Kurs: Programiranje igara sa programskim jezikom Pajton
008 čas: Program Vesanje.py
Teme: liste, metode sa listama, deljeno ukazivanje, rečnici

Upotreba lista

Liste su sekvence, poput ntorki, ali su liste promenjive (mogu se modifikovati).

program Inventar_heroja3.py

Za razliku od prethodne verzije u ovoj se neće koristiti ntorke već liste.

```
#Inventar_heroja3
#Prikazuje liste
inventar = ["mac", "oklop", "stit", "napitak zdravlja"]
print("Tvoje stvari:")
for stvar in inventar:
    print(stvar)
input("\nPritisni bilo koje dugme za nastavak.")
print("Imas", len(inventar), "stvari u inventaru.")
input("\nPritisni bilo koje dugme za nastavak.")
if "napitak zdravlja" in inventar:
    print("Borices se jos jedan dan.")
indeks = int(input("\nUnesi broj indeksa za neku stvar iz inventara: "))
print("Na indeksu", indeks, "je", inventar[indeks])
start = int(input("\nUnesi indeks za pocetak odsecka: "))
kraj = int(input("Unesi indeks za kraj odsecka: "))
print("inventar[", start, ":", kraj, "] je", end=" ")
print(inventar[start:kraj])
input("\nPritisni bilo koje dugme za nastavak.")
kovceg = ["zlato", "dragulji"]
print("Nasao si kovceg koji sadrzi:")
print(kovceg)
print("Dodao si sadrzaj kovcega u svoj inventar.")
inventar += kovceg
print("Tvoj inventar sada ima:")
print(inventar)
input("\nPritisni bilo koje dugme za nastavak.")
print("Zamenio si mac za samostrel.")
inventar[0] = "samostrel"
print("Tvoj inventar sada ima:")
print(inventar)
input("\nPritisni bilo koje dugme za nastavak.")
print("Za zlato i dragulje kupio si orb predvidjanja buducnosti.")
inventar[4:6] = ["orb predvidjanja buducnosti"]
print("Tvoj inventar sada ima:")
print(inventar)
input("\nPritisni bilo koje dugme za nastavak.")
print("U velikoj bici, tvoj stit se raspao.")
del inventar[2]
print("Tvoj inventar sada ima:")
print(inventar)
input("\nPritisni bilo koje dugme za nastavak.")
print("Tvoj samostrel i oklop su ukrali lopovi.")
del inventar[:2]
print("Tvoj inventar sada ima:")
print(inventar)
input("\n\nPritisni ENTER za izlazak iz igre.")
```

001 Kreiranje liste

```
inventar = ["mac", "oklop", "stit", "napitak zdravlja"]
print("Tvoje stvari:")
for stvar in inventar:
```

```
    print(stvar)
```
Ove linije koda kreiraju novu listu, dodeljuju novu listu promenjivoj inventar i štampaju svaki od elemenata. Vidi se da je razliku u odnosu na kreiranje ntorke u tome što se elementi liste smeštaju u pravougle zagrade.

002 Funkcija len() sa listama

Funkcija len() fukcioniše kao i kod ntorki, kao argument se koristi lista, poziv funkcije len vraća vrednost koja je broj elemenata u listi inventar.

```
print("Imas", len(inventar), "stvari u inventaru.")
```

003 Operator in sa listama

Operator in funkcioniše isto kao i kod ntorki, ovde se koristi za testiranje da li u listi postoji navedeni element:

```
if "napitak zdravlja" in inventar:
    print("Borices se jos jedan dan.")
```

004 Indeksiranje listi

Isto kao i kod ntorki, indeksiranje se koristi za pristup elementima liste prema njihovoj poziciji u sekvenci:

```
indeks = int(input("\nUnesi broj indeksa za neku stvar iz inventara: "))
print("Na indeksu", indeks, "je", inventar[indeks])
```

005 Odsecanje listi

Isko kao i kod ntorki, definisanjem dva broja (startni indeks i krajnji indeks odsečka) kreira se odsečak od liste:

```
start = int(input("\nUnesi indeks za pocetak odsecka: "))
kraj = int(input("Unesi indeks za kraj odsecka: "))
print("inventar[", start, ":", kraj, "] je", end=" ")
print(inventar[start:kraj])
```

006 Nadovezivanje listi

Nadovezivanje listi funkcioniše kao i kod ntroki. Sada se kreira lista i dodeljuje promenjivoj kovceg. I dalje se mogu nadovezivati samo sekvence istog tipa.

```
kovceg = ["zlato", "dragulji"]
print("Nasao si kovceg koji sadrzi:")
print(kovceg)
print("Dodao si sadrzaj kovcega u svoj inventar.")
inventar += kovceg
print("Tvoj inventar sada ima:")
print(inventar)
```

007 Promenjivost lista

Najveća razlika između lista i ntorki je u tome što su liste promenjive. Liste se mogu promeniti jednom kada se kreiraju i zato određene akcije sa listama se ne mogu ponoviti sa ntorkama.

008 Dodavanje novog elementa liste pomoću indeksa

Pošto su liste promenjive, postojeći element u listi može dobiti novu vrednost:

```
print("Zamenio si mac za samostrel.")
inventar[0] = "samostrel"
print("Tvoj inventar sada ima:")
print(inventar)
```

Linija : `inventar[0] = "samostrel"` dodeljuje novi string elementu liste inventar na poziciji 0. Na ovaj način se može promeniti vrednost elementa u listi ali se ne može kreirati novi element.

009 Dodela novog odsečka liste

Kao što se može dodeliti nova vrednost jednom elementu u listi, tako se može dodeliti i nova vrednost za odsečak. U programu se lista [“orb predvidjanja buducnosti”] dodeljuje odsečku inventar[4:6]:

```
print("Za zlato i dragulje kupio si orb predvidjanja buducnosti.")
inventar[4:6] = ["orb predvidjanja buducnosti"]
print("Tvoj inventar sada ima:")
print(inventar)
```

Ovaj iskaz dodele zamenjuje dve stavri inventar[4] i inventar[5] sa stringom “orb predvidjanja buducnosti”. Pošto je lista sa jednim elementom dodeljena odsečku liste sa dva elementa, to znači da se cela lista smanjila za jedan element ukupno.

010 Brisanje elementa liste

Element iz liste se briše sa komandom del:

```
print("U velikoj bici, tvoj stit se raspao.")
del inventar[2]
print("Tvoj inventar sada ima:")
print(inventar)
```

Kada se realizuje ovaj deo koda, element koji je bio na poziciji 2, string “stit” je odstranjen iz liste. Odstranjivanjem elementa iz liste se ne stvara praznina u listi. Dužina liste se smanjuje za jedan, a svi elementi posle odstranjenog se pomeraju prema prvom elementu za po jedno mesto. Tako da odstranjivanjem elementa liste za pozicije 2, element na poziciji 3 zauzima sada poziciju 2.

011 Brisanje odsečka liste

Odsečak liste se takođe može odstraniti:

```
print("Tvoj samostrel i oklop su ukrali lopovi.")
del inventar[:2]
print("Tvoj inventar sada ima:")
print(inventar)
```

Sa linijom `del inventar[:2]` se odstranjuje odsečak liste koji čine elementi samostrel i oklop iz inventar. Kao i pri odstranjivanju elementa iz liste, odstranjivanjme odsečka iz liste, dužina liste se smanjuje za broj elemenata u odsečku liste koji je odstranjen.

Metode sa listama

Metode u radu sa listama služe da se manipuliše njima. Kroz metode mogu se dodavati elementi, odstranjivati elementi bazirani na njihovoj vrednosti, sortirati liste, čak i preokrenuti redosled u listi.

program Najbolji_rezultati.py

```
#Najbolji_rezultati
#Prikazuje metode sa listama
rezultati = []
izbor = None
while izbor != "0":
    print(
    """
    Najbolji rezultati

    0 - Izlaz
    1 - Prikaz rezultata
    2 - Dodaj rezultat
    3 - Odstrani rezultat
    4 - Sortiraj rezultate
    """
    )
    izbor = input("Izbor: ")
    print()
    if izbor == "0":
        print("Dovidjenja.")
    elif izbor == "1":
         print("Najbolji rezultati")
```

```
        for bodovi in rezultati:
            print(bodovi)
    elif izbor == "2":
        bodovi = int(input("Koliki ti je rezultat?: "))
        rezultati.append(bodovi)
    elif izbor == "3":
        bodovi = int(input("Koji rezultat da odstranim?: "))
        if bodovi in rezultati:
            rezultati.remove(bodovi)
        else:
            print(bodovi, "nije na listi najboljih rezultata.")
    elif izbor == "4":
        rezultati.sort(reverse=True)
    else:
        print("Izvini, ali", izbor, "nije validan izbor.")
```

012 Prikaz rada programa

Posle komentara, kreirane su dve promenjive:

```
rezultati = []
izbor = None
```

rezultati je prazna lista u kojoj će se smeštati osvojeni bodovi a izbor predstavlja korisnikov izbor iz menija i inicijalizovan je na None.

Glavni deo programa je while petlja koja će se izvršavati sve dok se ne unese 0 kao vrednost za izlaz iz petlje.

Ako korisnik unese bilo koju drugu vrednost iz menija, izvršava se poseban deo programa koji obrađuje podatke na poseban način.

Za unos 1:

```
elif izbor == "1":
       print("Najbolji rezultati")
       for bodovi in rezultati:
            print(bodovi)
```

prikazaće se svi rezultati u listi rezultati.

Za unos 2:

```
elif izbor == "2":
       bodovi = int(input("Koliki ti je rezultat?: "))
        rezultati.append(bodovi)
```

program traži od korisnika unos novog rezultata i dodeljuje ga promenjivoj bodovi. Zatim preko linije: `rezultati.append(bodovi)` koristi metodu append() za dodavanje vrednosti iz bodovi na kraj liste rezultati. Sada će lista postati za jedan element duža.

Za unos 3:

```
elif izbor == "3":
       bodovi = int(input("Koji rezultat da odstranim?: "))
       if bodovi in rezultati:
            rezultati.remove(bodovi)
       else:
            print(bodovi, "nije na listi najboljih rezultata.")
```

kompjuter dobija od korisnika vrednost koja se smešta u promenjivu bodovi. Posle provere da li ta vrednost postoji u listi, element sa tom vrednosti se odstranjuje iz liste sa metodom remove(). Metod prolazi listom, počevši od pozicije 0 i traži vrednosti u rezultati listi. Kada metod pronađe prvo pojavljivanje te vrednosti, ona se briše iz liste. Ako je vrednost više od jednog puta u listi, odstranjuje se samo prvo pojavljivanje i lista je za jedan kraća.

Razlika između remove() i del() je u tome što remove() odstranjuje element iz liste bazirano na njegovoj vrednosti. Ako se pokuša odstraniti element iz liste čija vrednost ne postoji u listi, generisaće se greška koja prekida program.

Za unos 4:

```
elif izbor == "4":
        rezultati.sort(reverse=True)
```

čini da se sortiraju rezultati u listi. Metod sort() sortira elemente liste. Po difoltu, sort uređuje redosled po rastućoj vrednosti. Problem je ako se želi sortiranje po opadajućoj vrednosti. To se radi dodavanjem True kao vrednost parametra reverse u metodi sort.

Ako se ništa ne stavlja u zagrade metode sort, po difoltu će se sortirati podaci u rastućem redosledu.

Još neke metode sa listama:

count(vrednost) – vraća broj pojavljivanja vrednosti u listi

index(vrednost) – vraća poziciju prvog pojavljivanja vrednosti u listi

insert(i, vrednost) – ubacuje vrednost na poziciju i

pop([i]) – vraća vrednost na poziciji i a zatim i odstranjuje tu vrednost iz liste, ako se ne stavi i onda će se vratiti ta vrednost a zatim i odstraniti poslednji element i

013 Kada koristiti liste

Pošto su liste toliko fleksibilne, postavlja se pravo pitanje čemu služe ntorke? Ipak postoje mesta gde se treba koristiti ntorkama a ne listama:

- ntorke su brže nego liste. Pošto računar zna da se ne mogu promeniti, ntorke se mogu smeštati na način koji omogućava brže baratanje njima. U kompleksnim aplikacijama, sa velikim brojem linija koda i podatka, brzina jeste bitna
- Nepromenjivost ntorki čini ih savršenim za kreiranje konstanti pošto se neće menjati. Upotreba ntorki može dodati nivo sigurnosti i jasnoće u kod
- Ponekad se ntroke zahtevaju. U nekim slučajevima, Pajton zahteva nepromenjive vrednosti. Rečnici zahtevaju nepromenjive tipove, tako da ntorke mogu biti važne kod rečnika

Ali fleksibilnost lista ih čini mnogo primenjivijom nego ntorki.

Ugneždene sekvence

Rečeno je da se liste i ntorke sekvence bilo čega. Ako je to tačno, liste mogu sadržati druge liste ili ntroke, a ntroke mogu sadržati druge ntorke ili liste. Ako je to slučaj, one mogu graditi ugneždene (nestovane) sekvence. Nestovane sekvence su dobar način organizovanja kompleksnih kolekcija podataka.

program Najbolji_rezultati2.py

```
#Najbolji_rezultati2
#Prikazuje ugnezdene sekvence
rezultati = []
izbor = None
while izbor != "0":
    print(
    """
    Najbolji_rezultati2

    0 - Izlaz
    1 - Lista rezultata
    2 - Dodaj rezultat
    """
```

```
    )
    izbor = input("Izbor: ")
    print()
    if izbor == "0":
        print("Dovidjenja.")
    elif izbor == "1":
        print("Najbolji rezultati\n")
        print("IME\tREZULTAT")
        for ulaz in rezultati:
            bodovi, ime = ulaz
            print(ime, "\t", bodovi)
    elif izbor == "2":
        ime = input("Kako se zove igrac?: ")
        bodovi = int(input("Koji je rezultat igraca?: "))
        ulaz = (bodovi, ime)
        rezultati.append(ulaz)
        rezultati.sort(reverse=True)
        rezultati = rezultati[:5]
    else:
        print("Izvini, ali", izbor, "nije dobar izbor.")
```

014 Kreiranje ugneždene sekvence

Kreiranje nestovane sekvence lista ili ntorki se pravi kao i uvek: otkucaj svaki element iza kojeg je zarez. Razlika od dodadašnjih nestovanih sekvenci je što sada se uključuju čitave liste ili ntorke kao elementi:

```
>>> gnezdo = ["prvi", ("drugi", "treci"), ["cetvrti", "peti", "sesti"]]
>>> print(gnezdo)
['prvi', ('drugi', 'treci'), ['cetvrti', 'peti', 'sesti']]
```

U ovom primeru, iako postoji šest stringova, gnezdo ima samo tri elementa. Prvi je string “prvi“, drugi je ntorka `("drugi", "treci")` a treći elemement je lista`["cetvrti", "peti", "sesti"]`.

Pri kreiranju liste ili ntorke sa bilo kojim brojem lista ili ntorki, korisna ugneždena sekvenca često ima identičan oblik:

```
>>> bodovi = [("Ana", 1000), ("Miki", 1500), ("Zoki", 2000)]
>>> print(bodovi)
[('Ana', 1000), ('Miki', 1500), ('Zoki', 2000)]
```

Ovde je bodovi lista sa tri elementa. Svaki element je ntorka. Svaka ntorka ima tačno dva elementa koji su string i broj. U ovom primeru, sekvenca predstavlja tabelu najboljih rezultata sa imenom i rezultatom (kao svaka prava tabela rezultata). Ipak, kreiranje sve dublje nestovanih sekvenci nije dobra ideja, pa čak i iskusni programeri ne koriste ugneždene sekvence dublje od dva nivoa. Najčešće su nestovane sekvence sa jednim nivoom dubine.

015 Pristup nestovanim elementima

Pristup elementima nestovane sekvence je kao pristup bilo kojoj sekvenci, preko indeksiranja:

```
>>> bodovi = [("Ana", 1000), ("Miki", 1500), ("Zoki", 2000)]
>>> print(bodovi[0])
('Ana', 1000)
>>> print(bodovi[2])
('Zoki', 2000)
```

Svaki od pojedinačnih elemenata je ntorka, ali kako prići elementu elementa tj. elementu ntorke same? Jedan od načina je dodeliti ntorku promenjivoj i indeksirati ga:

```
>>> prvi_rez = bodovi[2]
>>> print(prvi_rez)
('Zoki', 2000)
>>> print(prvi_rez[0])
Zoki
```

Ali, postoji direktan način da se priđe elementu ntorke direktno iz bodovi:
print(bodovi[2][0]), što daje Zoki kao izlaz.
Pisanjem bodovi[2][0] daju se dva podatka, kojima se kaže da se uzme element iz bodovi na poziciji 2 (što je `('Zoki', 2000)`) a zatim unutar njega uzeti element na poziciji 0 (koji je Zoki). Na ovaj način se može koristiti višestruko indeksiranje sa ugneždenim sekvencama da bi se prišlo direktno nestovanom elementu.

016 Raspakovanje sekvence

Ako se zna koliko elemenata ima u sekvenci, može se u jednoj liniji dodeliti svakom od elemenata njegova promenjiva:

```
>>> ime, bod = ("Miki", 100)
>>> print(ime)
Miki
>>> print(bod)
100
```

Ovaj postupak se naziva raspakovanje (unpacking) i može se koristiti na bilo kojem tipu sekvence. Samo je neophodno koristiti isti broj promenjivih kao elemene sekvence, pošto bi se u drugom slučaju generisala greška.

017 Objašnjenje programa Najbolji_rezultati2.py

Za izbor 1:

```
elif izbor == "1":
        print("Najbolji rezultati\n")
        print("IME\tREZULTAT")
        for ulaz in rezultati:
            bodovi, ime = ulaz
            print(ime, "\t", bodovi)
```

kompjuter prolazi po svakom od elemenata u rezultati i raspakuje bodove i ime u obliku promenjivih bodovi, ime a zatim ih štampa.
Za izbor 2:

```
elif izbor == "2":
        ime = input("Kako se zove igrac?: ")
        bodovi = int(input("Koji je rezultat igraca?: "))
        ulaz = (bodovi, ime)
        rezultati.append(ulaz)
        rezultati.sort(reverse=True)
        rezultati = rezultati[:5]
```

kompjuter traži od korisnika novi broj bodova i ime. Sa time, kompjuter pravi ntorku ulaz. Izbor da se prvo ubace bodovi pa onda ime igrača je da bi se ulazi sortirali prvo prema bodovima pa onda prema imenu. Kompjuter zatim pridodaje novi ulaz u listu rezultati. Zatim se lista sortira po opadajućem redosledu a zatim se odsečak od pet prvih rezultata po bodovima dodeli promenjivoj rezultati što znači da su ostali elementi liste nestali sa liste.

018 Deljeno ukazivanje (shared references)

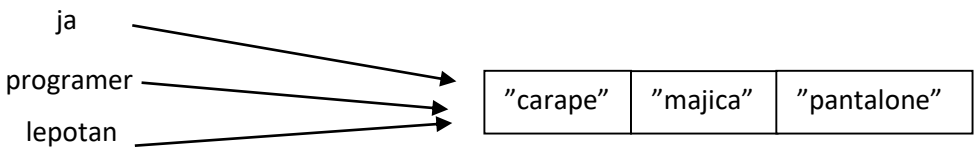
Ranije je prikazano da promenjiva ukazuje na vrednost. To znači da promenjiva ukazuje na mesto u memoriji gde je vrednost smeštena.
language = “Python”
ovako promenjiva language ukazuje na vrednost a to je string ”Python” negde u memoriji računara.

Za nepromenjive vrednosti poput brojeva, stringova i ntroki ovaj način rada nema neko posebno značenje. Drugačije je sa promenjivim vrednostima poput lista. Kada nekoliko

promenjivih ukazuje na istu promenjivu vrednost, oni dele istu referencu. Oni svi se odose na jednu, jedinstvenu kopiju te vrednosti. Promena u vrednosti kroz neku od tih promenjivih rezultuje u promeni za sve promenjive, pošto postoji samo jedna, deljena kopija.

Primer:

```
>>> ja = ["carape", "majica", "pantalone"]
>>> programer = ja
>>> lepotan = ja
>>> print(ja)
['carape', 'majica', 'pantalone']
>>> print(lepotan)
['carape', 'majica', 'pantalone']
>>> print(programer)
['carape', 'majica', 'pantalone']
```



Tri promenjive ukazuju na istu listu sa tri elementa u listi. To znača da promena u listi preko bilo koje od tri promenjivih će promeniti listu:

```
>>> programer[1] = ”dzemper”
>>> print(programer)
['carape', 'dzemper', 'pantalone']
>>> print(ja)
['carape', 'dzemper', 'pantalone']
>>> print(programer)
['carape', 'dzemper', 'pantalone']
```

Ovo pokazuje da treba biti oprezan u radu sa deljenim ukazivanjem sa promenjivim vrednostima.

Ipak ovaj efekat se može izbeći kopiranjem liste kroz odsecanje:

```
>>> ja = ["carape", "majica", "pantalone"]
>>> programer = ja[:]
>>> ja[1] = ”dzemper”
>>> print(ja)
['carape', 'dzemper', 'pantalone']
>>> print(programer)
['carape', 'majica', 'pantalone']
```

Vidi se da je promenjiva programer zadržala originalni izgled promenjive ja, dok je promenjiva ja promenila jedan element.

Rečnik

U programiranju se dosta vrši organizovanje podataka. Preko ntorki i lista se podaci mogu organizovati u sekvence. Rečnici takođe služe organizovanju podataka ali na drugačiji način. Sa rečnikom, ne smeštaju se podaci u sekvencu, već se smeštaju u parovima. To izgleda kao pravi rečnik gde svaki ulaz je par: reč i njena definicija. Kada se potraži po reči, dobija se njena definicija. Ovaj par u Pajtonu se naziva ključ (key) i vrednost (value).

program Prevodilac_sa_streberskog.py

```
#Prevodilac_sa_streberskog
#Prikazuje recnike
streberski = {"404": "nemam pojma. Poreklo iz veb poruke o gresci 404, znaci strana
nije pronadjena.",
        "Guglanje": "pretraga po internetu za dodatne informacije o osobi.",
```

```
        "Kuga tastature" : "kolekcija prasine u racunarskoj tastaturi.",
        "Truli link" : "proces preko kojeg link na veb strani postaje nepotreban.",
        "Servisiranje ubedjivanjem" : "akt udaranja po elektronskom uredjaju da bi
ovaj proradio.",
        "Deinstaliran" : "otpusten. Posebno popularno tokom doba ranog interneta."}
izbor = None
while izbor != "0":
    print(
    """
    Prevodilac sa streberskog

    0 - Izlaz
    1 - Pretraga streberskog izraza
    2 - Dodavanje streberskog izraza
    3 - Redefinisanje streberskog izraza
    4 - Brisanje streberskog izraza
    """
    )
    izbor = input("Izbor: ")
    print()
    if izbor == "0":
        print("Dovidjenja.")
    elif izbor == "1":
        izraz = input("Koji izraz treba da prevedem?: ")
        if izraz in streberski:
            definicija = streberski[izraz]
            print("\n", izraz, "znaci", definicija)
        else:
            print("\nIzvinite, ne znam", izraz)
    elif izbor == "2":
        izraz = input("Koji izraz zelis da dodam?: ")
        if izraz not in streberski:
            definicija = input("\nKoja je definicija?: ")
            streberski[izraz] = definicija
            print("\n", izraz, "je dodat.")
        else:
            print("\nOvaj izraz vec postoji! Probaj da ga redefinises.")
    elif izbor == "3":
        izraz = input("Koji izraz zelis da redefinisem?: ")
        if izraz in streberski:
            definicija = input("Koja je nova definicija?: ")
            streberski[izraz] = definicija
            print("\n", izraz, "je redefinisan.")
        else:
            print("\nTaj izraz ne postoji! Pokusaj da ga dodas.")
    elif izbor == "4":
        izraz = input("Koji izraz zelis da izbrisem?: ")
        if izraz in streberski:
            del streberski[izraz]
            print("\nU redu, izbrisan je", izraz)
        else:
            print("\nTo ne mogu da uradim!", izraz, "ne postoji u recniku.")
    else:
        print("\nIzvini, ali", izbor, "nije validan izbor.")
```

019 Kreiranje rečnika

Prvo se kreira nov rečnik sa izrazima i definicijama. Prvo su štreberski izrazi a zatim njihove definicije.

```
streberski = {"404": "nemam pojma. Poreklo iz veb poruke o gresci 404, znaci strana
nije pronadjena.",
        "Guglanje": "pretraga po internetu za dodatne informacije o osobi.",
        "Kuga tastature" : "kolekcija prasine u racunarskoj tastaturi.",
```

```
        "Truli link" : "proces preko kojeg link na veb strani postaje nepotreban.",
        "Servisiranje ubedjivanjem" : "akt udaranja po elektronskom uredjaju da bi
ovaj proradio.",
        "Deinstaliran" : "otpusten. Posebno popularno tokom doba ranog interneta."}
```

Kod kreira rečnik koji se zove streberski. Sastoji se od šest parova, koji se nazivaju stvari (items). Svaka stvar se sastoji od ključa i vrednosti. Ključevi su sa leve strane dvotačke a vrednosti su sa desne strane dvotačke. Ključevi su zaista pravi ključevi kojima se može doći do vrednosti.

Za kreiranje sopstvenog ključa treba pratiti ovakvu šemu. Otkucati ključ, dvotačku pa vrednost. Zarezi se koriste za razdvajanje parova ključ-vrednost, i treba zatvoriti sve u vitičaste zagrade. Poput ntorki i listi, može se koristiti više linija i to posle svakog zareza.

Pristup vrednostima rečnika

Najčešća stvar koja se radi sa rečnicima je korišćenje ključeva za pristup vrednostima. Postoji nekoliko načina za to.

020 Upotreba ključa za dobijanje vrednosti

Ovo je najjednostavniji način. Potrebno je samo posle naziva rečnika smestiti ključ u pravouglu zagradu:

```
>>> streberski["404"]
'nemam pojma. Poreklo iz veb poruke o gresci 404, znaci strana nije pronadjena.'
>>> streberski["Truli link"]
'proces preko kojeg link na veb strani postaje nepotreban.'
```

Postoji razlika u ovom načinu od indeksiranja sekvence. Kada se indeksira sekvenca, koristi se pozicija. Kada se traži vrednost u rečniku, gleda se ključ. Rečnici uopšte nemaju brojeve koji opisuju poziciju.

Vrednost ne može da se koristi za dobijanje ključa.

021 Testiranje ključa sa operatorom in pre dobijanja vrednosti

Pošto korišćenje nepostojećeg ključa dovodi do greške koja prekida program, najbolje je da se ne pristupa rečniku bez uzimanja u obzir nekih predostrožnosti. Najjednostavnije je proveriti da li zaista i postoji željeni ključ pre pokušaja dobijanja njegove vrednosti. Za to se koristi operator in:

```
>>> if "Igrajuci medved" in streberski:
...     print("Znam sta je to Igrajuci Medved")
... else:
...     print("Nemam ideju sta je to Igrajuci medved")
...
Nemam ideju sta je to Igrajuci medved
```

Pošto rečnik ne sadrži ključ Igrajuci medved, uslov je False i kompjuter daje poruku da taj ključ ne postoji. Na ovaj način se može samo proveriti da li postoji određeni ključ u rečniku, ne kakva je vrednost nekog ključa.

022 Upotreba get() metode za dobijanje vrednosti

Postoji još jedan način da se dobije vrednost iz rečnika. Može se koristiti rečnik metod get(). Metod ima ugrađenu sigurnosnu mrežu za upravljanje situacijama gde se traži vrednost ključa koji ne postoji. Ako ključ ne postoji, metod vraća difolt vrednost, koja se može definisati:

```
>>> print(streberski.get("Igrajuci medved", "Pojma nemam."))
Pojma nemam.
```

Upotrebljavajući get metod, garantovano se dobija vrednost. Ako je ovaj izraz bio u rečniku kao ključ, onda bi se dobila željena vrednost. Pošto je nema, nazad se dobija difolt vrednost koja je definisana, tj string ”Pojma nemam.”.

Da bi se koristio get() metoda, treba samo dati ključ koji se traži i opcionalna difolt vrednost. Ako ključ jeste u rečniku, dobija se njegova prava vrednost. Ako ključ nije u rečniku, dobija se difolt vrednost. Ali, ako se ne obezbedi difolt vrednost, onda se dobija None:

```
>>> print(streberski.get("Igrajuci medved")
None
```

Rad na programu

Ako korisnik unese 1:

```
elif izbor == "1":
        izraz = input("Koji izraz treba da prevedem?: ")
        if izraz in streberski:
            definicija = streberski[izraz]
            print("\n", izraz, "znaci", definicija)
        else:
            print("\nIzvinite, ne znam", izraz)
```

Ovaj deo programa traži izraz za prevođenje. Kompjuter proverava da li izraz postoji u rečniku. Ako postoji, program pristupa rečniku, koristi izraz kao ključ, dobija njegovu definiciju i štampa ih. Ako izraz nije u rečniku, kompjuter o tome obaveštava korisnika.

Ako korisnik unese 2:

```
elif izbor == "2":
        izraz = input("Koji izraz zelis da dodam?: ")
        if izraz not in streberski:
            definicija = input("\nKoja je definicija?: ")
            streberski[izraz] = definicija
            print("\n", izraz, "je dodat.")
        else:
            print("\nOvaj izraz vec postoji! Probaj da ga redefinises.")
```

Rečnici su promenjivi, tako da se mogu modifikovati. Ovaj deo programa dodaje nov izraz u rečnik. Kompjuter pita korisnika za nov izraz koji će dodati. Ako izraz već nije u rečniku, kompjuter uzima definiciju i dodaje par kroz liniju: `streberski[izraz] = definicija` .

Ovo kreira novu stvar u streberski rečniku. Na ovaj način se i dodeljuje nova stvar u rečniku. Kod je napisan tako da ako izraz već postoji u rečniku, neće se dodavati novi izraz.

Ako korisnik unese 3:

```
elif izbor == "3":
        izraz = input("Koji izraz zelis da redefinisem?: ")
        if izraz in streberski:
            definicija = input("Koja je nova definicija?: ")
            streberski[izraz] = definicija
            print("\n", izraz, "je redefinisan.")
        else:
            print("\nTaj izraz ne postoji! Pokusaj da ga dodas.")
```

ovaj deo koda služi za zamenu postojećeg para ključ-vrednost. Za samu zamenu koristi se ista linija koda koja se koristi pri dodavanju novog para: `streberski[izraz] = definicija` . Pajton zamenjuje trenutnu vrednost (definiciju) sa novom. Ako se dodeli vrednost u rečnik korišćenjem ključa koji već postoji, Pajton zamenjuje trenutnu vrednsot sa novom.

Ako korisnik unese 4:

```
elif izbor == "4":
        izraz = input("Koji izraz zelis da izbrisem?: ")
        if izraz in streberski:
            del streberski[izraz]
            print("\nU redu, izbrisan je", izraz)
        else:
            print("\nTo ne mogu da uradim!", izraz, "ne postoji u recniku.")
```

program traži od korisnika izraz koji će biti obrisan. Zatim, program proverava da li je taj izraz u rečniku pomoću operatora in. Ako jeste stvar se briše sa: `del streberski[izraz]` .
Ovime se briše stvar sa ključem izraz iz rečnika streberski. Može se obrisati bilo koja stvar u rečniku na ovaj način. Ako izraz ne postoji u rečniku, sa iskazom else će se obavestiti o tome korisnik. Pokušaj brisanja nepostojećeg ključa vodi do prijave o grešci i prekida programa.

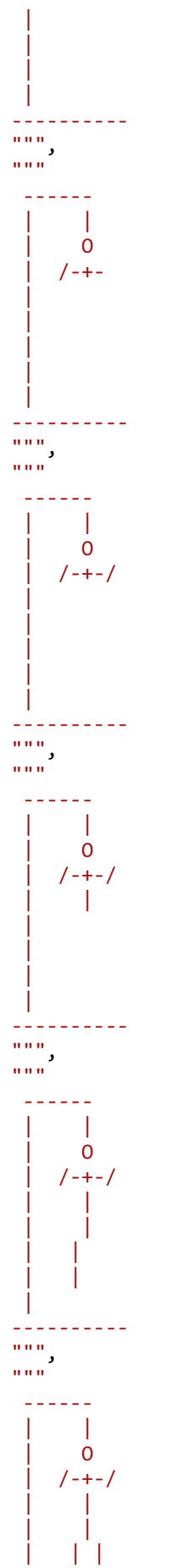
023 Razumevanje osobina rečnika

Rečnik ne može sadržavati više stvari sa istim ključem.
Ključ mora biti nepromenjiv. Može biti string, broj ili ntorka, što daje mnogo mogućnosti.
Vrednosti ne moraju biti jedinstvene. Takođe, vrednosti mogu biti promenjive i nepromenjive.
Druge metode rečnika:
keys() – vraća pogled (view) svih ključeva u rečniku
values() – vraća pogled svih vrednosti u rečniku
items() – vraća pogled svih stvari u rečniku; svaka stvar je ntorka sa dva elementa, gde je prvi element ključ a drugi element je vrednost ključa

> Pogledi rečnika – vraćeni od keys(), values(), items() – su slični listama. Mogu se iterirati sa for petljom. Ipak, nisu liste jer ne mogu da se indeksiraju. Pogledi su dinamički, što znači da njihov sadržaj nije nezavistan od njihovih rečnika.

Kod za program Vesanje.py

```
#Vesanje
#Kompjuter bira slucajnu rec i igrac bira jedno po jedno slovo.
#Igrac ima 7 pokusaja, i ako ne uspe zavrsice se slika vesanja.
import random
VESALA = (
"""
 ------
 |    |
 |
 |
 |
 |
 |
 |
----------
""",
"""
 ------
 |    |
 |    O
 |
 |
 |
 |
 |
----------
""",
"""
 ------
 |    |
 |    O
 |   -+-
 |
```



```
 |   | |
 |
----------
""")
MAKSIMUM_GRESAKA = len(VESALA) - 1
RECI = ("OPTERECEN", "KLJESTA", "SRBIJA", "STRAFTA", "PAJTON")
rec = random.choice(RECI)
do_sada = "-" * len(rec)
greska = 0
iskorisceno = []
print("Dobrodosli na igru Vesanje. Sretno!")
while greska < MAKSIMUM_GRESAKA and do_sada != rec:
    print(VESALA[greska])
    print("\nDo sada su koriscena sledeca slova:\n", iskorisceno)
    print("\nDo sada, rec je:\n", do_sada)
    pokusaj = input("\n\nUnesi tvoj pokusaj: ")
    pokusaj = pokusaj.upper()
    while pokusaj in iskorisceno:
        print("Vec si pokusao ovo slovo", pokusaj)
        pokusaj = input("Unesi tvoj pokusaj: ")
        pokusaj = pokusaj.upper()
    iskorisceno.append(pokusaj)
    if pokusaj in rec:
        print("\nDa!", pokusaj, "se nalazi u reci!")
        novo = ""
        for i in range(len(rec)):
            if pokusaj == rec[i]:
                novo += pokusaj
            else:
                novo += do_sada[i]
        do_sada = novo
    else:
        print("\nIzvini,", pokusaj, "nije trazena rec.")
        greska += 1
if greska == MAKSIMUM_GRESAKA:
    print(VESALA[greska])
    print("\nIgrac je obesen!")
else:
    print("\nRec je pogodjena!")
print("\nTo je bila rec", rec)
```

Tvoje stvari:

mac

oklop

stit

napitak zdravlja

Pritisni bilo koje dugme za nastavak.

Imas 4 stvari u inventaru.

Pritisni bilo koje dugme za nastavak.

Borices se jos jedan dan.

Unesi broj indeksa za neku stvar iz inventara: 2

Na indeksu 2 je stit

Unesi indeks za pocetak odsecka: 0

Unesi indeks za kraj odsecka: 2
inventar[0 : 2] je ['mac', 'oklop']

Pritisni bilo koje dugme za nastavak.
Nasao si kovceg koji sadrzi:
['zlato', 'dragulji']
Dodao si sadrzaj kovcega u svoj inventar.
Tvoj inventar sada ima:
['mac', 'oklop', 'stit', 'napitak zdravlja', 'zlato', 'dragulji']

Pritisni bilo koje dugme za nastavak.
Zamenio si mac za samostrel.
Tvoj inventar sada ima:
['samostrel', 'oklop', 'stit', 'napitak zdravlja', 'zlato', 'dragulji']

Pritisni bilo koje dugme za nastavak.
Za zlato i dragulje kupio si orb predvidjanja buducnosti.
Tvoj inventar sada ima:
['samostrel', 'oklop', 'stit', 'napitak zdravlja', 'orb predvidjanja buducnosti']

Pritisni bilo koje dugme za nastavak.
U velikoj bici, tvoj stit se raspao.
Tvoj inventar sada ima:
['samostrel', 'oklop', 'napitak zdravlja', 'orb predvidjanja buducnosti']

Pritisni bilo koje dugme za nastavak.
Tvoj samostrel i oklop su ukrali lopovi.
Tvoj inventar sada ima:
['napitak zdravlja', 'orb predvidjanja buducnosti']

Pritisni ENTER za izlazak iz igre.

024 Objašnjenje programa Vesanje.py

Na početku se definiše ntorka VESALA koja se sastoji od osam elemenata gde je svaki od elemenata string unutar trostrukih navodnika sastavljen od 12 linija ASCII znakova. Ova ntorka se u kodu koristi kao konstanta, jer se njen sadržaj neće menjati samo će se koristiti u kontekstu potreba prikaza stringova.
Zatim se kreira konstanta: `MAKSIMUM_GRESAKA = len(VESALA) - 1`
Maksimalan broj pogrešnih pokušaja je za jedan manji od dužine ntorke VESALA. To je zato što se prva slika, praznih vešala prikazuje i pre početka pogađanja. To znači da igrač ima pravo na sedam grešaka pre kraja igre.
Zatim se kreira ntorka: `RECI = ("OPTERECEN", "KLJESTA", "SRBIJA", "STRAFTA", "PAJTON")`
koja sadrži sve moguće reči koje računar može izabrati da ih igrač pogadja.

025 Inicijalizacija promenjivih

Koristi se funkcija random.choice() za odabiranje slučajne reči iz liste mogućih reči. Promenjivoj rec je dodeljena tajna reč.

```
rec = random.choice(RECI)
```

Kreirana je sledeća promenjiva do_sada, koja predstavlja sve što je igrač do sada pogađao. String počinje samo kao niz crtica, jedna za svako slovo u reči. Kada igrač tačno pogodi slovo, crtica u poziciji slova se zamenjuje sa samim slovom.

```
do_sada = "-" * len(rec)
```

Promenjiva greska ima dodeljeni broj 0 i vodi računa o broju pogrešnih pogađanja igrača.

```
greska = 0
```

Kreirana je prazna lista, `iskorisceno = []`koji sadrži sva slova koja je igrač pogađao.

026 Delovi igre

Igračev izbor se konvertuje u velika slova da bi mogao da se potraži u tajnoj reči koja je u velikim slovima.

```
pokusaj = input("\n\nUnesi tvoj pokusaj: ")
    pokusaj = pokusaj.upper()
    while pokusaj in iskorisceno:
        print("Vec si pokusao ovo slovo", pokusaj)
        pokusaj = input("Unesi tvoj pokusaj: ")
        pokusaj = pokusaj.upper()
    iskorisceno.append(pokusaj)
```

Posle toga, treba biti siguran da igrač nije već koristio to slovo. Ako jeste, onda igrač unosi novo slovo sve dok igrač ne unese ono koje još nije koristio. Kada igrač unese validan pokušaj, konvertuje se pokusaj u velika slova i dodaje listi iskorišćenih slova.

Sledeće, proverava se ako je pokusaj u tajnoj reči.

```
if pokusaj in rec:
        print("\nDa!", pokusaj, "se nalazi u reci!")
        novo = ""
        for i in range(len(rec)):
            if pokusaj == rec[i]:
                novo += pokusaj
            else:
                novo += do_sada[i]
        do_sada = novo
    else:
        print("\nIzvini,", pokusaj, "nije trazena rec.")
        greska += 1
```

Ako jeste, kaže se to igraču. Zatim se kreira nova verzija od do_sada da bi se uključila nova slova na svim mestima gde je to slovo u tajnoj reči.

Ako igračev pokušaj nije u reči, igrač se obaveštava i povećava se broj pokušaja za jedan.

Na kraju, ako broj pokušaja je dostigao maksimum, igrač je izgubio igru. Tada se iscrtava poslednja slika. U suprotnom se čestita igraču.