Dictionary

2025

1 Dictionary adatszerkezet

A `dict` (szótár) adatszerkezet kulcsokhoz rendelt értékek gyűjteménye (kulcs–érték adatszerkezet), ahol a kulcsoknak egyedieknek kell lenniük. A kulcsok általában string típusúak, de lehetnek más immutable tipusúak is (például számok, de lista nem). A kulcsok és az értékek is módosíthatók.

1.0.1 Szótár létrehozása

Az elemeket kapcsos `{}` zárójelek között kell megadni úgy, hogy a kulcs és az érték közé kettőspont `:` kerül.

```
szotar = {
 kulcs1 : ertek,
 kulcs2 : ertek,
 kulcs3 : ertek
}
```

> **Fontos**
> A `dict` szerkezet a Python 3.7 verziótól tartja a sorrendet, a 3.6 vagy korábbi verzióiban még rendezettlen volt.

[ ]:
```python
hallgato = {"nev": "Kiss Tamás", "neptun": "KDLRDR", "pontszam": 60}

print(f"Hallgató adatai: {hallgato}")
print(type(hallgato))
```

A szótár létrehozható a `dict()` konstruktorral is, azonban ebben az esetben a kulcsok és értékek közé az `=` jelet kell kitenni. Ilyenkor minden kulcs szöveg lesz.

[ ]:
```python
hallgato = dict(nev="Kiss Tamás", neptun="KDLRDR", pontszam=60)
print(f"Hallgató adatai: {hallgato}")
```

Egy szótár létrehozható úgy is, hogy a `fromkeys()` függvénynek megadunk egy tuple-t vagy listát, amely tartalmazza a kulcsokat. Ha nem adunk meg alapértelmezett értéket, akkor a `None` értéket kapják.

[ ]:
```python
kulcsok = ("nev", "neptun", "pontszam")

hallgato = dict.fromkeys(kulcsok)
print(f"Hallgató adatai: {hallgato}")
```

```
hallgato = dict.fromkeys(kulcsok, 0)
print(f"Hallgató adatai: {hallgato}")
```

1.0.2 Elemek elérése és módosítása

Az elemeket szögletes zárójelek `[]` között a kulcs alapján tudjuk elérni, ha a megadott kulcs nem létezik, akkor hibát fog adni.
A kulcshoz tartozó értékek lekérdezhetőek a `get()` függvény segítségével is, ha nem létezik a kulcs, akkor alapértelmezetten egy `None` értékkel tér vissza, de a `get()` függvény második argumentumaként megadhatjuk, hogy mi legyen az alapértelmezett érték, ha nincs olyan kulcs.

A `[]` operátorral kijelölt kulcsú elem módosítható. Ha a megadott kulcs nem létezik, akkor létrehozza a kulcsot a megadott értékkel. Adott kulcsú elem az `update()` függvény segítségével létrehozható és módosítható, attól függően, hogy létezik-e már a megadott kulcs.

Az `in` kulcsszó segítségével meg tudjuk vizsgálni, hogy a megadott kulcs szerepel-e a szótárban.

[ ]:
```
hallgato = {
  "nev": "Kiss Tamás",
  "neptun": "KDLRDR",
  "pontszam": 60
}

# érték lekérdezése
print(f"Hallgató neve: {hallgato["nev"]}")   # "nev" kulcshoz tartozó érték
print(f"Hallgató neptunkódja: {hallgato.get("neptun")}")   # "neptun" kulcshoz␣
 ↪tartozó érték
print(f"Hiányzó kulcs: {hallgato.get("jegy")}") # "jegy" kulcshoz tartozó érték
# ha nincs "jegy" kulcs, akkor legyen "Nem teljesítette" a visszaadott érték
print(f"Hiányzó kulcs alapértelmezett értékkel: {hallgato.get("jegy", "Nem␣
 ↪teljesítette")}")

# meglévő kulcs értékének módosítása
hallgato["neptun"] = "BATMAN"     # neptun módosítása "BATMAN"-ra
hallgato.update({"pontszam": 85})   # pontszam módosítása

# új kulcs létrehozása
hallgato["kar"] = "Gépészmérnök"    # "kar" kulcs hozzáadás
hallgato.update({"magassag": 185})  # "magassag" kulcs létrehozása

print(f"Hallgató adatai: {hallgato}")

# ellenőrzés, hogy az adott kulcs szerepel-e a szótárban
if "neptun" in hallgato:
    print("A 'neptun' kulcs szerepel a szótárban.")
```

1.0.3 Kulcsok és értékek nézetei

A szótár nézetek (views) olyan speciális objektumok, amelyek nem listák, hanem élő nézetek a szótár tartalmára. Ezek közvetlenül a szótár adataira mutatnak. Nem másolatot adnak, hanem egy ablakot a szótárra. Ha a szótár tartalma módosul, akkor a nézet automatikusan frissül.

A szótárban található kulcsok nézetét a `keys()` függvénnyel lehet lekérdezni.
A szótárban található értékek nézetét a `values()` függvénnyel lehet lekérdezni.
A szótárban található kulcs-érték párok nézetét az `items()` függvénnyel lehet lekérdezni. A kulcs-érték párok egy (kulcs, érték) tuple-be kerülnek.

[ ]:
```python
hallgato = {"nev": "Kiss Tamás", "neptun": "KDLRDR", "pontszam": 60}

# A szótár kulcsainak nézete
kulcsok_nezete = hallgato.keys()
print(f"A szótár kulcsainak nézete: {kulcsok_nezete}")

# A szótár értékeinek nézete
ertekek_nezete = hallgato.values()
print(f"A szótár értékeinek nézete: {ertekek_nezete}")

# A szótár elemeinek nézete
elemek_nezete = hallgato.items()
print(f"A szótár elemeinek nézete: {elemek_nezete}")

# változás követése
hallgato["pontszam"] = 85
print(f"A szótár értékeinek nézete: {ertekek_nezete}")
```

1.0.4 Elemek törlése

A `pop()` függvénnyel vagy a `del` kulcsszóval lehet eltávolítani egy adott kulcsú elemet a szótárból. Ha az adott kulcs nem található meg a szótárban, akkor hibát ad.
Ha az összes kulcsot törölni szeretnénk, akkor a `clear()` függvényt használjuk, ilyenkor egy üres szótárt fogunk kapni.

[ ]:
```python
hallgato = {"nev": "Kiss Tamás", "neptun": "KDLRDR", "pontszam": 60}

# "nev" kulcs törlése
hallgato.pop("nev")
# "neptun" kulcs törlése
del hallgato["neptun"]

print(f"Hallgató adatai: {hallgato}")
```

1.0.5 Bejárás

A `for` ciklusnak átadott szótár a kulcsokon halad végig. Bejárásra használhatók a nézetek (`keys()`, `values()`, `items()`) is. Az elemek nézete esetén a visszakapott tuple szétbontható kulcs és érték

változókra.

[ ]:
```python
hallgato = {"nev": "Kiss Tamás", "neptun": "KDLRDR", "pontszam": 60}

# Kulcsok bejárása
for kulcs in hallgato:  # hallgato.keys() is ugyanúgy a kulcsokat adja
    print(f"Kulcs: {kulcs}, Érték: {hallgato[kulcs]}")

# Értékek bejárása
for ertek in hallgato.values():
    print(f"Érték: {ertek}")

# Elemek bejárása
for kulcs, ertek in hallgato.items():
    print(f"Kulcs: {kulcs}, Érték: {ertek}")
```

1.0.6 Szótárkifejezés (dictionary comprehension)

Akárcsak a lista esetén a szótáraknál is van lehetőség az elemeinek átalakítására és szűrésére egy egyszerűsített kifejezés használatával, amivel kiváltható egy for ciklus.

`uj_szótár = {kifejezés(elem) for (kulcs, érték) in szótár.items()}`

[ ]:
```python
arak = {"AAPL": 121, "AMZN": 3380, "MSFT": 219, "BIIB": 280, "QDEL": 266,␣
 ↪"LVGO": 144}

# 2%-os áremelkedés
uj_arak = {}
for azonosito, ar in arak.items():
    uj_arak[azonosito] = ar * 1.02

print(uj_arak)

# 2%-os áremelkedés
uj_arak = {azonosito: ar * 1.02 for (azonosito, ar) in arak.items()}
print(uj_arak)
```

szűrés alkalmazása:
`uj_szótár = {kifejezés(elem) for (kulcs, érték) in szótár.items() if feltétel(érték)}`

[ ]:
```python
arak = {"AAPL": 121, "AMZN": 3380, "MSFT": 219, "BIIB": 280, "QDEL": 266,␣
 ↪"LVGO": 144}

# 200-nál nagyobb értékű termékek
szurt_arak = {}
for azonosito, ar in arak.items():
    if ar > 200:
        szurt_arak[azonosito] = ar
```

```
print(szurt_arak)

# 200-nál nagyobb értékű termékek
szurt_arak = {azonosito: ar for (azonosito, ar) in arak.items() if ar > 200}
print(szurt_arak)
```

1.0.7 Szótár másolása

Két szótár értékadása során pl.: `dic1 = dic2` csak a referencia másolódik át, így gyakorlatilag mindkettő szótár ugyanazt a területet használja.

A `copy()` függvény segítségével tudunk valódi másolatot készíteni, ilyenkor az össze kulcs-érték pár átmásolásra kerül. A `dict()` konstruktor hívásával is új másolat jön létre.

[ ]:
```
hallgato1 = {"nev": "Kiss Tamás", "neptun": "KDLRDR", "pontszam": 60}

hallgato2 = hallgato1  # ugyanaz a szótár 2 névvel
hallgato1["nev"] = "Molnár Gábor"
print(hallgato2)

hallgato3 = hallgato1.copy()  # ez már másolat
hallgato3["nev"] = "Nagy Máté"
print(hallgato3)

hallgato4 = dict(hallgato1)  # ez is lemásolja
hallgato3["nev"] = "Szabó Áron"
print(hallgato3)
```

1.0.8 Szótárak egymásba ágyazása

Egy szótár tartalmazhat értékként egy másik szótárt vagy akár listát is. (ez fordítva is igaz)

[ ]:
```
tankor = {
    "hallgato1": {"nev": "Kiss Tamás", "kor": 21},
    "hallgato2": {"nev": "Nagy Tamás", "kor": 20},
    "hallgato3": {"nev": "Molnar Tamás", "kor": 21},
}

# "hallgato2" adatai
print(tankor["hallgato2"])
# "hallgato2" neve
print(tankor["hallgato2"]["nev"])
```

1.1 CSV fájl használata szótár adatszerkezettel

A `csv.DictReader()` osztály egy olyan olvasót hoz létre, amelyben az adatok egy `dict` szerkezetben helyezkednek el. A fejlécben szereplő nevek lesznek a szótár kulcsai. A felismert kulcsokat a `fieldnames` változóban lehet megtekinteni.

[ ]:
```python
import csv

with open("./file/zh.txt", "r", encoding="utf-8") as csv_file:
    csv_reader = csv.DictReader(csv_file, delimiter="\t")

    print(", ".join(csv_reader.fieldnames))  # kulcsok a fejlécből

    sorszam = 0
    for sor in csv_reader:      # sorok szótárként értelmezve
        print(f"{sor["Neptunkód"]}, {sor["Pontszám"]}")
        sorszam += 1

    print(f"Összesen {sorszam} adat")
```

A `csv.DictWriter` osztály egy olyan írót hoz létre, amelynek meg kell adni a mezők kulcsait. Ennek ismeretében már kiírható a fejléc a `writeheader()` függvénnyel. A sorok a `writerow()` vagy `writerows()` függvények segítségével írhatók ki egy szótár kulcsai alapján.

[ ]:
```python
import csv

adatok = [
    {"Pontszám": 100, "Neptunkód": "BATMAN"},
    {"Pontszám": 48, "Neptunkód": "VERFES"},
    {"Pontszám": 68, "Neptunkód": "SECDED"},
    {"Pontszám": 39, "Neptunkód": "SCFRGR"},
    {"Pontszám": 89, "Neptunkód": "SEFJHI"},
    {"Pontszám": 76, "Neptunkód": "NKO8JK"},
]

with open("./file/zh.csv", "w", newline="") as csvfile:
    mezok = ["Neptunkód", "Pontszám"]
    writer = csv.DictWriter(csvfile, fieldnames=mezok) # kulcsok megadása

    writer.writeheader()        # fejléc kiírása
    writer.writerow({"Pontszám": 86, "Neptunkód": "DF56FG"})   # 1 sor kiírása
    writer.writerows(adatok)    # több sor kiírása
```

1.2 JSON adatszerkezet használata

A JSON (JavaScript Object Notation) egy olyan adatszerkezet, amely könnyen olvasható, és gyakran használják adatok cseréjére webes alkalmazások és szerverek között. Pythonban a `json` modul segítségével könnyen kezelhető.

1.2.1 Átalakítás json szerkezetre

A `json.dumps()` függvény képes átalakítani az összes Pythonban található egyszerű, vagy összetettebb adatszerkezetet json formátumú szöveggé.
Az `intent` tulajdonsággal be lehet állítani, hogy az ember számára olvasható megjelentés tagolásához mennyi üres karaktert használjon. Ha ezt elhagyjuk, akkor a lehető legkompaktabb

formátumot kapjuk.
Az `ensure_ascii` paraméter alapételmezetten igaz, ami azt jelenti, hogy az összes nem ASCII karaktert lecseréli escape karakterekre. Ha magyar ékezetes karaktereket szeretnénk használni, akkor ezt ki kell kapcsolni.

[ ]:
```python
import json

tankor = [
    {"nev": "Kiss Tamás", "kor": 21},
    {"nev": "Nagy Tamás", "kor": 20},
    {"nev": "Molnar Tamás", "kor": 21},
]

json_adatok = json.dumps(tankor, indent=4, ensure_ascii=False)

print(f"JSON adatok: {json_adatok}")
```

A `json.dump()` függvény segítségével közvetlenül egy fájlba is küldhetjük az átalakítás eredményét.

[ ]:
```python
import json

tankor = [
    {"nev": "Kiss Tamás", "kor": 21},
    {"nev": "Nagy Tamás", "kor": 20},
    {"nev": "Molnar Tamás", "kor": 21},
]

with open("./file/data.json", "w", encoding="utf-8") as json_file:
    json.dump(tankor, json_file, indent=2, ensure_ascii=False)
```

1.2.2 JSON betöltése

A `json.loads()` függvény képes egy json formátumú szöveget átalakítani egy python formátumú adatszerkezetté.

[ ]:
```python
import json

json_adat = '{"nev": "Nagy Tamás", "kor": 30, "varos": "Kecskemét", "hallgato":␣
 ↪true, "jegyek": [90, 85, 92], "hobbi": null}'

adatok = json.loads(json_adat)
print(adatok)
```

A `json.load()` függvény segítségével közvetlenül egy json fájl tartalmát konvertálhatjuk át python adatszerkezetre.

[ ]:
```python
import json

with open("./file/data.json", "r", encoding="utf-8") as json_file:
```

```
    adatok = json.load(json_file)

print(f"Beolvasott JSON adatok: {adatok}")
```

2 Gyakorló feladatok

1. Feladat

   Számold meg egy szövegben lévő szavak gyakoriságát, és hozz létre egy szótárt a szavak és azok gyakoriságaival.

[ ]:
```
szoveg = "Ez egy példa szöveg. Ebben a szövegben több példa szó is van."
szavak = szoveg.split()

gyakorisag = {}
for szo in szavak:
    szo = szo.strip(".,!?")
    gyakorisag[szo.lower()] = gyakorisag.get(szo.lower(), 0) + 1

print(gyakorisag)
for szo, db in gyakorisag.items():
    print(f"{szo}: {db} alkalommal fordul elő")
```

2. Feladat

   Távolítson el egy szótárból minden olyan kulcs-érték párt, ahol az érték kisebb, mint 2000.

[ ]:
```
programnyelvek = {
    "C": 1972,
    "C++": 1980,
    "Ada": 1983,
    "MATLAB": 1984,
    "LabVIEW": 1986,
    "Perl": 1987,
    "Tcl": 1988,
    "Wolfram Language": 1988,
    "Python": 1990,
    "Visual Basic": 1991,
    "Java": 1995,
    "Delphi": 1995,
    "JavaScript": 1995,
    "PHP": 1995,
    "C#": 2001,
    "Go": 2009,
    "Dart": 2011,
    "Kotlin": 2011,
    "Julia": 2012,
```

```
    "TypeScript": 2012,
    "Rust": 2015,
    "Carbon": 2022,
}

uj_nyelvek = {}
for nev, ev in programnyelvek.items():
    if ev >= 2000:
        uj_nyelvek[nev] = ev

print(uj_nyelvek)
```

3. Feladat

   Kombináljon két szótárat úgy, hogy azonos kulcsok esetén az értékeket összeadja.

[ ]:
```
kosar1 = {"alma": 1.5, "korte": 1.2, "banan": 0.5}
kosar2 = {"szilva": 2, "alma": 2.1, "korte": 1.1}

kozos = kosar1.copy()

for k, v in kosar2.items():
    kozos[k] = kozos.get(k, 0) + v

# kozos = {key: kosar1.get(key, 0) + kosar2.get(key, 0) for key in set(kosar1)␣
 ↪| set(kosar2)}

print(kozos)
```

4. Feladat

   Határozza meg egy szótárban szereplő hallgatók jegyeinek az átlagát.

[ ]:
```
eredmenyek = {
    "BATMAN": 4.5,
    "SDFERG": 3.0,
    "4DF5RH": 5.0,
    "LOU7RG6": 4.2,
    "SDRPH9": 3.9,
    "SWR9IP": 4.5,
}

osszeg = sum(eredmenyek.values())
osszes = len(eredmenyek)
print(f"Az eredméyek átlaga: {osszeg / osszes:.2f}")
```

5. Feladat

   Határozza meg a hallgatók tanulmányi átlagát a tantárgyak eredményeit tartalmazó szótár listáiból.

[ ]:
```
eredmenyek = [
    {"nev": "Szabó Tamás", "targy": "Matek", "jegy": 5, "kredit": 5},
    {"nev": "Szabó Tamás", "targy": "Statika", "jegy": 4, "kredit": 4},
    {"nev": "Szabó Tamás", "targy": "Gepelemek", "jegy": 5, "kredit": 6},
    {"nev": "Kiss Attila", "targy": "Matek", "jegy": 4, "kredit": 5},
    {"nev": "Kiss Attila", "targy": "Statika", "jegy": 4, "kredit": 4},
    {"nev": "Kiss Attila", "targy": "Gepelemek", "jegy": 4, "kredit": 6},
    {"nev": "Nagy Anita", "targy": "Matek", "jegy": 5, "kredit": 5},
    {"nev": "Nagy Anita", "targy": "Statika", "jegy": 5, "kredit": 4},
    {"nev": "Nagy Anita", "targy": "Gepelemek", "jegy": 3, "kredit": 6},
    {"nev": "Molnár Kata", "targy": "Matek", "jegy": 3, "kredit": 5},
    {"nev": "Molnár Kata", "targy": "Statika", "jegy": 4, "kredit": 4},
    {"nev": "Molnár Kata", "targy": "Gepelemek", "jegy": 5, "kredit": 6},
    {"nev": "Molnár Kata", "targy": "Merestechnika", "jegy": 5, "kredit": 4},
]

hallgatok = {}

for eredmeny in eredmenyek:
    hallgatok[eredmeny["nev"]] = hallgatok.get(eredmeny["nev"], 0) +␣
 ↪eredmeny["jegy"]

for nev in hallgatok.keys():
    db_targy = 0
    for eredmeny in eredmenyek:
        if eredmeny["nev"] == nev:
            db_targy += 1
    hallgatok[nev] /= db_targy

print(hallgatok)
```

6. Feladat

   Olvassa be az eredmenyek.json fájl tartalmát és írassa ki a hallgatók kredit indexét.

$$TÁ = \frac{\sum\left[\left(\text{teljesített kredit}\right)\times\text{érdemjegy}\right]}{\text{teljesített kredit}}$$

[ ]:
```
import json

with open("./file/eredmenyek.json", "r", encoding="utf-8") as json_file:
    eredmenyek = json.load(json_file)


hallgatok = {}

for eredmeny in eredmenyek:
```

```
    ki = eredmeny["jegy"] * eredmeny["kredit"] / 30
    if eredmeny["jegy"] > 1:
        hallgatok[eredmeny["nev"]] = hallgatok.get(eredmeny["nev"], 0) + ki

print(hallgatok)
```

7. Feladat

   Olvassa be az eredmenyek.json fájl tartalmát és írassa ki a hallgatók súlyozott tanulmányi átlagát.

$$TÁ = \frac{\sum\left[\left(\text{teljesített kredit}\right) \times \text{érdemjegy}\right]}{\text{teljesített kredit}}$$

[ ]:
```
import json

with open("./file/eredmenyek.json", "r", encoding="utf-8") as json_file:
    eredmenyek = json.load(json_file)


hallgatok = {}

for eredmeny in eredmenyek:
    szum = eredmeny["jegy"] * eredmeny["kredit"]
    if eredmeny["jegy"] > 1:
        hallgatok[eredmeny["nev"]] = hallgatok.get(eredmeny["nev"], 0) + szum

for nev in hallgatok.keys():
    osszekredit = 0
    for eredmeny in eredmenyek:
        if eredmeny["nev"] == nev and eredmeny["jegy"] > 1:
            osszekredit += eredmeny["kredit"]
    hallgatok[nev] /= osszekredit


print(hallgatok)
```