Žodynai

Įvadas

Žodynas (angl. *dictionary*) yra nerikiuotas raktas-reikšmė porų rinkinys. Jis suteikia galimybę struktūrizuotai susieti duomenų dalis, kad galėtume greitai rasti susijusias reikšmes.

Įsivaizduokite, kad turime tokią skaičiuoklės lentelę, kuri aprašo krepšinio žaidėją:

Vardas	**Pavardė**	**Komanda**	**Pozicija**	**Ūgis**
Romas	Petraitis	Žalgiris	Gynėjas	193

Pitono programoje, tokią lentelę, galime struktūrizuoti ir aprašyti žodyno pagalba dviem būdais. Pirmasis būdas - aprašomasis, kai lenktinių skliaustelių viduje pateikiam raktų ir jų reikšmių poras.

```
žaidėjas = {
    "vardas": "Romas",
    "pavardė": "Petraitis",
    "komanda": "Žalgiris",
    "pozicija": "Gynėjas",
    "ūgis": 193
}
```

Kaip matote, kairėje pusėje, kabutėse, yra viršutinės lentelės eilutės žodžiai. Šie žodžiai vadinami **raktais**. Dešinėje pusėje, už dvitaškio, matome antrosios eilutės duomenis, kurie vadinami **reikšmėmis**. Iš čia ir kilęs terminas *rakto-reikšmės pora*. Beje, raktai kartais vadinami **laukais**. Ir nors teisingiau būtų vadinti raktais, šiame kurse šiuos terminus naudosime kaip sinonimus.

Kitas būdas - žodyno sukūrimas panaudojant `dict` funkciją.

```
žaidėjas = dict(
    vardas = "Romas",
    pavardė = "Petraitis",
    komanda = "Žalgiris",
    pozicija = "Gynėjas",
    ūgis=193
)
```

Kaip matote, raktai aprašomi vardiniais funkcijų parametrais, o reikšmės šių parametrų reikšmėmis.

Gali kilti klausimas - tai kurį gi būdą naudoti!? Trumpas atsakymas - tai daugiau skonio reikalas :) Yra šiokių tokių niunsų, bet jie mūsų atveju yra neesminiai.

Žodyno raktai ir reikšmės

Raktai kaip eilutės

Įvade pamatėme, kad žodyno struktūros raktu gali būti eilutės. Prisiminkime:

```
žaidėjas = {
    "vardas": "Romas",
    "pavardė": "Petraitis",
    "komanda": "Žalgiris",
    "pozicija": "Gynėjas",
    "ūgis": 193
}
```

Tokiu atveju - reikšmė susieta su raktu yra pasiekiama naudojant laužtinius skliaustelius, kuriuose kabutėse nurodomas raktas.

```
žodyno_kintamasis["raktas"]
```

Duotu atveju, jei norime pasiekti žaidėjo pavardę, turime rašyti taip:

```
1. žaidėjas = {
2.     "vardas": "Romas",
3.     "pavardė": "Petraitis",
4.     "komanda": "Žalgiris",
5.     "pozicija": "Gynėjas",
6.     "ūgis": 193
7. }
8.
9. print(žaidėjas["pavardė"])
```

Paeksperimentuokite su šiuo kodu spausdindami kitų raktų reikšmes.

Raktai kaip skaičiai

Pasirodo, raktais gali būti ne tik eilutės, bet ir skaičiai. Panagrinėkite pavyzdį, kai lygio sunkumo raktai apibrėžiami skaičiumi:

```
1. lygio_sunkumas = {
2.     1: "labai lengvas",
3.     2: "lengvas",
4.     3: "vidutinis",
5.     4: "sunkus",
6.     5: "labai sunkus"
7. }
8. print(lygio_sunkumas[1])
```

Atkreipkite dėmesį, kad tie skaičiai nėra indeksai ir jie gali būti bet kokie. Raktais gali būti netgi skaičiai su kableliu. Taip pat - pastebėkite, kad skaitiniai raktai nėra rašomi kabutėse.

Kiti raktų tipai

Raktais gali būti ne tik eilutės ir skaičiai. Raktais gali būti ir kai kurios kitos struktūros, tokios kaip *bool* struktūra, *tuples* struktūra ir netgi *None*. Toliau pateikiamas pavyzdys su galimais raktų tipais:

```python
raktų_pavyzdys = {
    "raktas": "eilutė",     # raktui naudojama eilutė
    42: "skaičius",         # raktui naudojamas sveikas skaičius
    3.14: "pi reikšmė",     # raktui naudojamas skaičius su kableliu
    (1, 2, 3): "kamuolys",  # raktui naudojama tuple struktūra
    True: "tiesa",          # raktui naudojama bool struktūra
    None: "nieko"           # raktui naudojama None struktūra
}

print(raktų_pavyzdys)
```

Netinkami raktai

Vis tik - ne visos struktūos gali būti raktais. Keičiami (angl. *mutable*) duomenų tipai negali būti raktais. Tai: sąrašai (*list*), rinkiniai (*set*) ir tie patys žodynai (*dictionary*). Pažiūrėkite, kaip reaguoja Pitonas į šias struktūras:

```python
žodynas = {[1, 2, 3]: "klaida"}  # Sukels klaidą, nes raktui naudojamas sąrašas
# žodynas = {{1, 2, 3}: "klaida"}  # Sukels klaidą, nes raktui naudojamas rinkinys
# žodynas = {{1: "a", 2: "b"}: "klaida"}  # Sukels klaidą, nes raktui naudojamas žodynas
```

Atkomentuokite kitas eilutes, norėdami išbandyti kitus variantus.

Žodyno reikšmė - bet kuri struktūra

Skirtingai nuo raktų, žodyno reikšme gali būti bet kuri struktūra, net ir pats žodynas. Tai yra žodynas gali įtraukti ir kitus žodynus. Panagrinėkite moksleivio struktūros pavyzdį.

```python
moksleivis = {
    "vardas"     : "Jonas",                          # Eilutė (str)
    "pavardė"    : "Petrauskas",                     # Eilutė (str)
    "amzius"     : 14,                               # Sveikasis skaičius (int)
    "vidurkis"   : 8.5,                              # Skaičius su kableliu (float)
    "ar_aktyvus" : True,                             # Loginė reikšmė (bool)
    "hobiai"     : ["krepšinis", "programavimas"],   # Sąrašas (list)
    "pažymiai"   : {                                 # Kitas žodynas (dict)
        "matematika" : 9,
        "lietuvių"   : 8,
        "istorija"   : 7
    },
    "kontaktai": ("jonas@gmail.com", "+3706012321"), # Tuple
    "papildoma_info": None                            # None reikšmė
}

# Spausdiname visą žodyną
print("Moksleivio duomenys:")
print(moksleivis)
```

Kai kurie tipai, kaip aibė, *tuple* jums dar nežinomi. Netrukus su jais susipažinsime. Kol kas tiesiog žinokite, kad juos galima panaudoti žodyne :)

Žodyno reikšmių priskyrimas ir keitimas

Pradinis žodynas nebūtinai turi turėti reikšmės (lig šiol taip darėme). Pradinis žodynas gali būti tuščias:

```
žaidimai = {}
```

Žodynui reikšmes galime priskirti (arba jas pakeisti) naudodami priskytimo operatorių (=). Priskirkime žaidimus jų kategorijoms:

```
mano_žaidimai = {}
mano_žaidimai["šaudyklės"] = ["Call of Duty", "Doom"]
mano_žaidimai["strateginiai"] = ["Starcraft", "Dota", "Warcraft"]
mano_žaidimai["nuotykių"] = ["Indiana Jones", "Monkey Island"]

print(mano_žaidimai)
```

Pavyzdyje mes pildėme žodyną pridėdami po vieną raktą. Galime tai daryti ir efektyviau. Yra galimybė pridėti kelis raktus iš karto. Papildykime mūsų sąrašą papildoma (veiksmo žaidimų) kategorija ir pakeiskime strateginių žaidimų sąrašą:

```
mano_žaidimai = {}
mano_žaidimai["šaudyklės"] = ["Call of Duty", "Doom"]
mano_žaidimai["strateginiai"] = ["Starcraft", "Dota", "Warcraft"]
mano_žaidimai["nuotykių"] = ["Indiana Jones", "Monkey Island"]

# pridedam naują kategoriją
mano_žaidimai.update({"veiksmo": ["Sonic", "Mario"]})
# pakeičiam strateginių žaidimų kategoriją
mano_žaidimai.update({"strateginiai": ["Starcraft", "Starcraft 2"]})

print(mano_žaidimai)
```

Žodyno sudarymas iš sąrašų

Pabaigai noriu parodyti dar vieną įdomią ir naudingą techniką, kuri leidžia sukombinuoti atskirus sąrašus į žodyno struktūrą. Sakykime, turime dalykų ir jų įvertinimų sąrašus:

```
dalykai = ["matematika", "istorija", "programavimas"]
pažymiai = [9, 5, 10]
```

Tuomet šiuos du sąrašus galime paversti žodynu tokiu būdu:

dalykai = ["matematika", "istorija", "programavimas"]

```
pažymiai = [9, 5, 10]
mano_įvertinimai = {raktas:reikšmė for raktas, reikšmė in zip(dalykai, pažymiai)}
print(mano_įvertinimai)
```

Sąrašams sujungti čia naudojama **zip** funkcija. Sujungtas sąrašas yra iteruojamas (perrenkamos visos reikšmės). Žodynas konstruojamas raktui paimant reikšmę iš dalykų, o rakto reikšmė paimama iš pažymių.

Žodyno raktų ir reikšmių iteravimas

Mes žinome kaip pasiekti ir atspausdinti žodyno lauko reikšmę. Vis tik, žodynas nėra tiksliai apibrėžta struktūra (kaip klasė), todėl jo laukai (arba raktai) gali kisti. Ir mes ne visada žinosime, kokie jie yra. Todėl reikia mokėti pasitikrinti ir perrinkti visus žodyno raktus.

Saugus rakto reikšmės gavimas

Sakykime, turime klasės mokinių ir jų vidurkių žodyną. Kadangi mokiniai gali keistis (pvz. pereiti į kitą mokyklą), tai mes nesame užtikrinti, kad egzistuos mums žinomi raktai. Norėdami užsitikrinti, kad mokinys egzistuoja, naudosime žodyno metodą **get**:

```
# gražina reikšmę arba None
žodynas.get("raktas")
arba
# gražina reikšmę arba numatytą reikšmę
žodynas.get("raktas", "numatyta_reikšmė")
```

Šiame pavyzdyje panaudosime antrąjį variantą, kadangi paprasčiau :)

```
moksleiviai = {
    "Tomas"   : 8,
      "Rima"    : 7,
      "Petras"  : 9
}

print(moksleiviai["Tomas"])
# print(moksleiviai["Gintarė"])
print(moksleiviai.get("Gintarė", "Tokio mokinio(ės) nėra"))
```

Pirmą kartą kai paleisit - gausite `Key error: 'Gintarė'`, nes Gintarės žodyne nėra ir programa nustos veikti. Tuomet, užkomentuokite eilutę, kuri išduoda klaidą ir pažiūrėkite kaip veikia saugus rakto reikšmės gavimas.

Žodyno raktų iteravimas

Sakykime, mes norime atspausdinti moksleivių vardų sąrašą. Kitaip sakant - moksleivių žodyno raktų sąrašą, nes vardai apibrėžia raktus. Tai galime atlikti dviem būdais - iteruoti žodyną konvertuotą į sąrašą arba iteruoti žodyno raktus, kuriuos galima gauti panaudojus **keys()** metodą.

```
moksleiviai = {
    "Tomas"   : 8,
      "Rima"    : 7,
      "Petras"  : 9
}

# 1 atvejis - iteruojam žodyną konvertuotą į sąrašą
for moksleivis in list(moksleiviai):
    print(moksleivis)

# 2 atvejis - iteruojam žodyno raktus
for moksleivis in moksleiviai.keys():
    print(moksleivis)
```

Žodyno reikšmių iteravimas

Taip kaip mes galime iteruoti raktus, taip galime iteruoti ir reikšmes. Tiesiog vietoj metodo **keys** naudosime metodą **values**.

```
moksleiviai = {
    "Tomas"   : 8,
      "Rima"    : 7,
      "Petras"  : 9
}

for moksleivis in moksleiviai.values():
    print(moksleivis)
```

Žodyno raktų ir reikšmių iteravimas

Paimsime anksčiau nagrinėtą pavyzdį, kuriame turėjome milžinišką moksleivio struktūrą. Atspausdinus ją vienoje eilutėje tikrai turėjo būti sunku peržiūrėti raktus ir reikšmes. Patobulinsime spausdinimą perrinkdami visus moksleivio raktus (laukus).

```
moksleivis = {
    "vardas"      : "Jonas",
     "pavardė"     : "Petraitis",
    "amzius"      : 14,
    "vidurkis"    : 8.5,
    "ar_aktyvus"  : True,
    "hobiai"      : ["krepšinis", "skaitymas", "programavimas"],
    "pažymiai"    : {
        "matematika" : 9,
        "lietuvių"   : 8,
        "istorija"   : 7
    },
    "kontaktai"   : ("jonas@example.com", "+3706032123"),
    "papildoma_info": None
}

# Spausdiname visą žodyną
print("Moksleivio duomenys:")
for raktas, reikšmė in moksleivis.items():
    print(f"{raktas}: {reikšmė}")
```

Žodyno metodas **items** gražina raktų-reikšmių porų sąrašą. Kiekviena sąrašo reikšmė yra perrenkama cikle ir atspausdinama.

Žodyno raktų šalinimas

Visų raktų pašalinimas

Norint pašalinti visus žodyno laukus, tereikia panaudoti metodą **clear**.

Vieno rakto pašalinimas

Norint pašalinti vieną raktą, reikia naudoti metodus **del** arba **pop**. **del** metodas pašalina žinomą raktą. Jei rakto nėra, išduodama klaida `KeyError`.

Tuo tarpu **pop** metodas pašalina raktą ir gražina jo reikšmę. Jei rakto nėra, išduodama klaida `KeyError`. Tiesa, to galima išvengti nurodant numatytąją reikšmė.

Abejais atvejais reikia žinoti rakto reikšmę. Yra vienas metodas, **popitem()**, kuriam reikšmės žinoti nereikia. Jis pašalina paskutinį žodyno raktą ir gražina jo rakto ir reikšmės porą.

Panagrinėkite šiuos niuansus duotame pavyzdyje:

```
moksleiviai = {
    "Tomas"   : 5,
      "Rima"    : 6,
      "Jonas"    : 7,
      "Marytė"    : 8,
      "Petras"  : 9
}

print("del naudojimas:")
del(moksleiviai["Tomas"])
print(moksleiviai)

print("\npop naudojimas:")
print(moksleiviai.pop("Jonas"))
print(moksleiviai)

print("\npopitem naudojimas:")
print(moksleiviai.popitem())
print(moksleiviai)

print("\nclear naudojimas:")
moksleiviai.clear()
print(moksleiviai)
```

Žodynų sąrašai

Žodynų sąrašų naudojimas niekuo nesiskiria nuo kitų sąrašų tipų naudojimo. Prisiminkime sąrašus ir panagrinėkime įvairius atvejus, susijusius su tuo kaip naudojami žodynai sąrašuose.

Sąrašo deklaravimas ir indeksų naudojimas:

```
prekės = [
    {
        'pavadinimas': 'pieštukas',
        'kaina': 3.99
    },
    {
        'pavadinimas': 'knyga',
        'kaina': 14.49
    },
    {
        'pavadinimas': 'pienas',
        'kaina': 1.29
    }
]

print('Visos prekės:', prekės)
print('Antra prekė:', prekės[1])
print('Antros prekės kaina:', prekės[1]['kaina'])
```

Žodynų sąrašo iteravimas ir spausdinimas:

```
knygos = [
    { 'pavadinimas': 'Aplink saulę', 'kaina': 14.00, 'autorius': 'Jonas Jonaitis' },
    { 'pavadinimas': 'Miškininkas', 'kaina': 29.99, 'autorius': 'Petras Petrauskas' },
    { 'pavadinimas': 'Mistika abc', 'kaina': 10.90, 'autorius': 'Gintarė Gintarytė' },
    { 'pavadinimas': 'Smegenotyra', 'kaina': 25.49, 'autorius': 'Inga Ingutė' },
    { 'pavadinimas': 'Gyvulininkystė', 'kaina': 12.30, 'autorius': 'Ginčius Ginčiukas' },
]

print("Paprastas, neformatuotas spausdinimas")
for knyga in knygos:
    print(knyga)

print("\nFormatuotas spausdinimas")
for knyga in knygos:
    print(f"- {knyga['pavadinimas']} ({knyga['autorius']}) kaina {knyga['kaina']} €")
```

Žodyno su sudėtingesne struktūra apdorojimas:

```
studentai = [
    dict( vardas = 'Jonas', pazymiai = [8, 7, 8, 9, 6, 8] ),
    dict( vardas = 'Tomas', pazymiai = [10, 10, 9, 10, 9, 10] ),
    dict( vardas = 'Ugnė', pazymiai = [9, 10, 9, 8] )
]

for studentas in studentai:
    suma = sum(studentas['pazymiai'])
    studentas['vidurkis'] = round(suma / len(studentas['pazymiai']))

print(studentai)
print()

didziausio_vidurkio_studentas = studentai[0]

for studentas in studentai:
    if studentas['vidurkis'] > didziausio_vidurkio_studentas['vidurkis']:
        didziausio_vidurkio_studentas = studentas

print('Studentas(-ė) su didžiausiu vidurkiu:')
print(didziausio_vidurkio_studentas)

```

Ši programa suskaičiuoja kiekvieno studento vidurkį ir įrašo jį į naujai sukuriamą lauką *vidurkis*. Taip pat, suranda studentą su didžiausiu vidurkiu ir jį atspausdina.