

Kolekcije

Torke, skupovi i rečnici: tri nova načina da više vrednosti čuvamo zajedno

Danas pravimo: lika u igri kao rečnik i komande sa tastature

```
lik = {  
    "ime": "Cikica",  
    "zivot": 3,  
    "brzina": 2,  
}
```

Tri pitanja za zagrevanje

1. prošla tema: liste

```
a = [1, 2, 3]
b = a
b.append(4)
print(len(a))
```

Ispis: 4

2. stringovi

```
s = "Pikselko"
print(s[:3] + "!")
```

Ispis: Pik!

3. mešano

```
z = 0
for x in [3, 5, 2]:
    if x > 2:
        z += x
print(z)
```

Ispis: 8

Pet novih reči za danas

kolekcija

vrednost koja čuva više vrednosti zajedno, kao lista ili string

torka

uređena kolekcija koja se posle pravljenja ne može menjati

skup

kolekcija u kojoj se svaka vrednost javlja samo jednom, bez redosleda

rečnik

kolekcija parova ključ: vrednost

ključ

ime pod kojim u rečniku tražimo vrednost

Svaka kolekcija ima svoje zagrade

lista**[]**

```
boje = ["crvena", "plava"]
```

Već znamo: može da se menja.

torka**()**

```
tacka = (120, 80)
```

Kao lista, ali se ne menja.

skup**{ }**

```
oznake = {"mač", "štiti"}
```

Bez ponavljanja i bez redosleda.

rečnik**{ ključ: vrednost }**

```
lik = {"ime": "Cikica"}
```

Vrednost tražimo po ključu.

Torke

Kao lista, samo što se ne menja

Torka čuva koordinate i boje u jednoj promenljivoj

```
pozicija = (120, 80)
crvena = (255, 0, 0)
print(pozicija[0])
print(len(crvena))
print(crvena[-1])
x, y = pozicija
print(x + y)
```

ISPIS

```
120
3
0
200
```

- Indeksi počinju od 0, kao kod liste.
- `x, y = pozicija` raspakuje torku u dve promenljive.

Šta ispisuje ovaj program?

```
a = ("zelena")  
b = ("zelena",)  
print(type(a))  
print(type(b))
```

Jedina razlika između a i b je jedan znak.

A tačno

```
<class 'str'>  
<class 'tuple'>
```

B

```
<class 'tuple'>  
<class 'tuple'>
```

C

```
<class 'str'>  
<class 'str'>
```

D

```
<class 'tuple'>  
<class 'str'>
```

Torku ne menjamo, nego od nje pravimo novu

Pokušaj izmene

```
boja = (255, 200, 0)
boja[0] = 100
```

PORUKA O GREŠCI

```
TypeError: 'tuple' object
does not support item assignment
```

Isecanje i spajanje prave novu torku

```
torka = tuple(["crvena", "zelena", "žuta"])
print(torka[0])
nova = torka[1:3] + ("bela",)
print(nova)
print(torka)
```

ISPIS

```
crvena
('zelena', 'žuta', 'bela')
('crvena', 'zelena', 'žuta')
```

Torka u kojoj je lista: šta se dešava?

```
t = (1, [2, 3])  
t[1].append(4)  
print(t)
```

Torka se ne može menjati. A njen element?

Torka se nije promenila

i dalje ima iste dve stvari: broj 1 i istu listu.
Promenila se lista koja je u njoj.

A tačno
(1, [2, 3, 4])

B
TypeError

C
(1, [2, 3])

D
(1, [2, 3], 4)

Skupovi

Svaka vrednost samo jednom

Skup zadržava svaku boju samo jednom

```
pikseli = ["crna", "beļa", "crna",  
           "plava", "beļa", "crna"]  
boje = set(pikseli)  
print(len(pikseli))  
print(len(boje))  
print("plava" in boje)
```

ISPIS

6

3

True

Šest piksela, tri boje.

set() od liste pravi skup: duplikati nestaju. Operator in proverava da li je vrednost u skupu.

Prazan skup pravi `set()`, a `{}` je prazan rečnik

```
drugovi = {}  
print(type(drugovi))  
drugovi = set()  
drugovi.add("Mika")  
drugovi.add("Mika")  
print(drugovi, len(drugovi))
```

ISPIS

```
<class 'dict'>  
{'Mika'} 1
```

- `add` dodaje element u skup.
- Drugi „Mika“ se ne dodaje, jer ga skup već ima.

Unija, presek i razlika, kao na Venovom dijagramu

```
slika_a = {"crvena", "plava", "žuta"}  
slika_b = {"plava", "zelena"}  
print(sorted(slika_a | slika_b))  
print(sorted(slika_a & slika_b))  
print(sorted(slika_a - slika_b))
```

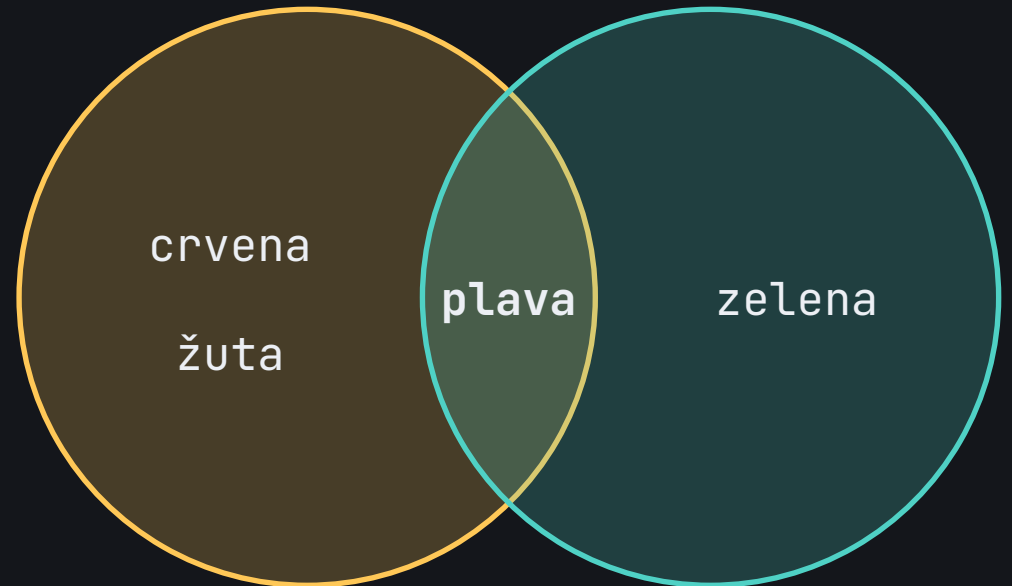
| unija: sve iz oba skupa
& presek: samo zajedničko
- razlika: u slika_a, a nije u slika_b

ISPIS

```
['crvena', 'plava', 'zelena', 'žuta']  
['plava']  
['crvena', 'žuta']
```

slika_a

slika_b



discard ćuti, remove javlja grešku, pop uzima bilo koji

```
boje = {"crvena", "plava"}  
boje.discard("zelena")  
print(len(boje))  
izbacena = boje.pop()  
print(len(boje))  
boje.remove("zelena")
```

ISPIS

```
2  
1  
KeyError: 'zelena'
```

- | | |
|-------------------|-------------------------------------|
| discard(x) | ako x nema, ništa se ne dešava |
| remove(x) | ako x nema, greška KeyError |
| pop() | uklanja i vraća proizvoljan element |

Koliko elemenata ima ovaj skup?

```
s = set("ananas")  
print(len(s))
```

Pomoć: set od stringa pravi skup njegovih slova.

A

6

B

tačno

3

C

2

D

TypeError

Rečnici

Vrednost tražimo po ključu

Rečnik povezuje ključ sa vrednošću

```
lik = {"ime": "Cikica", "zivot": 3, "brzina": 2}  
print(lik["ime"])  
print(lik["zivot"])
```



ISPIS

Cikica
3

Dodela menja postojeći par ili dodaje nov

```
lik = {"ime": "Cikica",  
      "zivot": 3,  
      "brzina": 2}  
print(lik["ime"])  
lik["zivot"] = lik["zivot"] - 1  
lik["boja"] = "plava"  
print(lik)  
print(len(lik))
```

- lik["ime"] čita vrednost.
- Ključ "zivot" postoji: vrednost se menja.
- Ključ "boja" ne postoji: dodaje se nov par.

ISPIS

```
Cikica  
{'ime': 'Cikica', 'zivot': 2, 'brzina': 2, 'boja': 'plava'}  
4
```

Šta ispisuje ovaj program?

```
lik = {"ime": "Cikica", "zivot": 3}  
print(lik["Ime"])
```

Pogledajte pažljivo kako je napisan ključ.

A

Cikica

B

None

C

tačno

KeyError: 'Ime'

D

NameError: Ime

Čudovište: KeyError



Kako izgleda

Program staje, a poslednji red poruke je `KeyError: 'Ime'`. Pod navodnicima je ključ koji nije pronađen.

Zašto se pojavljuje

Tražimo ključ koji rečnik nema: pogrešno slovo, veliko umesto malog slova ili ključ koji još nismo dodali.

Kako se pobeđuje

Ispišemo rečnik ili `lik.keys()` i uporedimo ključeve. Kad ključ možda ne postoji, koristimo `get` ili proveru `in`.

get vraća vrednost ili zamenu, bez greške

```
lik = {"ime": "Cikica", "zivot": 3}
print(lik.get("zivot"))
print(lik.get("brzina"))
print(lik.get("brzina", 1))
print(lik["brzina"])
```

ISPIS

```
3
None
1
KeyError: 'brzina'
```

get(ključ) vraća None kad ključa nema.

get(ključ, zamen) tada vraća zamen. Uglaste zagrade javljaju grešku KeyError.

Petlja for kroz rečnik: ključevi, vrednosti ili parovi

```
lik = {"ime": "Cikica", "zivot": 3, "brzina": 2}
```

for k in lik

daje ključeve

```
for k in lik:  
    print(k, end=" ")
```

```
ime zivot brzina
```

lik.values()

daje vrednosti

```
for v in lik.values():  
    print(v, end=" ")
```

```
Cikica 3 2
```

lik.items()

daje parove

```
for k, v in lik.items():  
    print(k, "=", v)
```

```
ime = Cikica  
zivot = 3  
brzina = 2
```

update dodaje nove parove i menja postojeće

```
lik = {"ime": "Cikica", "brzina": 2}  
nadogradnja = {"brzina": 4, "stit": True}  
lik.update(nadogradnja)  
print(lik)
```

ISPIS

```
{'ime': 'Cikica', 'brzina': 4, 'stit': True}
```

Lik je pokupio nadogradnju.

Brzina 2 postaje 4, jer ključ već postoji. Štit je nov ključ, pa se dodaje na kraj.

Četiri načina da nešto uklonimo iz rečnika

```
lik = {"ime": "Cikica", "zivot": 3,  
       "brzina": 2, "stit": True}  
z = lik.pop("zivot")  
par = lik.popitem()  
del lik["brzina"]  
print(z, par, lik)  
lik.clear()  
print(lik)
```

ISPIS

```
3 ('stit', True) {'ime': 'Cikica'}  
{}
```

pop(ključ)

uklanja par i vraća vrednost; ako ključa nema, greška
KeyError

popitem()

uklanja i vraća poslednji dodat par, kao torku

del lik[ključ]

uklanja par sa tim ključem

clear()

prazni rečnik; rečnik i dalje postoji

Dodela ne pravi kopiju rečnika, a `copy()` pravi

Dodela: isti rečnik, dva imena

```
lik = {"ime": "Cikica"}  
drugi = lik  
drugi["ime"] = "Bubica"  
print(lik["ime"])
```

ISPIS

Bubica

`copy()`: nov rečnik

```
lik = {"ime": "Cikica"}  
drugi = lik.copy()  
drugi["ime"] = "Bubica"  
print(lik["ime"])
```

ISPIS

Cikica

`dict(lik)` takođe pravi nov rečnik.

Isto smo videli kod lista: `b = a` ne pravi kopiju, nego drugo ime za istu listu.

Isti ključ dva puta: šta ostaje?

```
boje = {"nebo": "plava", "trava": "zelena",  
        "nebo": "siva"}  
print(len(boje), boje["nebo"])
```

Ključ "nebo" je naveden dva puta.

A

3 plava

B

tačno

2 siva

C

2 plava

D

SyntaxError

Pet kolekcija na jednom mestu

kolekcija	zagrade	uređena	promenljiva	ponavljanje	pristup
string	" "	da	ne	da	po indeksu
lista	[]	da	da	da	po indeksu
torka	()	da	ne	da	po indeksu
skup	set() ili {1, 2}	ne	da*	ne	in i for
rečnik	{ključ: vrednost}	da**	da	ključ ne, vrednost da	po ključu

* Skupu dodajemo i uklanjamo elemente, ali sami elementi moraju biti nepromenljivi (broj, string, torka).

** Rečnik čuva redosled dodavanja parova od Pythona 3.7.

Pravimo komande sa tastature za lika u igri

Cilj: za 15 minuta lik se pomera tasterima w, a, s, d, a q završava igru.

- 1 Lik je rečnik: ime, x i y
- 2 Rečnik komande: taster → pomeraj (torka)
- 3 while True: čitamo taster, q prekida
- 4 Nepoznat taster: proveravamo sa in
- 5 Raspakujemo pomeraj i menjamo x i y

KAKO IZGLEDA IGRA

```
Taster: d
Pozicija: 1 0
Taster: d
Pozicija: 2 0
Taster: s
Pozicija: 2 1
Taster: x
Nepoznat taster
Taster: q
```

Ceo program sa komandama

```
lik = {"ime": "Cikica", "x": 0, "y": 0}
komande = {"w": (0, -1), "s": (0, 1),
            "a": (-1, 0), "d": (1, 0)}

while True:
    taster = input("Taster: ")
    if taster == "q":
        break
    if taster not in komande:
        print("Nepoznat taster")
        continue
    dx, dy = komande[taster]
    lik["x"] += dx
    lik["y"] += dy
    print("Pozicija:", lik["x"], lik["y"])
```

komande[taster]

rečnik vraća torcu pomeraja

taster not in komande

provera pre čitanja, da ne bude KeyError

dx, dy = ...

raspakivanje torke

Pronađi grešku

```
boje = {"nebo": "plava", "sunce": "žuta"}  
for kljuc, vrednost in boje:  
    print(kljuc, vrednost)
```

PORUKA O GREŠCI

```
Traceback (most recent call last):  
  File "boje.py", line 2, in <module>  
    for kljuc, vrednost in boje:  
        ^^^^^^^^^^^^^^^^^^^  
ValueError: too many values to unpack (expected 2)
```

Program treba da ispiše svaki ključ i njegovu vrednost. Kako glasi ispravan drugi red?

line 2

red sa greškom

ValueError

vrednost ne odgovara onome što se traži

unpack

raspakivanje: „nebo“ ima 4 slova, a promenljive su 2

Izaberite nivo i počnite

LEVEL 1

Inventar

Lista predmeta ima ponavljanja. Ispišite koliko je različitih predmeta i da li je „mač“ među njima.

Pomoć: `set()` i `in`

LEVEL 2

Kartica lika

Napravite rečnik lika. Smanjite život za 1, dodajte oružje i ispišite sve parove.

Pomoć: `items()` u petlji `for`

LEVEL 3

Brojač boja

Za listu piksela napravite rečnik boja → broj pojavljivanja i ispišite najčešću boju.

Pomoć: `broj.get(boja, 0) + 1`

Sledeći put: funkcija koja vraća boju kao torku

```
import random

def nasumicna_boja():
    return (random.randint(0, 255),
            random.randint(0, 255),
            random.randint(0, 255))

print(nasumicna_boja())
```

JEDAN ISPIS

(128, 81, 116)

Funkciju `nasumicna_boja` pravimo u sledećoj temi. Svaki poziv vraća novu torku (r, g, b), a kasnije u PyGame-u tom bojom crtamo krugove.

Rezime

torka

```
tacka = (120, 80)
jedan = ("plava",)
```

Uređena, ne menja se. Jedan element traži zarez.

skup

```
boje = set(pikseli)
"plava" in boje
```

Bez ponavljanja i bez redosleda. Prazan skup: set().

rečnik

```
lik["zivot"] = 2
lik.get("brzina", 1)
```

Vrednost po ključu. Nepostojeći ključ: KeyError.

kopija i prolazak

```
drugi = lik.copy()
for k, v in lik.items():
```

Dodela ne kopira. items() daje parove.

Pitanje pre odlaska

```
d = {"a": 1, "b": 2}  
d["a"] = 3  
d["c"] = 1  
print(len(d))
```

Koji broj ispisuje program?

Odgovor pošaljite pre nego što izađete. Jedan broj je dovoljan.



Literatura

- Python dokumentacija, Tutorial, poglavlje 5 „Data Structures“ (torke, skupovi, rečnici): docs.python.org/3/tutorial
- Wentworth, P., Elkner, J., Downey, A. B., Meyers, C. How to Think Like a Computer Scientist: Learning with Python 3 (besplatna knjiga), poglavlja o torkama i rečnicima
- Sweigart, A. Invent Your Own Computer Games with Python, 4. izdanje, No Starch Press (besplatno na inventwithpython.com), poglavlje 9 (rečnici u igri „Vešala“)

Dodatna literatura i zanimljivi sajtovi

Petlja: Priručnik za Python

petlja.org · Priručnik za Python

Besplatan priručnik na srpskom, sa zadacima po nivoima težine.

Zvanični Python tutorijal

docs.python.org/3/tutorial

Tutorijal iz zvanične dokumentacije, na engleskom.

W3Schools: Python

w3schools.com/python

Kratke lekcije sa primerima koje menjate i pokrećete u pregledaču.

Exercism: Python

exercism.org/tracks/python

Besplatni zadaci za vežbu, poređani po temama.

Hvala!

Pitanja su uvek dobrodošla: na času, u četvrtu ili međim.

Slajdovi, primeri i kartica za podsećanje su na sajtu predmeta.

PREDAVANJA

dr Staša Vujičić Stanković

stasa.vujicic.stankovic@math.rs

www.matf.bg.ac.rs/~stasa.vujicic.stankovic

VEŽBE

Nevena Ćirić

poincare.matf.bg.ac.rs/~nevena.ciric/p1uu.html