Python adatszerkezetek

1. óra: Listák - alapműveletek

Elmélet

- Lista létrehozása szögletes zárójelek használatával.
- Elemek elérése indexek segítségével.
- Elemek módosítása adott indexen.
- Elemek törlése a listából.
- A lista hosszának meghatározása a `len()` függvénnyel.

Kód példa

```
# Üres lista létrehozása
gyumolcsok = []

# Lista létrehozása elemekkel
gyumolcsok = ["alma", "banán", "cseresznye"]

# Elem elérése index segítségével
print(gyumolcsok[1])  # banán

# Utolsó elem elérése index segítségével
print(gyumolcsok[-1])  # cseresznye

# Elem módosítása
gyumolcsok[1] = "narancs"
print(gyumolcsok)  # ['alma', 'narancs', 'cseresznye']

# Elem törlése
del gyumolcsok[0]
print(gyumolcsok)  # ['narancs', 'cseresznye']

# Lista hossza
print(len(gyumolcsok))  # 2
```

Önálló gyakorlati feladatok

1. Hozz létre egy listát kedvenc ételeidről, majd írd ki a listában található harmadik elemet.
2. Cseréld ki a lista utolsó elemét egy új ételre, és írd ki a listát.
3. Töröld a lista első elemét, és számold meg, hány elem maradt a listában.
4. Töröld a lista utolsó elemét.

5. Hozz létre egy listát kedvenc autó márkáidról. Ezután egy ciklus segítségével listázd ki őket egyenként és minden márka elé írj sorszámot.
6. Kérd be a felhasználótól, hogy melyik sorszámú márkát szeretné törölni a fenti listáról, majd jelenítsd meg újra a szűkített listát.

2. óra: Listák - metódusok

Elmélet

- `append()` : új elem hozzáadása a lista végéhez.
- `extend()` : több elem hozzáadása a lista végéhez.
- `insert()` : új elem beszúrása adott pozícióra.
- `remove()` : megadott elem eltávolítása.
- `sort()` : a lista elemeinek rendezése.
- `reverse()` : a lista elemeinek megfordítása.
- https://www.w3schools.com/python/python_ref_list.asp

Kód példa

```
szamok = [3, 1, 4, 1, 5, 9]

# Elem hozzáadása
szamok.append(2)
print(szamok)  # [3, 1, 4, 1, 5, 9, 2]

# Több elem hozzáadása
szamok.extend([6, 7])
print(szamok)  # [3, 1, 4, 1, 5, 9, 2, 6, 7]

# Elem beszúrása
szamok.insert(1, 8)
print(szamok)  # [3, 8, 1, 4, 1, 5, 9, 2, 6, 7]

# Elem törlése
szamok.remove(1)
print(szamok)  # [3, 8, 4, 1, 5, 9, 2, 6, 7]

# Lista rendezése
szamok.sort()
print(szamok)  # [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Lista megfordítása
szamok.reverse()
print(szamok)  # [9, 8, 7, 6, 5, 4, 3, 2, 1]

# Rajta van az 5-ös szám a listán?
if 5 in szamok:
    print("Az 5 szerepel a listán")
```

Gyakorlati feladatok

1. Hozz létre egy listát kedvenc filmjeidről, majd adj hozzá egy új filmet a lista végéhez.
2. Szúrd be a listába egy film címét a második pozícióra.
3. Rendezze a listát ábécé sorrendbe, és írd ki az eredményt.
4. Kérj be a felhasználótól 5 nevet, és tárold őket egy listában. Ezután írd ki a neveket fordított sorrendben.
5. Készíts egy üres listát, majd kérj be a felhasználótól számokat, amíg 0-t nem ad meg. Végül írd ki hány szám került fel a listára a lista hosszának lekérdezésével.
6. Készíts egy listát kedvenc könyveidről. Ezután kérj be egy könyvcímet a felhasználótól, és add hozzá a listához, ha még nem szerepel benne.

3. óra: Tuple

Elmélet

- Tuple létrehozása kerek zárójelek használatával.
- Elemeik nem módosíthatók (immutabilitás).
- Elem elérés index segítségével.
- Tuple-k gyorsabbak és kevesebb memóriát igényelnek, mint a listák.

Kód példa

```
gyumolcsok = ("alma", "banán", "cseresznye")

# Elem elérése
print(gyumolcsok[1])  # banán

# Hossz meghatározása
print(len(gyumolcsok))  # 3
```

Gyakorlati feladatok

1. Hozz létre egy Tuple-t kedvenc színeiddel, majd írd ki a második színt.
2. Készíts egy Tuple-t négy különböző számról, és írd ki a tuple hosszát.
3. Próbáld meg módosítani a Tuple egyik elemét. Mit tapasztalsz?
4. Próbáld meg törölni a Tuple egyik elemét. Mit tapasztalsz?
5. Hozz létre egy Tuple-t számokkal, és add össze a benne található értékeket egy for ciklussal
6. Hozz létre egy Tuple-t amely 1-től 9-ig tartalmaz számokat, vegyes sorrendben. Kérd be a felhasználótól, hogy melyik számot keresi, majd egy ciklussal keresd meg, és írd ki, hogy hányadik helyen van a szám a Tuple-ben.

4. óra: Szótárak - alapműveletek

Elmélet

- Szótár létrehozása kulcs-érték párok segítségével.
- Elemek elérése kulcs alapján.
- Kulcs-érték párok hozzáadása és módosítása.
- Szótár hossza a `len()` függvénnyel.

Kód példa

```
szotar = {"alma": "piros", "banán": "sárga"}

# Elem elérése
print(szotar["alma"])  # piros

# Új kulcs-érték pár hozzáadása
szotar["cseresznye"] = "piros"
print(szotar)  # {'alma': 'piros', 'banán': 'sárga', 'cseresznye': 'piros'}
```

Gyakorlati feladatok

1. Hozz létre egy szótárat kedvenc könyveiddel, ahol a kulcs a könyv címe, az érték pedig az oldalak száma.
2. Írd ki hány értékpárt tartalmaz a szótár.
3. Adj hozzá egy új kulcs-érték párt a szótárhoz.
4. Írd ki hány értékpárt tartalmaz most a szótár.
5. Kérdezd le az egyik könyv oldalainak számát, majd jelenítsd meg.
6. Vizsgáld meg mi történik, ha olyan könyvre hivatkozol ami nincs a listán?
7. Módosítsd az egyik könyv oldalszámát.
8. Írasd ki a teljes szótárat.

5. óra: Szótárak - metódusok

Elmélet

- `get()` : elem lekérése kulcs alapján.
- `pop()` : elem lekérése és törlése kulcs alapján.
- `clear()` : szótár teljes törlése.
- `copy()` : szótár másolása.
- https://www.w3schools.com/python/python_ref_dictionary.asp

Kód példa

```
# Szótár létrehozás
szotar = {"alma": "piros", "banán": "sárga", "szilva": "kék"}

# Érték lekérdezése (metódus nélkül)
print(szotar["szilva"])

# Érték lekérdezése get() metódussal
print(szotar.get("szilva"))

# Nem létező kulcsra hivatkozás
print(szotar["rosszkulcs"]) # Hibát ad!

# Nem létező kulcsra hivatkozás get() metódussal
print(szotar.get("rosszkulcs")) # None eredményt ad és nem hibát!

# Kulcs létezésének ellenőrzése get() metódussal
if szotar.get("rosszkulcs") is None:
    print("Nincs ilyen kulcs")

# Kulcs létezésének ellenőrzése in operátorral
if "rosszkulcs" not in szotar:
    print("Nincs ilyen kulcs")


# Érték lekérdezése és egyidejű törlése a pop() metódussal
szotar = {"alma": "piros", "banán": "sárga", "szilva": "kék"}
ertek = szotar.pop("banán")  # Érték visszaadása és törlés
print(ertek)  # Érték kiírása: sárga
print(szotar)  # Vegyük észre, hogy a szótárból törölve a kulcs-érték pár

# Teljes törlés
szotar.clear()
print(szotar)


# Szótár viselkedése egyenlőség operátorral
szotar1 = {"alma": "piros", "banán": "sárga", "szilva": "kék"}
szotar2 = szotar1

# Látszólag átmásolta, mert a két szótár egyenlőnek tűnik
print("szotar1=", szotar1)
print("szotar2=", szotar2)

# Azonban az egyiken való módosítás (pl. törlés) a másikat is módosítja!
# A két szótár a gyakorlatban nem másolódik át, hanem ugyanaz marad, csak más névvel.
del szotar2["szilva"]
print("szotar1=", szotar1)
print("szotar2=", szotar2)
```

```python
# Szótár viselkedése copy() metódussal
szotar1 = {"alma": "piros", "banán": "sárga", "szilva": "kék"}
szotar2 = szotar1.copy()

# Másolás után két szótár egyenlőnek tűnik
print("szotar1=", szotar1)
print("szotar2=", szotar2)

# Az egyiken való módosítás (pl. törlés) nem befolyásolja a másikat!
# Itt a két szótár a ténylegesen átmásolódik és függetlenek egymástól.
del szotar2["szilva"]
print("szotar1=", szotar1)
print("szotar2=", szotar2)
```

Gyakorlati feladatok

1. Hozz létre egy szótárat, amely egy bolt termékeit és áraikat tartalmazza. Írd ki a szótár hosszát. Töröld a szótár teljes tartalmát a clear() metódussal. Vizsgáld meg a szótár hosszát a törlés után.
2. Készíts egy szótárat autómodellekkel és áraikkal. Kérd be a felhasználótól egy autómodell nevét, és a pop() metódus segítségével kérdezd le és töröld a megadott autót a szótárból, kiírva az árát is. Ha az autó nem található, írj ki egy üzenetet. Ha elfogytak az autók a listából, akkor is adj üzenetet.
3. Készíts egy szótárat, amely emberek nevét és kedvenc színét tartalmazza. Kérj be egy nevet, majd a get() metódus segítségével jelenítsd meg a kedvenc színét. Ha a név nem szerepel a szótárban, az írd ki, hogy "Ismeretlen név".
4. A 2. feladatot módosítsd úgy, hogy az eredeti szótár megmaradjon. Készíts másolatot róla és azzal dolgozzon tovább a program.

6. óra: Listák, szótárak használata ciklusban

Elmélet

- Listák használata ciklusban
- Szótárak használata ciklusban

Kód példa

```
gyumolcsok = ["alma", "banán", "cseresznye"]

# Lista elemeinek kiírása a korábban tanult módszerrel
for i in range(len(gyumolcsok)):
    print(gyumolcsok[i])

# Lista elemeinek kiírása EGYSZERŰBB módszerrel
for gyumolcs in gyumolcsok:
    print(gyumolcs)


szotar = {"alma": "piros", "banán": "sárga", "szilva": "kék"}

# Szótar kulcsainak kiírása
for kulcs in szotar:
    print(kulcs)

# Szótar értékeinek kiírása
for ertek in szotar.values():
    print(ertek)

# Szótar kulcs/érték párjainak kiírása
for kulcs, ertek in szotar.items():
    print(f"{kulcs} = {ertek}")
```

Gyakorlati feladatok

1. Hozz létre egy listát különböző nevekkel. Írj egy ciklust, amely csak azokat a neveket jeleníti meg, amelyek legalább 5 karakter hosszúak. Írd ki ezeket a neveket.
2. Hozz létre egy szótárat, amely egy hét napjainak hőmérsékleteit tartalmazza. Írj egy ciklust, amely kiszámítja az átlaghőmérsékletet, majd kiírja azokat a napokat, amikor a hőmérséklet magasabb volt az átlagnál.
3. Készíts egy szótárat, amely különböző ételeket és azok kalóriatartalmát tartalmazza. Írj egy ciklust, amely minden 200 és 500 kalória közötti ételt megjelenít.
4. Készíts egy szótárat, amely termékek nevét és eredeti áraikat tartalmazza. Írj egy ciklust, amely minden termék árát csökkenti 20%-kal, majd egy új szótárban tárolja a kedvezményes árakat. Írd ki az eredeti és a kedvezményes árakat is.

7. óra: Halmazok

Elmélet

- Halmazok elemei egyediek és nem rendezettek.
- Egyedi elemek hozzáadása és törlése.

- Fontosabb halmazműveletek:
  - `union()` : egyesítés
  - `difference()` : kiveszi azokat az elemeket amelyek a többi halmazban szerepelnek
  - `intersection()` : csak azok az elemek maradnak, amelyek többi halmazban szerepelnek

Kód példa

```
gyumolcsok = {"alma", "banán", "cseresznye"}

# Elem hozzáadása
gyumolcsok.add("narancs")
print(gyumolcsok)

# Elem újbóli hozzáadása (nem lehetséges)
gyumolcsok.add("narancs")
print(gyumolcsok)

# Elem törlése
gyumolcsok.remove("banán")
print(gyumolcsok)

# Lista átalakítása halmazzá
# Vegyük észre, hogy a halmazba már nem kerülnek át a duplikátumok!
szamok_lista = [1, 2, 2, 3]
halmaz = set(szamok_lista)
print(halmaz)
```

```python
halmaz1 = {"alma", "banán", "cseresznye"}
halmaz2 = {"alma", "körte", "barack"}

# Egyesítés
halmaz3 = halmaz1.union(halmaz2)
print(halmaz3)  # {'cseresznye', 'banán', 'körte', 'barack', 'alma'}

# A fenti megoldható "|" operátorral is
halmaz3 = halmaz1 | halmaz2
print(halmaz3)

# Különbség: kiveszi azokat az elemeket amelyek a második halmazban szerepelnek
halmaz3 = halmaz1.difference(halmaz2)
print(halmaz3)  # {'banán', 'cseresznye'}

# A fenti megoldható "-" operátorral is
halmaz3 = halmaz1 - halmaz2
print(halmaz3)

# Közös elemek: csak azok az elemek maradnak, amelyek mindkét halmazban szerepelnek
halmaz3 = halmaz1.intersection(halmaz2)
print(halmaz3)  # {'alma'}

# A fenti megoldható "&" operátorral is
halmaz3 = halmaz1 & halmaz2
print(halmaz3)
```

Gyakorlati feladatok

1. Hozz létre egy halmazt kedvenc sportjaiddal
2. Adj hozzá egy új sportot.
3. Próbáld meg hozzáadni még egyszer. Mit tapasztalsz?
4. Hozz létre egy másik halmazt újabb sportokkal, majd egyesítsd őket egy új halmazba.
5. Kérj be a felhasználótól 5 számot. Halmaz segítségével szűrd ki belőle a duplikátumokat.
6. Készíts két halmazt tetszőleges számokkal. Írd ki halmazművelet segítségével azokat az elemeket, amelyek mindkét halmazban szerepelnek.
7. Készíts két halmazt tetszőleges számokkal. Az elsőből halmazművelettel vedd ki azokat az elemeket amelyek a második halmazban már szerepelnek.

8. óra: Összefoglalás, ismétlés

Python adatszerkezetek összehasonlító táblázata

Tulajdonság	Lista (List)	Szótár (Dictionary)	Tuple	Halmaz (Set)
Tipikus használat	Szekvenciális adatok, rendezett gyűjtemények	Kulcs-érték párok, leképezések	Nem változó adatok	Halmazműveletek
Üres létrehozás	`[]` vagy `list()`	`{}` vagy `dict()`	`()` vagy `tuple()`	`set()`
Szintaxis	`[1, 2, 3]`	`{'kulcs': 'érték'}`	`(1, 2, 3)`	`{1, 2, 3}`
Módosíthatóság	✓	✓	×	✓
Rendezettség	✓	✓	✓	×
Indexelés	✓	✓ (kulccsal)	✓	×
Duplikátumok	✓	A kulcsok egyediek	✓	×
Memóriahasználat	••	•••	•	••
Elemek elérése	Index alapján: `data[0]`	Kulcs alapján: `data["kulcs1"]`	Index alapján: `data[0]`	□ (csak iterálható)
Hozzáadás	`data.append(4)`	`data["kulcs3"] = "érték3"`	×	`data.add(4)`
Törlés	`data.remove(2)` vagy `del data[0]`	`del data["kulcs1"]` vagy `data.pop("kulcs1")`	×	`data.remove(2)` vagy `data.discard(2)`
Iterálás	Elemenként: `for elem in data`	Kulcsokon vagy kulcs-érték párokon: `for k, v in data.items()`	Elemenként: `for elem in data`	Elemenként: `for elem in data`
Előnyök	• Egyszerű indexelés • Dinamikusan módosítható • Sorrend megőrzése	• Gyors keresés • Rugalmas szerkezet • Könnyen bővíthető	• Alacsony memóriaigény • Gyors létrehozás	• Gyors elemkeresés • Garantált egyediség • Hatékony halmazműveletek
Hátrányok	• Lassú keresés nagy méretben • Memóriaigényes módosítások	• Nagy memóriaigény • Csak egyedi kulcsok • Komplex szerkezet	• Nem módosítható • Korlátozott műveletek	• Rendezetlen • Nem indexelhető • Nem támogat lista műveleteket

Kiegészítés

- **Hash-elhetőség**: Hash-elhető típusok például a `str`, `int`, `float`, és a tuple hash-elhető elemekkel.
- **Sebesség**: A szótárak és halmazok általában gyorsabbak keresésben, mivel belsőleg hash-táblákra épülnek. A listák és tuple-ök lineáris keresésen alapulnak.