Kurs: Programiranje igara sa programskim jezikom Pajton
002 čas: Program Beskorisni_podaci.py
Teme: tipovi podataka, promenjive, stringovi, funkcija ulaza, matematičke operacije

Kod za program Beskorisni_podaci.py

```
#Beskorisni_podaci
#Uzima licne podatke korisnika i stampa tacne ali beskorisne informacije o korisniku
ime = input("Zdravo. Kako se zoves? ")
godine = input("Koliko imas godina? ")
godine = int(godine)
tezina = int(input("Dobro, poslednje pitanje.
                   Koliko imas kilograma? "))

print("\nAko bi fini gospodin poslao tebi e-mail, obratio bi se tebi sa",
      ime.lower())
print("Ali ako bi gospodin bio ljut, nazvao bi te", ime.upper())

pozvan = ime * 5
print("\nAko bi dete trazilo tvoju paznju",)
print("tvoje ime bi postalo:")
print(pozvan)
sekunde = godine * 365 * 24 * 60 * 60
print("\nTi si preko", sekunde, "sekundi mator.")
mesec_tezina = tezina / 6
print("\nDa li znas da bi na Mesecu imao tezinu od",
      mesec_tezina, "kilograma?")
sunce_tezina = tezina * 27.1
print("Na Suncu, tvoja tezina bi bila", sunce_tezina, "(ali, pa... ne zadugo).")
```

017 021 018 004 013 001 007

Upotreba navodnika na stringovima

program Kraj_igre2.py

Sledeći kod je upgade verzija prethodnog koda Kraj_igre, i novi kod ima impresivniju verziju poruke koja upozorava igrača da je igri došao kraj:

```
#Kraj_igre2
#Prikazuje koriscenje navodnika u stringovima

print("Program 'Kraj_igre_2'")
print("Ista", "poruka", "kao i pre.")
print("Samo",
      "malo",
      "veca")
print("Evo", end=" ")
print("je...")
print("""
 _   __  _____       ___      _
| | / / |  _  \     /   |    | |
| |/ /  | |_| |    / /| |    | |
|   |   |  _  /   / ___ |    | |
| |\ \  | | \ \  / /  | |  __/ /
|_| \_\ |_|  \_\/_/   |_| |____/

 _   _____   _____   _____
| | /  ___| |  _  \ |  ___|
| | | |     | |_| | | |__
| | | |  _  |  _  / |  __|
| | | |_| | | | \ \ | |___
|_| \_____/ |_|  \_\|_____|
      """)
```

```
Program 'Kraj_igre_2'
Ista poruka kao i pre.
Samo malo veca
Evo je...

 _   __  _____       ___      _
| | / / |  _  \     /   |    | |
| |/ /  | |_| |    / /| |    | |
|   |   |  _  /   / ___ |    | |
| |\ \  | | \ \  / /  | |  __/ /
|_| \_\ |_|  \_\/_/   |_| |____/

 _   _____   _____   _____
| | /  ___| |  _  \ |  ___|
| | | |     | |_| | | |__
| | | |  _  |  _  / |  __|
| | | |_| | | | \ \ | |___
|_| \_____/ |_|  \_\|_____|

Press any key to continue . . .
```

Jednostavni stringovi se prave tako što se tekst ograniči navodnicima sa obe strane. Može se koristiti ili par apostrofa (’ ’) ili par navodnika (“ “).
U stringu: `"Program 'Kraj_igre_2'"` se koriste i apostrofi i navodnici, ali na izlazu se pojavljuju samo apostrofi pošto su oni deo stringa. Navodnici nisu deo stringa, oni su ograničavači, pokazuju računaru odakle dokle je string. Zato, ako se navodnici koriste kao ograničavači, mogu se apostrofi koristiti kao deo stringa.
Kada se jedan tip navođenja koristi za ograničavanje stringa taj isti tip se ne može koristiti i kao deo stringa.

001 Štampanje više vrednosti istovremeno

Moguće je štampati više vrednosti istovremeno pozivom samo jedne funkcije print, ako su argumenti odvojeni međusobno zarezima: `print("Ista", "poruka", "kao i pre.")`
Ovde se dostavljaju tri argumenta funkciji: `"Ista"`, `"poruka"` i `"kao i pre."` , što realizuje pojavu teksta na izlazu: `Ista poruka kao i pre.` Svaka vrednost je odštampana sa praznim mestom kao separatorom. Ovo je difolt ponašanje funkcije print.
Kada postoji lista argumenata, može se startovati nova linija posle bilo kojeg od zarez separatora u listi. Sledeće tri linije koda formiraju jedan iskaz koji štampa jednu liniju teksta `Samo malo veca`. Posle svakog zarez separatora startuje nova linija u kodu:

```
print("Samo",
      "malo",
      "veca")
```

Ponekad je korisno rastaviti listu argumenata na više linija zbog lakšeg čitanja koda.

002 Formatiranje stringa

Po difoltu, funkcija print štampa karakter nove linije kao svoju poslednju vrednost. To znači da bi poziv sledeće funkcije print prikazao tekst na toj novoj liniji. Najčešće, to je ono što se i želi postići ali bolje je ako se može samostalno specificirati kako će izgledati string. Npr, može se specificirati da će prazno mesto biti poslednji karakter koji će se odštampati (umesto karaktera nove linije) pri pozivu print funkcije. To bi značilo da bi poziv sledeće funkcije print odštampao vrednost posle praznog mesta prethodne funkcije print. Upravo se to događa u sledećem slučaju:

```
print("Evo", end=" ")
print("je...")
```

Ovaj kod štampa tekst “Evo je…” u jednoj liniji. To je zato što u prvoj funkciji print je specificirano prazno mesto kao poslednji string koji će se odštampati. Zato, iskaz štampa tekst “Evo ” (uključujući prazno mesto posle slova o) ali ne i karakter nove linije. Sledeći iskaz print počinje štampanje teksta “je…” odmah posle praznog mesta. Ovo je izvedeno dodavanjem praznog mesta u end parametar funkcije print (end = ” ”). Može se specificirati da bilo kakav string bude odštampan kao poslednja vrednost u print iskazu.

003 Stringa sa tri navodnika

Poslednja funkcija print u kodu štampa veliki deo teksta ograničavajući tekst sa tri navodnika. Ovo se naziva trostruko navedeni string (triple-quoted string). I ovde nije bitno da li se koriste apostrofi ili navodnici.
Trostruko navedeni stringovi se mogu prostirati preko većeg broja linija. Oni će se odštampati tačno onako kako su i otkucani.

ASCII umetnost su slike napravljene samo od znakova sa tastature. ASCII je skraćenica od American Standard Code for Information Interchange. To je kod koji predstavlja 128 standardnih karaktera. Istorijski ova vrsta umetnosti datira još od starih kucaćih mašina, od 1898. godine.

004 Eskejp sekvence sa stringovima

Eskejp sekvence omogućavaju smeštanje specijalnih karaktera u stringove. Oni daju veću kontrolu i fleksibilnost tekstu koji se prikazuje. Svaka eskejp sekvenca se sastoji od dva znaka: kose crte (\, backslash) i drugog karaktera.

Unos programa zahvalnice

Programi često imaju i zahvalnice svim ljudima koji su zaslužni za izradu programa.

```
#Zahvalnice
#Demonstrira koriscenje eskejp sekvenci
print("\t\t\tZahvalnice")
print("\t\t\t \\ \\ \\ \\ \\ \\ \\")
print("\t\t\t\tnapravio")
print("\t\t\tNajbolji Programer")
print("\t\t\t \\ \\ \\ \\ \\ \\ \\")
print("\nPosebno se zahvaljujem:")
print("Mom brici, Miletu \'Velikom\', koji nikad ne kaze \"ne\'moz.\"")
#zvuk sistemske zujalice
print("\a")
```

```
                    Zahvalnice
                     \ \ \ \ \ \ \
                            napravio
                    Najbolji Programer
                     \ \ \ \ \ \ \

Posebno se zahvaljujem:
Mom brici, Miletu 'Velikom', koji nikad ne kaze "ne'moz."

Press any key to continue . . .
```

005 Tabulatora

Ponekad je potrebno odvojiti tekst od leve margine gde se obično štampa. U tekst procesorima se koristi Tab dugme za isti posao. Sa stringovima, može se koristiti eskejp sekvenca za tabulator: \t. To se dešava u liniji:

```
print("\t\t\tZahvalnice")
```

Vidi se da se koriste tri tab eskejp sekvence. Tako da, kada program štampa string, on štampa tri tabulatora a zatim i tekst. Tabulatori se koriste za uređenje teksta u vidu tabela.

006 Štampanje kose crte (\ backslash)

Može se postaviti pitanje kako da se odštampa kosa crta (\ backslash) ako kompjuter vidi ovaj karakter kao početak eskejp sekvence. Rešenje je u tome da se kucaju dve kose crte: \\. Zato sledeća linija koda: `print("\t\t\t \\ \\ \\ \\ \\ \\ \\")` štampa tri taba pa sedam kosih crta (pošto su napisani sedam parova \\), odvojeni praznim mestima.

007 Ubacivanje nove linije

Sekvenca nove linije se obeležava \n. Upotrebom ove sekvence, ubacuje se karakter nove linije u string na mesto gde se želi preći u novu liniju na izlazu. Karakter nove linije se može koristiti na početku stringa da bi se odvojio od prethodno prikazanog teksta:

`print("\nPosebno se zahvaljujem:")`. Računar vidi \n sekvencu, prelazi u novi red, a zatim štampa tekst.

008 Ubacivanje navoda

Ako je potrebno u string ubaciti isti navodnik sa kojim se string i ograničava, mora se koristiti sekvenca \ (\' ili \"). Zato se u sledećoj liniji koda mogu koristiti obe vrste navodnika:

```
print("Mom brici, Miletu \'Velikom\', koji nikad ne kaze \"ne\'moz.\"")
```

Na izlazu će se desiti sledeće:

- `\'Velikom\'` se štampa kao `'Velikom'`
- Svaka sekvenca \' se štampa kao jedan apostrof
- `\"ne\'moz.\"` se štampa kao `"ne'moz."`
- Svaka \" sekvenca se štampa kao navodnik

Dok se program izvršava začuće se ton . Pošto linija: `print("\a")` uključuje sistemski zvuk. To se dešava zbog karaktera sistemskog zvona. To će se desiti samo ako se program startuje direktno iz operativnog sistema a ne iz okruženja.

Nadovezivanje i ponavljanje stringova

program Sasavi_stringovi.py

```
# Sasavi_stringovi
# Prikazuje nadovezivanje i ponavljanje stringova
print("Mogu se nadovezati dva " + "stringa sa '+' operatorom.")
print("\nOvaj string " + "mozda ne " + "izgleda stra" + "sno impresivno. " \
    + "Ali ono " + "sta se ne zna" + " je da\n" + "je to jedan zai" \
    + "s" + "ta" + " dug string, kreiran nadovezivanjem " \
    + "od " + "dvadeset dva\n" + "razlicita stringa, podeljenih na " \
    + "sest linija." + " Da li je neko" + " impresioniran? " + "Dobro,\n" \
    + "ovaj " + "zaista " + "dugacak" + " string je sada zavrsen!")
print("\nAko neko zaista voli string, moze da ga ponovi. Na primer,")
print("ko ne voli pitu? Naravno svi je vole. Ali ako je neko zaista")
print("voli, trebalo bi da to i kaze kao da to i misli:")
print("Pita" * 10)
```

```
Mogu se nadovezati dva stringa sa '+' operatorom.

Ovaj string mozda ne izgleda strasno impresivno. Ali ono sta se ne zna je da
je to jedan zaista dug string, kreiran nadovezivanjem od dvadeset dva
razlicita stringa, podeljenih na sest linija. Da li je neko impresioniran? Dobro,
ovaj zaista dugacak string je sada zavrsen!

Ako neko zaista voli string, moze da ga ponovi. Na primer,
ko ne voli pitu? Naravno svi je vole. Ali ako je neko zaista
voli, trebalo bi da to i kaze kao da to i misli:
PitaPitaPitaPitaPitaPitaPitaPitaPitaPita
Press any key to continue . . .
```

009 Nadovezivanje stringova

Nadovezivanje (concatenating) stringova označava njihovo pridruživanje čime se kreira nov string:

```
print("Mogu se nadovezati dva " + "stringa sa '+' operatorom.")
```

Operator + udružuje dva stringa: `"Mogu se nadovezati dva "` i `"stringa sa '+' operatorom."` čime oni formiraju novi, veći string.

Kada se udruže dva stringa, njihove tačne vrednosti su udružene, bez praznog mesta ili znaka separatora između njih (“dobar“ + “dan“ daje “dobardan“ a ne “dobar dan“).

010 Karaktera nastavka linije

Uopšteno, piše se jedna po jedna linija koda. Ali to nije obavezno. Može se raširiti jedan iskaz preko više linija. Za to se koristi karakter nastavka linije (\, line continuation character). Može se staviti bilo gde u string gde bi se inače stavilo prazno mesto. Kompjuter sve to vidi kao jednu dugačku liniju koda.

Računaru nije bitna dužina linije koda, ali ljudima jeste. Ako je linija isuviše duga ili programer oseća da bi bilo jasnije podeliti liniju na više linija, tada se koristi karakter nastavka linije.

011 Ponavljanje stringova

Ova ideja je predstavljena na sledeći način u programu:

```
print("Pita" * 10)
```

Ova linija kreira novi string: PitaPitaPitaPitaPitaPitaPitaPitaPitaPita i štampa ga. Ovo je naravno, string “Pita“ ponovljen 10 puta.

Poput operatora nadovezivanja, * operator ponavljanja (repetition operator) je dosta intuitivan. Isti simbol se koristi u računarima za množenje brojeva, pa je zato ponavljanje stringova sa njime logično. Potrebno je samo u zagradi navesti string i pomnožiti ga željeni broj puta.

Rad sa brojevima

Do sada su korišćeni stringovi za predstavljanje teksta. Ali string je samo jedan tip vrednosti. Jedan od osnovnih ali najvažniji način predstavljanja vrednosti su brojevi. Brojevi se koriste skoro u svakom programu. Od zapisivanja najboljih rezultata u igri do podataka u fininsijskom paketu, brojevi su nezamenjivi.

program Problem_sa_recima.py

```
#Problem_sa_recima
#Prikazuje brojeve i matematiku
print("Ako 2000 kg teska zenka nilskog konja rodi mladunce od 100 kg,")
print("a zatim pojede 50 kg hrane, koliku ce onda da ima tezinu?")
input("Pritisni bilo koje dugme da bi saznao.")
print("2000 - 100 + 50 =", 2000 - 100 + 50)

print("\nAko avanturista se vrati sa uspesne avanture i kupi svakom od")
print("6 pratilaca po 3 flase piva, koliko je ukupno flasa kupio?")
input("Pritisni bilo koje dugme da bi saznao.")
print("6 * 3 =", 6 * 3)

print("\nAko racun u restoranu iznosi 1900 dinara zajedno sa baksisem,")
print("a ti i tvoji prijatelji ga delite na 4 jednaka dela, koliki ce")
print("biti pojedinacni delovi za placanje?")
input("Pritisni bilo koje dugme da bi saznao.")
print("1900 / 4 =", 1900 / 4)

print("\nAko 4 pirata pronadju kovceg sa 107 zlatnika i oni podele")
print("blago podjednako medju sobom, koliko ce celih zlatnika svaki")
print("od njih dobiti?")
input("Pritisni bilo koje dugme da bi saznao.")
print("107 // 4 =", 107 // 4)

print("\nAko ta ista grupa pirata na jednake delove podeli kovceg")
print("sa 107 zlatnika, koliko zlatnika ce preostati?")
input("Pritisni bilo koje dugme da bi saznao.")
print("107 % 4 =", 107 % 4)
```

```
Ako 2000 kg teska zenka nilskog konja rodi mladunce od 100 kg,
a zatim pojede 50 kg hrane, koliku ce onda da ima tezinu?
Pritisni bilo koje dugme da bi saznao.
2000 - 100 + 50 = 1950

Ako avanturista se vrati sa uspesne avanture i kupi svakom od
6 pratilaca po 3 flase piva, koliko je ukupno flasa kupio?
Pritisni bilo koje dugme da bi saznao.
6 * 3 = 18

Ako racun u restoranu iznosi 1900 dinara zajedno sa baksisem,
a ti i tvoji prijatelji ga delite na 4 jednaka dela, koliki ce
biti pojedinacni delovi za placanje?
Pritisni bilo koje dugme da bi saznao.
1900 / 4 = 475.0

Ako 4 pirata pronadju kovceg sa 107 zlatnika i oni podele
blago podjednako medju sobom, koliko ce celih zlatnika svaki
od njih dobiti?
Pritisni bilo koje dugme da bi saznao.
107 // 4 = 26

Ako ta ista grupa pirata na jednake delove podeli kovceg
sa 107 zlatnika, koliko zlatnika ce preostati?
Pritisni bilo koje dugme da bi saznao.
107 % 4 = 3
Press any key to continue . . .
```

012 Tipovi brojeva

Program Problem_sa_recima koristi brojeve i to dva različita tipa brojeva. Programeri u Pajtonu mogu da koriste nekoliko različitih tipova brojeva. Najčešće korišćeni tipovi brojeva su baš ova dva koja se koriste u ovom programu: celobrojni (integers) i brojevi sa pomerajućom tačkom (floating-points numbers ili skraćeno floats). Integers cu brojevi bez decimalnog dela (nemaju decimalanu tačku) (1, 27, -100). Floats su brojevi sa decimalnom tačkom (2.376, -99.1, 1.0).

013 Matematički operatori

Upotrebom matematičkih operatora, računar postaje kalkulator. Npr, u 2000 – 100 + 50, 100 se oduzima od 2000 a zatim se dodaje 50 pre nego se odštampa vrednost 1950. Tehnički, određuje se (evaluate) izraz (expression) 2000 – 100 + 50, što daje rezultat od 1950. Izraz je samo niz vrednosti udruženih sa operatorima koji se može jednostavnije napisati sa drugom vrednosti.Kod 6 * 3 množi 6 sa 3 i štampa rezultat od 18. Kod 1900 / 4 deli 1900 sa 4 i štampa vrednost sa decimalnom tačkom od 475.0.

U kodu 107 // 4, koristi se operator // (two forward slashes). Kada se ovaj operator ovde koristi on predstavlja celobrojno deljenje, deljenje gde je rezultat uvek ceo broj (decimalni deo se ignoriše). Ovo je suprotno od operatora / (pravo deljenje) gde se rezultat u decimalnom obliku ne ignoriše. To znači da 107 // 4 daje 26.

U kodu 107 % 4, simbol % se koristi kao operator modula, koji daje rezultat ostatka od celobrojnog deljenja. Tako da 107 % 4 daje 3, što je ostatak deljenja 107 / 4.

U primeru, 7 / 3, rezultat je 2.3333333333333335. Iako je rezultat poprilično precizan, on nije potpuno tačan. Ovakav rezultat je koristan za najveći broj upotreba, ali nije 100% tačan. Ovo je čest problem kod rezultata dobijenih pravim deljenjem.

Promenjive

Korišćenjem promenjivih se mogu čuvati i manipulisati sa podacima, što je osnovni aspekt programiranja.

program Pozdrav.py

```
#Pozdrav
#Prikazuje koriscenje promenjivih
ime = "Marko"
print(ime)
print("Pozdrav,", ime)
```

```
Marko
Pozdrav, Marko
Press any key to continue . . .
```

014 Kreiranje promenjivih

Promenjive su način za označavanje i pristup informacijama. Umesto da je neophodno tačno znati gde u memoriji računara se nalazi neka informacija, koristi se promenjiva za pristup toj informaciji. Pre korišćenja promenjive, ona se mora kreirati: `ime = "Marko"`

Ova linija se naziva iskaz dodele (assignment statement). Ona kreira promenjivu sa imenom ime i dodeljuje joj vrednost. Kaže se da promenjiva ime ukazuje na (references) string "Marko". U principu, iskazi dodele dodeljuju vrednost promenjivoj. Ako promenjiva prethodno nije kreirana (kao u ovom slučaju), kreira se a zatim joj se i dodeljuje vrednost.

> Iskaz dodele smešta vrednost sa desne strane znaka jednakosti u memoriju računara dok ime promenjive sa leve strane znaka jednakosti samo ukazuje na vrednost (ne smešta direktno vrednost).

015 Upotreba promenjive

Kada se promenjiva jednom kreira, ona ukazuje na neku vrednost. Pogodnost za rad sa promenjivima je da se ona može koristiti kao da se koristi vrednost. Tako da linija:
print(ime), zapravo štampa string "Marko" kao da je napisano print("Marko").
Linija: `print("Pozdrav,", ime)` štampa string "Pozdrav", prazno mesto, pa "Marko". U ovom slučaju, može se koristiti promenjiva ime umesto vrednosti "Marko" da se dobije isti rezultat.

016 Imenovanje promenjive

Programer odlučuje kako će se zvati promenjiva. Za prethodni program programer je izabrao naziv – ime, ali je mogao i drugačije da nazove promenjivu: osoba, covek ili alfa123456 i program bi opet dobro radio. Postoji samo nekoliko pravila pri kreiranju imena promenjivih a najvažnija su:

1. Ime promenjive može sadržati samo cifre, slova i donje crte (_, underscore)
2. Ime promenjive ne može startovati sa cifrom

Osim ovih tvrdih pravila postoje i uputstva za kreiranje dobrih imena promenjivih:

- Treba izaberi opisno ime: imena promenjivih bi trebalo da su dovoljno jasna tako da drugi programer lako zaključi šta predstavljaju. Npr, bolje je rezultat nego r, score nego s; izuzetak je korišćenje promenjive koja se koristi samo u malom segmentu koda, tada je u redu nazvati je x
- Treba budi dosledan: postoje različite škole misli o tome kako pisati ime promenjive koje se sastoji od više reči; da li najbolji_rezultat ili najboljiRezultat bolje? Bitno je samo da se programer dosledno drži stila imenovanja promenjivih
- Pratiti tradiciju jezika: neke konvencije imenovanja su tradicionalne, npr u mnogim jezicima imena promenjivih počinju sa malim slovom. Takođe je tradicija da se ime

promenjive ne započinje sa underscore pošto takva imena imaju posebno značenje u Pajtonu

- Voditi računa o dužini imena: Ovo može da bude u suprotnosti sa uputstvom da se izabere opisno ime. Zar nije bolje dati ime kontrola_privatnog_bankovnog_racuna ? Možda ne. Dugačka imena vode u probleme jer čine kod teškim za čitanje, osim toga povećava se mogućnost greške u kucanju. Zato, treba zadržati ime promenjive ispod 15 karaktera.

Kod je samostalno dokumentovan ako je tako napisan da mu nisu potrebni nikakvi komentari. Pokušaj da se daju dobra imena promenjivih je korak prema samostalno dokumentovanom kodu.

Dobijanje korisnikovog ulaza

Svaki program na kraju krajeva na neki način komunicira sa korisnikom. Promenjive se koriste za uzimanje, smeštanje i manipulaciju korisnikovim ulaznim podacima.

program Licni_pozdrav.py

```
#Licni_pozdrav
#Prikazuje uzimanje korisnikovog ulaza
ime = input("Zdravo. Kako se zoves? ")
print(ime)
print("Pozdrav,", ime)
```

```
Zdravo. Kako se zoves? Marko
Marko
Pozdrav, Marko
```

017 Funkcija input

Linija: `ime = input("Zdravo. Kako se zoves? ")`

Na levoj strani jednakosti je promenjiva ime i njoj se dodeljuje vrednost sa desne strane jednakosti. Ali, na desnoj strani jednakosti je poziv funkciji input. Funkcija input dobija nekakav tekst od korisnika. Dobija string argument od korisnika preko uputstva korisniku. Sada je argument koji se dostavlja funkciji input string: `"Zdravo. Kako se zoves? "` Funkcija input koristi string da bi upozorila korisnika i čeka da korisnik nešto unese. Kada korisnik klikne na ENTER, funkcija input vraća ono šta je uneto u vidu stringa. Taj string, povratna vrednost poziva funkcije, je ono šta se smešta u promenjivu ime.

Isto tako je moguće napisati: `input("Zdravo. Kako se zoves? ")` . Ali sada se uneta vrednost ne dodeljuje nikakvoj promenjivoj pa će ta vrednost ostati nezapamćena od strane programa. Ovo ima smisla koristiti kada se samo očekuje od korisnika da potvrde nešto klikom na ENTER npr: `input("Pritisni bilo koje dugme za nastavak igre.")`

Korišćenje string metoda

Pajton ima veliki broj alata za rad sa stringovima. Jedna vrsta alata za rada sa stringovima su metode. String metode omogućavaju kreiranje novog stringa iz starog stringa.

Program Manipulacija_citatima.py

```
#Manipulacija_citatima
#Prikazuje string metode
#citat IBM Predsednika, Wotson Tomasa iz 1943. godine
citat = "Mislim da postoji svetsko trziste za mozda pet kompjutera."
print("Originalni citat:")
print(citat)
print("\nVelikim slovima:")
print(citat.upper())
print("\nMalim slovima:")
print(citat.lower())
```

```
print("\nKao naslov:")
print(citat.title())
print("\nSa sitnom izmenom:")
print(citat.replace("pet", "milione"))
print("\nOriginalni citat je jos uvek:")
print(citat)
```

```
Originalni citat:
Mislim da postoji svetsko trziste za mozda pet kompjutera.

Velikim slovima:
MISLIM DA POSTOJI SVETSKO TRZISTE ZA MOZDA PET KOMPJUTERA.

Malim slovima:
mislim da postoji svetsko trziste za mozda pet kompjutera.

Kao naslov:
Mislim Da Postoji Svetsko Trziste Za Mozda Pet Kompjutera.

Sa sitnom izmenom:
Mislim da postoji svetsko trziste za mozda milione kompjutera.

Originalni citat je jos uvek:
Mislim da postoji svetsko trziste za mozda pet kompjutera.
```

018 Kreiranje novog stringa sa string metodom

Linija: `print(citat.upper())`

radi sledeće: štampa verziju citata sa svim slovima kao velikim slovima. To se dešava kroz korišćenje string metode, upper(). String metoda je kao sposobnost koju string ima. Tako da, citat ima sposobnost kreiranja novog stringa, verziju sebe sa velim slovima, kroz metod upper(). Kada se to desi, on vraća novi string a linija postaje ekvivalentna sa:

```
print("MISLIM DA POSTOJI SVETSKO TRZISTE ZA MOZDA PET KOMPJUTERA.")
```

Metode su slične sa funkcijama (zato i metode imaju zagrade). Glavna razlika je što ugrađene (built-in) funkcije, poput input(), se mogu same pozvati. Ali string metode moraju da se pozovu kroz tačno određeni string. Zato nema smisla pisati: print(upper())

Metod se startuje ili priziva (invoke) pisanjem naziva stringa, dodavanjem tačke (dot), iza koje sledi ime metode i zagrada. Zagrade u metodi takođe mogu imati argumente kao i u funkcijama ali konkretno metoda upper() ne prihvata nikakve argumente.

Linija: `print(citat.lower())`

priziva metodu lower() stringa citat da bi kreirala verziju istog stringa sa malim slovima.

Linija: `print(citat.title())`

štampa verziju stringa citat koja liči na naslov. Metoda title() vraća string gde prva slova svake reči su velika a ostatak reči su malim slovima.

Linija: `print(citat.replace("pet", "milione"))`

štampa novi string, gde svako pojavljivanje stringa “pet“ u stringu citat se zamenjuje sa stringom “milione“.

Metod replace() traži najmanje još dve informacije: stari tekst koji će se zameniti i novi tekst sa kojim će se zameniti. Dva argumenta se odvajaju zarezom. Može se dodati i opcioni treći argument, ceo broj, koji kaže metodi koliki maksimalni broj puta da se izvrši zamena.

Linija: `print("\nOriginalni citat je jos uvek:")`

pokazuje da se originalni string nije promenio pošto string metode kreiraju novi string i ne utiču na originalno stanje stringa.

019 Dodatne string metode

Metoda swapcase() vraća novi string u kojem je zamenjena veličina svakog slova originalnog stringa.

Metoda capitalize() vraća novi string gde je samo prvo slovo originalnog stringa veliko a ostala su mala slova.

Metoda strip() vraća novi string gde su sve bela mesta (prazna mesta, tabovi, karakteri nove linije) na početku i na kraju stringa skinute iz stringa

Upotreba pravog tipa

Bitno je znati gde treba koristiti koji od tipova podataka (integer, float, string). Takođe je bitno znati kako raditi sa određenim tipovima.

program Cuvar_para_losa_verzija.py

Ideja programa je kreirati alat koji računa koliko se troši novca a koliko ih je na raspolaganju. Za potrebe učenja, program je namerno napravljen da ne daje tačne rezultate što će se viditi na izlazu programa.

```
#Cuvar_para_losa_verzija
#Prikazuje logicke greske
print("""
            Cuvar Para
Racuna tvoje mesecne izdatke tako da tvoj novac ne nestane
(i ti budes primoran da nadjes pravi posao).
Unesi svoje mesecne izdatke. Posto si bogat, ignorisi
sitan novac i koristi samo hiljade dinara.
""")
kola = input("Nadogradnja Lambordzinija: ")
stan = input("Apartman na Menhetnu: ")
avion = input("Najam privatnog aviona: ")
poklon = input("Pokloni: ")
hrana = input("Vecere u restoranima: ")
posluga = input("Tim posluge (batler, kuvar, sofer, pomocnik): ")
guru = input("Licni guru i trener: ")
igre = input("Kompjuterske igre: ")
ukupno = kola + stan + avion + poklon + hrana + posluga + guru + igre
print("\nUkupno:", ukupno)
```

```
            Cuvar Para
Racuna tvoje mesecne izdatke tako da tvoj novac ne nestane
(i ti budes primoran da nadjes pravi posao).
Unesi svoje mesecne izdatke. Posto si bogat, ignorisi
sitan novac i koristi samo hiljade dinara.

Nadogradnja Lambordzinija: 120
Apartman na Menhetnu: 550
Najam privatnog aviona: 450
Pokloni: 340
Vecere u restoranima: 230
Tim posluge (batler, kuvar, sofer, pomocnik): 450
Licni guru i trener: 567
Kompjuterske igre: 1080

Ukupno: 120550450340230450567671080
```

020 Pronalaženje logičkih grešaka

Jedna od najtežih grešaka za pronalaženje u kodovima su logičke greške. Pošto se program ne prekida, korisnik ima utisak da je sve u redu sa izlazom programa. Da bi se ovakve greške pronašle, potrebno je uočiti ponašanje programa.

U ovom slučaju, izlaz programa priča priču. Dugačak broj očigledno nije suma svih brojeva koje je korisnik uneo. Iako je promenjiva ukupno nastala dejstvom operatora + nad svim unetim brojevima. To se desilo zato što funkcija input() vraća string. To znači da svaki broj koji se unese se tretira kao string a ne kao brojčana vrednost. Zato linija:

```
ukupno = kola + stan + avion + poklon + hrana + posluga + guru + igre
```

ne sabira brojeve već nadovezuje stringove.

Sada, kada se zna šta je problem, kao ga rešiti? Nekako je potrebno pretvoriti (konvertovati) string vrednosti u brojeve pa će program raditi ono što je potrebno.

> Simbol + radi i sa stringovima i sa celim brojevima ono što je logično. Korišćenje istog operatora sa vrednostima različitih tipova se naziva preopterećenje operatora (operator overloading). Ako se dobro upotrebi proptерećenje operatora daje dobar kod.

Konverzija vrednosti

Rešenje programa Cuvar_para_losa_verzija.py je konverzija string vrednosti koje funkcija input() vraća, u brojčane vrednosti. Logično je da svaki uneti string odmah konvertuje u celobrojnu vrednost pre nego se bilo šta radi sa njime.

Program Cuvar_para.py

```
#Cuvar_para
#Prikazuje konverziju tipova podataka
print("""
            Cuvar Para
Racuna tvoje mesecne izdatke tako da tvoj novac ne nestane
(i ti budes primoran da nadjes pravi posao).
Unesi svoje mesecne izdatke. Posto si bogat, ignorisi
sitan novac i koristi samo hiljade dinara.
""")
kola = input("Nadogradnja Lambordzinija: ")
kola = int(kola)
stan = int(input("Apartman na Menhetnu: "))
avion = int(input("Najam privatnog aviona: "))
poklon = int(input("Pokloni: "))
hrana = int(input("Vecere u restoranima: "))
posluga = int(input("Tim posluge (batler, kuvar, sofer, pomocnik): "))
guru = int(input("Licni guru i trener: "))
igre = int(input("Kompjuterske igre: "))
ukupno = kola + stan + avion + poklon + hrana + posluga + guru + igre
print("\nUkupno:", ukupno, "hiljada dinara.")
```

```
           Cuvar Para
Racuna tvoje mesecne izdatke tako da tvoj novac ne nestane
(i ti budes primoran da nadjes pravi posao).
Unesi svoje mesecne izdatke. Posto si bogat, ignorisi
sitan novac i koristi samo hiljade dinara.

Nadogradnja Lambordzinija: 345
Apartman na Menhetnu: 123
Najam privatnog aviona: 45
Pokloni: 67
Vecere u restoranima: 32
Tim posluge (batler, kuvar, sofer, pomocnik): 90
Licni guru i trener: 45
Kompjuterske igre: 678

Ukupno: 1425 hiljada dinara.
```

021 Konverzija stringova u celobrojne vrednosti

Postoji nekoliko funkcija koje konvertuju različite tipove podataka. Funkcija koja konvertuje neku vrednost u celobrojnu:

```
kola = input("Nadogradnja Lambordzinija: ")
kola = int(kola)
```

Prva linija dobija ulaz od korisnika kao string i dodeljuje tu vrednost promenjivoj kola. Druga linija radi konverziju. Funkcija int() uzima string na koji upućuje kola i vraća njenu celobrojnu verziju. Zatim, kola dobija novu celobrojnu vrednost.

U sledećih sedam linija se u jednoj liniji dešava i uzimanje korisnikovog ulaznog stringa i njegova konverzija u celobrojnu vrednost:

```
avion = int(input("Najam privatnog aviona: "))
```

To je omogućeno zbog poziva dve funkcije, input() i int(), koje su nestovane (ugnježđene). Nestovanje poziva funkcija znači stavljanje jedne unutar druge. Ovo je dozvoljeno sve dok vraćena vrednost unutrašnje funkcije se može koristiti kao argument spoljne funkcije. U ovom slučaju, vraćena vrednost funkcije input() je string, a string je dozvoljen kao argument funkcije int() za konverziju u celobrojnu vrednost.

Primeri:

float(“10.0“) daje: 10.0

int(“10“) daje: 10

str(10) daje: ’10’

022 Proširenih operatora dodele

Prošireni (augmented) operatori dodele se sastoje uvek od dva znaka operatora, od kojih je drugi uvek znak jednakosti.

Kada je potrebno vrednost u promenjivoj promeniti na osnovu već postojeće vrednosti u istoj promenjivoj, efikasnije je koristiti proširene operatore dodele. Npr: a = a + 10, može se napisati sa proširenim operatorom dodele kao a += 10.

*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5

Program Beskorisni_podaci.py

```
#Beskorisni_podaci
#Dobija licne podatke od korisnika i onda
#stampa tacne ali beskorisne informacije o korisniku
ime = input("Zdravo. Kako se zoves? ")
godine = input("How old are you? ")
godine = int(godine)
tezina = int(input("Dobro, poslednje pitanje. Koliko imas kilograma? "))
print("\nAko bi pesnik poslao tebi e-mail, obratio bi se tebi sa",
      ime.lower())
print("Ali ako bi pesnik bio ljut, nazvao bi te", ime.upper())
pozvan = ime * 5
print("\nAko bi dete trazilo tvoju paznju",)
print("tvoje ime bi postalo:")
print(pozvan)
sekunde = godine * 365 * 24 * 60 * 60
print("\nTi si preko", sekunde, "sekundi mator.")
mesec_tezina = tezina / 6
print("\nDa li znas da bi na Mesecu imao tezinu od",
      mesec_tezina, "kilograma?")
```

```
sunce_tezina = tezina * 27.1
print("Na Suncu, tvoja tezina bi bila", sunce_tezina, "(ali, pa... ne zadugo).")
```

```
Zdravo. Kako se zoves? Miki
How old are you? 55
Dobro, poslednje pitanje. Koliko imas kilograma? 100

Ako bi pesnik poslao tebi e-mail, obratio bi se tebi sa miki
Ali ako bi pesnik bio ljut, nazvao bi te MIKI

Ako bi dete trazilo tvoju paznju
tvoje ime bi postalo:
MikiMikiMikiMikiMiki

Ti si preko 1734480000 sekundi mator.

Da li znas da bi na Mesecu imao tezinu od 16.666666666666668 kilograma?
Na Suncu, tvoja tezina bi bila 2710.0 (ali, pa... ne zadugo).
```

023 Uzimanje korisnikovih ulaza

Korišćenjem funkcije input() program dobija podatke od korisnika: ime, godine i težinu:

```
ime = input("Zdravo. Kako se zoves? ")
godine = input("Koliko imas godina? ")
godine = int(godine)
tezina = int(input("Dobro, poslednje pitanje. Koliko imas kilograma? "))
```

Funkcija input() uvek vraća string, pa pošto godine i tezina su brojevi oni moraju da se konvertuju. Proces je razbijen na dve linije za promenjivu godine. Kod promenjive tezina, sve je urađeno u jednoj liniji nestovanjem poziva funkcija. Namerno je urađeno na dva načina. U realnim programima treba izabrati jedan način i držati se njega.

024 Štampanje verzija imena sa malim i velikim slovima

Sledeće linije štampaju verzije promenjive ime sa velikim i malim slovima korišćenjem string metoda:

```
print("\nAko bi pesnik poslao tebi e-mail, obratio bi se tebi sa",
      ime.lower())
print("Ali ako bi pesnik bio ljut, nazvao bi te", ime.upper())
```

025 Štampanje imena pet puta

Program prikazuje korisnikovo ime pet puta koristeći ponavljanje stringa:

```
pozvan = ime * 5
print("\nAko bi dete trazilo tvoju paznju",)
print("tvoje ime bi postalo:")
print(pozvan)
```

Promenjivoj pozvan je data vrednost promenjive ime, ponovljena pet puta.

026 Računanje sekundi

Starost korisnika u sekundama se računa i štampa sa:

```
sekunde = godine * 365 * 24 * 60 * 60
print("\nTi si preko", sekunde, "sekundi mator.")
```

Pošto je 365 dana u godini, 24 sati u danu, 60 minuta u satu i 60 sekundi u minuti, godine se množe sa proizvodom 365*24*60*60. Dobijena vrednost se dodeljuje promenjivoj sekunde.

027 Računanje mesec_tezina i sunce_tezina

Računanje i prikaz težina korisnika na Mesecu i Suncu:

```
mesec_tezina = tezina / 6
print("\nDa li znas da bi na Mesecu imao tezinu od", mesec_tezina, "kilograma?")
sunce_tezina = tezina * 27.1
print("Na Suncu, tvoja tezina bi bila", sunce_tezina, "(ali, pa... ne zadugo).")
```

Pošto Mesec ima 6 puta slabiju gravitaciju od Zemljine, mesec_tezina dobija vrednost tezina / 6. Pošto je gravitacija na Suncu 27.1 puta veća nego na Zemlji, tezina se množi sa 27.1 i dodeljuje sunce_tezina promenjivoj.