

## Beleške sa časa – 3.nedelja

### Terminologija

programska paradigma = način programiranja, programski stil = familija programskih jezika

kastovanje = konvertovanje tipa = promena tipa

blok kôda = grupa naredbi

indentacija = uvlačenje linija kôda = poravnanje linija kôda

### Tipovi podataka

- podaci sa kojima baratamo u Python-u su organizovani u tipove

- jedan tip podataka karakteriše:

- vrsta podataka koje opisuje (numerički podaci, tekstualni podaci, logičke konstante, kolekcije podataka itd.)
- način reprezentacije (kako se ti podaci predstavljaju pomoću bitova u računaru)
- broj bitova koji se koriste za reprezentaciju (odakle sledi opseg mogućih vrednosti)
- skup operacija koje se mogu primeniti nad podacima tog tipa

- programski jezik Python podržava rad sa:

- tekstualnim podacima kroz tip podataka **str**
- numeričkim podacima kroz tri tipa podataka **int**, **float** i **complex**
- logičkim konstantama kroz tip **bool**
- podacima koji predstavljaju kolekcije kroz pet tipova podataka **list**, **tuple**, **set**, **dict** i **range**

BUILT-IN DATA TYPES IN PYTHON	
• str	x = "Pycoders"
• int	x = 17
• float	x = 17.5
• complex	x = 7j
• bool	x = True
• list	x = ["red", "blue"]
• tuple	x = ("apple", "banana")
• set	x = {"apple", "banana"}
• dict	x = {"name" : "John"}
• range	range(6)

### Objektno-orijentisani aspekt Python jezika

- pored toga što je Python skript jezik (otuda fajlovi sa ekstenzijom .py nazivamo Python skriptama), on takođe pripada i paradigmi objektno-orijentisanih jezika

- **Def:** Objektno-orijentisano programiranje je programska paradigma zasnovana na pojmu OBJEKATA. Izvršavanje programa se zasniva na promeni stanja pojedinačnih objekata i njihovoj međusobnoj interakciji.

- **Def:** Objekat je integralna celina podataka (atributa) i funkcija (metoda) za rad sa njima.

- **METOD = FUNKCIJA KOJA PRIPADA NEKOM OBJEKTU, KOJA SE POZIVA NAD OBJEKTOM**

- **Def:** Klasa je skup objekata sa zajedničkim svojstvima, koji se ponašaju na isti način. Klasa definiše šablon za kreiranje objekata, tj. opisuje strukturu objekta.

- primerak (instanca) klase je konkretan objekat date klase, sa konkretnim vrednostima atributa (podataka)

- **KLASOM JE DEFINISAN TIP OBJEKATA, A SVAKI OBJEKAT PRIMERAK (INSTANCA) IMA KONKRETE VREDNOSTI PODATAKA**

- **VAZNO:** u Python-u je gotovo sve objekat tj. primerak neke klase

### Tipovi podataka kao klase objekata

- Klasa je zapravo konstrukt kojim se definiše novi tip podataka (objekata)
- Za svaki od ugrađenih tipova podataka postoji istoimena klasa koja ih definiše

### Tip **str**

- tip **str** opisuje nizove karaktera kojima se predstavljaju tekstualni podaci (od engleskog *string*, niz karaktera)
- stringovske konstante se navode između jednostrukih, dvostrukih ili trostrukih navodnika
- ne postoji razlika između jednostrukih i dvostrukih navodnika, dok se u slučaju trostrukih navodnika string može prostirati u više redova
- interni način reprezentacije tipa **str** podrazumeva da se nizovi karaktera koji definišu stringovsku konstantu čuvaju sekvencijalno u memoriji, pri čemu se svaki karakter pojedinačno predstavlja pomoću Unicode kodne šeme
- moguće je definisati i tzv. *prazan string*, tj. stringovsku konstantu koja ne sadrži ni jedan karakter
- nad podacima ovog tipa mogu se primenjivati relacijske operacije (<, >, ...), operator + (tzv. operator konkateniranja), operator [] (tzv. operator indeksiranja) i operator [ : ] (tzv. slajsing operator)

### Tip **int**

- tip **int** opisuje celobrojne vrednosti (od engleskog *integer*, ceo broj)
- podrazumeva se da su vrednosti ovog tipa označene
- interni način reprezentacije tipa **int** je takav da ne postoji ograničenje za minimalan i maksimalan broj cifara
- nad podacima ovog tipa mogu se primenjivati aritmetičke operacije (+, -, \*, /, %, ...), relacijske operacije (<, >, ...) i logičke operacije (and, or, not)
- kada vrednosti tipa **int** učestvuju u logičkim izrazima, tada se sve nenula vrednosti implicitno kastuju u logičku konstantu **True**, a nula vrednost u konstantu **False**

### Tip **float**

- tip **float** opisuje realne brojeve
- interni način reprezentacije tipa **float** podrazumeva predstavljanje realnih brojeva u tzv. pokretnom zarezu, odakle potiče i motivacija za naziv ovog tipa podataka (od engleskog *floating-point representation*)
- tip float zauzima 64 bita
- definiše konstante za predstavljanje beskonačnih vrednosti koje se zadaju na sledeći način:

**float('inf')**  
**float('-inf')**

- nad podacima ovog tipa mogu se primenjivati aritmetičke operacije (+, -, \*, /, %, ...), relacijske operacije (<, >, ...) i logičke operacije (and, or, not)
- kada vrednosti tipa **int** učestvuju u logičkim izrazima, tada se sve nenula vrednosti implicitno kastuju u logičku konstantu **True**, a nula vrednost u konstantu **False**
- za primenu operacija kada su jedan ili oba operanda beskonačne vrednosti primenjuju se uobičajena matematička pravila

Name	Key type
Positive infinity	$\infty$
Negative infinity	$-\infty$
Infinity difference	$\infty - \infty$ is undefined
Zero product	$0 \cdot \infty$ is undefined
Infinity quotient	$\infty / \infty$ is undefined
Real number sum	$x + \infty = \infty$ , for $x \in \mathbb{R}$
Positive number product	$x \cdot \infty = \infty$ , for $x > 0$

### Tip bool

- tip **bool** opisuje logičke konstante **True** i **False** (od engleskog *boolean*, po imenu engleskog matematičara koji se bavio oblašću logike, *George Boole*)
- nad podacima ovog tipa mogu se primenjivati relacijske logičke operacije (and, or, not), aritmetičke operacije (+, -, \*, /, %, ...) i relacijske operacije (<, >, ...)
- kada logičke konstante učestvuju u aritmetičkim i relacijskim izrazima, tada se **True** konstatna implicitno kastuje u vrednost **1**, a **False** konstanta u vrednost **0**

### Promenljive i tipovi podataka

- promenljive su mesta u memoriji u kojima se čuvaju podaci
- promenljive imaju **tip**, **ime** i **vrednost**
- vrednost promenljive može da se menja za vreme izvršavanja programa
- u Python-u ne mora eksplicitno da se navede kog je tipa promenljiva, već se to implicitno određuje na osnovu tipa njoj pridružene vrednosti
- u Python-u tip promenljive može da se menja za vreme izvršavanja programa (ovo ne važi za sve programske jezike, već samo za neke)
- informacija o tipu promenljive se može dobiti pomoću **type** funkcije standardne biblioteke
- svakoj promenljivoj se može pridružiti objekat bilo kog tipa
- kažemo da je promenljiva onogo tipa kojeg je njoj pridruženi objekat

### Kastovanje (implicitno i eksplicitno)

- promenljivama se može menjati tip, što se drugačije naziva kastovanje
- za svaki od ugrađenih tipova podataka postoji funkcija koja vrši kastovanje u taj tip
- eksplicitni kastovanje podrazumeva pozivanje odgovarajuće funkcije za kastovanje
- implicitno kastovanje se vrši automatski prilikom prevođenja programa kada se na mestu nekog argumenta funkcije ili operanda za primenu neke operacije očekuje određeni tip podataka a naveden je drugačiji tip podataka
- implicitno kastovanje se vrši samo ukoliko su očekivani i dobijeni tip podataka kompatibilni (npr. float i int)

### Funkcija int()

- funkcija **int()** konvertuje prosleđeni argument u ceo broj
- ukoliko je prosleđeni argument funkcije tipa **float** (realan broj), cifre posle decimalne tačke će biti uklonjene
- kompleksne brojeve nije moguće konvertovati u cele korišćenjem ove funkcije
- ako je prosleđeni argument funkcije tipa **str** koji sadrži samo cifre i opciono počinje znakom + ili -, funkcija **int()** će uspešno izvršiti konverziju, a u suprotnom će izazvati grešku u programu

### Funkcija float()

- funkcija **float()** konvertuje prosleđeni argument u realni broj
- ukoliko je prosleđeni argument funkcije tipa **int**, funkcija će izvršiti konverziju celobrojne vrednosti u odgovarajuću realni broj koji iza decimalne tačke ima sve nule
- kompleksne brojeve nije moguće konvertovati u cele korišćenjem ove funkcije
- ako je prosleđeni argument funkcije tipa **str** koji sadrži samo cifre, opciono počinje znakom + ili - i sadrži decimalnu tačku, funkcija **float()** će uspešno izvršiti konverziju, a u suprotnom će izazvati grešku u programu

### Funkcija str()

- funkcija **str()** konvertuje prosleđeni argument u string pridruživanjem odgovarajuće tekstualne reprezentacije

### Funkcija bool()

- funkcija **bool()** konvertuje prosleđeni argument u odgovarajuću logičku promenljivu
- ukoliko je prosleđeni argument funkcije numeričkog tipa (**int**, **float** ili **complex**), funkcija će izvršiti konverziju u nenula vrednosti u logičku konstantu **True**, a vrednost 0 u logičku konstantu **False**
- ako je prosleđeni argument funkcije tipa **str** koji sadrži jedan ili više karaktera, biće konvertovan u logičku konstantu **True**, dok se prazan string konvertuje u logičku konstantu **False**

### Blokovi kôda i indentacija

- blok kôda ili blok naredbi predstavlja grupu sekvencijalnih naredbi koje čine jednu celinu
- blok kôda se u Python-u definiše kao skup linija kôda (naredbi) sa istom količinom indentacije tj. istim brojem belina ispred linija kôda (uvlačenje kôda)
- indentacija linija kôda je broj belina pre početka aktualnog kôda, i to je Python-ov način grupiranja naredbi u programskom kôdu, tj. način odvajanja blokova kôda od ostatka programa
- indentacija u Python-u, dakle, označava kada blokovi naredbi u kôdu počinju i završavaju
- za indentaciju bloka naredbi može se koristiti razmak ili tabulator
- nije dozvoljeno kombinovati simbole za indentaciju (upotrebljavati u kôdu malo jedan simbol beline, pa malo drugi), potrebno je odlučiti se za jednu vrstu belina kojom će se vršiti indentacija i koristiti isti u celom kôdu programa
- **dosledno uvlačenje koda je bitno u Python-u jer definiše blokove koda**
- za razliku od nekih drugih programskih jezika, dosledna indentacija u Python-u je obavezna i ukoliko se ne vrši na ispravan način, program neće moći da se prevede, ma koliko god sam kôd programa bio ispravan
- blokovi kôda mogu u sebi sadržati druge blokove kôda sa većom količinom indentacije
- sve linije koda koje želimo staviti unutar istog bloka naredbi moramo uvući za isti broj belina
- povećanje broja razmaka dolazi nakon znaka dvotačke (:) koja označava početak bloka naredbi, dok smanjivanje broja razmaka označava kraj trenutnog bloka naredbi

### Obrada grešaka (**try-except** blok)

- u Python-u obrada grešaka koje nastaju u toku izvršavanja koda postiže se upotrebom **try-except** blokova
- **try** blok sadrži kôd koji potencijalno može izazvati grešku, dok se u **except** bloku navode naredbe koje treba izvršiti u slučaju da se greška dogodi, tj. naredbe kojima se ta greška obrađuje
- ukoliko se greška dogodi u nekoj od naredbi iz **try** bloka, prekida se izvršavanje naredbi iz **try** bloka na tom mestu i izvršavaju se naredbe iz **except** bloka
- ukoliko do greške ne dođe prilikom izvršavanja naredbi u **try** bloku, neće se izvršiti naredbe iz **except** bloka