (1–7. sor). Próbáljuk ki.

Feladatok

1. Tároljuk egy listában a hét napjait. Írassuk ki a képernyőre a hét napjait sorrendben, egymás mellé, szóközzel elválasztva.
2. Tároljuk egy listában a hét napjait. Írassunk ki a képernyőre véletlenszerűen 15 db nap nevét egymás mellé, szóközzel elválasztva.
3. Az amőbajáték ábrázolásához háromféle karaktert használunk: 'X', 'O', és '.' Utóbbival az üres helyeket ábrázoljuk. Tároljuk egy listában a felhasznált karaktereket. Írassuk ki a képernyőre az amőbajáték ábrázolásához használt karaktereket egymás mellé.
4. Töltsük fel egy lista 50 elemét egy dobókocka véletlenszerű értékeivel, majd írassuk ki a képernyőre ilyen módon: 1 3 1 4 4 5 6 2... A feltöltés és a kiíratás két külön ciklussal történjen.
5. Töltsük fel egy lista 100 elemét fej vagy írás véletlenszerű értékeivel, majd írassuk ki a képernyőre ilyen módon: *fej írás* írás fej... A feltöltés és a kiíratás két külön ciklussal történjen.

Fogalma

A karakterlánc (string) egy különleges karakterlistának tekinthető. Ennek megfelelően minden karakterére hivatkozhatunk a karakterlánc nevével és a betű sorszámával. Tárolja például a *nev* nevű karakterlánc azt a nevet, hogy András. Ebben az esetben a nev(2) értéke a d betű lesz. A számozás itt is a 0-tól indul.



A karakterlánc azért különleges karaktertömb, mert tartalmaz egy lezáró karaktert, amit \0-val jelölünk. Ennek segítségével tudják megállapítani a Python beépített függvényei, hol a karakterlánc vége.

Alapvető műveletek karakterláncokkal

A karakterlánc akárhányadik karaktere lekérdezhető ilyen módon. Például: *Ki: nev(2)*(4. sor).

```
1) w = "isz"
2) hb = ""
3) hb = "h" + w + "ed" → hiszed
4) print (w[2])      →      z
5) w[2] = 'i'
```

Ellenben a Pythonban nem változtathatunk meg egyszerű értékadással olyan karaktert, ami része egy karakterláncnak (5. sor). Azonban a kívánt hatást elérhetjük, ha a lecserélendő karaktert először töröljük, majd a helyére beszúrjuk a kívánt karaktert.

Üres karakterláncot a 2. sorban található módon hozhatunk létre.

A karakterláncok összefűzhetők a + jel segítségével (3. sor). A művelet eredménye egy karakterlánc, ami balról jobbra sorrendben egymás után tartalmazza a kiindulási karakterláncokat. A *w* változóban az *isz* karakterlánc tárolódik, így hb = "h" +"isz"+ "ed" = "hiszed".

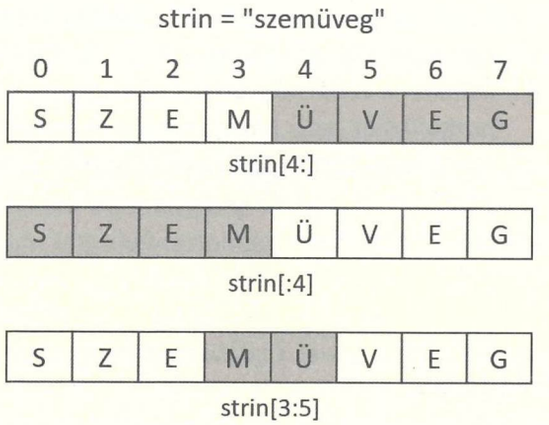
A szövegösszefűzést szaknyelven **konkatenációnak** is hívják.

50. mintafeladat – ismerkedés a karakterláncokkal

1. Álljunk a parancsértelmező ablakba (**Python 3.8.0 Shell**).
2. A prompt jel >>> mögött villog a kurzor. Oda írjuk majd be a parancsokat, mindegyik után megnyomva az **Enter** billentyűt.
3. Készítsünk egy *str1* nevű karakterláncot, ami ezt tartalmazza: *isz*. Írjuk be, majd üssünk **Entert**:
   ```
   str1 = "isz"
   ```
4. Hozzunk létre egy *hb* nevű, üres karakterláncot. Írjuk be, majd üssünk **Entert**:
   ```
   hb = ""
   ```
5. A *hb* karakterláncban fűzzük egymás után a *h* betűt, az *str1* változó tartalmát, majd az *ed* karakterláncot. Az összefűzést a Pythonban a + jel jelöli. A művelet eredménye egy karakterlánc, ami balról jobbra sorrendben egymás után tartalmazza a kiindulási karakterláncokat. Írjuk be, majd üssünk **Entert**:
   ```
   hb = "h" + str1 + "ed"
   ```
6. Írassuk ki a *hb* változó tartalmát. Az *str1* változóban az *isz* karakterlánc tárolódik, így hb = "h" +"isz"+"ed" = "hiszed". Írjuk be, majd üssünk **Entert**:
   ```
   print(hb)
   ```
7. Írassuk ki a *hb* változó 3. karakterét. (Z betűt kell kapnunk eredményként.) Írjuk be, majd üssünk **Entert**:
   ```
   print(hb[3])
   ```
8. Próbáljuk megváltoztatni a *hb* változó 0. karakterét *v* betűre. Írjuk be, majd üssünk **Entert**:
   ```
   hb[0] = "v"
   ```
9. Hibaüzenetet fogunk kapni: *'str' object does not support item assignment*. Sajnos a karakterek nem helyettesíthetők olyan egyszerűen, mint a lista elemei.

Szeletelés

A karakterláncnak hivatkozhatunk a tartományaira a mellékelt ábra alapján. Ilyenkor a szögletes zárójelbe két számot írhatunk kettősponttal elválasztva (pl. strin[3:5]). Az első szám a tartomány kezdő pozícióját jelzi. A második szám a végpozíciónál eggyel nagyobb. Az ü a 4. karakter, helyette az 5-öt kell megadni.



Ha a második számot hagyjuk el, a tartomány a karakterlánc végéig tart (pl. strin[4:]).

Ha az első számot hagyjuk el, a tartomány a karakterlánc elejétől kezdődik (pl. strin[:4]).

Az itt megismert technikát szeletelésnek hívják, és a listákra is végrehajtható.

51. mintafeladat – alapvető karakterlánc műveletek

1. Álljunk a parancsértelmező ablakba (**Python 3.8.0 Shell**).
2. A prompt jel >>> mögött villog a kurzor. Oda írjuk majd be a parancsokat, mindegyik után megnyomva az **Enter** billentyűt.
3. Készítsünk egy *str1* nevű karakterláncot, ami ezt tartalmazza: *hiszed*. Írjuk be, majd üssünk **Entert**:
   ```
   str1 = "hiszed"
   ```
4. Határozzuk meg a karakterlánc hosszát, azaz hogy hány karakterből áll. (6 lesz a helyes eredmény.) Írjuk be, majd üssünk **Entert**:
   ```
   print(len(str1))
   ```
5. Írassunk ki a karakterlánc 2. (azaz 1 indexű) karakterétől számítva 3 karaktert. A 3 karakter a harmadik pozícióban végződik. Ennél eggyel nagyobb számot kell megadni. Ezért a kettőspont mögötti szám a 4. (*isz* a helyes eredmény). Írjuk be, majd üssünk **Entert**:
   ```
   print(str1[1:4])
   ```

6. Töröljük a karakterlánc első (azaz 0 indexű) karakterét. Ezt úgy tehetjük meg, hogy a többi karaktert (0 indexű utániakat, tehát az 1-től kezdve) áttöltjük az eredeti változóba. (Ezt kapjuk: *iszed.*) Írjuk be, majd üssünk **Entert**:
```
str1 = str1[1:]
print(str1)
```
7. Cseréljük ki a karakterlánc negyedik (azaz 3 indexű) karakterét o betűre. Ezt úgy tehetjük meg, hogy a 0.-tól kezdődően kivágjuk az első 3 karaktert (a 3 karakter a 2.-nál végződik, ennél eggyel nagyobbat kell megadni, tehát 3-at), hozzáfűzzük az o betűt, majd a 4. karaktertől kezdve mellé illesztjük a szó maradékát. (Ezt kapjuk: *iszod.*) Írjuk be, majd üssünk **Entert**:
```
str1 = str1[:3]+"o"+str1[4:]
print(str1)
```

Feladatok

1. Kérjünk be egy szót, majd írassuk ki a hosszát.
2. Kérjünk be egy szót, majd írassuk ki a kezdőbetűjét.
3. Kérjünk be egy szót, majd írassuk ki az utolsó betűjét.
4. Kérjünk be egy ötbetűs szót, és írassuk ki a középső három karakterét.
5. Kérjünk be egy budapesti irányítószámot, és írjuk ki, melyik kerületben található. A budapesti irányítószámok középső két karaktere adja az irányítószámot.
6. Készítsünk egy karakterláncot, ami a *leszel* szót tartalmazza. Minden feladatrész után írjuk ki a karakterláncot. Ne használjunk segédváltozót.
   a. Változtassuk meg a kezdőbetűt L-re.
   b. Változtassuk meg az utolsó betűt k-ra.
   c. Töröljük a kezdő- és utolsó betűjét.
   d. Bővítsük a szó végét egy m betűvel.
7. Kérjük be a felhasználó vezeték-, majd utónevét két külön változóba. Fűzzük össze a vezetéknév első karakterét a teljes utónevével. Írassuk ki az így keletkezett karakterláncot csupa kisbetűvel. (Például: Bodor Elek→belek)
8. Kérjük be a felhasználó vezeték-, majd utónevét két külön változóba. Fűzzük össze a vezeték- és utónevének első három karakterét, majd egészítsük ki 01-gyel. Írassuk ki az így keletkezett karakterláncot csupa kisbetűvel. (Például: Kiss Ferenc→kisfer01)
9. Kérjünk be egy szót. Írassuk ki, hogy 'e' betűvel kezdődik-e.
10. Kérjünk be két szót. Írassuk ki őket hosszúság szerint növekvő sorrendben.
11. Kérjünk be két ötbetűs szót, hasonlítsuk össze a három belső karaktert, és írjuk ki azt, amelyik ábécésorrendben előrébb van.