Kurs: Programiranje igara sa programskim jezikom Pajton
006 čas: Program Izmesane_reci.py
Teme: for petlja, string operatori i funkcije, konstante, ntorke

Kod za program Izmesane_reci.py

```
#Izmesane_reci
#Kompjuter slucajnim izborom bira rec a zatim izmesa slova te reci
#igrac pogadja originalnu rec
import random
RECI = ("python", "jumble", "easy", "difficult", "answer", "xylophone")
rec = random.choice(RECI)
tacna_rec = rec
izmesana =""
while rec:
    pozicija = random.randrange(len(rec))
    izmesana += rec[pozicija]
    rec = rec[:pozicija] + rec[(pozicija + 1):]

print(
"""
           Dobrodosli u Ismesane Reci!

        Pogodi rec od ispreturanih slova.
          (Pritisni ENTER za kraj igre.)
""")
print("Izmesana slova:", izmesana)
pokusaj = input("\nTvoj pokusaj: ")
while pokusaj != tacna_rec and pokusaj != "":
    print("Izvini, ali to nije tacno.")
    pokusaj = input("Tvoj pokusaj: ")

if pokusaj == tacna_rec:
    print("To je to! Pogodak!\n")

print("Hvala na igranju.")
```

030 018 010 015

Upotreba for petlje

Petlja for ponavlja kod bazirano na sekvenci (nizu), ili redosledu u listi stvari. Ova petlja ponavlja sadržaj tela petlje za svaki element u sekvenci, po redosledu. Kada dostigne kraj sekvence, petlja staje sa radom. Npr, ako petlja se kreće po spisku naziva filmova, ona uzima prvi film po redosledu u sekvenci, ispiše njegov naziv, uzima sledeći iz sekvence, itd.

program Zapetljani_string.py

Program uzima reč od korisnika a zatim štampa njena slova po redosledu na razdvojenim linijama.

```
#Zapetljani_string
#Prikazuje rad for petlje sa stringovima
rec = input("Unesi jednu rec: ")
print("\nOvo su slove te reci:")
for slovo in rec:
    print(slovo)
```

```
Unesi jednu rec: kodovi

Ovo su slove te reci:
k
o
d
o
v
i
```

001 Objašnjenje rada for petlje

Sve sekvence se sastoje od elementa. String je sekvenca u kojoj svaki element je po jedan karakter. U slučaju stringa “kodovi“, prvi element je slovo ’k’, drugi element je slovo ’o’ itd.

Petlja for prolazi kroz (kaže se iterira preko) sekvence jedan po jedan element. U programu, petlja iteriria preko stringa "kodovi", jedno po jedno slovo. Petlja for koristi promenjivu koja dobija svaki sledeći element u sekvenci. U programu, slovo je ime promenjive koja dobija svako slovo iz stringa. Unutar tela petlje, petlja može nešto korisno da uradi sa svakim pojedinačnim elementom. U ovom primeru, samo se ispisuje vrednost promenjive slovo.
Tako, kada petlja startuje, slovo promenjiva je kreirana i dobija prvi karakter iz promenjive reci koji je slovo 'k'. Zatim, u telu petlje, iskaz **print** štampa k. Sledeće, kada telo petlje završi, kontrola se vraća na vrh petlje i promenjiva slovo dobija novi znak iz promenjive reci, koji je 'o'. Računar prikazuje o i petlja se nastavlja sve dok se svaki karakter u stringu ne prikaže.

002 Kreiranje for petlje

Da bi se kreirala for petlja, može se korsititi primer iz prethodnog programa.

```
for slovo in rec:
    print(slovo)
```

Započinje sa **for**, zatim se koristi promenjiva koja dobija elemente iz sekvence, zatim službena reč **in** i na kraju sama sekvenca po kojoj će petlja da iterira (na kraju je dvotačka).
U sledećem redu započinje telo petlje, tj ono što će se u svakom ciklusu petlje izvršiti.

Brojanje sa for petljom

Često se javlja zahtev za brojanjem nečega u programiranju. U kombinaciji sa **for** petljom, može se koristiti funkcija **range**() za razna prebrojavanja.

Program Brojac.py

```
#Brojac
#Prikazuje funkciju range()
print("Brojanje:")
for i in range(10):
    print(i, end=" ")

print("\n\nBrojanje po pet:")
for i in range(0, 50, 5):
    print(i, end=" ")

print("\n\nBrojanje unazad:")
for i in range(10, 0, -1):
    print(i, end=' ')

input("\n\nPritisni bilo koje dugme za izlaz.\n")
```

```
Brojanje:
0 1 2 3 4 5 6 7 8 9

Brojanje po pet:
0 5 10 15 20 25 30 35 40 45

Brojanje unazad:
10 9 8 7 6 5 4 3 2 1

Pritisni bilo koje dugme za izlaz.
```

003 Brojanje unapred

Prva petlja broji unapred:

```
for i in range(10):
    print(i, end=" ")
```

Petlja for iterira preko sekvence čije vrednosti generiše funkcija range(). Funkcija range() vraća sekvencu brojeva. Ako se u range() ubaci pozitivan ceo broj, ona vraća sekvencu koja startuje od 0 pa sve do (ne uključujući) tog pozitivnog broja. Zato range(10) vraća sekvencu: 0 1 2 3 4 5 6 7 8 9.

004 Brojanje po pet

Druga petlja broji po pet:

```
for i in range(0, 50, 5):
    print(i, end=" ")
```

Ako se u funkciju range() upišu tri vrednosti, ova funkcija ih vidi kao početnu tačku, krajnju tačku i broj po kojem se broji (korak). Početna tačka je uvek prva vrednost u sekvenci, dok se krajnja tačka nikad ne nalazi u sekvenci. Zato, dobijena sekvenca je 0 5 10 15 20 25 30 35 40

45. Sekvenca se završava sa 45 iako je krajnja tačka 50 (pošto se krajnja tačka ne uključuje u sekvencu). Da bi se 50 uključilo u sekvencu treba izabrati za krajnju tačku neku veću vrednost, npr 51.

005 Brojanje unazad

Treća petlja broji unazad:

```
for i in range(10, 0, -1):
    print(i, end=' ')
```

To je omogućeno postavljanjem poslednje vrednosti na -1. Ovo govori funkciji da krene od početne tačke prema krajnjoj tački dodajući -1 svaki put. Zato se proizvodi sekvenca 10 9 8 7 6 5 4 3 2 1.

Moguće je da u telu petlje nema nikakve promenjive. Samo se želi ponoviti neka akcija određeni broj puta. Da bi se to i ostvarilo, kreira se telo for petlje na čije izvršavanje nikakva promenjiva ne utiče. Npr, ispisivanje teksta određeni broj puta:

```
for i in range(10):
    print('Zdravo')
```

Korišćenje operatora sekvenci i funkcija sa stringovima

Stringovi su vrsta sekvenci, sa slovima kao pojedinačnim karakterima. Pajton koristi korisne funkcije i operatore koji rade sa bilo kojom vrstom sekvenci, uključujući i stringove. Ove funkcije i operatori mogu dati osnovne ali bitne stvari o sekvenci, poput dužine sekvence ili da li se određeni element nalazi u sekvenci.

program Analizator_poruka.py

```
#Analizator_poruka
#Prikazuje funkciju len() i operator in
poruka = input("Unesi poruku: ")
print("\nDuzina tvoje poruke je:", len(poruka))
print("\nNajcesce korisceno slovo u srpskom jeziku, 'a',")
if "a" in poruka:
    print("nalazi se u tvojoj poruci.")
else:
    print("ne nalazi se u tvojoj poruci.")
```

```
Unesi poruku: Ko peva zlo ne misli

Duzina tvoje poruke je: 20

Najcesce korisceno slovo u srpskom jeziku, 'a',
nalazi se u tvojoj poruci.
```

006 Korišćenje funkcije len()

Kada se dobije korisnička poruka i smesti u promenjivu poruka, dobijanje dužine poruke se realizuje linijom: `print("\nDuzina tvoje poruke je:", len(poruka))`

Može se pridodati bilo koja sekvenca u len() funkciju i ona će da vrati dužinu te sekvence. Dužina sekvence je zapravo broj elemenata u sekvenci. Pošto je u promenjivoj poruka string, elementi stringa su pojedinačni karakteri i dužina stringa je broj karaktera u stringu (broji se svaki znak uključujući prazno mesto ili znaci interpunkcije). U ovom slučaju to je 10 znakova.

007 Korišćenje in operatora

Sledeći deo programa proverava da li se najčešće korišćeno slovo u srpskom jeziku nalazi u promenjivoj poruka:

```
if "a" in poruka:
```

```
    print("nalazi se u tvojoj poruci.")
else:
    print("ne nalazi se u tvojoj poruci.")
```

Uslov u if iskazu je `"a" in poruka` . Ualov je tačan ako se znak a nalazi u promenjivoj poruka. Ako je neki element u sekvenci, kaže se da je član sekvence.

Može se koristiti in bilo gde za proveru da li je neki element član sekvence.

Indeksiranje stringova

Upotrebom petlje for, može se proći kroz string, znak po znak u redosledu pojavljivanja. Ovo se naziva sekvencionalni pristup, što znači da se prolazi kroz sekvencu element po element. Sekvencionalni pristup je u redu sem u slučajevima kada je potreban element koji se nalazi na kraju dugačke sekvence. Kada se direktno pristupa željenom elementu u sekvenci, to se naziva slučajan (random) pristup. On omogućava dobijanje bilo kog elementa u sekvenci direktno. Da bi se to omogućilo, neophodno je postojanje indeksiranja. Preko indeksiranja, određuje se pozicija (indeks) elementa u sekvenci i omogućava pristup tom elementu na toj poziciji.

program Direktan_pristup.py

```
#Direktan_pristup
#Prikazuje indeksiranje stringova
import random
rec = "indeks"
print("Rec je: ", rec, "\n")
visoko = len(rec)
nisko = -len(rec)
for i in range(10):
    pozicija = random.randrange(nisko, visoko)
    print("rec[", pozicija, "]\t", rec[pozicija])
```

```
Rec je:  indeks

rec[ 1 ]        n
rec[ 2 ]        d
rec[ -6 ]       i
rec[ 2 ]        d
rec[ 4 ]        k
rec[ -2 ]       k
rec[ -3 ]       e
rec[ 5 ]        s
rec[ -5 ]       n
rec[ -3 ]       e
```

008 Pozicije sa pozitivnim brojevima

Čim se promenjivoj dodeli string: `rec = "indeks"` odmah se kreira sekvenca gde svaki znak ima numerisanu poziciju. Prvo slovo “i”, je na poziciji 0. Drugo slovo “n” je na poziciji 1, itd.

Pristup pojedinačnom znaku u stringu je jednostavno. Za pristup slovu na poziciji 0 u promenjivoj rec, samo se koristi: rec[0].

Zato print(rec[0]) daje na ekranu slovo i. Dok print(rec[5]) daje na ekranu slovo s. Zato, dužina stringa daje istovremeno i indeks poslednjeg znaka u stringu, umanjenog za 1.

009 Pozicije sa negativnim brojevima

Postoji način pristupa elementima u sekvenci korišćenjem pozicija sa negativnim brojevima. Kod pozicija sa pozitivnim brojevima, pozicije započinju od prve pozicije u sekvenci koja se označava sa 0. Za stringove to znači da se započinje odbrojavanje od prvog slova. Kod pozicija

sa negativnim brojevima, početak brojanja je od kraja stringa. Za stringove to znači, da se započinje brojanje od poslednjeg znaka u stringu i da se ide prema početku stringa.
Zato print(rec[-1]) daje na ekranu slovo s. Dok print(rec[-2]) daje na ekranu slovo k. Zato, se sa indeksom -6, označava prvi znak u stringu indeks.

010 Pristup slučajno izabranom elementu stringa

Za pristup slučajno izabranom slovu iz stringa "indeks", treba generisati slučajne brojeve. To se izvodi linijom: import random.
Zatim, traži se način za dobar izbor broja koji će predstavljati poziciju u promenjivoj rec, negativnu ili pozitivnu. Program treba da je sposoban da generiše slučajan broj između -6 i 5, uključujući i njih. Tome služi funkcija random.randrange(), koja može da uzme dva broja i da proizvede slučajan broj između njih:

```
visoko = len(rec)
nisko = -len(rec)
```

Promenjiva visoko dobija vrednost 6, posto string "indeks" ima 6 slova u sebi. Promenjiva nisko dobija negativnu vrednost dužine stringa u promenjivoj rec, a to je -6. To prestavlja opseg iz kojeg će se generisati slučajan broj (zapravo između -6 i 5).
Format funkcije:
pozicija = random.randrange(nisko, visoko)
može da generiše brojeve: -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5 (jer 6 je samo ograničavač).
Na kraju, postoji for petlja koja se izvršava tačno 10 puta. U telu petlje, program generiše slučajan broj za poziciju vrednosti i štampa tu poziciju i odgovarajući element u sekvenci:

```
for i in range(10):
    pozicija = random.randrange(nisko, visoko)
    print("rec[", pozicija, "]\t", rec[pozicija])
```

011 Razumevanje nepromenjivosti stringova

Sekvence se dele na promenjive (mutable) i nepromenjive (immutable). Stringovi su nepromenjive sekvence. Zato, string "Kraj igre!" se ne može promeniti jednom kada se kreira.

```
>>> ime = "Miki"
>>> print(ime)
Miki
>>> ime = "Jova"
>>> print(ime)
Jova
```

Ovaj primer možda pokazuje suprotno, jer promenjiva ime je dobila dva različita stringa, dakle, srtingovi su se promenili. Zar ne ?!
Ne, na ovaj način se nijedan od stringova nije promenio. Kreirana su dva različita stringa. Prvo je kreiran string "Miki" i on je dodeljen promenjivoj ime. Zatim je kreiran string "Jova" pa je on dodeljen promenjivoj ime. Na kraju ime ukazuje samo na poslednji dodeljen string (Jova).
Posledica nepromenjivosti stringa je što se stringu ne može dodeliti novi karakter kroz indeksiranje:

```
>>> rec = "igrica"
>>> rec[0] = "1"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Prikazan je pokušaj da se promeni prvi karakter u stringu ali je prikazana greška, interpreter kaže da objekat string ne podržava dodelu nove vrednosti u stringu.
Ipak nepromenjivost stringa ne sprečava da se kreiraju novi stringovi na osnovu starog stringa.

Kreiranje novog stringa

program Bez_samoglasnika.py

```
#Bez_samoglasnika
#Prikazuje kreiranje novog stringa sa for petljom
poruka = input("Unesi poruku: ")
nova_poruka = ""
SAMOGLASNICI = "aeiou"
print()
for slovo in poruka:
    if slovo.lower() not in SAMOGLASNICI:
        nova_poruka += slovo
        print("Kreiran je novi string:", nova_poruka)
print("\nTvoja poruka bez samoglasnika je:", nova_poruka)
```

```
Unesi poruku: Ja sam dobar decko

Kreiran je novi string: J
Kreiran je novi string: J
Kreiran je novi string: J s
Kreiran je novi string: J sm
Kreiran je novi string: J sm
Kreiran je novi string: J sm d
Kreiran je novi string: J sm db
Kreiran je novi string: J sm dbr
Kreiran je novi string: J sm dbr
Kreiran je novi string: J sm dbr d
Kreiran je novi string: J sm dbr dc
Kreiran je novi string: J sm dbr dck

Tvoja poruka bez samoglasnika je: J sm dbr dck
```

012 Konstante

Sa linijom : `SAMOGLASNICI = "aeiou"`
promenjiva SAMOGLASNICI dobija string “aeiou“, sastavljen samo od samoglasnika. Naziv promenjive je posebno ispisan, sva slova su velika. Ovo je posebna promenjiva kada su sva slova velika, i zove se konstanta. Reč konstanta se odnosi na vrednost koju promenjiva dobija, a to znači da se očekuje da tokom rada programa njena vrednost se neće promeniti (konstantna vrednost).
Konstante su značajne za programere iz dva razloga: čine kod jasnijim i manje se vrednosti u kodu menjaju.

> I pored nepisanog pravila da se promenjiiva koja ukazuje na konstantnu vrednost piše svim velikim slovima, to ipak nije sintaksno pravilo. Zato treba biti pažljiv i proveriti da li se ova konvencija zaista i pridržavala u određenom kodu.

013 Kreiranje novog stringa od postojećeg stringa

Pravi deo posla u kodu počinje kreiranjem petlje. Program kreira novu poruku, bez samoglasnika dok se petlja izvršava. Ipak svaki put, kopjuter proverava sledeće slovo u originalnoj poruci. Ako to nije samoglasnik, dodaje ga novoj poruci koju kreira. Ako jeste samoglasnik, program se prebacuje na sledeće slovo. Pošto je poznato da program ne može zaista da doda karakter u string, zato preciznije rečeno, kada program pronađe znak koji nije samoglasnik, on pridodaje taj karakter novoj poruci koju ima do sada da bi kreirao novi string.

Deo koda koji to radi:

```
for slovo in poruka:
    if slovo.lower() not in SAMOGLASNICI:
        nova_poruka += slovo
        print("Kreiran je novi string:", nova_poruka)
```

Postoje dve nove ideju u petlji.

Prva, Pajton je izbirljiv kada radi sa stringovima i karakterima. “A“ nije isto što i “a“. Pošto je promenjivoj SAMOGLASNICI dodeljen string koji sadrži samo mala slova, potrebno je osigurati da se samo mala slova proveravaju pri korišćenju in operatora. Zato se i koristi `slovo.lower()`.

> Često je neophodno konverovati stringove na istu veličinu slova jer se dopušta korisniku da samo unese željenu string vrednost bez obzira na veličinu slova. Problem iste veličine slova se ostavlja da reši programer konverzijom stringa u kodu.

Primer: treba uporediti dva stringa, ime i pobednik, da bi se proverilo da li su identični, i nije bitno kakve su veličine slova. Može se kreirati uslov:

ime.lower() == pobednik.lower()

Uslov je tačan ako obe promenjive imaju stringove sa svim istim karakterima bez obzira na veličinu slova. To znači da “Maki“ == “maki“, “MAKI“ == “maki“, “MaKi“ == “mAkA“.

Druga, korišćenje proširenih operatora dodele, +=, za pridruživanje stringova. Dakle, umesto `nova_poruka = nova_poruka + slovo`, koristi se: `nova_poruka += slovo`.

Odsecanje stringova

Indeksiranje je korisna tehnika, ali programer nije ograničen za kopiranje samo jednog po jednog elementa iz sekvence. Mogu se napraviti kopije kontinualne sekcije elemenata (zovu se odsečci (slices)). Mogu se kopirati (ili odseći) jedan element (poput indeksiranja) ili dela sekvence (npr, srednja tri elementa u sekvenci). Čak se može kreirati i odsečak koji je kopija cele sekvence. Tako, za stringove, to znači da se može odseći bilo šta, od jednog karaktera, preko grupe karaktera, do celog stringa.

program Odsecanje_pice.py

```
#Odsecanje_pice
#Prikazuje odsecanje stringa
rec = "pica"
print(
"""
Odsecanje 'Tabela Varanja'
0   1   2   3   4
+---+---+---+---+
| p | i | c | a |
+---+---+---+---+
-4 -3  -2  -1
""")
print("Uneti pocetni i zavrsni indeks za tvoj odsecak 'pice'.")
print("Pritisni ENTER dugme kod 'Start' za izlaz.")
start = None
while start != "":
    start = (input("\nStart: "))
    if start:
        start = int(start)
        kraj = int(input("Kraj: "))
        print("rec[", start, ":", kraj, "] je", end=" ")
        print(rec[start:kraj])
```

```
Odsecanje 'Tabela Varanja'
0   1   2   3   4
+---+---+---+---+
| p | i | c | a |
+---+---+---+---+
-4 -3  -2  -1

Uneti pocetni i zavrsni indeks za tvoj odsecak 'pice'.
Pritisni ENTER dugme kod 'Start' za izlaz.

Start: 1
Kraj: 3
rec[ 1 : 3 ] je ic

Start: -2
Kraj: -4
rec[ -2 : -4 ] je

Start: -4
Kraj: -2
rec[ -4 : -2 ] je pi

Start:
Press any key to continue . . .
```

014 Upotreba None

Nova ideja u liniji: `start = None`

Ovde se pojavljuje nova vrednost, koja se zove None (nijedan, niko, ništa). None je Pajtonov način za predstavljanje ničega. None čini dobar način prihvatanja vrednosti. Takođe ono procenjuje na False kada se koristi kao uslov. Ovde se koristi za inicijalizaciju promenjive start za korišćenje u uslovu petllje while.

015 Objašnjenje odsecanja

Kreiranje odsecanja je slično indeksiranju. Ali umesto korišćenja jednog broja kao pozicije, ovde se daju startna i krajnja pozicija. Svaki element između dve tačke postaje deo odsečka. U odsecanju se može koristiti i kombinacija pozitivnih i negativnih brojeva pozicija.

```
>>> rec = "pica"
>>> print(rec[0:4])
pica
>>> print(rec[1:3])
ic
>>> print(rec[-4:-2])
pi
>>> print(rec[-4:3])
pic
```

Ako se pokuša dobiti nemoguć odsečak, kada je start pozicija manja od kraj pozicije, kao rec[2:1], neće se prekinuti program niti pojaviti poruka o grešci. Umesto toga, Pajton će samo vratiti praznu sekvencu, tj prazan string.

016 Kreiranje odsečaka

Unutar programa, štampa se sintaksa za kreiranje odsečka bazirano na početnoj i krajnjoj poziciji koju korisnik unosi, preko linije:

```
print("rec[", start, ":", kraj, "] je", end=" ")
```

a zatim se taj odesečak, tj string i štampa: `print(rec[start:kraj])`

017 Kraća verzija odsecanja

Iako se može dobiti bilo kakav odsečak pomoću dva broja, postoji nekoliko kraćih načina dobijanja odsečaka. Ako se izostavi početna pozicija, smatra se da je početna pozicija prvi element u sekvenci (prvi znak u stringu). Ako se izostavi krajnja pozicija, smatra se da je krajnja pozicija poslednji element u sekvenci (poslednji znak u stringu). Ako se oba broja izostave, smatra se da je odsečak zapravo cela sekvenca.

```
>>> rec = "pica"
>>> print(rec[0:4])
pica
>>> print(rec[0:])
pica
>>> print(rec[2:5])
ca
>>> print(rec[2:4])
ca
>>> print(rec[:])
pica
```

018 Kreiranje ntorki

ntorke (tuples) su tip sekvenci, poput stringova. Ali za razliku od stringova, koji mogu sadržavati samo znake, ntorke mogu sadržavati elemente bilo kojih tipova. U ntorke se mogu smastiti stringovi, brojevi, grafičke slike, zvučni fajlovi. Bilo šta što se može dodeliti promenjivoj, može se grupisati i smestiti kao sekvenca u ntorci.

Program Inventar_heroja.py

Program služi za održavanje inventara lika heroja u klasičnim RPG (role-playing game) igrama.

```
#Inventar_heroja
#Prikazuje kreiranje ntorki
#kreira praznu ntorku
inventar = ()
#koristiti ntorku kao uslov
if not inventar:
    print("Tvoj inventar je prazan.")
#kriranje ntorke sa nekoliko stvari
inventar = ("mac", "oklop", "stit", "napitak zdravlja")
#stampaj ntorku
print("\nU inventaru je:")
print(inventar)
#stampaj svaki element u ntorci
print("\nTvoje stvari:")
for stvar in inventar:
    print(stvar)
```

```
Tvoj inventar je prazan.

U inventaru je:
('mac', 'oklop', 'stit', 'napitak zdravlja')

Tvoje stvari:
mac
oklop
stit
napitak zdravlja
```

019 Kreiranje prazne ntorke

Za kreiranje ntorke, koristi se sekvenca vrednosti, odvojeni zarezima, unutar zagrada. Prazna ntorka ima sve sem vrednosti u zagradi: inventar = ()

020 Kreiranje ntorke kao uslova

U linijama se ntorka koristi kao uslov:

```
if not inventar:
    print("Tvoj inventar je prazan.")
```

Kao uslov, prazna ntorka je False. Ntorka sa najmanje jednim elementom je True. Program ispisuje tekst iz primera pošto je uslov `not inventar` True.

021 Kreiranje ntroki sa elementima

U programu:

```
inventar = ("mac", "oklop", "stit", "napitak zdravlja")
```

Prvi element ntorke je “mac” a poslednji je “napitak zdravlja”.

022 Štampanje ntorke

Iako ntorka može sadržavati više elemenata, može se odštampati cela ntorka kao da je bilo koja pojedinačna vrednost:

```
print("\nU inventaru je:")
print(inventar)
```

Kopjuter prikazuje sve elemente, okružene zagradama.

023 Prolazak kroz elemente ntorke sa petljom

U primeru:

```
print("\nTvoje stvari:")
for stvar in inventar:
    print(stvar)
```

Petlja štampa svaki element (string) u inventaru na drugoj liniji. Ova petlja se može koristiti za prolazak petljom kroz elemente bilo koje sekvence.

Iako je ovo ntorka gde su svi elementi istog tipa (stringovi), ntorke ne moraju biti sa vrednostima istog tipa.

Upotreba ntorki

Pošto su ntorke samo još jedna vrsta sekvenci, sve što važi za stringove važi i za ntorke. Može se dobiti dužina ntorke, odštampati svaki element preko for petlje, koristiti in operator, itd.

program Inventar_heroja2.py

```
#Inventar_heroja2
#Prikazuje rad sa ntorkama
inventar = ("mac", "oklop", "stit", "napitak zdravlja")
print("Tvoje stvari:")
for stvar in inventar:
    print(stvar)

input("\nPritisni ENTER za nastavak.")
print("Imas", len(inventar), "stvari u inventaru.")
input("\nPritisni ENTER za nastavak.")
if "napitak zdravlja" in inventar:
    print("Zivot se nastavlja za jos borbi.")
indeks = int(input("\nUnesi broj indeksa za stvar u inventaru: "))
print("Na indeksu", indeks, "je", inventar[indeks])
start = int(input("\nUnesi broj indeksa za pocetak odsecka: "))
kraj = int(input("Unesi broj indeksa za kraj odsecka: "))
print("inventar[", start, ":", kraj, "] je", end=" ")
print(inventar[start:kraj])
input("\nPritisni ENTER za nastavak.")
kovceg = ("zlato", "dragulji")
print("Nasao si kovceg. U kovcegu je:")
print(kovceg)
print("Dodao si sadrzaj kovcega u tvoj inventar.")
inventar += kovceg
```

```
print("Tvoj inventar trenutno sadrzi:")
print(inventar)
Tvoje stvari:
mac
oklop
stit
napitak zdravlja

Pritisni ENTER za nastavak.
Imas 4 stvari u inventaru.

Pritisni ENTER za nastavak.
Zivot se nastavlja za jos borbi.

Unesi broj indeksa za stvar u inventaru: 3
Na indeksu 3 je napitak zdravlja

Unesi broj indeksa za pocetak odsecka: 1
Unesi broj indeksa za kraj odsecka: 2
inventar[ 1 : 2 ] je ('oklop',)

Pritisni ENTER za nastavak.
Nasao si kovceg. U kovcegu je:
('zlato', 'dragulji')
Dodao si sadrzaj kovcega u tvoj inventar.
Tvoj inventar trenutno sadrzi:
('mac', 'oklop', 'stit', 'napitak zdravlja', 'zlato', 'dragulji')
```

024 Funkcija len sa ntorkama

Funkcija len radi sa ntorkama kao i sa stringovima. Ako je potrebno znati dužinu ntorke, ona se smešta unutar zagrada. Funkcija vraća broj elemenata ntorke. Prazna ntorka ili bilo koja prazna sekvenca ima dužinu 0.

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

025 in operator sa ntorkama

Kao i sa stringovima, in operator se koristi sa ntorkama za testiranje postojanja elementa. I ovde se in operator koristi za kreiranje uslova.

```
if "napitak zdravlja" in inventar:
    print("Zivot se nastavlja za jos borbi.")
```

Uslov `"napitak zdravlja" in inventar` testira ako ceo string `"napitak zdravlja"` je element u inventaru.

026 Indeksiranje ntorki

Indeksiranje ntorki je identično s indeksiranjem stringova. Specificira se broj pozicije u zagradama, za pristup određenom elementu.

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

027 Odsecanje ntorki

Odsecanje se odvija kao kod stringova. Postoje početna i krajnja pozicija. Rezultat je ntorka koja sadrži svaki element između te dve pozicije.

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

028 Nepromenjivost ntorki

Poput stringova i ntorke su nepromenjive. To znači da se ntorka ne može promeniti. Ali kao i sa stringovima, mogu se kreirati nove ntorke od postojećih.

029 Nadovezivanje ntorki

Nadovezivanje ntorki se izvodi na isti način kao i stringovi. Jednostavno se udružuju sa + operatorom:

```
kovceg = ("zlato", "dragulji")
print("Nasao si kovceg. U kovcegu je:")
print(kovceg)
print("Dodao si sadrzaj kovcega u tvoj inventar.")
inventar += kovceg
print("Tvoj inventar trenutno sadrzi:")
print(inventar)
```

Prvo se kreira nova ntorka, kovceg, sa dva string elementa, “zlato“ i “dragulji“. Posle štampanja sadržaja ntorke, koristi se proširени operator dodele += za nadovezivanje inventora sa kovčegom i dodelom rezultata nazad u inventar. Nije modifikovana originalna ntorka dodeljen inventaru (pošto je to nemoguće, pošto su ntorke nepromenjive). Umesto toga, prošireni operator dodele je kreirao novu ntorku sa elementima iz inventara i kovčega i dodelio to inventaru.

program Izmesane_reci.py

```
#Izmesane_reci
#Kompjuter slucajnim izborom bira rec a zatim je promesa
#igrac pogadja originalnu rec
import random
RECI = ("python", "jumble", "easy", "difficult", "answer", "xylophone")
rec = random.choice(RECI)
tacna_rec = rec
izmesana =""
while rec:
    pozicija = random.randrange(len(rec))
    izmesana += rec[pozicija]
    rec = rec[:pozicija] + rec[(pozicija + 1):]

print(
"""
           Dobrodosli u Ismesane Reci!

        Pogodi rec od ispreturanih slova.
          (Pritisni ENTER za kraj igre.)
""")
print("Izmesana slova:", izmesana)
pokusaj = input("\nTvoj pokusaj: ")
while pokusaj != tacna_rec and pokusaj != "":
    print("Izvini, ali to nije tacno.")
    pokusaj = input("Tvoj pokusaj: ")

if pokusaj == tacna_rec:
    print("To je to! Pogodak!\n")

print("Hvala na igranju.")
```

```
          Dobrodosli u Ismesane Reci!

        Pogodi rec od ispreturanih slova.
          (Pritisni ENTER za kraj igre.)

Izmesana slova: pynxeloho

Tvoj pokusaj: ne znam
Izvini, ali to nije tacno.
Tvoj pokusaj: a da li je
Izvini, ali to nije tacno.
Tvoj pokusaj: a ovo
Izvini, ali to nije tacno.
Tvoj pokusaj: onda je
Izvini, ali to nije tacno.
Tvoj pokusaj: xylophone
To je to! Pogodak!

Hvala na igranju.
```

030 Postavljanje osnovnih parametara programa

Prvo se importuje random modul.

Zatim se koristi ntorka za kreiranje sekvence reči. Promenjiva se naziva RECI, čime ukazuje da će se u programu korisitit kao konstanta.

```
RECI = ("python", "jumble", "easy", "difficult", "answer", "xylophone")
```

Zatim se koristi nova funkcija, random.choice(), koja dohvata slučajno izabranu reč iz ntorke RECI: `rec = random.choice(RECI)`

Sa njome kompjuter gleda u sekvencu i bira slučajnim izborom jedan element iz sekvence. Kada kompjuter izabere jedan element, dodeljuje ga promenjivoj rec. To je reč koju igrač treba da pogodi. Na kraju se promenjivoj tacna_rec dodeljuje vrednost iz promenjive rec, da bi se sačuvala tačna reč u memoriji.

031 Planiranje programa

Sledeći deo programa koristi nove koncepte. To je deo koji zapravo pravi izmešanost u reči u odnosu na originalnu reč.

Pre kodiranja izvršeno je planiranje programa u pseudokodu:

Kreirati praznu rec

Dok izabrana reč ima slova u njoj

    Izvući slučajno izabrana slova iz izabrane reči

    Dodati slučajna slova u izmešanu reč

Ova priprema je u redu ali može biti problema sa semantikom. Pošto su stringovi nepromenjivi, ne može se zaista izvući slučajno slovo iz stringa koji je korisnik uneo. Ali, može se kreirati novi string koji ne sadrži slučajno generisano izabrano slovo. I dok se ne može dodati slučajno slovo u izmešanu string, može se kreirati nov string nadovezivanjem trenutne iznešane reči sa izvučenim slovom.

032 Kreiranje praznog izmešanog stringa

Prazan string se kreiara sa: `izmesana =""`

Ova promenjiva će upućivati na krajnju izmešanu reč.

033 Postavljanje petlje

Proces kreiranja izmešane reči se kontroliše sa while petljom: while rec:

Postavljena je petlja da bi se nastavila sve dok je rec jednako sa praznim stringom. Ovo je korisno, pošto svaki put kada se petlja izvrši, kompjuter kreira novu verziju promenjive rec sa

još jednim izbačenim slovom i ponovo ju dodeljuje promenjivoj rec. Na kraju krajeva, rec će postati prazan string i mešanje će se završiti.

034 Generisanje slučajne pozicije u reči

Prva linija u telu petlje generiše slučajnu poziciju u reči, baziranu na njenoj dužini:

```
pozicija = random.randrange(len(rec))
```

Tako, slovo rec[pozicija] je ono koje će biti izbačeno iz rec i dodato u promenjivu izmesana.

035 Kreiranje nove verzije od izmešana

Sledeća linija u petlji kreira novu verziju stringa izmesana. Postaje jednaka svojm starom obliku, plus slovo rec[pozicija].

```
izmesana += rec[pozicija]
```

036 Kreiranje nove verzije od rec

Sledeća linija u petlji:

```
rec = rec[:pozicija] + rec[(pozicija + 1):]
```

kreira novu verziju rec minus jedno slovo na poziciji pozicija. Korišćenjem odsecanja, kompjuter kreira dva nova stringa iz rec. Prvi isečak, rec[:pozicija], je svako slovo do pozicije pozicija. Sledeći odsečak, rec[(pozicija+1):] je svako slovo posle rec[pozicija]. Ova dva stringa su udruženi i dedeljeni u rec, koja je sada jednaka sva slova sem slovo na rec[pozicija].

037 Uzimanje igačevog pokušaja pogađanja

Kompjuter nastavlja sa pitanjima za pokušaj pogađanja dok igrač ne da tačnu reč ili ne pritisne ENTER:

```
pokusaj = input("\nTvoj pokusaj: ")
while pokusaj != tacna_rec and pokusaj != "":
    print("Izvini, ali to nije tacno.")
    pokusaj = input("Tvoj pokusaj: ")
```