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.

Računalno razmišljanje - ključni elementi

• Dekompozicija – rastavi problem na manje djelove.

• Uočavanje uzoraka - uoči 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.

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.

Računalno razmišljanje - Primjer: otvaranje trgovine s IT i gaming opremom

```
- otvaranje obrta
    - sakupljanje informacija (računovodstvo, knjigovodstvo)
        - fiskalizacija
        - software za fiskalizaciju
    - sakupljanje informacija o obavezama samostalnog zaposlenja
    - papirologija
- pronalazak prostora
    - prema lokaciji
        - target korisnika
            - mlađi ljudi
            - blizina škola, fakulteta
        - lokacije s gustom naseljenosti
    - prema količini skladišnog prostora
        - određivanje količine komponenta prema popularnosti
- nabava stvari potrebnih za rad
    - blagajna
    - software
    - fiskalizacija
        - gore navedene stvari provjeriti s dobavljačem komponenta?
- nabava komponenti za prodaju
    - popularnost komponenta
    - dobavljač
        - brzina isporuke
        - dostupnost količina
- reklama:
    - program lojalnosti
    - izrada letaka
        - slike
    - izrada web stranice (web shop?)
        - slike
```

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 neispravan unos za username jer vec postoji u bazi podataka


dok nije passwordIspravan
      upisi password
      upisi ponoviPassword
      ako je password = ponviPassword I password validan
            passwordIspravan
      inace
            ispisi poruku da password ne sadrzava obavezne znakove, duljinu ili
            se passwordi ne podudaraju

registraj korisnika
kraj
```

Zadatak: Pseudo kod

• Napisati pseudo kod za komponentu dohvata vrijednosti s meteo stanice (temperatura, tlak i vlaga) koje dolaze u CSV obliku iz modula MeteoService.
• Provjeriti ispravnosti podataka i spremiti ih u txt datoteku.
• Pripaziti na provjere!

```
Pocetak

meteoPodaciRaw = meteoService dohvati podatke
temperatura, vlaga, tlak = podijeli vrijednosti po ','

ako temperatura nije prazna i temperatura je decimalni broj:
      temperatura isprivna

ako vlaga nije prazna i nalazi se u rasponu 0-100:
      vlaga ispravna

ako tlak nije prazan i tlak je u rasponu 900-1100:
      tlak ispravan
ako su temperatura, vlaga i tlak ispravni:
      ako file za zapis postoji:
            appendaj file s meteoPodaciRaw
      inace:
            kreiraj file i zapisi podatke
inace:
      ispisi „podaci nisu ispravni i ne uzimamo ih u analizu“
kraj
```

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 rješavanjesvakodnevnih problema.
D) Računalno razmišljanje zahtijeva razmišljanje kao računalo, odnosno na način da se sve može izračunati.

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.
• Postoje tri različita tipa:
    • **Standards** – opisuju nove značajke ili implementacije u Pythonu
    • **Informational** – smjernice i informacije za korisnike
    • **Process** - objašnjenje procesa u Pythonu

Varijable i tipovi podataka

• Varijable – predstavljaju skup podataka korištenih u programskom kodu, a predstavlju "human readable" nazive memorije računala kao mjesto gdje se informacija tj. Podatak čuva tijekom izvođenja koda.
• Što čini varijablu?
    • **Tip** (npr. String, Integer, …)
    • **Podatak** (npr. stvarna vrijednost "neki tekst", 5, …)
    • **Naziv** (naziv varijable definiran od osobe koja razvija)

• 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	pass	class	from
print	continue	global	raise	def
if	return	del	import	try
elif	in	while	else	is
with	except	lambda	yield	for

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

```
1Broj: int = 5    krivo
broj1: int = 5    točno
Broj1: int = 5    točno

!broj: int = 5    krivo
Broj!: int = 5    krivo
broj@: int = 5    krivo

prviBroj: int = 5      točno
prvi_broj: int = 5     točno
prvi broj: int = 5     krivo (razmak)
prvi-broj: int = 5     krivo zbog -
is: int = 5            krivo, ključna riječ
```

Tipovi podatka

- **Osnovni tipovi podataka u Python-u**
  - String (str) - tekst, pise se unutar ' ili " navodnika
    ```
    tekst: str = 'Ovo je tekst'
    ```
  - Integer (int) - cijeli brojevi
    ```
    broj: int = 12
    ```
  - Float (float) - decimalni brojevi
    ```
    decimalniBroj: float = 123.4
    ```
  - Boolean (bool) - True/False
    ```
    istina: bool = True
    ```
- **Kolekcije**
  - Objekti koji mogu poprimiti više različitih objekta i spremiti ih kao cijelinu.
  - Omogućuju dohvat i iteraciju kroz objekte da bi im mogli pristupiti.
- **Lista**
  - Definira se preko uglatih zagrada [ ] ili pozivom konstruktora list()
  - Promjenjiv tip podatka (možemo naknadno dodavati, mjenjati ili brisati vrijednosti).
- **Tuple**
  - Definira se preko ( ) zagrada i nepromjenjiv je.
- **Set**
  - Definira se preko { } zagrada ili pozivom konstruktora set()
  - Set, za razliku od liste, odbija duplikate (ne može spremiti iste vrijednosti ili obriše iste vrijednosti ako je kreiran iz liste)
- **Dictionary**
  - Definira se preko { } zagrada ili pozivom konstruktora dict()

  *Napomena: iako se i dict i set definiraju preko { } zagrada, razlikuju se po tome da set sprema vrijednosti preko indexa (kao i lista i tuple), dok dict koristi key-value.

```
listaBrojeva = list()
listaBrojeva.append(1)
listaBrojeva.append(2)
listaBrojeva.append(2)

setBrojeva = set(listaBrojeva)
print(f'Lista: {listaBrojeva}, Set: {setBrojeva)')

tuplePrimjer = ('ponedjeljak', 'utorak', 'srijeda', 'četvrtak', 'petak',
'subota', 'nedjelja')
#tuple koristimo za podatke koje ne mjenjamo i nemamo potrebe mjenjati

dictionaryOsoba = {'ime':'Pero', 'prezime':'Perić'}
```

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('Upisite ime: ')
prezime = input('Upisite prezime: ')
email = input('Upisite email: ')
username = input('Upisite username: ')
password = input('Upisite password: ')
```

Primjer1

```
print('Ime: ' +  ime)
print('Prezime: ' +  prezime)
print('Email: ' +  email)
print('Username: ' +  username)
print('Password: ' +  password)
print('*' * 50)
```

Primjer2

```
print('Ime: {}\nPrezime: {}\nEmail: {}\nUsername: {}\nPassword: {}'.format(ime,
prezime, email, username, password))
```

Primjer3

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

Zadatak

• Kreirajte kolekciju (dict) podataka o očitanju s meteo bazne stanice s preddefiniranim proizvoljnim vrijednostima: 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'Temperatura: {meteo['vlaga']}')
print(f'Temperatura: {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{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)
```

Objektno orijentirano programiranje (OOP)

• Jedan od pristupa računalnog programiranja.
• Princip programiranja gdje je rješenje bazirano na skupu objekta koji međusobno komuniciraju i na taj način riješavaju zadan problem.
• OOP se ističe jasnim i definiranim arhitekturama aplikacije, jasnim i čitljivim kodom, malom količinom redundancije, odnosno velikom količinom reusabilnog koda

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.

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 Pythonom).
• Module možemo kreirati mi, ali mogu biti i skinuti s interneta (npr. pip).
• Module koristimo kako bismo imali bolje orgraniziran kod koje možemo tako lakše ispravljati, proširiti, debuggirati i lakše se snalaziti u istom.

Zadatak
• Kreirajte klasu Vozilo:
    • Atributi:
        • jeUpaljeno: bool (default: False)
    • Metode:

- upali()
- vozi(brzina: int)

• Metoda upali() postavlja atribut jeUpaljeno na True
• Metoda vozi ispisuje poruku "Vozilo putuje brzinom {brzina} km/h" ako je vozilo upaljeno, ako vozilo nije upaljeno, a pozvana je metoda vozi() ispisati poruku, "Vozilo nije upaljeno“

```
class Vozilo:

    def __init__(self):
        self.jeUpaljeno = False

    def vozi(self, brzina):
        if self.jeUpaljeno:
            print(f'Vozilo putuje brzinom {brzina} km/h')
        else:
            print('Vozilo nije upaljeno')

    def upali(self):
        self.jeUpaljeno = True

if __name__ == '__main__':
    automobil = Vozilo()
    automobil.vozi(50)
    automobil.upali()
    automobil.vozi(50)
```

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).

```
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({self.vrsta.value}; procesor{self.procesor}, "\
               f"Ram{self._ram}, Disk: {self._disk.value}/{self._diskSize}, "\
               f"externa graficka: {'ima' if self.graficka else 'nema'}"
```

```
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(VrstaDiska.SSD, diskSize=1024)

print(racunalo1)
print(racunalo2)
```

SQLite

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

SQL

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

CRUD – create, retrieve/read, update, delete

```
INSERT INTO [imeTablice] (kolona1, kolona2, kolona3)
VALUES (vrijednosKolona1, 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;
```

Zadatak: Registracija korisnika

• Kreirati dvije tablice:
    • Tablica **gradovi** sadrži:
    • id, naziv, zip (poštanski broj)
    • Sva polja obavezna
    • Tablica **korisnici** treba sadržavati:
        • **id, ime, prezime, username, password, email, mobitel, grad_id**
        • sve polja su obavezna osim mobitela
• grad_id je relacija na tablicu gradovi
• ubaciti minimalno 2 grada i minimalno 5 korisnika (minimalno 2 korisnika u jednom gradu)
• dohvatiti sve korisnike po grad_id (koji ima vise od 1 korisnika)
• dodatno (bonus), dohvatiti sve korisnike prema nazivu grada koristeći join

```
CREATE TABLE IF NOT EXISTS gradovi (
        id INTEGER PRIMARY KEY,
        naziv TEXT NOT NULL,
        zip INTEGER NOT NULL);

INSERT INTO gradovi (naziv, zip) VALUES
 ('Rijeka', 51000),
 ('Zagreb', 10000),
 ('Split', 21000);

CREATE TABLE IF NOT EXISTS korisnici (
        id INTEGER PRIMARY KEY,
        ime TEXT NOT NULL,
        prezime TEXT NOT NULL,
        username TEXT NOT NULL UNIQUE,
        password TEXT NOT NULL,
        email TEXT NOT NULL,
        mobitel TEXT,
        grad_id INTEGER NOT NULL,
        FOREIGN KEY (grad_id) REFERENCES gradovi(id)
 );

INSERT INTO korisnici (ime, prezime, username, password, email, grad_id, mobitel)
VALUES
   ('ana', 'anic', 'aanic', 'ana123', 'ana@email.com', 1, '098123456'),
   ('ivo', 'ivic', 'iivic', 'ivo123', 'ivo@email.com', 1, '098123457'),
   ('marko', 'markic', 'mmarikic', 'marko123', 'marko@email.com', 2, '098123457'),
   ('lana', 'lanic', 'llanic', 'lana123', 'lana@email.com', 2, '098123458'),
   ('jure', 'juric', 'jjuric', 'jure123', 'jure@email.com', 3, '098123466');
```

```
SELECT * FROM korisnici WHERE grad_id = 1;


SELECT ime, prezime, username FROM korisnici
INNER JOIN gradovi ON korisnici.grad_id = gradovi.id
WHERE naziv = 'Rijeka';


SELECT * FROM korisnici
WHERE grad_id == 2 AND grad_id IN (
   SELECT grad_id
   FROM korisnici
   GROUP BY grad_id
   HAVING COUNT(*) > 1);
```

- Ako treba u pythonu izvesti sve naredbe:

```
import sqlite3

conn = sqlite3.connect('database.db')

cursor = conn.cursor()
query = 'sql query ovdje'

cursor.execute(query)

result = cursor.fetchall()
cursor.close()

conn.commit() # spremiti promjene u bazu, za pisanje, kad citamo ne treba
conn.close()
```

```
select_from_table_query='''
    SELECT * FROM Employees WHERE name = ?
    '''
cursor.execute(select_from_table_query, ('Jure Jurić',))
```

```
insert_into_table_query = '''
INSERT INTO Employees (name, email)
VALUES (?, ?);
'''
```

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 međusobno su povezani relacijama u obliku nekih identifikatora. Podaci su konzistentni, ne ponavaljaju 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

IoT

Koja su područja primjene IoT-a?

• Industrija (robotizacija, automatizacija)

• Potrošači

• Trgovina, oglašavanje, financije

• Zdravstvena industrija

• Transport i logistika

• Poljoprivreda

• Okoliš i ekologija

• Energija

• Smart City, Vojska …

Koja su najpoznatija IoT računala? – Arduino, Raspberry Pi

IoT – Internet of Things

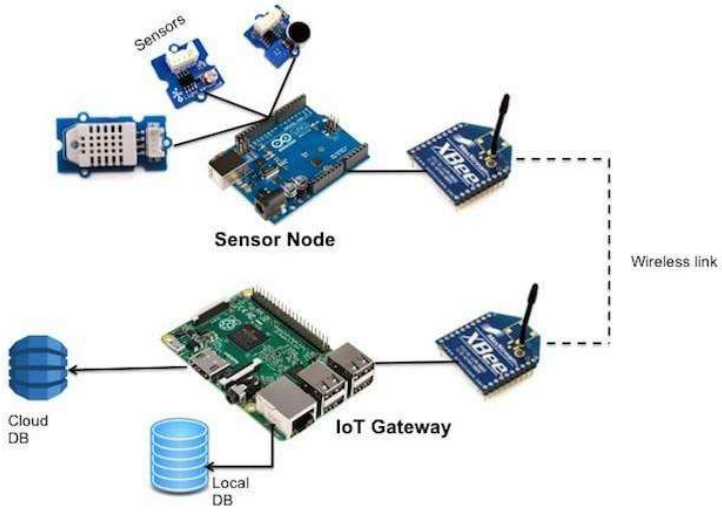
• Naziv za mrežu elektroničkih sustava povezanih putem interneta.

• Sustavi se sastoje od hardware-a i software-a.

• Komponente 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 harwareom 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 Sense HAT povezali s Python-om, importamo modul (unutar RasbianOS-a).

- from sense_emu import SenseHat

• Gdje potom možemo koristiti emulirani hardware preko SenseHat() objekta

• Možemo koristiti:

    • humidity, temperature, pressure, joystick, accelerometer, gyroscope, display

Zadatak

Napišite program koji će beskonačno ispisivati vaše ime i prezime na ekranu SenseHAT emulatora.

```
from sense_emu import SenseHat
hat = SenseHat()
while True:
      hat.show_message('Ime Prezime')
```

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 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

xpoints = np.array([1,8])
xpoints = np.array([3,10])
plt.plot(xpoints, ypoints)
plt.show()
```

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

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

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.

Zadatak

- Pomoću NumPy-a generirati 100 nasumičnih brojeva u rasponu 1 – 100 i ispisati prosjek istih.

```
import numpy as np

niz = np.random.randint(0,100,100)
print(niz)
print(np.average(niz))
```