Unos vrijednosti od korisnika – input()

- Vrijednost varijabli često trebamo dobiti od korisnika našeg programa
- Za unos vrijednosti varijable od korisnika koristi se naredba **input()**

```
ime = input('Upišite ime')
```

```
>>> help(input)
Help on built-in function input in module builtins:

input(prompt=None, /)
    Read a string from standard input.  The trailing newline is stripped.

    The prompt string, if given, is printed to standard output without a
    trailing newline before reading input.

    If the user hits EOF (*nix: Ctrl-D, Windows: Ctrl-Z+Return), raise
EOFError.
    On *nix systems, readline is used if available.
```

Vježba – varijable, print() i input()

- Zatražite od korisnika unos dva broja.
  - Nakon unosa brojeva, ispišite:
    zbroj, razliku, umnožak, količnik (rezultat djeljnja), potenciranje te modulo unesenih brojeva
  - Svaka operacija treba biti ispisana u novom redu, a ispis treba imati uključene brojeve, znak računske operacije te rezultat.
  - PRIMJER ISPISA:
  - 5 + 8 = 13
  - 5 - 8 = -3
  - NAPOMENA Za sada kod unosa neka kod prvog unosa drugi broj NE bude 0 (nula), jer nije dopušteno dijeliti s nulom. To svakako pokušajte napraviti, ali NE u prvom pokušaju.

- **PROBLEM!!! Dogodila se greška!**

Greške u programskom kôdu

- Ljudi nisu savršeni.
- Kôd koji ljudi napišu nije savršen, ima grešaka.
- Kôd koji napišu roboti, koje su napravili ljudi, također ima grešaka

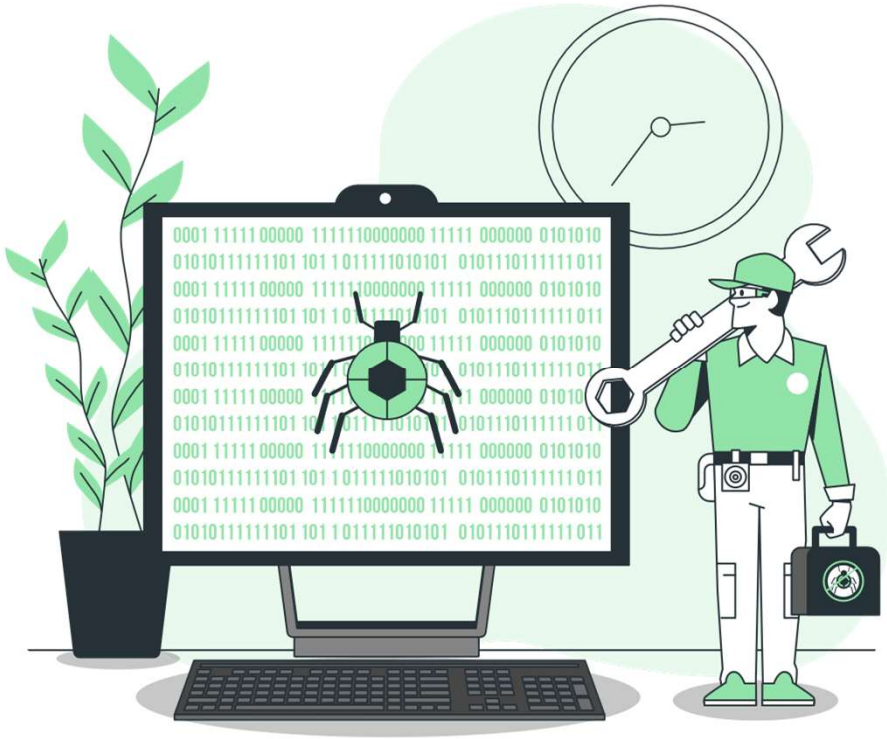
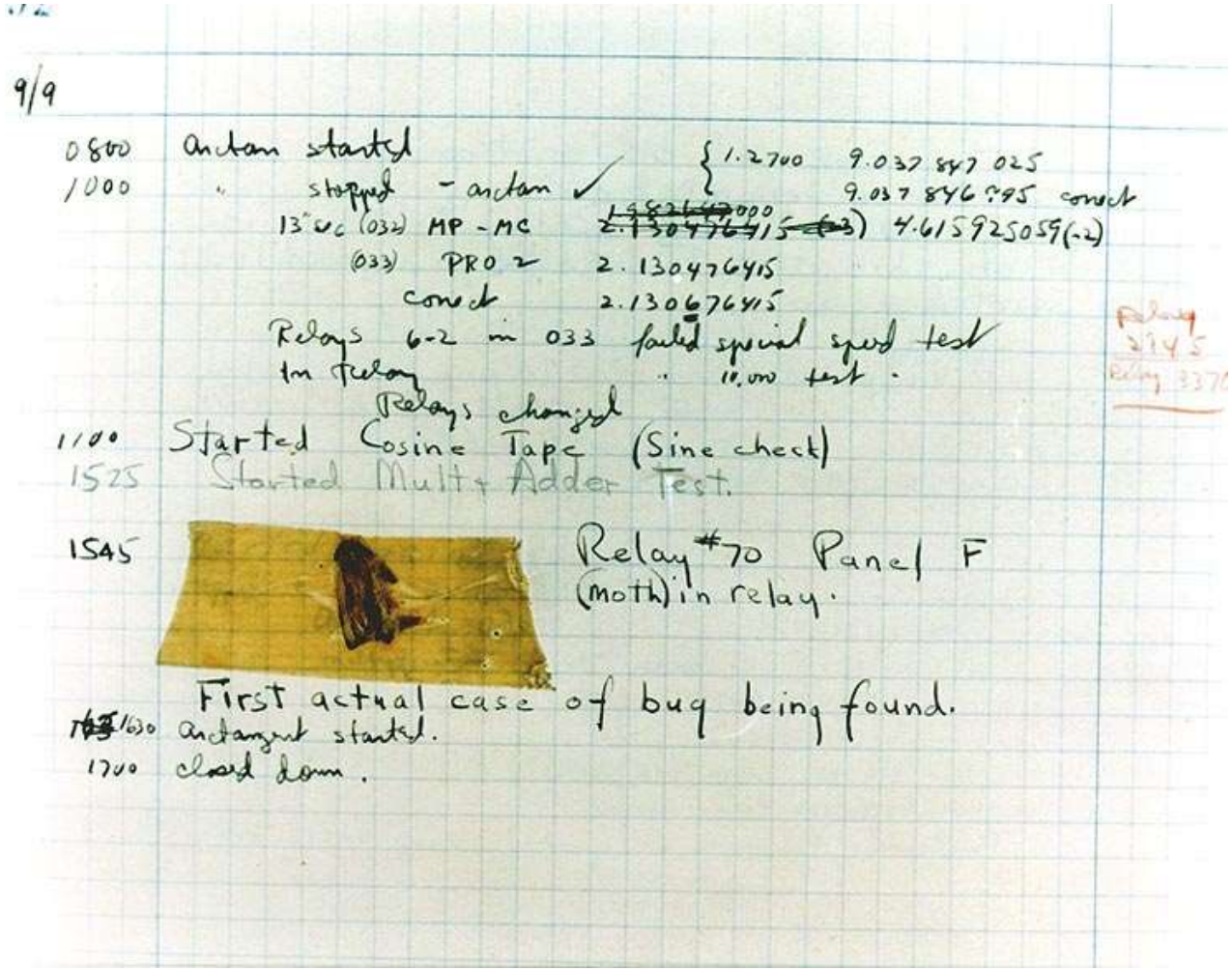


Illustration by Freepik Storyset

BUG

Znate li priču o bubi u računalu zbog koje je računalo prestalo raditi i zbog koje se kolokvijalno greške u kôdu nazivaju BUG.



First Computer Bug, 1945 - Software bug - Wikipedia

Greške u kôdu – tipovi grešaka

- **Syntax Error** – greške u sintaksi. Krivo napisane naredbe, nazivi funkcija … Najčešće program NE možete pokrenuti s ovom greškom, ali IDE točno ukazuje gdje se nalazi ovakva greška.
- **Runtime Error** ili **Exception Error** – greške koje nastaju prilikom izvođenja programa. Ponekad ih ne uočite odmah (primjer pokretanje zadnjeg zadatka). Malo ih je teže otkriti.
- **Bugs** – greške koje je najteže otkriti. To su greške koje predstavljaju krivo funkcioniranje programa. Primjer: uz cijenu se ne obračunava PDV, nego su sve cijene iste.
  Dakle, program funkcionira, kôd je ispravno napisan, ali program ne radi ono što bi trebao.

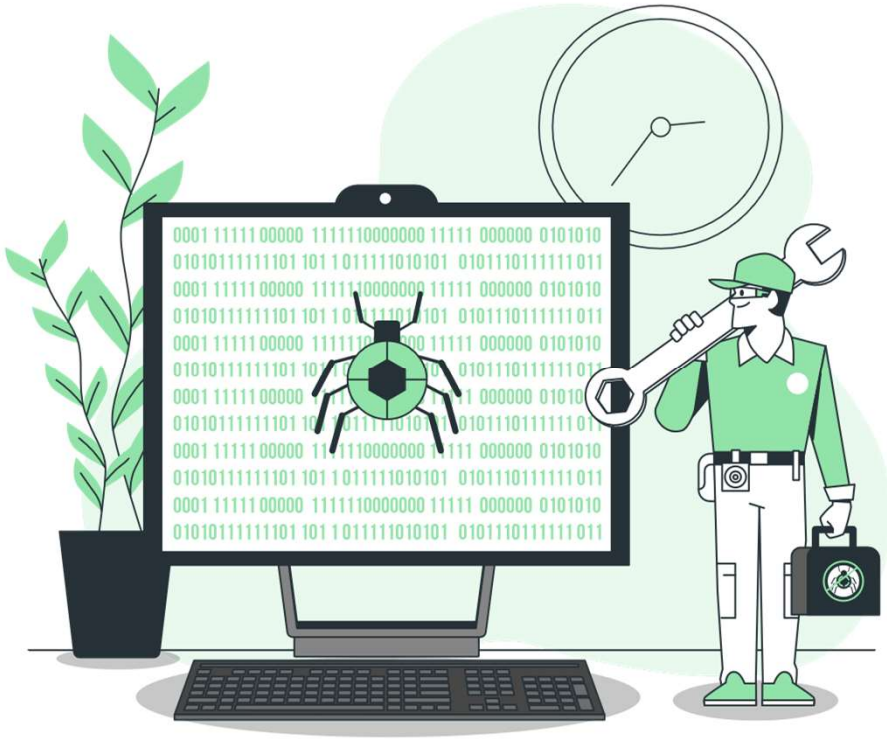


Illustration by Freepik Storyset

Konverzija tipova podataka

- Ponekad je nužno napraviti konverziju iz jednog tipa podatka u drugi tip
- Najčešće se konverzija radi iz teksta (string) u broj (integer ili float) ili obrnuto (Primjer kada trebamo vrijednost varijable zapisati u datoteku).
- Postoje i naredbe za konverziju brojeva između brojevnih sustava – binarni, oktalni, dekadski, heksadekadski

Konverzije tipova podatka

- Integer – **int('string_koji_konvertiramo')** – cijeli broj
- Float – **float('string_koji_konvertiramo')** – decimalni broj,
- String – **str(broj_ili_objekt_koji_konvertiramo)** – pretvara u tekst,
- Boolean – **bool('True')** – string u boolean
  OPREZ!!! Ova funkcija će pretvoriti bilo koji string koji ima znakove u True, osim jedne iznimke. To je 'False'.
- Character – **chr(broj_koji_konvertiramo)** – pretvara u Unicode znak
- Unicode Character – **ord('string_koji_konvertiramo')** – pretvara u cijeli broj koji predstavlja Unicode znak

Konverzija između brojevnih sustava

- Dodatne funkcije za konverziju između brojevnih sustava (NE tipova podataka)

- Iz dekadskog u binarni – **bin(cijeli_broj_koji_konvertiramo)**
- Iz dekadskog u oktalni – **oct(cijeli_broj_koji_konvertiramo)**
- Iz dekadskog u heksadekadski – **hex(cijeli_broj_koji_konvertiramo)**
- Iz … natrag u dekadski – **int('broj_koji_konvertiramo', baza)**
  - Broj koji konvertiramo treba biti tipa string pa zato imamo navodnike
  - Baza predstavlja bazu sustava iz kojeg konvertiramo. Binarni je 2, oktalni je 8, heksadekadski je 16

Vježba – varijable, print(), input(), konverzija

- Prepravite prethodne vježbe i zadatke tako da vrijednosti varijabli tražite od korisnika te napravite potrebnu konverziju.
  - *Otvorite .py datoteke u kojima imate riješene zadatke te ih prepravite tako da koristite i naredbu input(). Zatim ih možete pohraniti pod novim imenom ili ostaviti isti naziv.*
- IP adresa je adresa svakog računala na mreži koja se sastoji od četiri broja između 0 i 256. Primjer IP adrese: 192.168.0.184
  - IP adresu iz primjera ispišite u binarnom, oktalno i heksadekadskom obliku.
  - SAVJET: Za sada koristite zasebnu varijablu za svaki od četiri broja, odnosno dijela (okteta) IP adrese, ali ispišite ih u istom obliku kako je navedeno u primjeru (192.168.0.184).
  - Ispis treba napraviti za sve oblike brojevnih sustava.
- Na stranici https://www.color-hex.com/color-palette/33532 imate boje Google logotipa. Pomoću odgovarajućih naredbi za konverziju pokušajte pretvoriti RBG zapise u HEX boja i obratno.
  - Primjer:
  - Naziv boje HEX zapis RGB (Red Green Blue)
  - CRVENA *#EA4335* (234, 67, 53)
  - Za HEX zapis EA-43-35 trebate dobiti RGB zapis 234-67-53
  - NAPOMENA Zanemarite početni # znak u HEX zapisu sa stranice.
  - NAPOMENA HEX zapis čine tri grupe po dva znaka EA-43-35, svaka dva znaka čine jednu boju RGB

Vježba – prepravite ove zadatke…

- … tako da za unos vrijednosti pitate korisnika:
  - Ako automobil troši 5.3 litara na 100 km i ako je cijena goriva 9.56 kn po litri (nije važno kojeg goriva), izračunajte koliko košta 1 km vožnje automobilom. Prikažite mjesečni trošak (30 dana) odlaska na posao automobilom koji je udaljen 20 km u jednom smjeru.
  - Imate 10000 kn i možete zaboraviti na njih na 15 godina. Ako Vam banka nudi 2.5% godišnju kamatu za taj iznos, koliko ćete zaraditi nakon 15 godina. Jednostavni kamatni račun k = C * p * t
    - k = iznos kamata odnosno prinos
    - C = iznos glavnice
    - p = godišnja kamatna stopa – NAPOMENA: 5% = 5 / 100 = **0.05**
    - t = vrijeme u godinama

Tip podataka string ili TEKST

- STRING je tekstualni tip podataka
  - Ime i prezime, ulica, grad, država, poruka …

- Vrijednost string varijable deklariramo pomoću navodnika
  - Svejedno je koje navodnike koristimo, ali uvijek moramo koristiti isti tip navodnika na početku i završetku teksta
- Za unos vrijednosti string podatka od korisnika koristimo input()
- NE možemo mijenjati string tip podatka. String je nepromjenjiv IMMUTABLE – Testirajte.
- Konverzija u drugi tip podataka i obrnuto (int(), float(), str() ...)

Format stringa – format()

- Cilj pisati slično načinu pisanja rečenice

1. Napišimo rečenicu koju želimo imati kao tekst
   ```
   'Puno ime i prezime je: Petar Perić'
   ```
2. Nakon zadnjeg navodnika stavimo točku i upišemo format()
   ```
   'Puno ime i prezime je: Petar Perić'.format()
   ```
3. Na mjesto vrijednosti upisujemo {} koliko god nam treba, a unutar format() dodamo nazive varijabli
   ```
   'Puno ime i prezime je: {} {}'.format(ime, prezime)
   ```
4. Sve to pohranimo u neku varijablu
   ```
   puno_ime = 'Puno ime i prezime je: {} {}'.format(ime, prezime)
   print(puno_ime)
   ```

Format stringa – kraći format

- Kraći i potencijalno jasniji način formatiranja teksta

1. Umjesto format() na kraju navodnika, koristimo slovo f prije prvog navodnika
   ```
   f'Puno ime i prezime je: Petar Perić'
   ```
2. Isto kao i kod prvog načina, gdje smo pisali { } zagrade pišemo ih i dalje, samo što sada zagrade ne ostavljamo prazne nego u njih upisujemo nazive varijabli
   ```
   f'Puno ime i prezime je: {ime} {prezime}'
   ```
3. I na kraju sve to opet pohranimo u neku varijablu
   ```
   puno_ime = f'Puno ime i prezime je: {ime} {prezime}'
   print(puno_ime)
   ```

Format stringa – "\" Escape Character

- "\" ili Backslesh ili Excape Character
- Omogućava dodavanje nekih posebnih znakova unutar teksta
- Nakon \ znaka se, bez razmaka dodaje znak koji želimo dodati
- Desno u tabeli NISU sve kombinacije već najčešće korištene
- Backslash neki prevode kao obrnuta kosa crta … ili obrnuti kroz (obrnuti "/"). Mi ćemo koristiti backslash

Kombinacija	Naziv	Namjena
\\	Backslesh	Ispisuje JEDAN znak \ Prvi služi kao signal da dolazi poseban znak
\'	Jednostruki navodnik	Ispisuje jedan jednostruku navodnik
\"	Dvostruki navodnik	Ispisuje jedan dvostruki navodnik
\n	Novi red	Ispis teksta nastavlja u novom redu. Efekt isti kao i tipka ENTER
\t	Tab	Pomjeranje teksta u desno. Efekt isti kao tipka TAB

Vježba – rad sa string-ovima

- Pomoću nekog od načina formatiranja teksta ispišite podatke o osobi i filmu iz prethodnih primjera.
- Ako string ne možemo mijenjati, kako onda možemo raditi zbrajanje, odnosno kombiniranje dva i više stringova u jedan?

Kolekcije podataka

- Adresar
- Lista propuštenih poziva
- To-Do lista
- Diskografija albuma nekog glazbenika/glazbenice
- Popis stvari koje želim napraviti prije 60. rođendana
- TEMP folder na računalu
- …

Illustration by Freepik Storyset

Python kolekcije podataka

- Lista – lista[1, 2, 3, ‘Petar Perić’, True]
  - Lista koristi [ ] zagrade
  - Svaki element liste ima svoj jedinstveni indeks ili redni broj
  - Indeks brojevi počinju od 0 (nula)
- Dictionary / Rječnik – dict{1 : ‘Ban Josip Jelačić’, }
  - Rječnik koristi { } zagrade
  - Elementi su par ključ : vrijednost
  - Svaki element nema indeks, ali ima ključ. Zato ključ MORA biti jedinstven. Nema duplikata.
- Tuple / N-terac – tuple(1, 2, 3)
  - Tuple koristi ( ) obične zagrade
  - Tuple je NEPROMJENJIV. Kada kreirate Tuple, ne možete kasnije dodati ili maknuti element.
  - Svaki elementi N-terca ima svoj jedinstveni indeks ili redni broj
  - Indeks brojevi također počinju od 0 (nula)

Python kolekcije podataka – lista

- Lista koristi [ ] zagrade
- Svaki elementi liste ima svoj jedinstveni indeks ili redni broj
- Indeks brojevi počinju od 0 (nula)
  - `lista = [154, 'tekst', 'jos jedan tekst', 3.14, True, 'Pa opet tekst']`
  - `prazna_lista = []`
- Pristup pojedinačnom elementu liste
  - `prvi_element = zadaci[0]`
  - `drugi_element = zadaci[1]`
  - `treci_element = zadaci[2]`
- String – lista znakova

Grupiranje instrukcija u blokove

- Python za grupiranje kôda, umjesto { } zagrada, koristi uvlaku (tab) ili 4 razmaka (space).
- O ovome ne treba voditi brigu ukoliko koristite program za pisanje kôda jer se nakon **:** znaka za početak bloka, automatski novi red uvuče za potrebni broj razmaka
- Ovo je jedan od načina kako Python želi povećati čitljivost programskog kôda. Kao paragraf.
- Lako se vizualno izdvajaju cjeline, odnosno blokovi kôda.

Primjer kôda:

```
for element in kolekcija:
    instrukcije nad element
```

FOR petlja

- Petlja koja neku instrukciju ponavlja određeni broj puta.
- Ključna riječ for
- FOR petlja se može čitati kao da računalu zadajemo zadatak:

„**Za svako** *ime* **unutar** *imenika*:
ispiši *ime* na ekran„

```
for element in kolekcija:
    aktivnost nad element
```

element predstavlja varijablu koja se koristi samo dok se izvršava FOR petlja, nakon toga ta varijabla postaje nedostupna.

Naziv varijable *element* je proizvoljan.

Naziv kolekcije *kolekcija* je predefiniran, odnosno to je naziv varijable u koju smo prethodno pohranili neku kolekciju podataka (*elemenata*)

Raspon brojeva – range()

- Ugrađena funkcija koja vraća listu brojeva unutar zadanog raspona
- range(n) – vraća brojeve u rasponu od 0 (nula) do n uz korak 1
- range(n, m) – vraća brojeve u rasponu od n (uključen) do m (m nije uključen) uz korak 1
- range(n, m, k) – vraća brojeve u rasponu od n (uključen) do m (m nije uključen) uz korak k

```
range(5) – 0, 1, 2, 3, 4, 5
range(5, 10) – 5, 6, 7, 8, 9
range(5, 10, 2) – 5, 7, 9
```

Lista – osnovne naredbe

- *naziv_liste.**append(novi_element)*** – naredba za dodavanje novog elementa na kraj liste
- ***naziv_liste[indeks]*** = *nova_vrijednost* – naredba za izmjenu vrijednosti pohranjene u element liste na poziciji *indeks*
- *broj_elemenata_liste* = ***len(naziv_liste)*** – len() naredba dohvaća broj elemenata pohranjenih u listi

- *Provjera je li tekst zaista lista znakova?*

Slice – lista[START : STOP : STEP]

- Način kreiranja nove liste na osnovu manipulacije elementima prethodno kreirane liste. Primjer:
  - izdvojiti zadnja dva elementa liste
  - izdvojiti svaki treći element liste
- Po sintaksi jako slično range() naredbi.

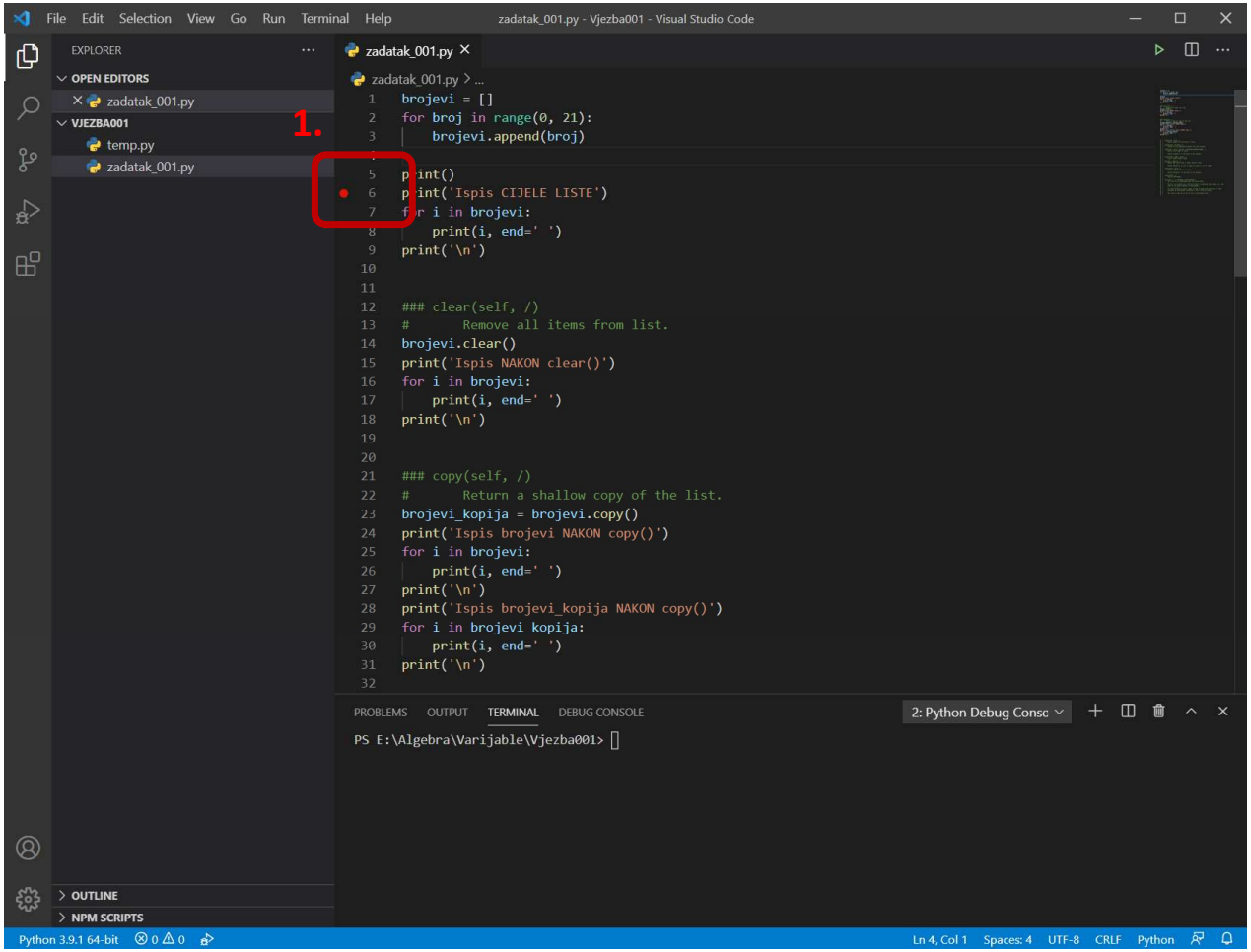
- Primjeri na listi brojeva od 1 do 100.

Lista – ostale naredbe

- *naziv_liste**.clear()*** – naredba za brisanje svih elemenata liste
- *nova_lista = naziv_liste**.copy()*** – naredba za kopiranje liste
- **PROBLEM!!! .copy() NE kopira listu!**

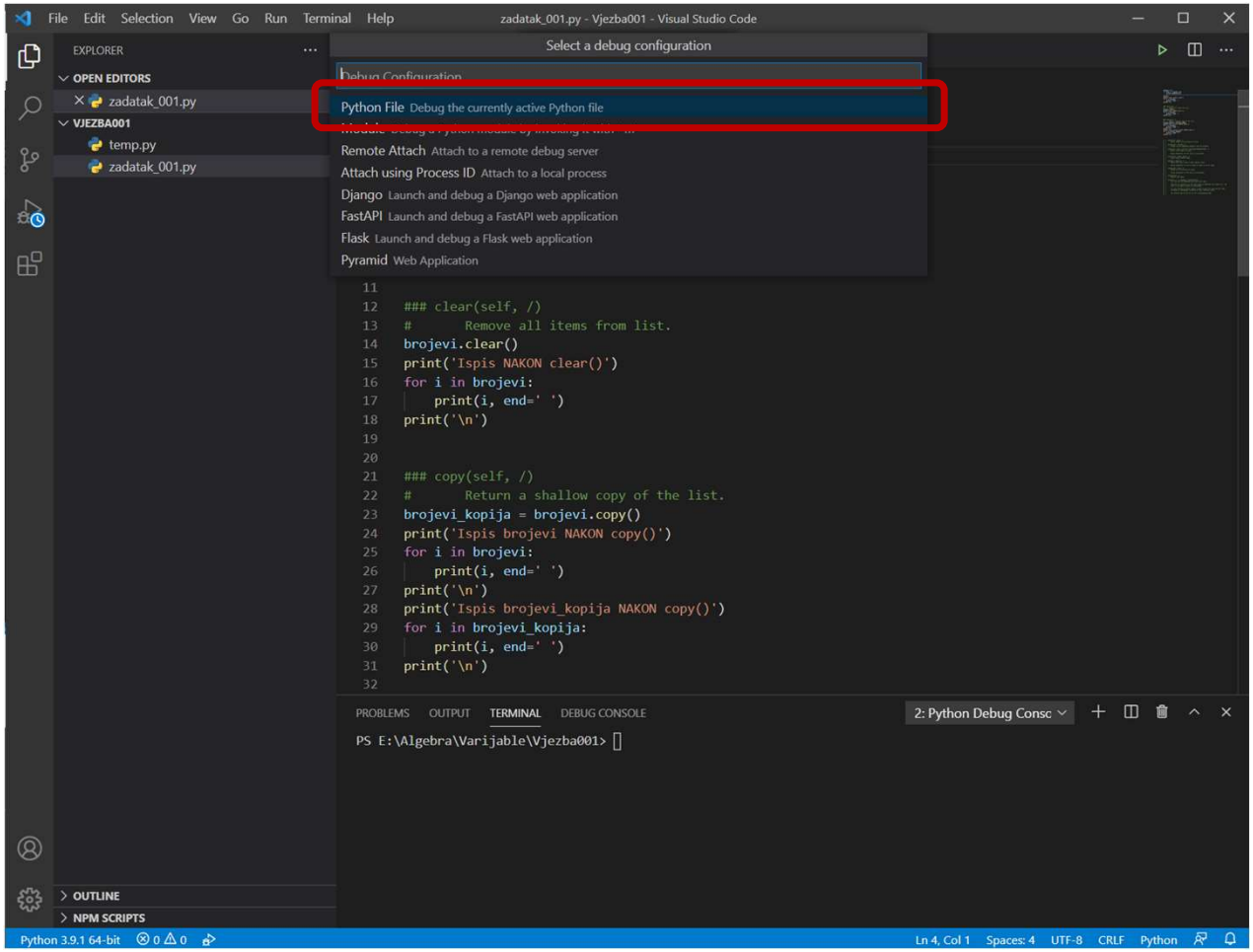
Debugging

1. Kliknite na stupac lijevo od linije kôda. U našem primjeru to je linija 6. Kada kliknete na taj stupac ispred te linije će se pojaviti točka.
   Ta točka se zove Break Point i to je točka u kojoj će se ZAUSTAVITI izvršavanje programa.
2. Pokrenite program u DEBUGGING načinu rada jednostavno tipkom F5 na tipkovnici ili iz izbornika
   Run -> Start Debugging



Debugging

1. Nakon klika na F5, VS Code će vam ponuditi što želite debugirati?
2. Odaberite "**Python File** Debug the currently active Python file"

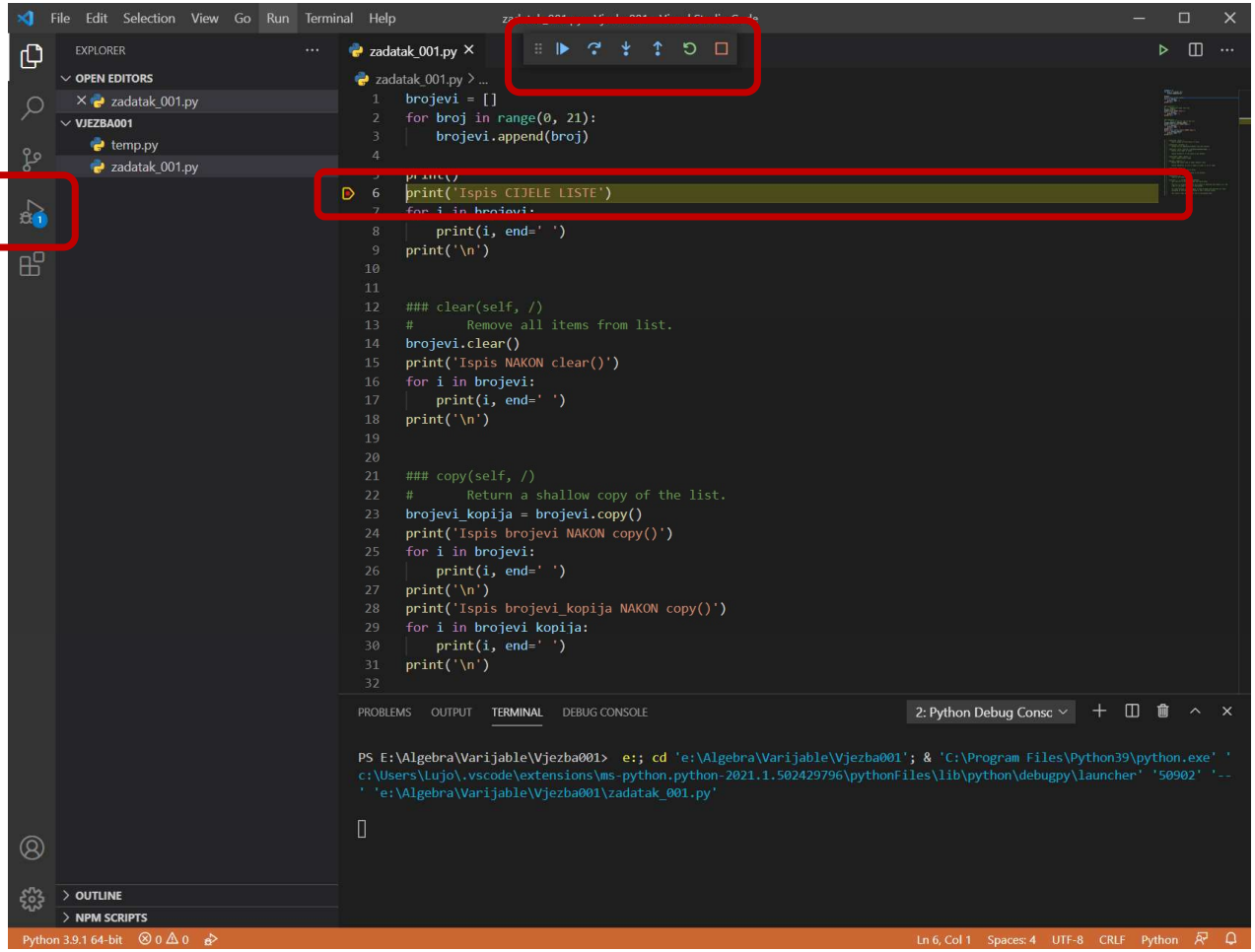


Debugging

1. Prvo kliknite na ikonu koja prikazuje aktivne programe.

Ostali crveni okviri prikazuju:

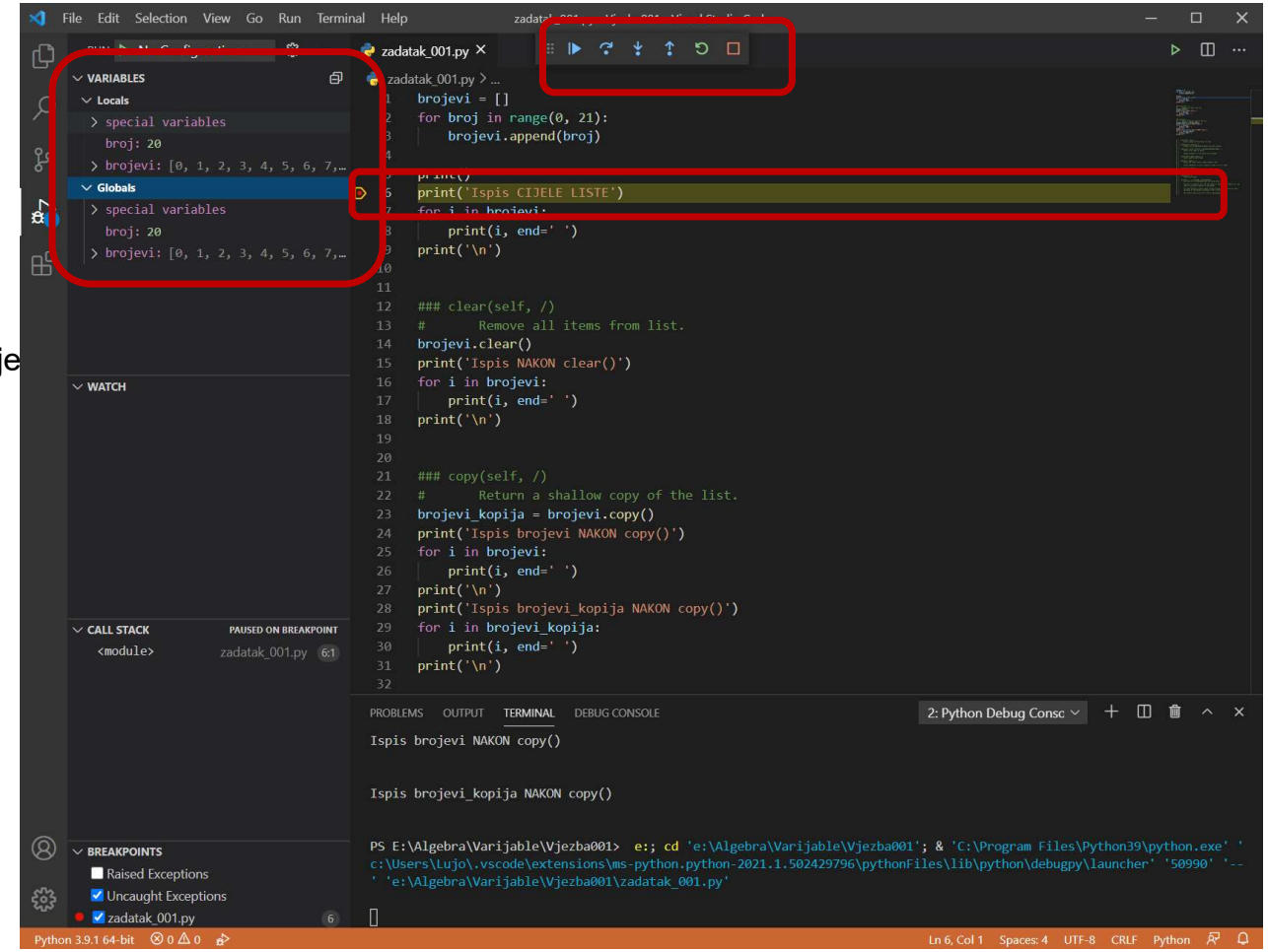
- Ikone za upravljanje pokretanjem programa. Opis na slijedećem slideu.
- Liniju u kojoj je program zaustavljen, a kada se pomoću upravljačkih ikona izvršavanje programa pokrene na slijedeću liniju, onda će ta slijedeća linija biti označena.
    - VAŽNO: Kôd u označenoj liniji NIJE se izvršio. To znači da je izvršavanje programa ZAUSTAVLJENO na POČETKU te linije.



Debugging

- Ikone za upravljanje pokretanjem programa. Opis s lijeva na desno:
  - Ikona s točkicama je za pozicioniranje ovog okvira - zanemarite
  - Continue (F5) – nastavi izvršavanje programa do kraja.
  - **Step Over (F10)** – ovo je ikona koju ćete najčešće koristiti. Pokreće izvršavanje programa red po red.
  - Step Into (F11)
  - Step Out (Shift + F11)
  - Restart (Ctrl + Shift + F5)
  - Stop (Shift + F5)

Stupac VARIABLES prikazuje trenutnu vrijednost lokalnih i globalnih varijabli. Ako nemate prikaz kao na slici, raširite prikaz klikom na > znak.



Lista – ostale naredbe

- *naziv_liste*.**clear()** – naredba za brisanje svih elemenata liste
- *nova_lista* = *naziv_liste*.**copy()** – naredba za kopiranje liste
- *broj_ponavljanja_u_listi* = *naziv_liste*.**cunt(element)** – naredba koja prebrojava koliko se puta element pojavljuje u listi
- *naziv_liste*.**extend(nova_lista)** – naredba koja proširuje postojeću listu novom listom
- *indeks_elementa* = *naziv_liste*.**index(element)** – naredba koja dohvaća indeks pozicije na kojoj se nalazi element
- *naziv_liste*.**insert(indeks, element)** – naredba za umetanje elementa na poziciju točno ispred pozicije označene navedenim indeksom
- *naziv_liste*.**sort()** – naredba za sortiranje elemenata liste
- *naziv_liste*.**reverse()** – naredba za sortiranje elemenata liste obrnutim redoslijedom