



Distance Learning System



# Tipovi programiranja

Object Oriented programming in Python

[vladimir.maric@link.co.rs](mailto:vladimir.maric@link.co.rs)

# Šta su tipovi programiranja

---

- Tipovi programiranja su koncepti koje programeri primenjuju prilikom upotrebe nekog programskog jezika i kreiranja programa
- Ovi koncepti su rezultat odluke programera, projekta, ili same tehnologije (programskog jezika i pratećih platformi)
- Neki programski jezici podržavaju samo jednu varijantu programiranja, dok ih neki podržavaju više

# Koji sve tipovi programiranja postoje?

- Neki od najupečatljivijih tipova programiranja danas su: **imperativno**, **deklarativno**, **proceduralno**, **objektno**, **funkcionalno** (nisu deo iste kategorizacije)
- Osim pomenutih, postoje i manje formalni tipovi programiranja, koji se radije mogu tretirati kao projektni pristupi: test driven development, pair programming, agile development
- Ima takođe i duhovitih pristupa, poput: rubber duck programming/debugging
- I naravno, mnogi drugi, manje ili više formalni tipovi

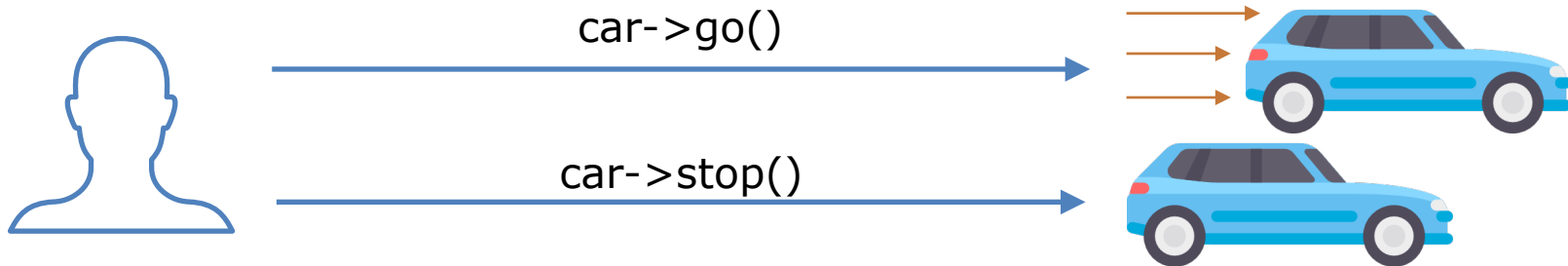
# Šta je **imperativno** programiranje?

---

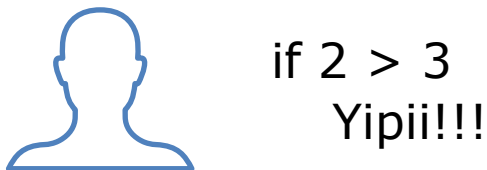
- Imperativno programiranje podrazumeva konkretnu implementaciju onoga što želimo da program uradi
- Ovo se realizuje kroz seriju definisanih koraka koje program treba da izvrši
- Većina “pravih” programskih jezika je imperativnog karaktera
  - **Proceduralno** i **objektno** programiranje spadaju u kategoriju imperativnog programiranja

# Gde koristimo imperativno programiranje

## U radu sa objektima



## U direktnom rešavanju logičkih problema



# Poznatiji imperativni jezici

---

Mašinski jezik

Fortran

Basic

Cobol

C

...

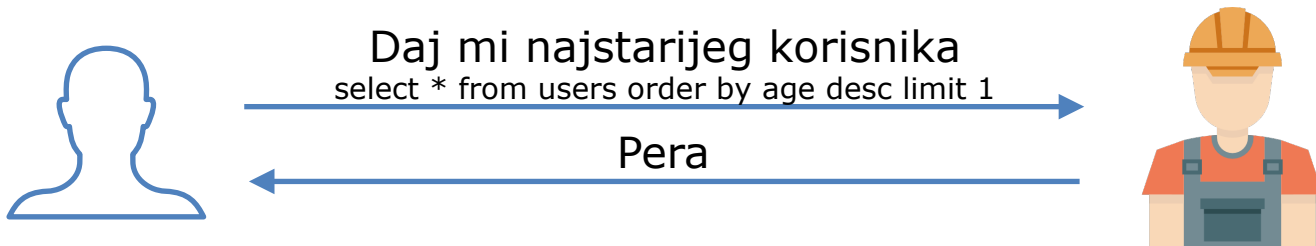
# Šta je **deklarativno** programiranje?

---

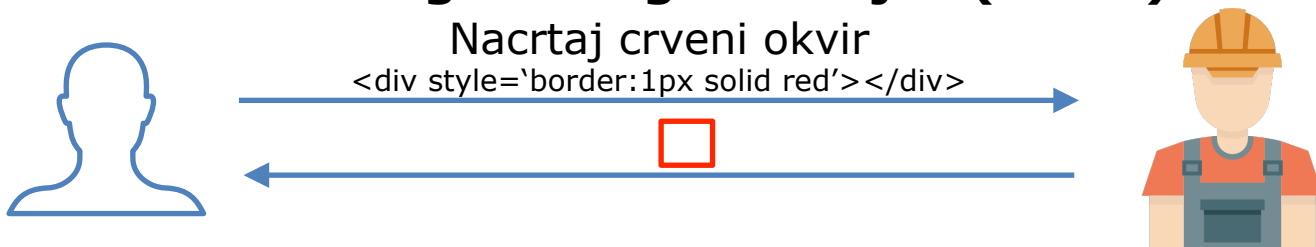
- Deklarativno programiranje ne zahteva bavljenje implementacijom logike procesiranja, već radije traženjem krajnjeg rezultata procesiranja
- Deklarativno programiranje je manje programiranje, a više upotreba već gotovih elemenata
- Naravno, ispod deklarativnog koncepta, mora postojati implementacija, ali je ona bazirana na poznatim apstrakcijama pa se njome ne moramo baviti
  - **Funkcionalno programiranje, manipulacija bazama podataka**, kao i **markup** "programiranje" spadaju u deklarativne tipove programiranja

# Gde koristimo deklarativno programiranje

## U upravljanju bazama podataka (SQL)



## U izradi grafičkog interfejsa (HTML)





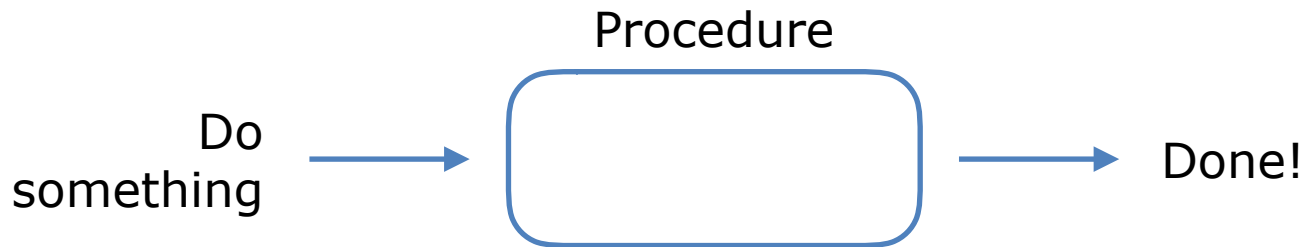
# Poznatiji deklarativni jezici

---

HTML i ostali XML derivati  
SQL varijacije  
...

# Šta je **proceduralno** programiranje?

- Proceduralno programiranje podrazumeva upotrebu procedura u cilju grupisanja i procesiranja koda
- Proceduralni pristup povećava preglednost koda, smanjuje količinu koda i pojednostavljuje njegovu upotrebu

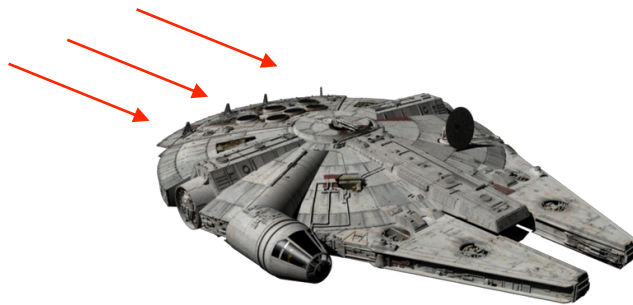


# Šta je **objektno** programiranje?

- Objektno programiranje koristi objekat kao komponentu programa
- Objekti imaju **atribute** i **operacije**
- Menjanjem atributa objekta i izvršavanjem njegovih operacija, menjaju se njegova stanja, a samim tim i stanja programa



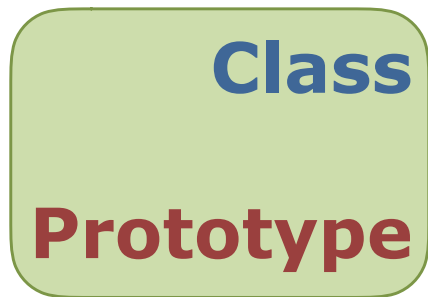
brzina = 0



brzina = 45

# Varijante objektnog programiranja

---



Koriste se klase kao šabloni za kreiranje objekata



Kao šabloni se koriste postojeći objekti

**Concurrent**



Više operacija se obavlja paralelno nad deljenim resursima

**Actor**

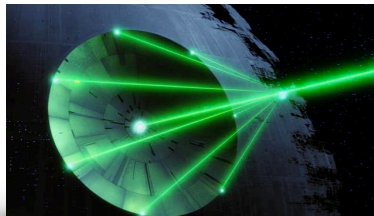


Resursi se ne dele između objekata

*<https://anthonylebrun.silvrback.com/actors-vs-objects>*

# Tri najbitnija koncepta objektnog programiranja

- Enkapsulacija



*Niko nema pojma kako zvezda smrti radi. Ali nikoga nije ni briga, sve dok može da puca.*

- Nasleđivanje



*Zvezda smrti 2 je ista, ali još bolja od zvezde smrti 1*

- Polimorfizam



*Obojica znaju silu, ali je totalno drugačije koriste*

# Enkapsulacija

- Pojam enkapsulacije podrazumeva sakrivanje kompleksne logike neke funkcionalnosti.

Za ovo je odličan primer automobil. Kada okrenemo ključ u automobilu, dolazi do startovanja motora. Ono što mi, kao vozači, znamo, je da treba da okrenemo ključ da bismo aktivirali motor. Ali, ono što ceo proces aktivacije motora podrazumeva, mnogo je kompleksnije od okretanja ključa. Kreator automobila sakrio je kompleksnu logiku aktivacije motora od nas i dao nam pristup njegovim najbitnijim funkcijama.

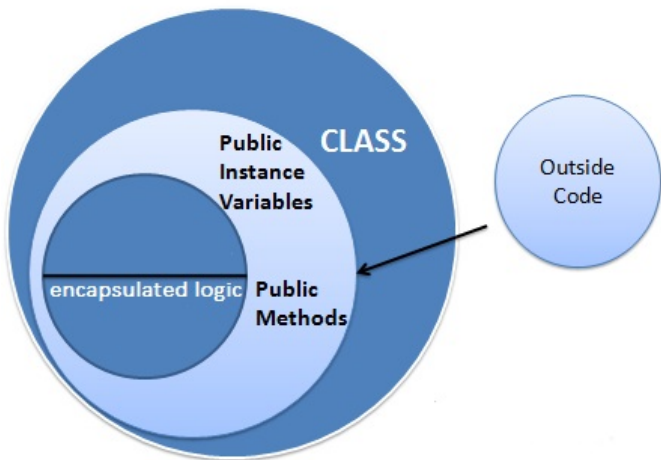
Nije ga briga ko je za volanom



Nije ga briga šta se događa u pozadini

# Enkapsulacija

- Ako postoji klasa Person, i poseduje metodu run, tada najverovatnije nećemo hteti da znamo šta se u toj metodi događa, već hoćemo samo da vidimo krajnji rezultat, a to je trčanje.



```
somePerson = Person()  
somePerson.run()
```

Ne zanima nas kako ovaj metod funkcioniše, sve dok nam daje željene rezultate

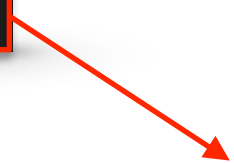
# Enkapsulacija

---

- Metode i svojstva ugrađenih Python klasa upravo oslikavaju primer enkapsulacije. Mi u našim programima koristimo ove metode i svojstva bez ikakvog znanja o načinu njihovog funkcionisanja.

*main.py*

```
import sys  
sys.exit(0)
```



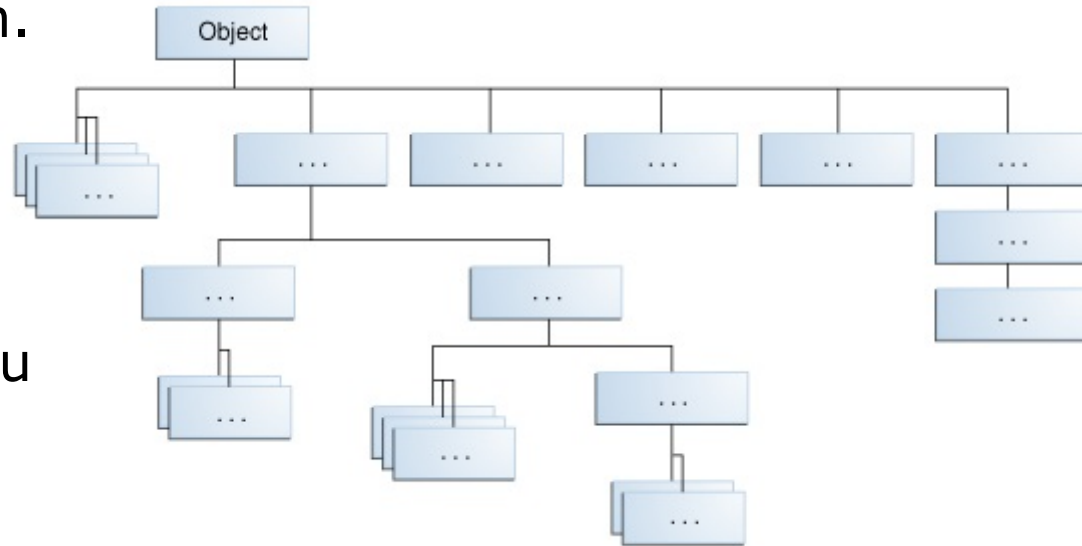
*sys.pyi*

```
def exit(arg: object = ...) -> NoReturn:  
    raise SystemExit()
```



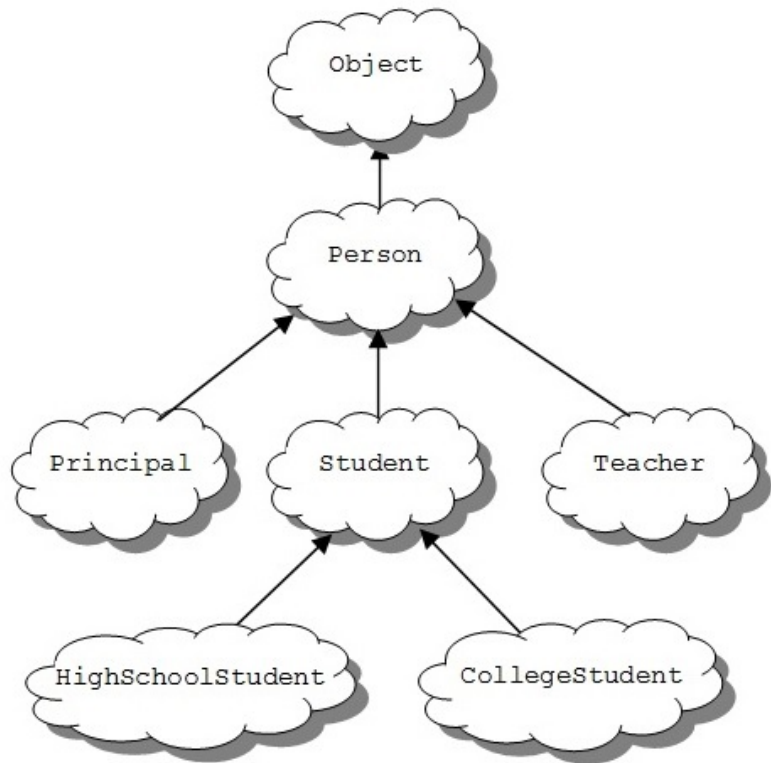
# Nasleđivanje (inheritance)

- Nasleđivanje definiše odnos između klasa, a samim tim i objekata u jednom objektno-orientisanom sistemu.
- Sve klase u Python-u imaju neku drugu klasu iznad, koja se može nazvati njihovim roditeljem.
- Specijalna klasa, jedina koja nema svog roditelja je klasa koja se zove object.
- Svi drugi objekti u Python-u predstavljaju naslednike ovog tipa.



# Nasleđivanje (inheritance)

- Možemo reći da smo svi mi instance jedne klase Osoba (Person). Ipak svako od nas može obavljati različite poslove, pa se na toj osnovi mogu kreirati novi skupovi osoba u zavisnosti od posla koji obavljaju (Principal, Student, Teacher).
- Student može biti na visokoj školi ili koledžu. Ako bismo to preveli na klasni model objektno-orijentisanog jezika, može se reći da klasa HighSchoolStudent nasleđuje klasu Student, a da klasa Student nasleđuje klasu Person.
- Svaka klasa koja nema kreiranog roditelja, kao svoju nadklasu ima klasu object. Ova struktura je prikazana na ilustraciji.



# Ikonice

---

<https://www.flaticon.com/authors/freepik>