



Priprema za završni pismeni ispit

Python developer

Što sadrži termin pripreme za ispit?

- Temelji računalnog razmišljanja, primjena elemenata računalnog razmišljanja i pseudokod.
- Povijest Pythona i uvod u programski jezik Python.
- Varijable, tipovi podataka, funkcije u Pythonu i upravljanje greškama u programskom kôdu.
- Moduli u Pythonu.
- Baze podataka i SQL.
- Python u području IoT.
- Python u podatkovnoj znanosti.

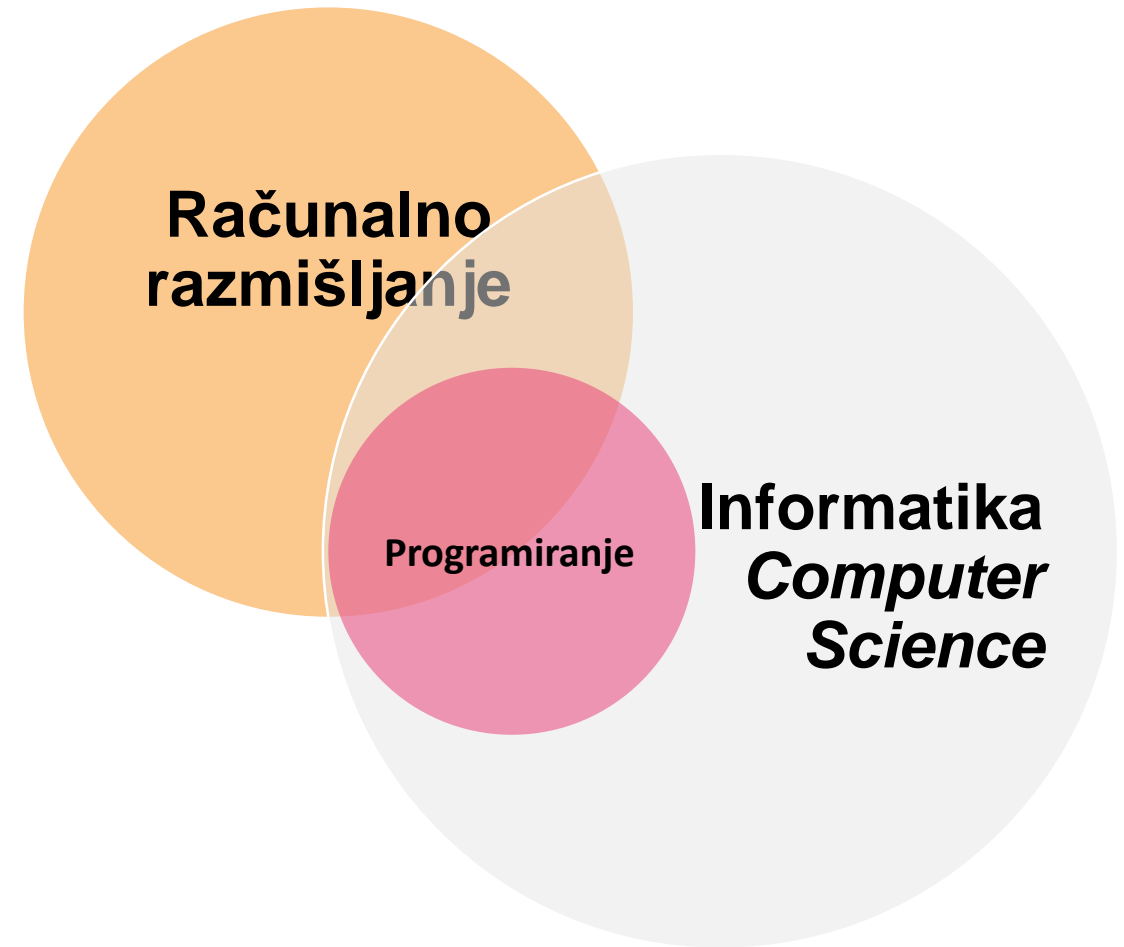
Važno!

- Jednako kao na nastavi, kako biste riješili određene zadatke na ispitu koji vas očekuje koristit ćete IDE alat Visual Studio Code.
- Rješenja iz VSC-a kopirat ćete u ispitnu aplikaciju.
 - Više o navedenom možete pronaći u “*Uputama za pismeni završni ispit*”.

Uvod u računalno razmišljanje

Prisjetimo se: Što je kodiranje, a što programiranje?

- **Kodiranje:** pisanje programskog koda na osnovu prethodno dobivenih instrukcija, odnosno zahtjeva.
- **Programiranje:** širi pojam koji obuhvaća kodiranje, ali i razradu instrukcija, odnosno predstavlja razradu svih potrebnih koraka koji će dovesti do rješenja problema.
- *Kodiranje je dio programiranja.*



Računalno razmišljanje

- Računalno razmišljanje je način rješavanja problema koji se može primijeniti na rješavanje problema iz života, ne samo problema povezanih s računarstvom.
- To je misaoni proces tijekom kojeg definiramo problem te njegove manje dijelove na način da se rješenje može opisati kao slijed jednoznačno definiranih koraka.
- Računalnim razmišljanjem možemo doći do rješenja za kompleksni problem te isti predstaviti na način kojeg mogu razumjeti osobe, računala ili oboje.

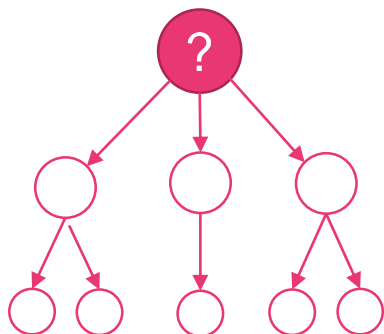
Koji su ključni elementi računalnog razmišljanja?

Računalno razmišljanje - ključni elementi

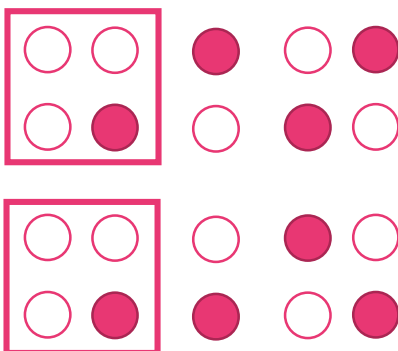
- **Dekompozicija** – rastavi problem na manje djelove.
- **Uočavanje uzoraka** - uočí ponavljanja i sličnosti.
- **Apstrakcija** – identifikacija i fokus samo na važne elemente rješenja.
- **Algoritam** – uputa korak po korak kako doći do rješenja.

* *Provjera kvalitete rješenja* - ne ulazi u osnovne elemente, ali je nužan korak za kvalitetno i ispravno rješavanje problema.

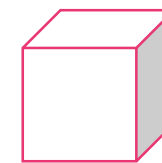
Računalno razmišljanje



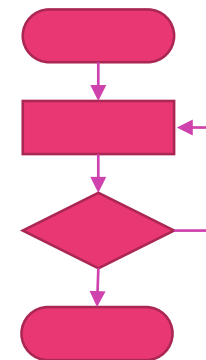
Rastavljanje



Uzorak



Apstrakcija



Algoritam

Zadatak

Što od navedenog nije točno u vezi računalnog razmišljanja?
(više je odgovora točno)

- A) Računalno razmišljanje je najbolje za programe razvijene u Pythonu, ali može se koristiti i uz druge programske jezike.
- B) Računalno razmišljanje je primjenjivo samo na probleme za koje se mogu izraditi računalni programi.
- C) Računalno razmišljanje se može primjenjivati za rješavanje svakodnevnih problema.
- D) Računalno razmišljanje zahtijeva razmišljanje kao računalno, odnosno na način da se sve može izračunati.

Zadatok

Pseudo kod

- Programski kod koji koristi slobodan jezik kako bi opisao neki problem ili algoritam.
- Nije vezan za određeni programski jezik.
- Pseudo kôd može sličiti programskom kôdu nekog programskog jezika, ali pseudo kôd NEMA definiranu sintaksu.

Važno je zapamtiti da se programski kôd piše po točno definiranim pravilima i može se pokretati i izvršavati na računalu, dok pseudo kôd nema nikakva pravila.

Pseudo kod - primjer

Zadatak:

Napisati pseudokod za komponentu registracije korisnika na neku aplikaciju (registracija koristi ime, prezime, email, username i password). Sva navedena polja su obvezna.

Razmisliti i napraviti određene provjere kako bi utvrdili da su svi podaci valjani i da je registracija moguća.

```
pocetak
upisi ime
upisi prezime
dok nije emailIspravan
    upisi email
    ako je email validan i email ne postoji u bazi podataka
        emailIspravan
    inace
        ispisi grešku za korisnika da se upisan email ne može registrirati
dok nije usernameIspravan
    upisi username
    ako username ne postoji u bazi:
        usernameIspravan
    inace
        ispisi neispravn unos za username jer vec postoji u bazi podataka
dok nije passwordIspravan
    upisi password
    upisi ponoviPassword
    ako je password = ponoviPassword i password validan
        passwordIspravan
    inace
        ispisi poruku da password ne sadrzava obavezne znakove, duljinu ili se
        passwordi ne podudaraju
registraj korisnika
kraj
```

Osnove programiranja u Pythonu

O Python-u

- Python je programski jezik opće namjene.
- Nastao je krajem 80-tih godina prošlog stoljeća, a prva javna inačica objavljena je 1991. godine.
- Osmislio ga je programer **Guido van Rossum** u Nizozemskoj.
- Ime Python proizašlo je iz televizijske serije **Monty Python's Flying Circus**.



Python PEP

- **Python Enhancement Proposals**

- skup dokumenta i pravila koji definiraju dizajn, pravila i konvenciju pisanja Python koda napisanih za Python korisnike.

Varijable i tipovi podataka

- Varijable – predstavljaju skup podataka korištenih u programskom kodu, a predstavljaju **"human readable"** nazive memorije računala kao mjesto gdje se informacija tj. podatak čuva tijekom izvođenja koda.
- Varijablu čine:
 - Tip (npr. String, Integer, ...)
 - Podatak (npr. stvarna vrijednost "neki tekst", 5, ...)
 - Naziv (naziv varijable definiran od osobe koja razvija)
 - **Ne smije početi brojem; Ne smije biti ključna riječ iz Pythona; Ne smije imati razmak; Ne smije imati specijalne znakove !, ?, #, @ itd.**

Naziv varijable

- Naziv varijable:
 - NE smije biti isti kao neka od ključnih riječi
 - NE smije početi brojkom
 - NE smije imati razmak
 - NE smije imati posebne znakove: !, ?, @, #, \$, %, ...

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Varijable u Pythonu – naziv varijable

- Preporuke za imenovanje varijabli.
- Naziv varijable bi trebao:
 - Opisivati podatak koji je pohranjen u varijabli
 - Imati znak „_” (podvlaka ili underscore) umjesto razmaka.
*Opcionalno bez razmaka tako da svaka riječ počinje velikim slovom *camelCase*
 - Imati sve znakove napisane malim slovima
 - Bez (hrvatskih) dijakritičkih znakova
 - Zadržati dosljednost pa kako je varijabla dio programskog kôda, bolje je za naziv varijable koristiti engleski jezik, ali nikako NE miješati malo engleski, malo hrvatski ili neki drugi jezik.

Naziv varijable	Podatak koji čuva
\$x	'Patar Perić' – NE
ime_prezime	'Petar Perić' – DA
A	23,2° C – NE
unutarnja_temperatura	28,9° C – DA
vanjska_temperatura	23,2° C – DA
polumjer kružnice	24.45 – NE
polumjerKruznice*	52 – DA

Naslov



Zadatak

- *Napišite program za unos podataka za registraciju korisnika. Registracija: ime, prezime, email, username i password.*
- *Svaku vrijednost ispišite u novi red tako da prvo ispišete opis podatka, a zatim njegovu vrijednost.*

* Za rješenje zadatka koristiti VSC.

```
ime = input("Unesite ime: ")
prezime = input("Unesite prezime: ")
email = input("Unesite email: ")
username = input("Unesite username: ")
password = input("Unesite password: ")

# Primjer 1
print("Ime: " + ime)
print("Prezime: " + prezime)
print("Email: " + email)
print("Username: " + username)
print("Password: " + password)
print("*" * 50)

# Primjer 2
print("Ime: {}\nPrezime: {}\nEmail: {}\nUsername: {}\nPassword: {}".format(ime, prezime, email, username, password))
print("*" * 50)

# Primjer 3
print(f"Ime: {ime}\nPrezime: {prezime}\nEmail: {email}\nUsername: {username}\nPassword: {password}")
print("*" * 50)
```

Zadatak

- Kreirajte kolekciju podataka o očitanju s meteo bazne stanice: Meteo: temperatura, tlak, vlaga, zrak-kvaliteta. Svaku vrijednost ispišite u novi red, tako da prvo ispišete opis podatka, a zatim njegovu vrijednost.*

```
meteo = {  
    "temperatura": 23.3,  
    "tlak": 1009,  
    "vlaga": 60,  
    "zrak-kvaliteta": "loša"  
}  
  
# Primjer 1  
  
print(f"Temperatura: {meteo['temperatura']}")  
print(f"Tlak: {meteo['tlak']}")  
print(f"Vlaga: {meteo['vlaga']}")  
print(f"Zrak-kvaliteta: {meteo['zrak-kvaliteta']}")  
print("*" * 50)  
  
# Primjer 2  
temperatura = "temperatura"  
tlak = "tlak"  
vlaga = "vlaga"  
kvaliteta = "zrak-kvaliteta"  
  
print(f"{temperatura.capitalize()}: {meteo[temperatura]}\n{tlak.capitalize()}: {meteo[tlak]}\n"  
      f"{vlaga.capitalize()}: {meteo[vlaga]}\n{kvaliteta.capitalize()}: {meteo[kvaliteta]}")  
print("*" * 50)  
  
# Primjer 3  
for kljuc in meteo.keys():  
    print(f"{kljuc.capitalize()}: {meteo[kljuc]}")  
print("*" * 50)
```

Programiranje u Pythonu

Klase i objekti

- **Klasa** - korisnički definirani tip podatka kojim se modeliraju objekti sličnih svojstva.
 - Predstavlja predložak, nacrt na temelju kojeg će se definirati varijable unutar programskog koda.
- **Objekt** – stvarna instanca neke klase (koja se nalazi u memoriji).
 - Svaki objekt je definiran stanjem i ponašanjem definiranim u klasi.

Zadatak

- *Kreirajte klasu koja čuva podatke o računalu:
Svako računalo ima podatke o:*
 - *procesor*
 - *količina RAM memorije*
 - *vrsta diska*
 - *veličina diska*
 - *vrsta (prijenosno/stolno)*
 - *externa grafička kartica (bool)*

*Omogućiti jednostavni tekstualni prikaz (reprezentaciju) objekta računala u konzoli
Izmjenu količine RAM memorije i diska riješiti koristeći metode za ažuriranje RAM-a,
odnosno ažuriranje diska (vrsta diska i količina memorije).*

Rješenje

```
from enum import Enum

class VrstaDiska(Enum):
    SSD = "SSD"
    HDD = "HDD"

class VrstaRacunala(Enum):
    PRIJENOSNO = "PRIJENOSNO"
    STOLNO = "STOLNO"

class Racunalo:

    def __init__(self, procesor, ram, disk: VrstaDiska, diskSize, vrsta: VrstaRacunala, imaExternuGraficku: bool):
        self.procesor = procesor
        self._ram = ram
        self._disk = disk
        self._diskSize = diskSize
        self.vrsta = vrsta
        self.graficka = imaExternuGraficku

    def azurirajRAMmemoriju(self, kolicina):
        self._ram = kolicina

    def azurirajDisk(self, disk: VrstaDiska=None, diskSize=None):
        if disk is not None:
            self._disk = disk

        if diskSize is not None:
            self._diskSize = diskSize

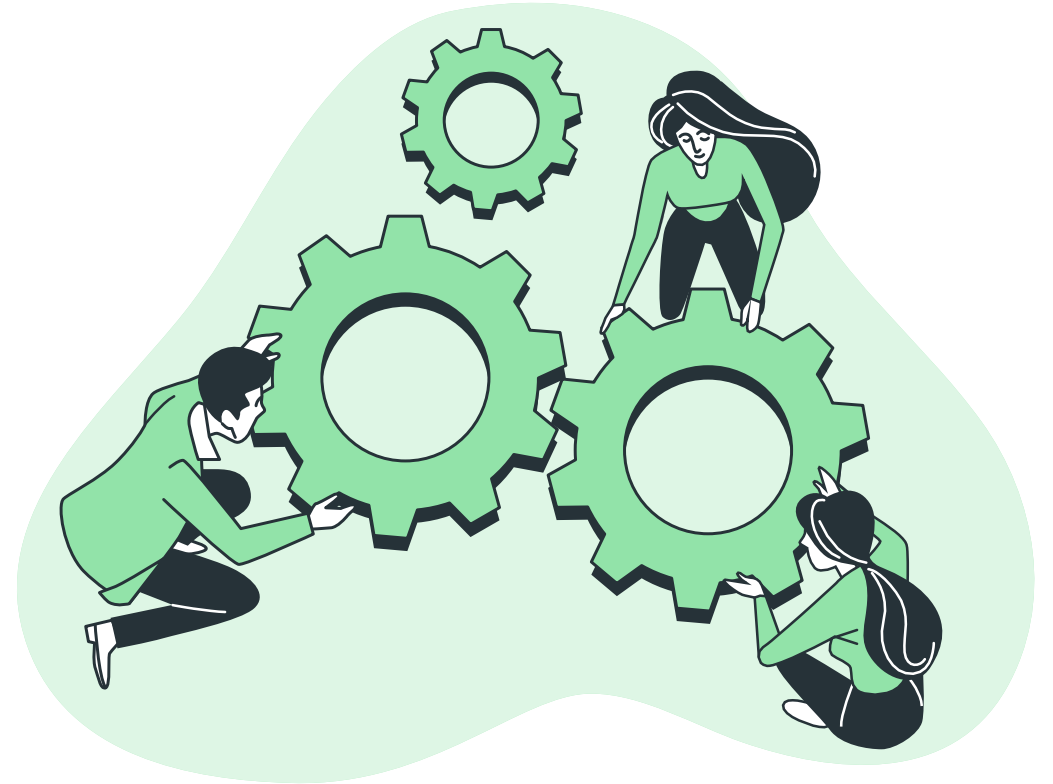
    def __repr__(self):
        return f"Racunalo(" \
            f"{self.vrsta.value}; procesor: {self.procesor}, RAM: {self._ram}, " \
            f"Disk: {self._disk.value}/{self._diskSize}, externa graficka: {'ima' if self.graficka else 'nema'}" \
            f")"
```

```
racunalo1 = Racunalo("Intel i5", 16, VrstaDiska.SSD, 512, VrstaRacunala.PRIJENOSNO, False)
racunalo2 = Racunalo("Ryzen 7", 16, VrstaDiska.HDD, 512, VrstaRacunala.STOLNO, True)
racunalo1.azurirajRAMmemoriju(32)
racunalo1.azurirajDisk(diskSize=1024)
racunalo2.azurirajDisk(disk=VrstaDiska.SSD, diskSize=1024)

print(racunalo1)
print(racunalo2)
```

Python moduli

- Moduli u Python-u omogućavaju odvajanje funkcionalnosti u zasebne cijeline i datoteke
 - Modul može biti jedna datoteka ili jedan direktorij
 - Postoje ugrađeni moduli (dolaze s Python-om), module možemo kreirati mi, ali mogu biti i skinuti s interneta (npr. pip)
 - Module koristimo kako bi imali bolje organiziran kod koje možemo tako lakše ispravljati, proširiti, debuggirati i lakše se snalaziti u istom



Programiranje u Pythonu

– rad s bazama podataka

SQL

Structured Query Language

- Upitni programski jezik visoke razine.
- Dizajniran za rad s bazama podataka.
- '*' kao i u Pythonu je zamjena za "sve".

• CRUD

```
INSERT INTO [imeTablice] (kolona1, kolona2, kolona3)  
VALUES (vrijednostKolona1, vrijednostKolona2, vrijednostKolona3);
```

```
SELECT [kolona1], [kolona3] FROM [imeTablice];  
SELECT * FROM [imeTablice];
```

```
UPDATE [imeTablice] SET kolona1=novaVrijednostKolona1,  
kolona3=novaVrijednostKolona3 WHERE kolona2=vrijednostKolona2;
```

```
DELETE FROM [imeTablice] WHERE kolona2=vrijednostKolona2;
```

SQLite

- SQLite – mala, jednostavna, brza, SQL Database aplikacija.
- Ne traži instalaciju.
- Uključen u Python instalaciju i većinu Linux distribucija, uključujući i MacOS.



Zadatok

Zadatak

**Koje dva osnovna tipa baza podataka se danas najčešće koriste?
Opišite prednosti i mane svakog od njih.**

Dva najzastupljenija tipa baza podataka su:

- *Relacijske baze - podaci su podijeljeni u tabele i međudobno su povezani relacijama u obliku nekih identifikatora. Podaci su konzistentni, ne ponavljaju se, pohrana podataka je otporna na ispade sustava*
- *NoSQL baze - to su baze primarno namijenjene za visoke performanse, pohranjuju podatke kao cjelinu, bez da ih podijele po tabelama i povežu relacijama, podaci su pohranjeni u drugačije oblike od tabela.*

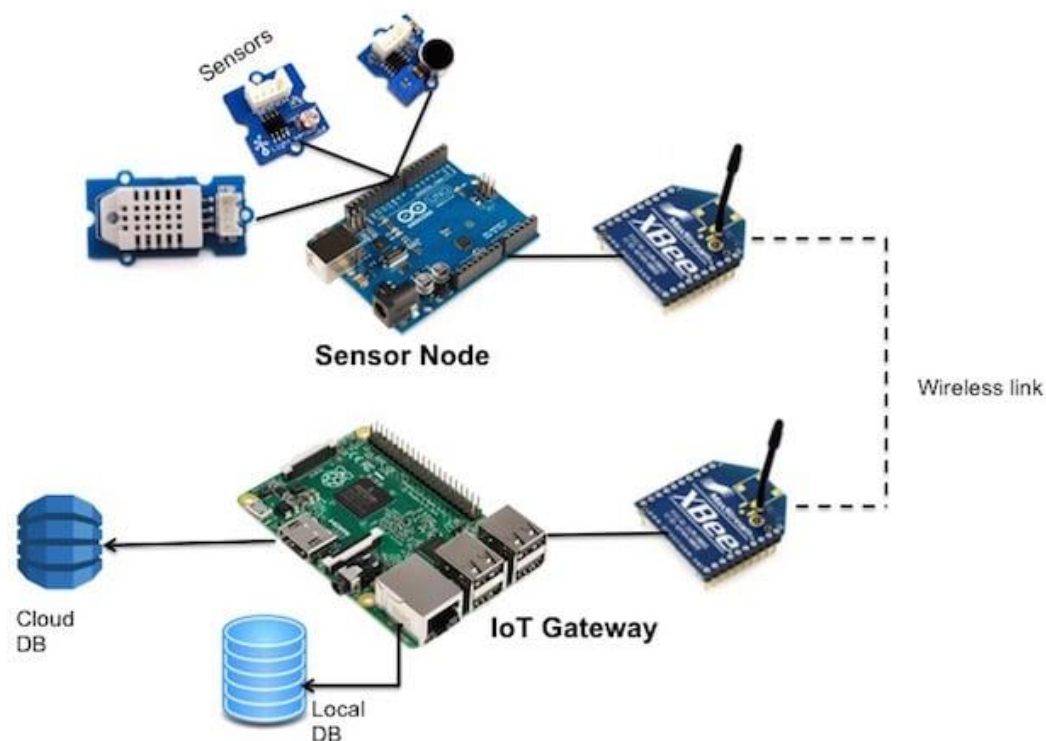
Python u području IoT

**Koja su područja primjene IoT-a?
Koja su najpoznatija IoT računala?**

IoT – *Internet of Things*

- Naziv za mrežu elektroničkih sustava povezanih putem interneta.
- Sustavi se sastoje od hardware-a i software-a.
- Hardware koji se koristi u IoT-u:
 - Microcontroller (npr. Arduino) ili Single-board computer (npr. Raspberry Pi)
 - Senzori (PIR, ultrazvuk, temperatura, tlak, vlaga i sl.)
 - Radio veza (radio moduli – npr. LoRa, Sigfox, XBee)
 - Internet (cloud – sa softwareom).

IoT – primjer sustava - flow



- Microcontroller (npr. Arduino) na koji su spojeni senzori (žicom) ili napravljen custom PCB sa zalemljenim senzorima na pločici.
- Najčešće upogonjen baterijom gdje je potrebno paziti kako napraviti implementaciju zbog štednje energije.
- S obzirom da takvih uređaja može biti mnogo u sustavu i da spajanje svakog od ovakvih uređaja na internet može biti skupo ili nemoguće koristi se radio modul specifičnih frekvencija gdje putem radio signala svi uređaji (Node-ovi) šalju putem radio signala digitalne podatke pretvorene iz fizikalnih veličina (senzori) do centralnog mjesta (collector/gateway) najčešće Raspberry Pi koji se napaja iz električne mreže i ima konekciju na Internet, da distribuira podatke u Cloud na obradu (od svih Node-ova iz sustava).

IoT – Sense HAT

- Kad nismo u mogućnosti raditi s hardware-om postoje razni simulatori s kojima možemo učiti kako rad s hardwareom funkcionira - otprilike :)
- Za zamjenu Raspberry Pi-a koristimo RaspbianOS Desktop inačicu (**RaspbianOS** kao takav, je stvarni operacijski sustav koji se koristi na Raspberry Pi uređajima, samo je prilagođen za drugačiju infrastrukturu ARM).
- Sense HAT – postoji kao stvaran hardware, ali mi koristimo emuliranu software-sku inačicu.

- Da bi Sei

- from

```
from sense_emu import SenseHat
```

- Gdje pot

- Možemo

- humi

```
hat = SenseHat()
```

```
while True:
```

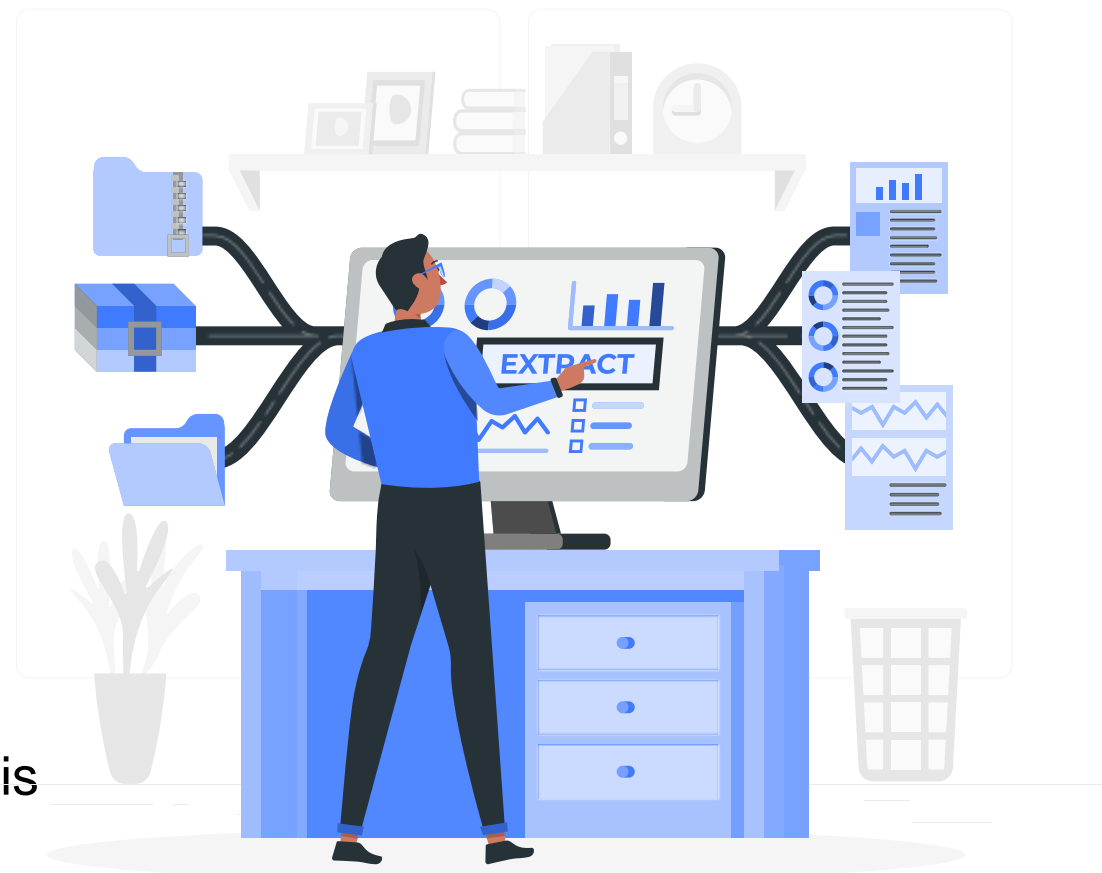
```
    hat.show_message("Ime Prezime")
```

Zadatak

Python u podatkovnoj znanosti

Python u Data science-u

- Python je danas jedan od češće korištenih programskih jezika u podatkovnoj znanosti.
- Podatkovna znanost uči nas radu s podacima, točnije, kako:
 - Prilagoditi format podataka
 - Očistiti podatke
 - Uzorkovati podatke prema odgovarajućoj skupini
 - Komunicirati rezultate kroz vizualizaciju, opis i sažetu interpretaciju rezultata.



Python DS - moduli

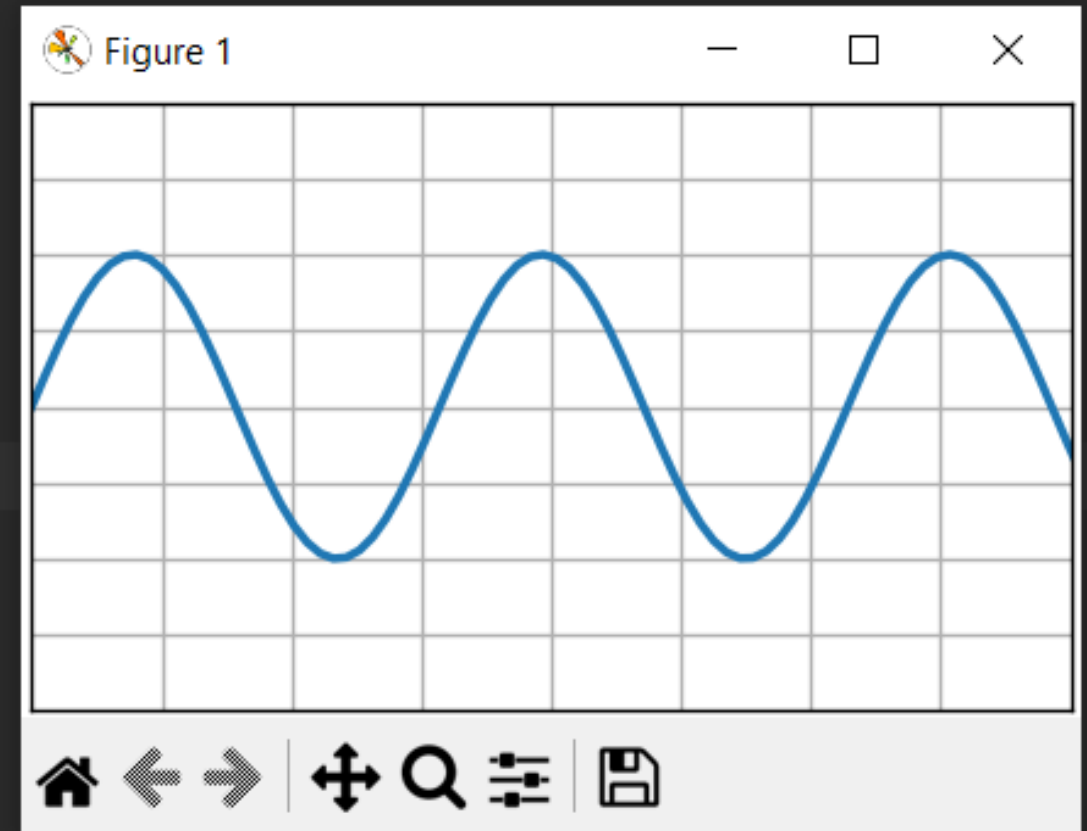
- Pandas
 - Naziv proizašao iz riječi Panel Data
 - Alat za analiziranje podataka u Pythonu
 - Podatke sprema u DataFrame, a zanimljiv je jer u DataFrame možemo ubaciti podatke iz većine poznatih data formata (SQL baza, Excel, CSV i sl.).
- Matplotlib
 - Koristimo za vizualizaciju i izradu grafova
 - Jednostavna upraba
 - Grafovi visoke kvalitete
 - Moguća ugradnja u bilo koji Python GUI.

```
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('_mpl-gallery')

x = np.linspace(0, 10, 100)
y = 4 + 2 * np.sin(2 * x)
fig, ax = plt.subplots()
ax.plot(x, y, linewidth=2.0)
ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```

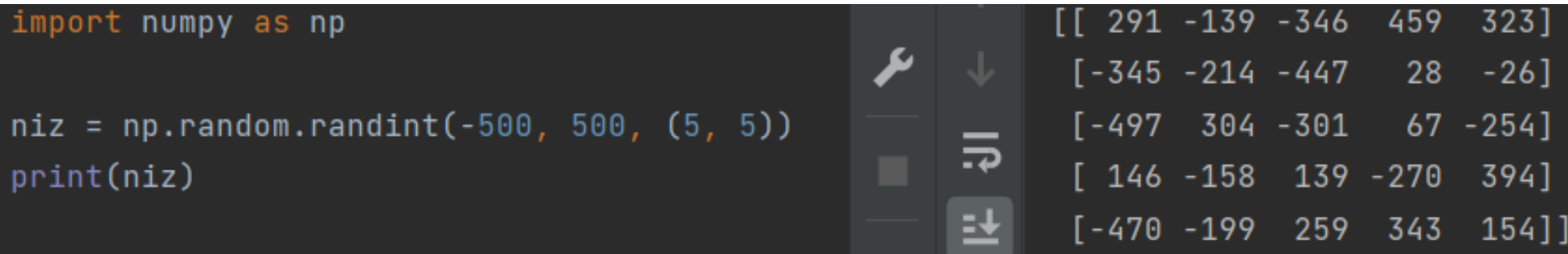


Python DS - moduli

- Numpy
 - Numerical Python
 - Temeljni paket za znanstvene proračune
 - Zanimljiv jer daje lakoću pisanja koja dolazi iz high-level jezika Python, sa snagom i brzinom low-level jezika (poput C-a).
- **Primjer: generiraj matricu 5x5 ispunjenu s nasumičnim brojevima u rasponu -500, 500**

```
import numpy as np

niz = np.random.randint(-500, 500, (5, 5))
print(niz)
```



291	-139	-346	459	323
-345	-214	-447	28	-26
-497	304	-301	67	-254
146	-158	139	-270	394
-470	-199	259	343	154

Zadatak

Koja je razlika između Pandas i NumPy alata?

Pandas je Python biblioteka za obradu heterogenih tipova podataka, a NumPy je namijenjen za numeričke proračune pa je orijentiran na numeričke tipove podataka.

Zadatok

Pitanja?





Hvala na
pažnji!