Workplace and Process Safety in Next Generation Europe - Teaching for Learning Safety4EU

Project No. 101126832

Coordinator: Dr Milan Protić

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Safety4EU projekat
L3. Analiza podataka, programiranje i interpretacija

Uvod u Pajton

Milan Gocić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

O Pajtonu

Co-funded by the European Union

Pajton je osmislio holandski programer **Guido van Rossum** krajem osamdestih godina prošlog veka. Jezik je dobio ime po popularnoj engleskoj seriji Monty Python. Pajton je interpretirani, objektno-orijentisani jezik visokog nivoa, namenjen za pravljenje svih vrsta aplikacija – od inženjerskih i naučnih, do poslovnih i veb primena.

Dostupan je za različite operativne sisteme poput Windows-a, Linux-a i Mac OS-a.

Uz osnovnu instalaciju dobija se i besplatno razvojno okruženje IDLE (Integrated Development and Learning Environment) koje uključuje interaktivni interpreter, editor teksta i alate za testiranje koda.

Kako je Pajton jezik koji se interpretira, odgovarajuća razvojna okruženja imaju u svom sastavu ugrađen interpreter.

Zašto Pajton

Co-funded by the European Union

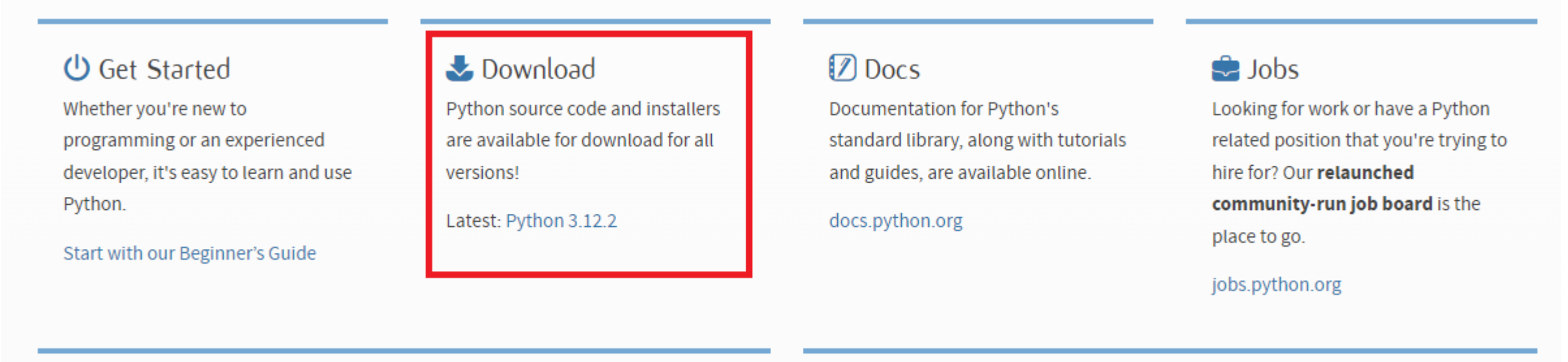
- Pajton je softver otvorenog koda, što znači da je besplatan; uključen je u većinu Linux distribucija.
- Pajton je dostupan za sve glavne operativne sisteme (Linux, Unix, WindoWs, Mac OS, i tako dalje). Program napisan na jednom sistemu radi bez modifikacija na svim sistemima.
- Pajton je lakši za učenje i proizvodi čitljiviji kod od većine programskih jezika.
- Pajton i njegove ekstenzije se lako instaliraju.

Instalacija Pajtona

Co-funded by the European Union

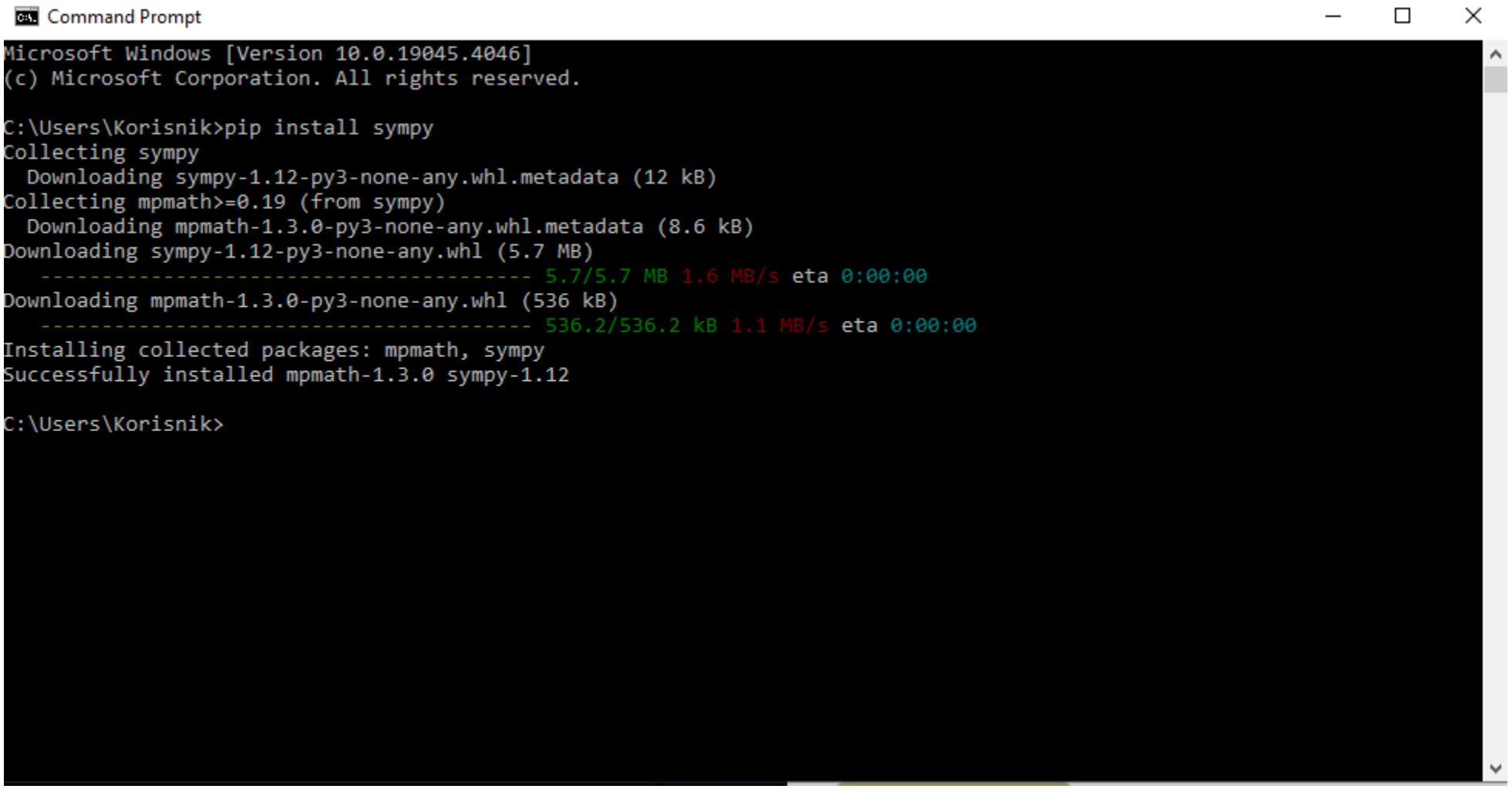
Da biste instalirali Python prvo je potrebno da odete na sajt **www.python.org/** na kome ćete naći aktuelnu verziju Python-a. Idite na sekciju „Downloads“ i tu kliknite na dugme na kome će biti aktulena verzija Python-a (u ovom slučaju verzija „Python 3.12.2“) da biste započeli sa preuzimanjem instalacionog programa za Windows.

U trenutku pisanja ovog predavanja aktuelna verzija Python-a je bila 3.12.2.



Instalacija paketa

Co-funded by the European Union



pip install sympy

Simbol >>>

Co-funded by the European Union

Simbol >>> označava tekuću liniju za unos komande koja se, po pritisku na taster ENTER, izvršava od strane interpretera, a njen rezultat ispisuje u sledećem redu.

```
>>> 1 + 2
3

a = 1
b = 2
c = 2 * (a + b)
c
6
```

Trenutne vrednosti pojedinih promenljivih mogu se prikazati navođenjem njihovog imena (npr. c). Ako se navede promenljiva - koja prethodno nije definisana, interpreter prijavljuje grešku.

```
y
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    y
NameError: name 'y' is not defined
```

Objekat

Co-funded by the European Union

Objekat je definisan svojim tipom, identitetom i vrednošću. **Tipom** se određuje skup svih vrednosti koje objekti iz tipa mogu imati, kao i skup svih operacija u kojima ti objekti mogu učestvovati. Tip objekta zove se još i klasom. Klasa predstavlja sve objekte iste vrste, na primer sve realne brojeve. **Identitet** objekta odnosi se na njegovu memorijsku adresu (lokaciju u memoriji) i ne može se promeniti pošto se objekat prvi put kreira. Tip (klasa) svakog objekta može se odrediti funkcijom **type()**, a identitet funkcijom **id()**.

```
# tip (klasa) objekta koji predstavlja broj 100
type(100)
<class 'int’>

# identitet (memorijska adresa) objekta sa vrednošću 100
id(100)
1676122721616
```

Osnovni predefinisani tipovi

Co-funded by the European Union

Tip (ili klasa)	Oznaka	Domen vrednosti	Primeri vrednosti
celobrojni	int	Celi brojevi	0, 1, -12, 23_156
realni	float	Realni brojevi	1.34, .05, -25.84
kompleksni	complex	Kompleksni brojevi	-5j, -0.8 – 2.34j
logički	bool	{False, True}	False, True
tekstualni	str	Niz znakova u Unicode-u	'Programski jezik Python'
torka	tuple	Nepromenljiv niz proizvoljnih objekata	(), (1.65, True, 'ab')
lista	list	Promenljiv niz proizvoljnih objekata	[], [1, 3], ['ab', 2, 2+6j]
skup	set	Neuređen skup proizvoljnih nepromenljivih objekata. Duplikati nisu dozvoljeni.	{}, {1, 2}, {'A', {1, 3}}
rečnik	dict	Parovi ključ-vrednost. Vrednosti proizvoljni objekti, ključevi proizvoljni nepromenljivi objekti.	{11000: 'Beograd’, 18000: 'Niš'}
nepostojeće	NoneType	{None}	None

Simbol _ može se koristiti u brojnom zapisu da poveća čitljivost za velike brojeve (23_156 je 23156).
Tipovi (klase) str, tuple, list, set i dict, predstavljaju kolekcije objekata - to su objekti koji sadrže druge objekte.

Unos i štampanje podataka

Co-funded by the European Union

Unos podataka se tretira kao tekst (niska karaktera), a ako predstavlja ispravan zapis celog ili realnog broja, on se može naknadno konvertovati u brojevnu vrednost. Jedan od načina da se tekst konvertuje u celobrojnu vrednost je funkcija **int**, a u realnu vrednost je funkcija **float**.

```
x = int(input())
y = float(input())
```

Ako se pri ovakvom učitavanju u jednoj liniji unesu dva broja razdvojena razmakom, dolazi do greške pri pretvaranju teksta u broj.

```
print('Unesite ceo broj: ')
ceo_broj = int(input())
print('Unesite realan broj: ')
realan_broj = float(input())
print('Ceo broj: ', ceo_broj, 'Realan broj: ', realan_broj)
```

Primer

Co-funded by the European Union

Izračunati težinu tela na površini Zemlje na osnovu poznate mase u kilogramima.

Algoritam:

1) uneti masu tela u kg i zapamtiti je u promenljivoj m
2) F = m * 9.81
3) prikazati F

```
# izračunava silu težine na površini Zemlje
# na osnovu mase tela u kg
m = input('Unesite masu tela u kg: ') # treba uneti broj
print('sila težine je ', float(m) * 9.81, 'N')
```

```
Unesite masu tela u kg: 5
sila težine je  49.050000000000004 N
```

je oznaka za komentar. Komentari treba da pojasne određene delove programa za osobu koja će održavati program ili nastaviti rad na njemu.

Izrazi, promenljive i dodela vrednosti

Co-funded by the European Union

Objekti se, upotrebom različitih operacija, mogu kombinovati u izraze. Rezultat izračunavanja izraza takođe predstavlja objekat određenog tipa. Uobičajeni prioritet izračunavanja može se promeniti korišćenjem zagrada.

```
O = 2 * 4.0 * 3.14
```

Promenljiva O ne sadrži objekat, već adresu memorijske lokacije na kojoj se objekat nalazi - istu onu adresu koja predstavlja identitet objekta. U tom smislu promenljiva predstavlja **objektnu referencu** putem koje se pristupa objektu.

```
O = 'A'
```

Sada se uspostavlja novo vezivanje promenljive O za objekat tekstualnog tipa čija je vrednost znak 'A'. Ovde se uočava dinamička priroda promenljivih u Pajtonu prema tipu objekta na koje ukazuju

Izrazi, promenljive i dodela vrednosti

Co-funded by the European Union

Imena promenljivih u Pajtonu mogu biti sastavljena od velikih i malih slova engleskog alfabeta, cifara i simbola _. Imena ne mogu početi cifrom i osetljiva su na velika i mala slova.

Imenovanje treba da odgovara značenju podatka (objekta) na koji promenljiva ukazuje.

Sugestija je da imena počinju malim slovima, kao i da ukoliko ime sadrži više reči, da se one razdvajaju simbolom _ (na primer, broj_indeksa).

U Pajtonu se može koristi naredba višestruke dodele vrednosti:

```
a, b, tekst = 3, 18, 'tekst'
print(a, b, tekst)
3 18 tekst
```

Na levoj strani jednakosti nalazi se lista promenljivih razdvojenih zapetom, a na desnoj, lista izraza čije se izračunate vrednosti pridružuju promenljivim.

Primer promenljive

Co-funded by the European Union

```
x = int(input('Unesite vrednost za x\n'))
proizvod = 3 * x
print(proizvod)
```

Nazivi imena grčkih promenljiva u Pajtonu

```
beta = 2.35
theta = 60.0
```

Razmena vrednosti

Co-funded by the European Union

```
a = input('Prva vrednost = ')
b = input('Druga vrednost = ')

print('pre razmene a =', a, 'b =', b)
a, b = b, a # razmena
print('posle razmene a =', a, 'b =', b)
```

```
Prva vrednost = 15
Druga vrednost = 26
pre razmene a = 15 b = 26
posle razmene a = 26 b = 15
```

Aritmetički operatori

Co-funded by the European Union

Operator	Opis
x + y	Sabiranje
x - y	Oduzimanje
x * y	Množenje
x / y	Deljenje. Rezultat je uvek float
x // y	Celobrojno deljenje (3 // 2 = 1)
x % y	Ostatak pri celobrojnom deljenju (7 % 4 = 3)
x ** y	Stepenovanje

Prioritet operatora *, /, // i % je viši od prioriteta operatora + i -, dok su svi navedeni operatori levo asocijativni (računske operacije istog prioriteta se izvode s leva na desno). Prioritet i asocijativnost se mogu promeniti navođenjem zagrada.

Primer prioriteta operatora

Co-funded by the European Union

PEMDAS (Please Excuse My Dear Aunt Sally) - parentheses, exponents, multiplication, division, addition, and subtraction

I slučaj

```
vrednost_izraza = 2 * (3 + 2)**2 / 5 - 4
print(vrednost_izraza) # štampa 6.0
```

II slučaj

```
vrednost_izraza = 2 * ((3 + 2)**2 / 5) - 4
print(vrednost_izraza) # štampa 6.0
```

Kodiranje aritmetičkih izraza

Co-funded by the European Union

Kodirati sledeće aritmetičke izraze:

a) $ab + ac$

b) $3\left(\frac{2a - 8b}{c}\right)$

c) $\frac{a + b}{k - 3}(3k + 1)$

Rešenje:

a) a * b + a * c

b) 3 * (2 * a – 8 * b) / c

c) (a + b) * (3 * k + 1) / (k – 3)

Logički operatori

Co-funded by
the European Union

Operacija	Opis
x and y	Logičko i
x or y	Logičko ili
not x	Logičko ne

Operatori poređenja

Co-funded by the European Union

Operator	Opis
x == y	Jednako
x != y	Različito
x < y	Manje
x > y	Veće
x <= y	Manje jednako
x >= y	Veće jednako
x is y	Provera identiteta. True, ako x i y ukazuju na isti objekat

Maksimalna vrednost

Co-funded by
the European Union

```
x = float(input('x = '))
y = float(input('y = '))
print('veći broj je', x * (x > y) + y * (x <= y) )

x = 1.2
y = 2.8
veći broj je 2.8
```

Ispis realnih brojeva

Co-funded by the European Union

Prilikom ispisa realnih brojeva moguće je prikazati zaokružene vrednosti na određeni broj decimala. Jedan od načina je da se to uradi je da se vrednosti pre ispisa pretvore u tekst funkcijom **format**.

```
pi = 3.14159265
s = format(pi, '.2f')
print(s)
```

```
3.14
```

ili

```
print(format(pi, '.2f'))
```

Ispis realnih brojeva

Co-funded by the European Union

Kada je potrebno da se precizno kontroliše ispis više brojeva u jednom redu, može se ispred niske u funkciji **print** navesti slovo **f**. Tada unutar niske u vitičastim zagradama može da se navede ime promenljive i opis ispisivanja.

```
ime = 'Milan'
prosek = 4.83562
print(f'{ime} ima prosek {prosek:.2f}')
```

```
Milan ima prosek 4.84
```

Ugrađene matematičke funkcije

Co-funded by the European Union

Ugrađene matematičke funkcije su označene sa **math**.

math.fabs(x) – apsolutna vrednost $|x|$
math.pow(x, y) – izračunava stepen x^y
math.sqrt(x) – kvadratni koren
math.pi – konstanta π

Kodiranje izraza

Co-funded by the European Union

Pri kodiranju matematičkih izraza postoje tri specijalna slučaja:

Predstavljanje stepena argumenta funkcije: Na primer, ukoliko je argument funkcije sinus trećeg stepena, imaće izraz sledeći oblik

$$\sin(x^3) \cong math.\sin(math.pow(x, 3))$$

Predstavljanje stepena funkcije: Na primer, ukoliko je funkcija sinus trećeg stepena, imaće izraz sledeći oblik

$$\sin^3(x) \cong math.pow(math.\sin(x), 3)$$

Predstavljanje korena funkcije koji nije kvadratni: Na primer, ukoliko je potrebno predstaviti peti koren kvadrata promenljive x, imaće izraz sledeći oblik

$$\sqrt[5]{x^2} = x^{\frac{2}{5}} \cong math.pow(x, 2.0/5.0)$$

Izračunavanje vrednosti izraza

Co-funded by the European Union

Ako je a = 0.2, b = 0.5 i x = 1 izračunati vrednost izraza

$$z = \ln \sqrt{\left|\sin(a + e^{\frac{x}{2}}) + \cos\left(b - arctg\frac{x}{2}\right)\right|}$$

```
import math
a = 0.2; b = 0.5; x = 1
z = math.log(math.sqrt(math.fabs(math.sin(a + math.exp(x/2)) + math.cos(b – math.atan(x/2)))))
z
0.3367186419376566
```

Izračunavanje vrednosti izraza

Co-funded by the European Union

Ako je x = 1.5 i y = 2.8 izračunati vrednost izraza

$$z = \ln \sqrt{tg^2 x + 0{,}2 + \frac{\cos y}{e^{xy}}}$$

```
import math
x = 1.5; y = 2.8
z = math.log(math.sqrt(math.pow(math.tan(x), 2) + 0.2 + math.cos(y) / math.exp(x * y)))
z
2.646742643815521
```

Izračunavanje vrednosti izraza

Co-funded by the European Union

Safety 4EU

Ako je x = 1.5 izračunati vrednost izraza

$$y = \frac{\sqrt[3]{\cos^2 x} - \left|\ln^2\left(\sin x^2\right)\right|}{\sqrt[3]{\left(x^2 + 2{,}5\right)^4}}.$$

```
import math
x = 1.5
y = (math.pow(math.cos(x), 2.0/3.0) – math.fabs(math.pow(math.log(math.sin(x * x)), 2)))/math.pow(x * x + 2.5, 4.0/3.0)
y
0.01353491740857625 7
```

Logaritam

Co-funded by the European Union

Naći x ako je

$$2^x = 8$$

$x = log_2 8$

```
from math import log
x = log(8, 2)
print(x)
```

3.0

Safety4EU projekat
L3. Analiza podataka, programiranje i interpretacija

Algoritamske strukture

Milan Gocić

Co-funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Algoritam definicija

Co-funded by the European Union

Postupak transformacije ulazne informacije u izlaznu informaciju naziva se **algoritam**.

Algoritam predstavlja konačan skup precizno definisanih pravila za rešavanje nekog problema.

Algoritam je opis za rešavanje nekog problema.

Algoritam je plan za izradu programa.

Niklaus Wirth je dao sledeću definiciju:

Program = Algoritam + Podaci

Abu Abdulah Muhamed bin Musa al-Hvarizmi

Co-funded by the European Union



Muhamed Al Horezmi (780. – 840.) je persijski naučnik, matematičar, astronom i pisac. Poznat je kao pronalazač algebre, koja je ime dobila po naslovu jedne njegove knjige. Zaslužan je za širenje indijskih brojeva po Zapadu.

Napisao je knjigu *Al Horezmi o indijskoj veštini računanja*, gde u arapsku matematiku uvodi indijske cifre i decimalni brojni sistem.

Kasnije je ova knjiga prevedena na latinski, ali je ovoga puta naslov "**Algoritni de numero indorum**" što je loš prevod. Ubrzo se reč **algoritam** (lat. **Algorithmus**) odomaćila na način računa sa ovim novim brojkama (sada već nazvane arapskim, a ne indijskim). Inače ovo je ujedno i nastanak reči **Algoritam**.

Osobine algoritma

Co-funded by the European Union

Osnovne osobine koje treba da poseduje svaki algoritam su:

- diskretnost,
- determinisanost,
- rezultativnost i
- masovnost.

Diskretnost znači da svakom algoritamskom koraku odgovara diskretni vremenski period.

Determinisanost znači da je rezultat operacije pojedinog algoritamskog koraka jednoznačan.

Rezultativnost obezbeđuje dobijanje željenog rezultata posle konačnog broja algoritamskih koraka.

Masovnost je osobina kojom se obezbeđuje rešavanje šire klase problema jednim algoritmom.

Način predstavljanja algoritma

Co-funded by the European Union

Algoritmi se mogu predstaviti na neki od sledećih načina:

- dijagram toka podataka – blok šema,
- pseudo jezici, odnosno, pseudokod (Pseudokod predstavlja međukorak između svakodnevnog jezika i programskih jezika),
- programski jezici,
- Prostova mašina ,
- Tjuringova mašina,
- rekurzivne funkcije.

Dijagram toka podataka – blok šema

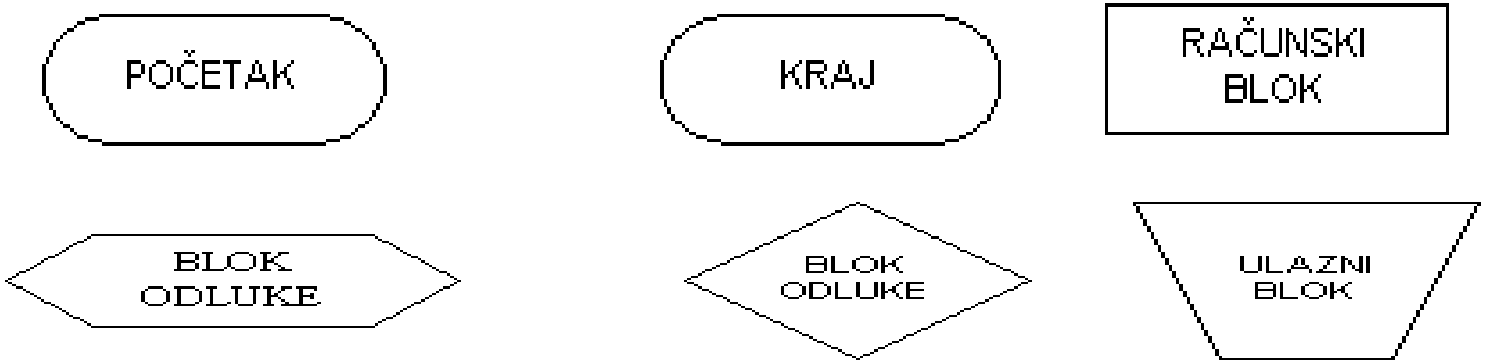
Co-funded by the European Union

Algoritam se najčešće predstavlja u obliku blok šeme sa jasno definisanim nizom radnji, korak po korak.

Grafički zapis algoritma naziva se ***algoritamska blok šema*.**

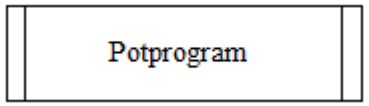
Grafički simboli algoritma

Co-funded by the European Union



Konektor (izlaz u drugi deo algoritma ili ulaz iz drugog dela algoritma)

Strelica za povezivanje elemenata i označavanje smera toka algoritma



Elementarne algoritamske strukture

Co-funded by the European Union

Pod algoritamskom (programskom) strukturom podrazumeva se više koraka (komandi programskog jezika) koji čine jednu celinu.

Postoje tri elementarne algoritamske strukture:

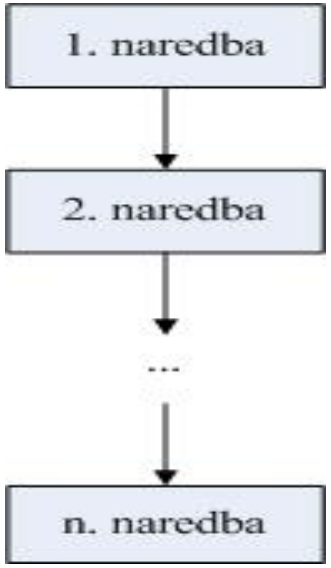
- linijska,
- razgranata i
- ciklična.

Linijska struktura

Co-funded by the European Union

Naredbe koje se pri izvršavanju programa uvek izvršavaju u navedenom redosledu (sekvencijalno) čine **linijsku strukturu**.

Linijski programi su programi u kojima se javljaju naredbe ulaza, obrade i izlaza. Ovi programi imaju smisla u slučajevima kada za razne vrednosti ulaznih veličina treba izvršiti složena izračunavanja.



Izračunavanje vrednosti izraza

Co-funded by the European Union

Sastaviti algoritam i napisati program kojim se izračunava vrednost funkcije

$$f = \left(\frac{x + 3z + y}{2x} \right)^4 - \frac{x}{x + 3z - y}$$

na osnovu unetih vrednosti za x, y i z. Na izlazu štampati vrednost funkcije.

Tekstualni opis algoritma

Co-funded by the European Union

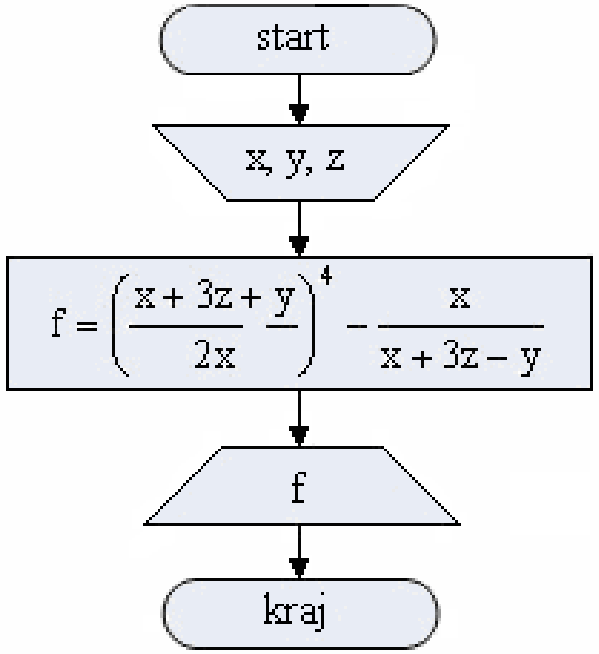
1. ulaz: učitati x, y, z

2. obrada: izračunati $f = \left(\frac{x + 3z + y}{2x} \right)^4 - \frac{x}{x + 3z - y}$

3. izlaz: ispisati f

Algoritam

Co-funded by the European Union



Program

Co-funded by the European Union

```
import math
# Unos podataka
x = float(input('x = '))
y = float(input('y = '))
z = float(input('z = '))
# Izračunavanje vrednosti funkcije f
f = math.pow((x + 3 * z + y) / (2 * x), 4) - x / (x + 3 * z - y)
# Štampanje rezultata
print('Vrednost funkcije f je ',  f)
```

Izlazni rezultati programa

Co-funded by
the European Union

x = 2.6

y = 3.5

z = 4.8

Vrednost funkcije f je 241.35464197224226

Rastojanje između centra opisanog i upisanog kruga

Co-funded by the European Union

Trougao je zadat dužinama stranica a, b i c. Sastaviti algoritam i napisati program za izračunavanje:

* poluprečnika opisanog kruga $R = \frac{a \cdot b \cdot c}{4 \cdot P},$

pri čemu je $P = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}, s = \frac{a+b+c}{2},$

* poluprečnika upisanog kruga $r = \frac{a \cdot b \cdot c}{2 \cdot R \cdot s}$

* rastojanja između centra opisanog i upisanog kruga $d = \sqrt{R \cdot (R - 2 \cdot r)}$

Tekstualni opis algoritma

Co-funded by the European Union

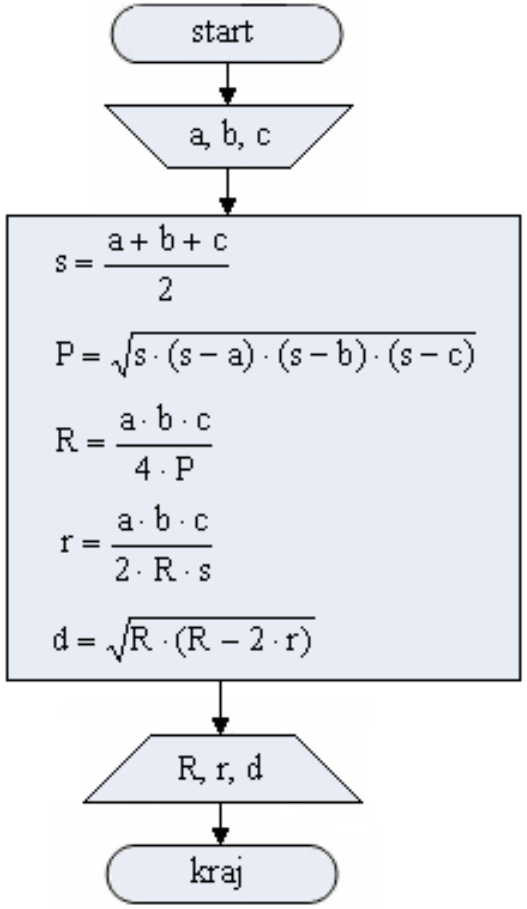
1. ulaz: učitati stranice trougla a, b i c

2. obrada: izračunati poluprečnike opisanog i upisanog kruga i rastojanje između centra opisanog i upisanog kruga

3. izlaz: ispisati vrednosti poluprečnika opisanog i upisanog kruga i rastojanja između centra opisanog i upisanog kruga

Algoritam

Co-funded by the European Union



Program

Co-funded by the European Union

```
import math
# a, b, c - stranice trougla
# s - poluobim trougla
# p - površina trougla
# R - poluprečnik opisanog kruga
# r - poluprečnik upisanog kruga
# d - rastojanje izmedju centra opisanog i upisanog kruga

# Unos podataka

a = float(input('a = '))

b = float(input('b = '))

c = float(input('c = '))
# Izračunavanje vrednosti
s = (a + b + c) / 2
p = math.sqrt(s * (s - a) * (s - b) * (s - c))
R = a * b * c / (4 * p)
r = a * b * c / (2 * R * s)
d = math.sqrt(R * (R - 2 * r))

# Štampanje rezultata
print('R = ',  R, 'r = ', r, 'd = ', d)
```

Izlazni rezultati programa

Co-funded by the European Union

a = 10

b = 6

c = 5

R = 6.57951694959769 r = 2.1712405933672376 d = 3.836484682442342 6

Dijagonala pravougaonika

Co-funded by
the European Union

Napisati program kojim se izračunava dijagonala pravougaonika za unete stranice pravougaonika a i b.

Tekstualni opis algoritma

Co-funded by
the European Union

1. ulaz: učitati stranice pravougaonika a i b

2. obrada: izračunati dijagonalu pravougaonika

3. izlaz: ispisati vrednost dijagonale pravougaonika

Program

Co-funded by
the European Union

```
import math
# Unos podataka
a = float(input('a = '))
b = float(input('b = '))
# Štampanje rezultata
print('Dijagonala pravougaonika je ', math.sqrt(a * a + b * b))
```

Izlazni rezultati programa

Co-funded by
the European Union

a = 4

b = 5

Dijagonala pravougaonika je 6.4031242374328485

Galoni u litre

Co-funded by
the European Union

Napisati program za pretvaranje galona u litre. Napomena: 1 galon = 4,54 l.

Tekstualni opis algoritma

Co-funded by the European Union

1. ulaz: učitati količinu u galonima g

2. obrada: izračunati količinu u litrima

l = g * 4.54

3. izlaz: ispisati vrednost količine u litrima

Program

Co-funded by
the European Union

```
g = float(input('Unesite zapreminu u galonima: '))
# Štampanje rezultata
print('Zapremina u litrima je ', g * 4.54)
```

Izlazni rezultati programa

Co-funded by
the European Union

Unesite zapreminu u galonima: 10

Zapremina u litrima je 45.4

Rastojanje između tačaka

Co-funded by the European Union

Učitati koordinate dve tačke $M_1(x_1, y_1)$ i $M_2(x_2, y_2)$ i izračunati njihovu međusobnu udaljenost.

Opis rešenja: Posmatra se trougao koji formiraju tačke $M_1(x_1, y_1)$ i $M_2(x_2, y_2)$, hipotenuza je duž između ovih tačaka. Katete su $\Delta x = x_2 - x_1$, druga kateta trougla je $\Delta y = y_2 - y_1$. Na osnovu Pitagorine teoreme hipotenuza se računa: hipotenuza $d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$ odnosno kvadratni koren iz sume kvadrata.

Tekstualni opis algoritma

Co-funded by the European Union

1. ulaz: učitati koordinate tačaka x1, y1, x2, y2

2. obrada: izračunati rastojanje d

3. izlaz: ispisati vrednost rastojanja d

Program

Co-funded by the European Union

```
import math

x1 = float(input('Unesite x1 tačke M: '))
y1 = float(input('Unesite y1 tačke M: '))
x2 = float(input('Unesite x2 tačke N: '))
y2 = float(input('Unesite y2 tačke N: '))

d = math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2))

print('Rastojanje između tačaka M i N je ', d)
```

Izlazni rezultati programa

Co-funded by the European Union

Safety 4EU

Unesite x1 tačke M: 1

Unesite y1 tačke M: 1

Unesite x2 tačke N: 5

Unesite y2 tačke N: 4

Rastojanje između tačaka M i N je 5.0

Program

Co-funded by the European Union

```
import math

(x1, y1) = map(float, input().split())
(x2, y2) = map(float, input().split())

d = math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2))

print('Rastojanje između tačaka M i N je ', d)
```

Izlazni rezultati programa

Co-funded by
the European Union

1 1

5 4

Rastojanje između tačaka M i N je 5.0

Dvocifreni broj

Co-funded by the European Union

Napisati program za izdvajanje cifara dvocifrenog broja.

Opis rešenja: Pri pretvaranju se koristi osobina pozicionog sistema. Dvocifren broj

$broj = d * 10^1 + j * 10^0$

Ostatak deljenja dvocifrenog broja sa 10 (broj % 10) dobijamo jedinice broja (u ovom slučaju dvocifrenog). Celobrojnim deljenom sa 10 (broj / 10) dvocifrenog broja se dobija cifra desetica.

Tekstualni opis algoritma

Co-funded by the European Union

1. ulaz: učitati dvocifreni broj a

2. obrada: izračunati cifru desetice i jedinice

```
d = a // 10; # izdvajanje cifre desetica
j = a % 10; # izdvajanje cifre jedinica
```

3. izlaz: ispisati a, d, j

Program

Co-funded by the European Union

```
a = int(input('Unesite dvocifreni broj: '))
d = a // 10
j = a % 10

print('Za uneti dvocifreni broj ', a, ' cifra na mestu desetica je ', d, '  dok je cifra na mestu jedinica ', j)
```

Izlazni rezultati programa

Co-funded by the European Union

Unesite dvocifreni broj: 45

Za uneti dvocifreni broj 45 cifra na mestu desetica je 4   dok je cifra na mestu jedinica 5

Logički operatori

Co-funded by the European Union

Napisati program kojim se izračunava vrednost funkcije

$$z = \neg(x \wedge y) \vee x \wedge y$$

na osnovu unetih logičkih vrednosti promenljivih x i y. Na izlazu štampati vrednost funkcije.

Program

Co-funded by
the European Union

```
x = bool(input('Unesite vrednost za x: '))
y = bool(input('Unesite vrednost za y: '))

z = not(x and y) or x and y

print('z je ', z)
```

Izlazni rezultati programa

Co-funded by
the European Union

Unesite vrednost za x: True

Unesite vrednost za y: False

z je True

Vreme

Co-funded by the European Union

Napisati program koji vreme učitano u satima, minutima i sekundama izračunava i pretvara u sekundama.

Tekstualni opis algoritma

Co-funded by the European Union

1. ulaz: učitati sate, minute i sekunde

2. obrada: izračunati vreme u sekundama

vreme = 3600 * s + 60 * m + sec

3. izlaz: ispisati vrednost vremena u sekundama

Program

Co-funded by the European Union

```
s = int(input('Unesite sate: '))
m = int(input('Unesite minute: '))
sec = int(input('Unesite sekunde: '))

print('Vreme izraženo u sekundama je ', s * 3600 + m * 60 + sec)
```

Izlazni rezultati programa

Co-funded by the European Union

Unesite sate: 2

Unesite minute: 23

Unesite sekunde: 15

Vreme izraženo u sekundama je 8595

Završetak posla

Co-funded by the European Union

Grupa koju čini *n* radnika uradi posao za *s* sati. Napisati program kojim se određuje za koliko sati će posao biti završen ako se priključi još *m* radnika, a svaki radnik radi posao podjednako efikasno.

Ako sa s_n označimo broj sati n radnika, a sa x obeležimo broj sati za koje posao uradi veća grupa radnika, do rešenja dolazimo rešavanjem proporcije $n: (n + m) = x: s_n$ odnosno $x = \frac{n \cdot s_n}{n+m}$

```
n = int(input()) # početni broj radnika
sn = float(input()) # vreme za koje urade posao
m = int(input()) # dodatni broj radnika
x = (n * sn) / (n + m) # vreme za koje svi zajedno urade posao
print(format(x, '.2f’))
```

1
5
2
1.67

Zbir cifara trocifrenog broja

Co-funded by the European Union

Napisati program kojim se učitava trocifreni broj i ispisuje zbir svih cifara tog broja.

Ulaz: 123 Izlaz: 6

```
# učitavanje polaznog broja
broj = int(input())
# izračunavanje zbira cifara
cifra_jedinica = (broj // 1) % 10
cifra_desetica = (broj // 10) % 10
cifra_stotina = (broj // 100) % 10
zbir_cifara = cifra_jedinica + cifra_desetica + cifra_stotina
# prikaz rezultata
print(zbir_cifara)
```

Razgranata struktura

Co-funded by the European Union

Naredbe koje omogućuju promenu redosleda izvršavanja naredbi programa zovu se **naredbe prelaska**. Prema načinu na koji se obrazuje prelazak u programu, ove naredbe se dele na:

Naredbe bezuslovnog prelaska omogućuju prelazak iz jednog dela programa na bilo koju izvršnu naredbu u istoj programskoj jedinici.

Naredbe uslovnog prelaska omogućuju grananja u programu, tako da se pri izvršavanju programa prolazi samo kroz jednu od mogućih grana. Ovakva programska struktura zove se **razgranata struktura**. Broj grana u razgranatoj strukturi može biti različit (dve, tri ili više), što zavisi od naredbe prelaska kojom se ostvaruje grananje u programu.

if iskaz

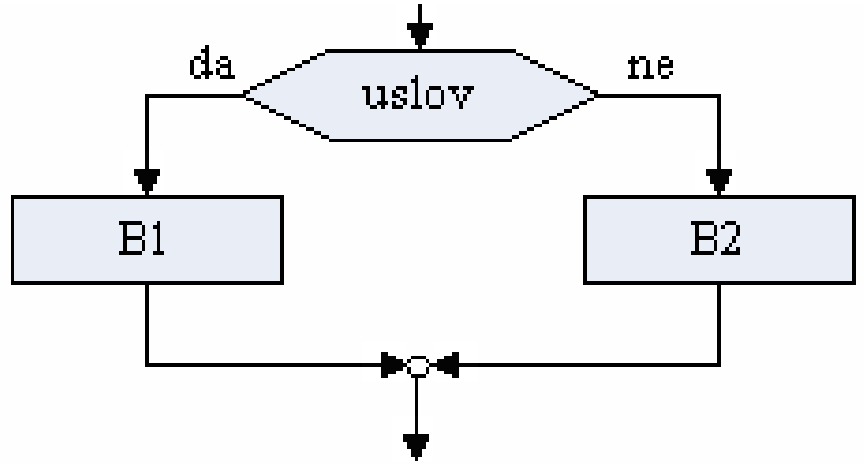
Sintaksa **if-else** strukture ima oblik

if uslov:

    B1

else:

    B2



Primer

Sastaviti algoritam i napisati program kojim se izračunava vrednost funkcije

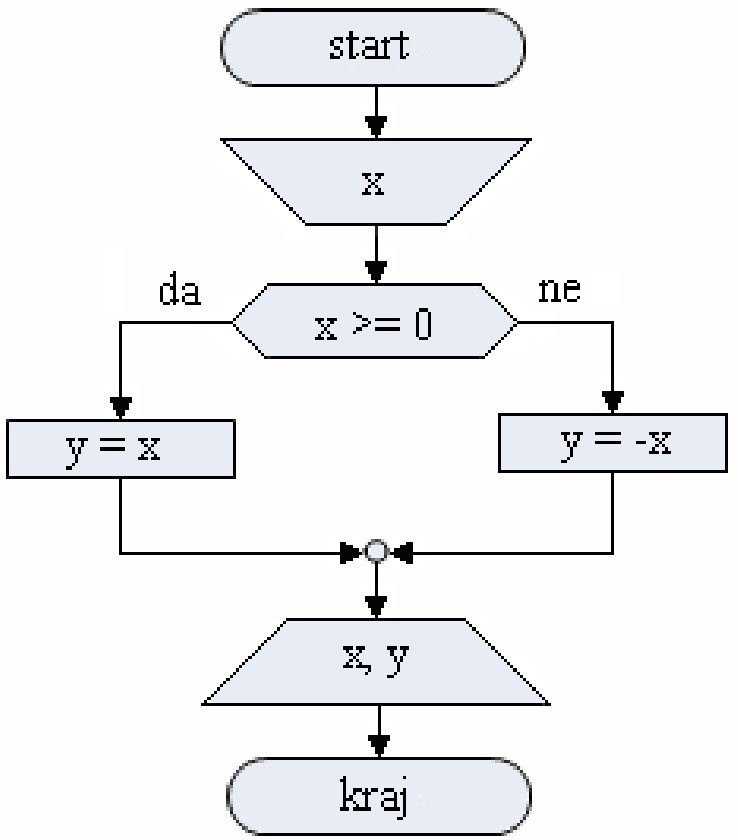
$$y = |x| = \begin{cases} x, & x \geq 0, \\ -x, & x < 0. \end{cases}$$

Na izlazu štampati rezultat oblika

Za uneto X = xxx.xxx dobija se Y = xxx.xxx.

Algoritam

Co-funded by the European Union



Program

Co-funded by the European Union

```
x = float(input('x = '))

if x >= 0:
    y = x
else:
    y = -x

print('Za uneto x = ',  x, ' dobija se y = ', y)
```

Izlazni rezultati programa

Co-funded by
the European Union

x = -26.3

Za uneto x = -26.3 dobija se y = 26.3

Uslovni izraz

Co-funded by the European Union

Umesto naredbe grananja pogodnije je iskoristiti uslovni izraz (izraz grananja).

Uslovni izraz ima sledeći oblik:

rezultat_tacno **if** uslov **else** rezultat_netacno

Ovaj operator je ternarni, odnosno ima tri argumenta: prvi je vrednost izraza ako je uslov ispunjen, drugi argument je uslov od koga zavisi konačna vrednost izraza, dok se trećim argumentom zadaje vrednost izraza ako uslov nije ispunjen.

```
if broj % 2 == 0:
    print(' paran ')
else:
    print(' neparan ')
```

```
print(' paran ' if broj % 2 == 0 else ' neparan ')
```

Primer

Co-funded by the European Union

Napisati program kojim se za unete vrednosti *a* i *b* izračunava z prema izrazu

$$z = \begin{cases} a + b, a\ neparno, b\ parno \\ a * b, za\ ostale\ slučajeve \end{cases}$$

Program

Co-funded by
the European Union

```
a = int(input('a = '))
b = int(input('b = '))

if (a % 2 != 0) & (b % 2 == 0):
    z = a + b
else:
    z = a * b
print('z = ',  z)
```

Izlazni rezultati programa

Co-funded by the European Union

I slučaj

a = 9

b = 10

z = 19

II slučaj

a = 10

b = 12

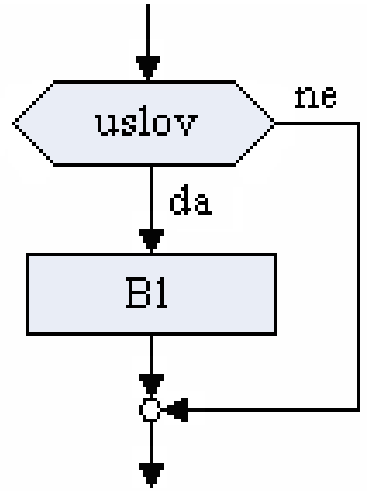
z = 120

Logičko if

Co-funded by the European Union

if uslov:

B1



Primer

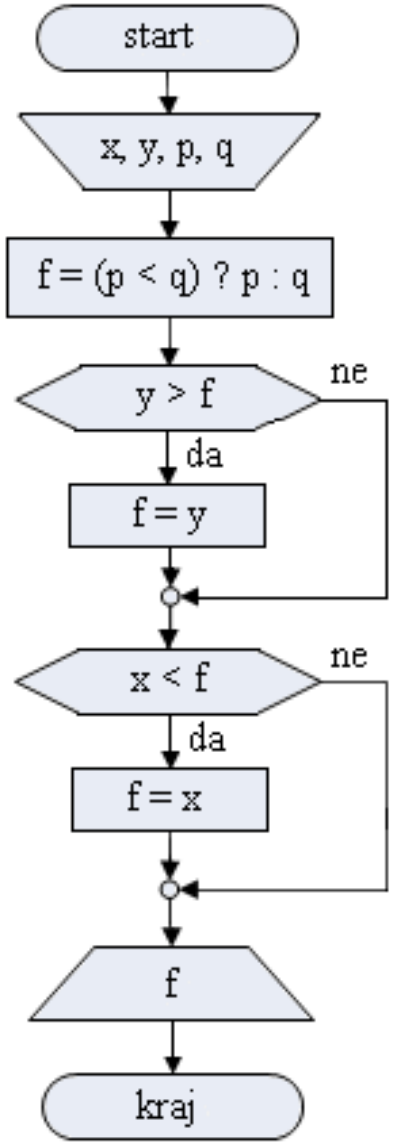
Co-funded by the European Union

Sastaviti algoritam i napisati program za izračunavanje vrednosti funkcije sa vrednostima x, y, p i q

$$f(x, y, p, q) = \min(x, \max(y, \min(p, q)))$$

Algoritam

Co-funded by the European Union



Program

Co-funded by
the European Union

```
x = int(input('x = '))
y = int(input('y = '))
p = int(input('p = '))
q = int(input('q = '))

f = p if p < q else q

if y > f:
        f = y
if x < f:
        f = x

print('f = ',  f)
```

Izlazni rezultati programa

Co-funded by
the European Union

x = 10

y = 8

p = 6

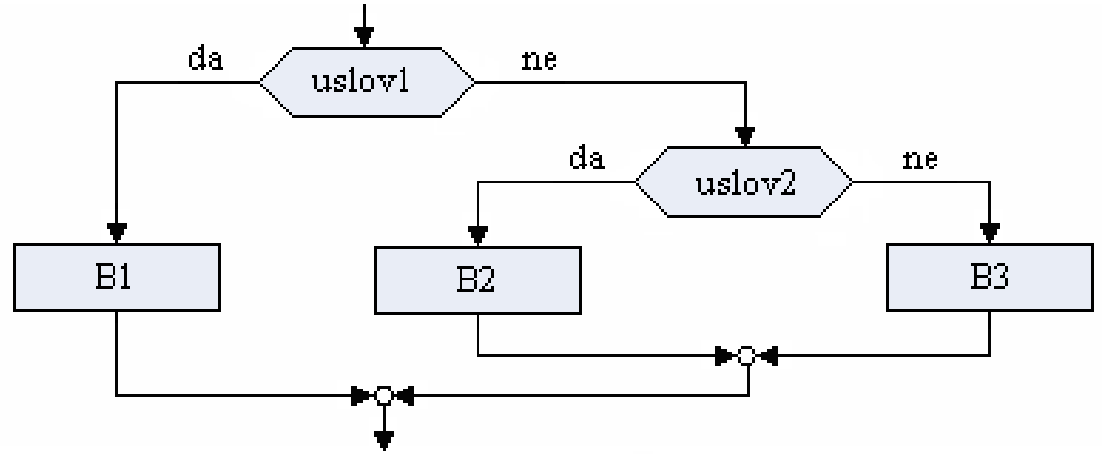
q = 5

f = 8

if-else-if struktura

Sintaksa **if-else-if** strukture ima oblik

```
if uslov1:
    B1
elif uslov2:
    B2
else:
    B3
```



Primer

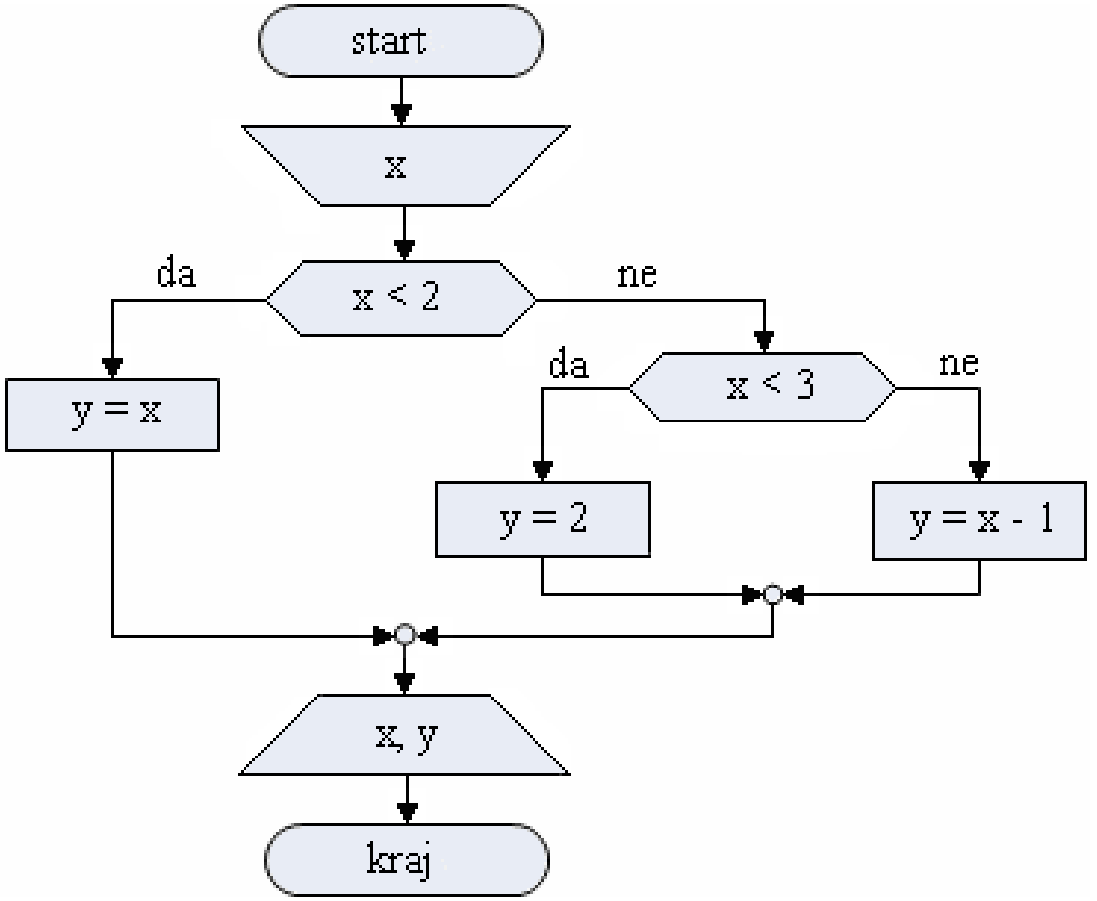
Co-funded by the European Union

Napisati program kojim se izračunava vrednost funkcije

$$y = \begin{cases} x, & x < 2, \\ 2, & 2 \le x < 3, \\ x - 1, & x \ge 3. \end{cases}$$

Za unetu vrednost argumenta x štampati vrednost argumenta i vrednost funkcije.

Algoritam



Program

Co-funded by the European Union

```
x = float(input('x = '))

if x < 2:
    y = x
elif x < 3:
    y = 2.0
else:
    y = x - 1

print('Za uneto X = ',  x, ' dobija se Y = ', y)
```

Izlazni rezultati programa

Co-funded by
the European Union

x = 2.5

Za uneto X = 2.5 dobija se Y = 2.0

Ciklična struktura

Co-funded by the European Union

Ako se niz naredbi u programu može izvršiti više puta u toku izvršavanja programa, tada se kaže da ove naredbe obrazuju **cikličnu strukturu**.

Među naredbama koje čine cikličnu strukturu mora postojati bar jedna naredba takva da omogućava izlazak iz ciklusa.

Uslov pod kojim se izlazi iz ciklusa zove se izlazni kriterijum ciklusa. Ako je izlazni kriterijum broj ponavljanja ciklusa, tada se ciklus zove **brojački ciklus**. Ako je izlazni kriterijum dostignuta tačnost u procesu računanja, tada se ciklus zove **iterativni ciklus**.

Iterativni ciklusi

Co-funded by the European Union

Kod ove vrste ciklusa kriterijum za izlazak iz ciklusa je dostignuta tačnost u procesu računanja.

Iterativni ciklusi se najčešće koriste pri programiranju iterativnih numeričkih postupaka.

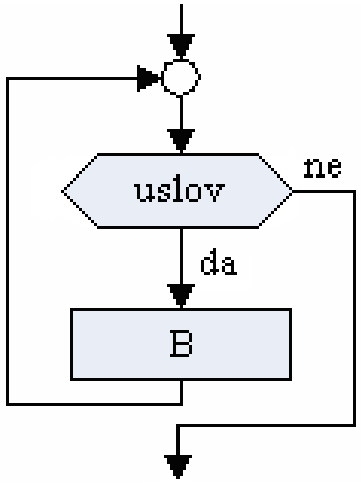
Za realizaciju iterativnih ciklusa koristi se petlja **while**.

While petlja

Sintaksa **while** petlje ima oblik

while uslov:

B



Primer

Co-funded by the European Union

Napisati program za iterativno rešavanje sistema jednačina

$$4x^2 - 3xy + x - 1 = 0$$
$$x - 2y^2 + 3 = 0$$

koristeći sledeće formule:

$$x_{n+1} = \frac{1}{2}\sqrt{x_n(3y_n - 1) + 1}, \qquad y_{n+1} = \sqrt{\frac{x_n + 3}{2}}.$$

Iterativni proces prekinuti kada se ispune uslovi

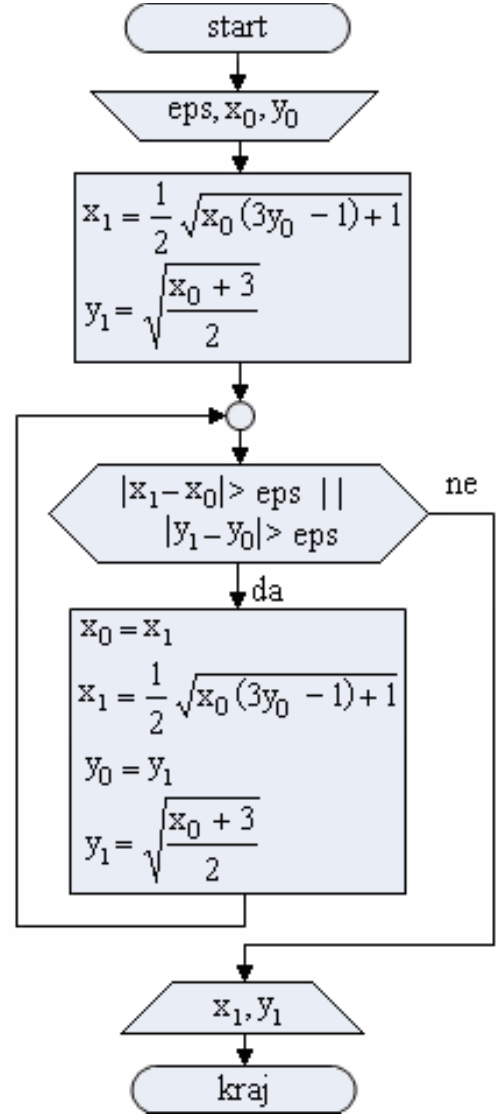
$$|x_{n+1} - x_n| \le \varepsilon \quad \text{i} \quad |y_{n+1} - y_n| \le \varepsilon.$$

Na ulazu je data vrednost za ε i početne vrednosti za x_0 i y_0. Na izlazu štampati rešenja sistema jednačina.

Zadatak rešiti koristeći while petlju.

Algoritam

Co-funded by the European Union



Program

Co-funded by the European Union

```
import math

eps = float(input('eps = '))
x0 = float(input('x0 = '))
y0 = float(input('y0 = '))

x1 = 0.5 * math.sqrt(x0 * (3 * y0 - 1) + 1)
y1 = math.sqrt((x0 + 3) / 2)

while (math.fabs(x1 - x0) > eps or math.fabs(y1 - y0) > eps):
        x0 = x1
        x1 = 0.5 * math.sqrt(x0 * (3 * y0 - 1) + 1)
        y0 = y1
        y1 = math.sqrt((x0 + 3) / 2)

print('Rešenja sistema jednačina su x = ',  x1, ' i y = ', y1)

eps = 1.0E-6
x0 = 1.2
y0 = 3.4
Rešenja sistema jednačina su x =  1.054903978501030 1  i y =  1.423886330144 88
```

Primer

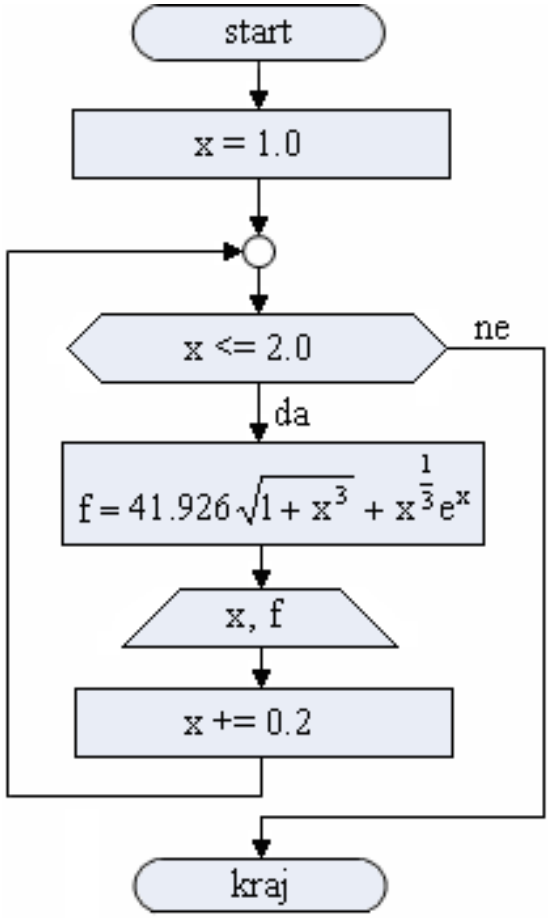
Izračunati vrednost funkcije $f(x) = 41,926\sqrt{1+x^3} + \sqrt[3]{x}e^x$

za x = 1,00(0,2)2,00. Izlazna lista neka izgleda

X	F(X)
x.xx	xxx.xx
⋮	⋮

Program napisati pomoću **while** ciklusa

Algoritam



Program

Co-funded by the European Union

```
import math


print("\tX\tF(X)")
x = 1.0
while (x <= 2.0):
    f = 41.926 * math.sqrt(1 + math.pow(x, 3)) + math.pow(x, 1.0 / 3.0) * math.exp(x)
    print("\t", format(x, '.2f'), "\t", format(f, '.5f'))
    x += 0.2
```

Izlazni rezultati programa

Co-funded by the European Union

X	F(X)
1.00	62.01060
1.20	72.77589
1.40	85.66088
1.60	100.43820
1.80	116.94561
2.00	135.08763

Brojački ciklusi

Co-funded by the European Union

Kod brojačkih ciklusa kriterijum za izlazak iz ciklusa je broj ponavljanja ciklusa.

Vrlo česta greška pri programiranju ovih ciklusa je da se programski ciklus ponavlja jedanput manje nego što je to potrebno.

Za realizaciju brojačkog ciklusa koristi se **for** petlja.

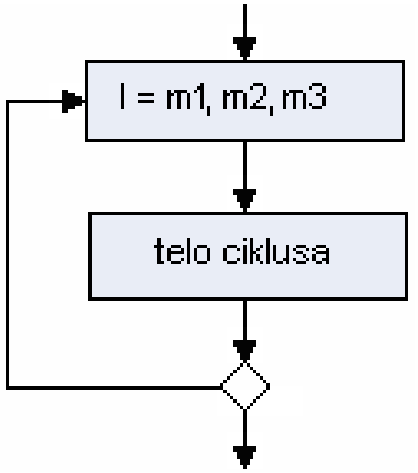
For petlja

Co-funded by the European Union

Sintaksa **for** petlje ima oblik

for i in range(inicijalizacija; uslov; iteracija):

telo ciklusa



pri čemu je ***I*** brojač (kontrolna ili ciklusna promenljiva), ***m1*** početak brojanja, ***m2*** kraj brojanja, a ***m3*** korak brojanja.

poziv funkcije	generisana sekvenca
`range(n)`	za `n > 0: 0, 1, 2, ..., n-1`
`range(start, stop)`	za `stop > start:` `start, start+1, start+2, ..., stop-1`
`range(start, stop, korak)`	za `stop > start` i `korak > 0:` `start, start+korak, start+2*korak, ..., n` važi `stop - korak <= n < stop`
`range(start, stop, korak)`	za `stop < start` i `korak < 0:` `start, start+korak, start+2*korak, ..., n` važi `stop - korak >= n > stop`

For petlja

Co-funded by the European Union

U zavisnosti od parametra ***m3*** postoji:

brojanje unapred (primenjuje se inkrement brojača)

```
for i in range(1, 10, 2):
    print(i)
```

1 3 5 7 9

i brojanje unazad (primenjuje se dekrement brojača)

```
for i in range(10, 1, -2):
    print(i)
```

10 8 6 4 2

Sumiranje

Co-funded by the European Union

$S=\sum_{i=1}^{5} 2i$

```
S = sum(2 * i for i in range(1, 6))
print(S)
```

30

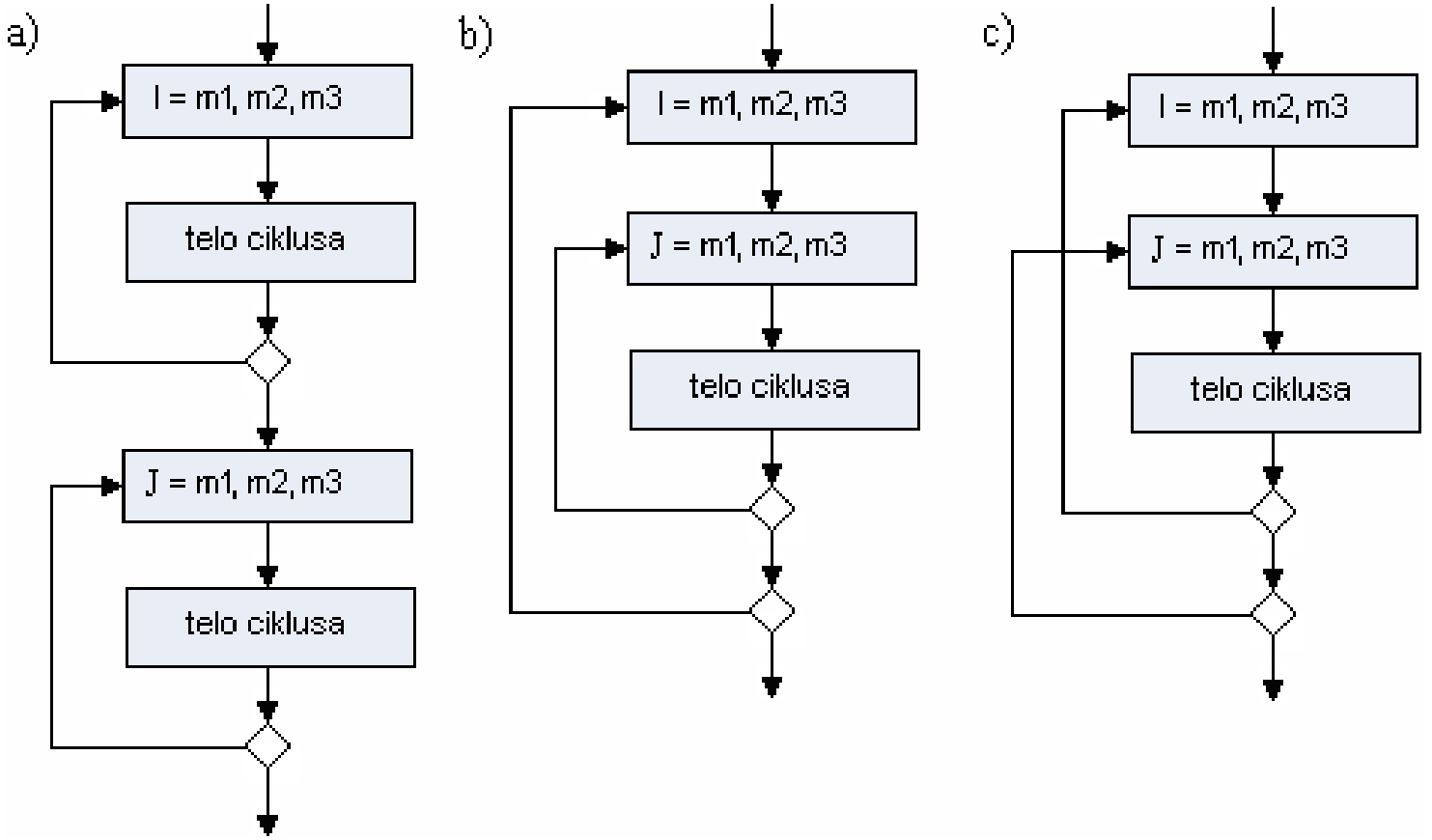
$S=\sum_{i=1}^{n} 10xi$

```
x = [1, 4, 6, 2]
n = len(x)
S = sum(10 * x[i] for i in range(0, n))
print(S)
```

130

Dozvoljene i nedozvoljene kombinacije brojačkog ciklusa

Co-funded by the European Union



Primer

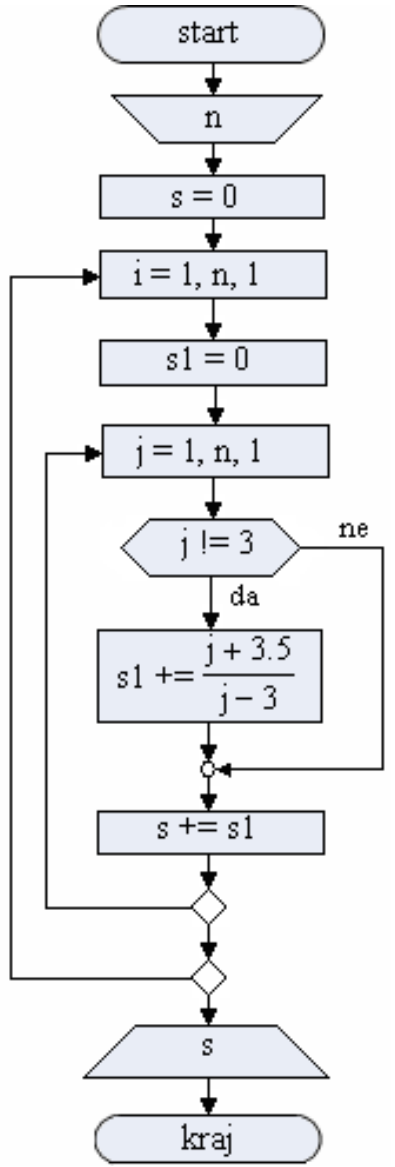
Co-funded by the European Union

Napisati program za izračunavanje vrednosti sume

$$S = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq 3}}^{n} \frac{j + 3,5}{j - 3}.$$

Algoritam

Co-funded by the European Union



Program

Co-funded by the European Union

```
s = 0

n = int(input('n = '))

for i in range(1, n + 1):
        s1 = 0
        for j in range(1, n + 1):
                if (j != 3):
                        s1 += (j + 3.5) / (j - 3.0)
        s += s1

print("Dobijena vrednost sume je ", s)

n = 50
Dobijena vrednost sume je  3404.838248562508
```

Safety4EU projekat
L3. Analiza podataka, programiranje i interpretacija

Funkcija

Milan Gocić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Definisanje funkcija

Co-funded by the European Union

Pored bibliotečkih funkcija može i korisnik da definiše funkcije. Za odgovarajući broj ulaznih parametara funkcije izračunavaju rezultujuću vrednost.

```
# funkcija izračunava obim pravougaonika čije su stranice dužine a i b
def obim(a, b):
  return 2 * a + 2 * b

print(obim(2, 3))
```
10

def obim(a, b): je definicija funkcije koje se naziva **obim** i prima dve ulazne vrednosti koje se nazivaju **a** i **b**. Telo funkcije se piše uvučeno i opisuje kako se izračunava rezultat na osnovu ulaznih vrednosti. Kada je izračunat konačan rezultat, funkcija vraća taj rezultata pomoću naredbe **return**. U pozivu se navode argumenti kojima se inicijalizuje vrednost parametara.

Postoje i funkcije koje ne vraćaju vrednost. One nešto izračunaju i rezultat ispišu na ekran.

Primer funkcije

Co-funded by the European Union

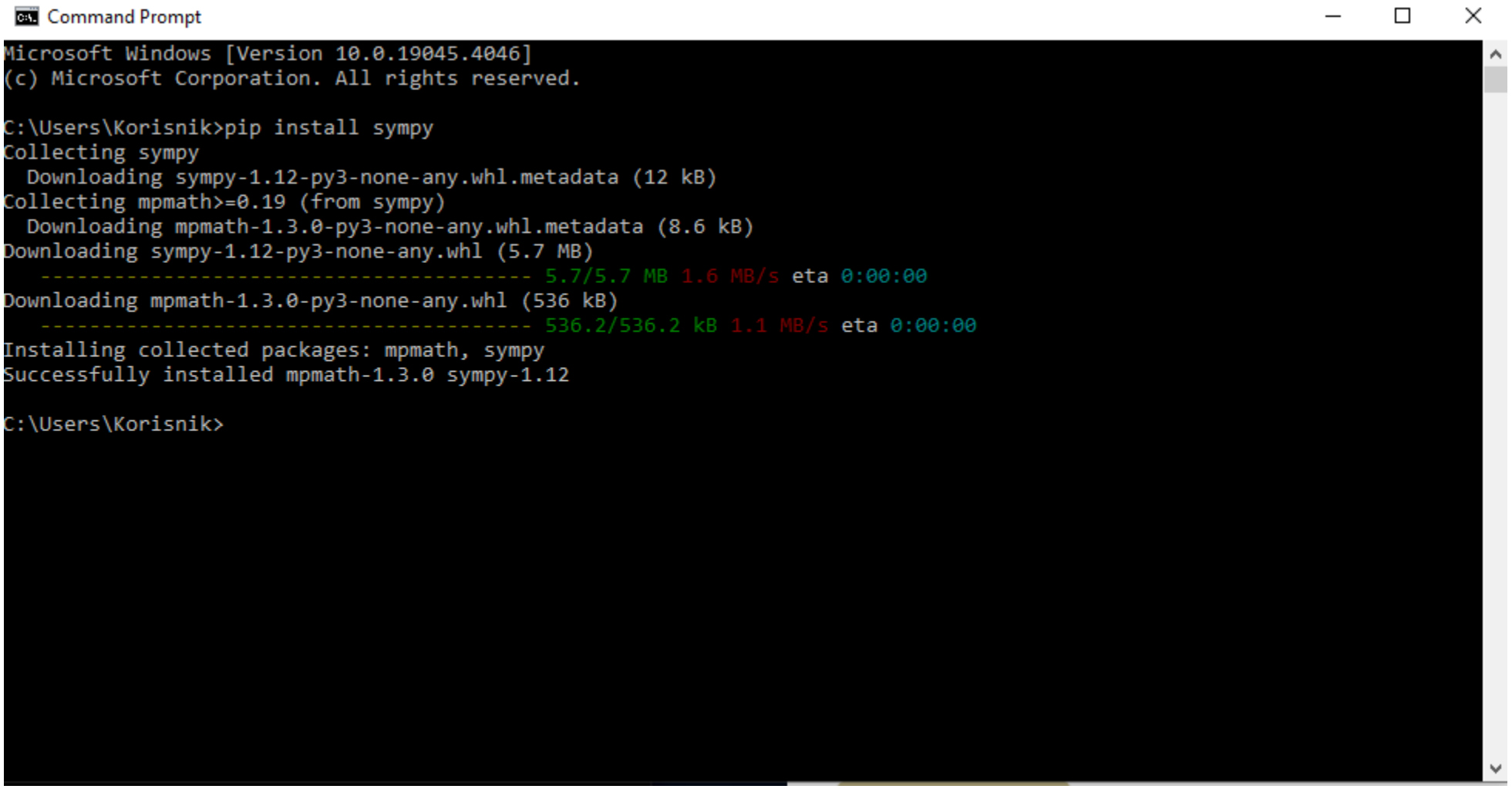
```
def f(x):
        return 2 * x + 1

x_vrednosti = [0, 1, 2, 3]

for x in x_vrednosti:
        y = f(x)
        print(y)
```

Instalacija paketa

Co-funded by the European Union

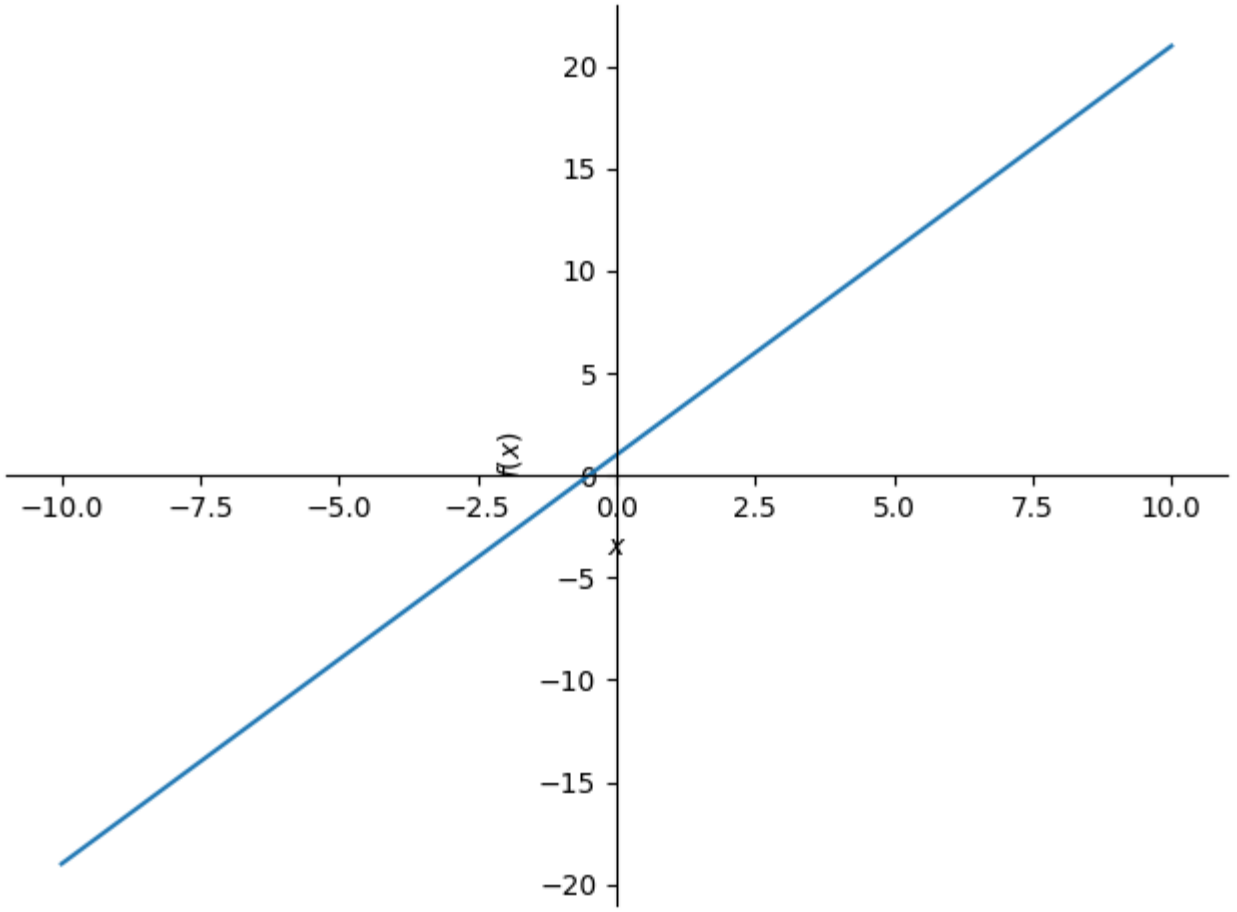


pip install sympy

Iscrtavanje grafika funkcije

Co-funded by the European Union

```
from sympy import *
x = symbols('x')
f = 2 * x + 1
plot(f)
```



Izračunavanje nagiba funkcije

Co-funded by the European Union

```
from sympy import *

x = symbols('x')

f = 3 * x**2 + 1

dx_f = diff(f)

print(dx_f)

print(dx_f.subs(x, 3))
```

```
18
```

Možemo pozvati funkciju subs() da zamenimo promenljivu x sa vrednošću 3.

Više povratnih vrednosti

Co-funded by the European Union

Parametri funkcije su njene ulazne vrednosti, dok je vrednost vraćena pomoću **return** njena izlazna vrednost.

```
def u_metre_i_centimetre(duzina):
    metri = duzina // 100
    centimentri = duzina % 100
    return metri, centimentri

m, cm = u_metre_i_centimetre(356)
print(m, cm)
```

```
3 56
```

Isti efekat se postiže i ako se koriste uređeni parovi.

```
def u_metre_i_centimetre(duzina):
    metri = duzina // 100
    centimentri = duzina % 100
    return (metri, centimentri)

(m, cm) = u_metre_i_centimetre(356)
print(m, cm)
```

Površina trougla na osnovu zadatih temena

Co-funded by the European Union

Napisati program koji izračunava površinu trougla ako su poznate koordinate njegovih temena.

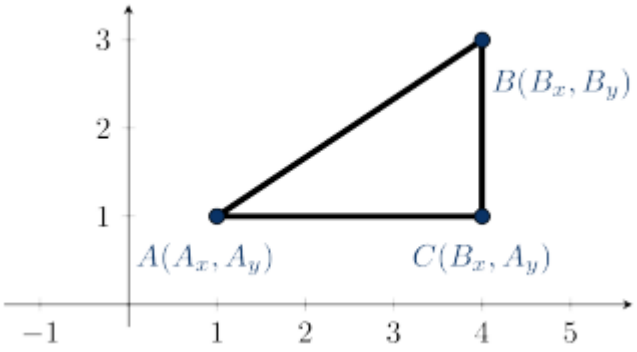
Sa standardnog ulaza unosi se 6 realnih brojeva. Prva dva predstavljaju *x* i *y* koordinatu temena A, druga dva *x* i *y* koordinatu temena B i poslednja dva predstavljaju *x* i *y* koordinatu temena C.

Zadatak se rešava primenom Heronovog obrasca za izračunavanje površine trougla na osnovu dimenzija stranica.

$$P = \sqrt{s \cdot (s-a) \cdot (s-b) \cdot (s-c)}, s = \frac{a+b+c}{2}$$

Rastojanje između tačaka zadatih svojim koordinatama se određuje primenom sledećeg izraza

$$|AB| = \sqrt{(B_x - A_x)^2 + (B_y - A_y)^2}$$



Površina trougla na osnovu zadatih temena

Co-funded by the European Union

```
import math

# rastojanje od tačke (x1, y1) do tačke (x2, y2)
def rastojanje(x1, y1, x2, y2):
    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2)

# površina trougla čije su dužine stranica a, b i c
def povrsina_trougla(a, b, c):
    # Heronov obrazac
    s = (a + b + c) / 2
    return math.sqrt(s * (s - a) * (s - b) * (s - c))

# koordinate temena
Ax = float(input()); Ay = float(input())
Bx = float(input()); By = float(input())
Cx = float(input()); Cy = float(input())
```

Površina trougla na osnovu zadatih temena

Co-funded by the European Union

```
# dužine stranica trougla
a = rastojanje(Bx, By, Cx, Cy); b = rastojanje(Ax, Ay, Cx, Cy); c = rastojanje(Ax, Ay, Bx, By);
# površina trougla
P = povrsina_trougla(a, b, c);

print(format(P, '.5f'))
```

Konverzija km u milje

Co-funded by the European Union

```
def konvertuj(d):
        c = 1.609344
        d_konv = d / c
        return d_konv

for i in range(10):
        print(10 * i, 'km = ', konvertuj(10 * i), 'mi')
```

```
0 km =  0.0 mi
10 km =  6.2137119223733395 mi
20 km =  12.427423844746679 mi
30 km =  18.64113576712002 mi
40 km =  24.854847689493358 mi
50 km =  31.068559611866696 mi
60 km =  37.28227153424004 mi
70 km =  43.495983456613374 mi
80 km =  49.70969537898716 mi
90 km =  55.92340730136005 mi
```

Faktorijel broja rekurzivnim postupkom

Co-funded by the European Union

Neka se složenost problema može okarakterisati prirodnim brojem n, pri čemu složenijem problemu odgovara veće n. Ako rešenje problema složenosti n zavisi od rešenja istog problema složenosti k, pri čemu je n > k, te ako je poznato bazno rešenje istog problema (za složenost reda 1), onda se u rešavanju može primeniti rekurzivni algoritam.

Napisati rekurzivni postupak za nalaženje faktorijela broja.

```
def faktorijel(n):
        if n == 1:
                return 1
        else:
                return n * faktorijel(n - 1)

for n in range(1, 11):
        print(faktorijel(n), end = '  ')
```

```
1  2  6  24  120  720  5040  40320  362880  3628800
```

Fibonačijev niz rekurzivnim postupkom

Co-funded by the European Union

$$F(n) = F(n-1) + F(n-2)$$

```
def fibonaci(n):
        if n == 1:
                return 0
        elif n == 2:
                return 1
        else:
                return fibonaci(n - 1) + fibonaci(n - 2)

for n in range(1, 11):
        print(fibonaci(n), end = '  ')
```

```
0  1  1  2  3  5  8  13  21  34
```

Safety4EU projekat
L3. Analiza podataka, programiranje i interpretacija

Pregled matematičkog računa

Milan Gocić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Granična vrednost

Co-funded by the European Union

Odrediti graničnu vrednost $\lim_{x\to\infty}\frac{1}{x}$.

```
from sympy import *
x = symbols('x')
f = 1/x
rezultat = limit(f, x, oo)
print(rezultat)
```

0

Odrediti graničnu vrednost $\lim_{x\to 0}\frac{\sin 3x}{x}$.

```
from sympy import *
x = symbols('x')
f = sin(3 * x)/x
rezultat = limit(f, x, 0)
print(rezultat)
```

3

Granična vrednost

Co-funded by the European Union

Odrediti graničnu vrednost $\lim_{x\to 0} e^{3x} - \frac{1}{x}$.

```
from sympy import *
x = symbols('x')
f = exp(3 * x) - 1/x
rezultat = limit(f, x, 0)
print(rezultat)
```

-oo

Odrediti graničnu vrednost $\lim_{x\to \infty} e^{-2x}\sqrt{x}$.

```
from sympy import *
x = symbols('x')
f = exp(-2 * x) * sqrt(x)
rezultat = limit(f, x, 00)
print(rezultat)
```

0

Diferenciranje

Co-funded by the European Union

Odrediti prvi izvod funkcije $f(x) = x^2$.

```
from sympy import *
x = symbols('x')

f = x ** 2

dx_f = diff(f)

print(dx_f)
```

```
2 * x
```

Diferenciranje

Co-funded by the European Union

Odrediti prvi izvod funkcije $f(x) = e^x + \cos x.$

```
from sympy import *
x = symbols('x')

f = exp(x) + cos(x)

dx_f = diff(f)

print(dx_f)
```

```
exp(x) - sin(x)
```

Parcijalni izvod funkcije

Co-funded by the European Union

Odrediti parcijalni izvod funkcije f (x, y) = 2 x^3 + 3 y^3.

```
from sympy import *
from sympy.plotting import plot3d

x, y = symbols('x y')

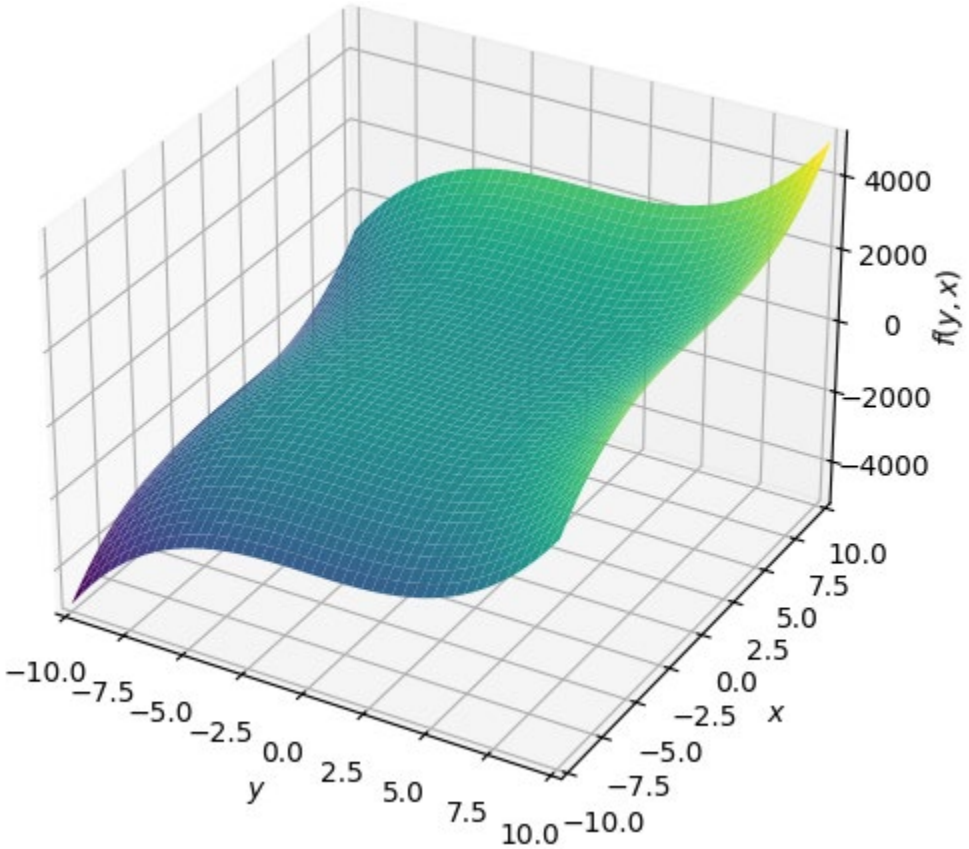
f = 2 * x**3 + 3 * y**3

dx_f = diff(f, x)
dy_f = diff(f, y)

print(dx_f)
print(dy_f)

plot3d(f)

6*x**2
9*y**2
```



Integracija

Co-funded by the European Union

Naći $\int_0^1 (x^2 + 1)dx$.

```
from sympy import *

x = symbols('x')

f = x**2 + 1

oblast = integrate(f, (x, 0, 1))

print(oblast)
```

4/3

Integracija

Co-funded by the European Union

Naći $\int_0^1 \frac{x}{\sqrt{x}} dx$.

```
from sympy import *

x = symbols('x')

f = x / sqrt(x)

oblast = integrate(f, (x, 0, 1))

print(oblast)
```

2/3

N-ti koren

Co-funded by the European Union

Izračunati $\sqrt[n]{a}$.

Rešenje problema zasniva se na Njutn-Rapsonovoj metodi za određivanje nula polinoma

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0.$$

$$x_{i+1} = x_i - \frac{P(x_i)}{P'(x_i)}$$

$$x_{i+1} = x_i - \frac{x_i^n - a}{n x_i^{n-1}}$$

N-ti koren

Co-funded by the European Union

```
import math
a = float(input('a = '))
n = int(input('n = '))

# tačnost rešenja
eps = float(input('eps = '))

if n <= 0:
        print('n mora biti veći od nule!')
elif a < 0 and n % 2 == 0:
        print('ne postoji realni parni koren iz negativnog broja')
else:
        # početna aproksimacija n-tog korena
        koren = a / n
        while math.fabs(koren**n - a) > eps:
                koren = koren - (koren**n - a) / (n * koren**(n-1))
        print(n, 'koren za', a, 'je', koren)
```

Safety4EU projekat
L3. Analiza podataka, programiranje i interpretacija

Verovatnoća i statistika

Milan Gocić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Definisanje pojma

Co-funded by the European Union

Probability is about quantifying predictions of events yet to happen, whereas **likelihood** is measuring the frequency of events that already occurred.

In statistics and machine learning, we often use likelihood (the past) in the form of data to predict probability (the future).

Probability is purely theoretical of how likely an event is to happen and does not require data. Statistics, on the other hand, cannot exist without data and uses it to discover probability and provides tools to describe data.

Binomna raspodela

Co-funded by the European Union

```
from scipy.stats import binom

n = 10
p = 0.9

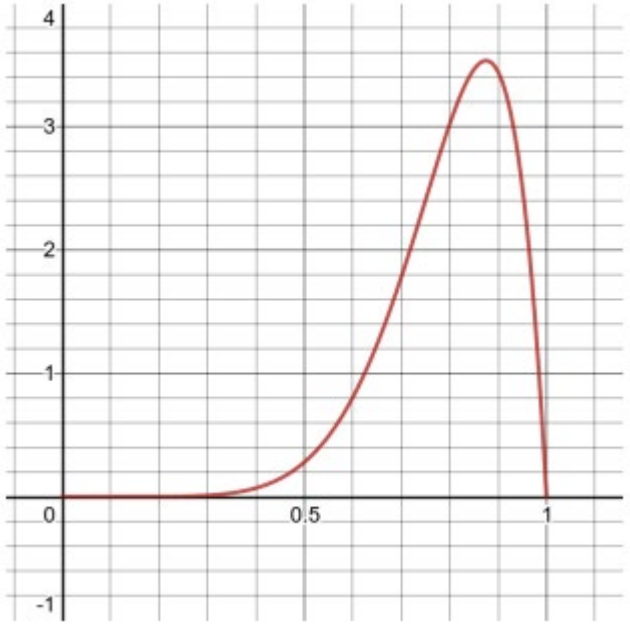
for k in range(n + 1):
      probability = binom.pmf(k, n, p)
      print("{0} - {1}".format(k, probability))
```

```
0 - 9.999999999999977e-11
1 - 8.999999999999976e-09
2 - 3.6449999999999933e-07
3 - 8.747999999999988e-06
4 - 0.00013778099999999974
5 - 0.0014880347999999982
6 - 0.011160260999999989
7 - 0.057395627999999997
8 - 0.1937102444999998
9 - 0.3874204889999976
10 - 0.3486784401000015
```

PMF je probability mass function

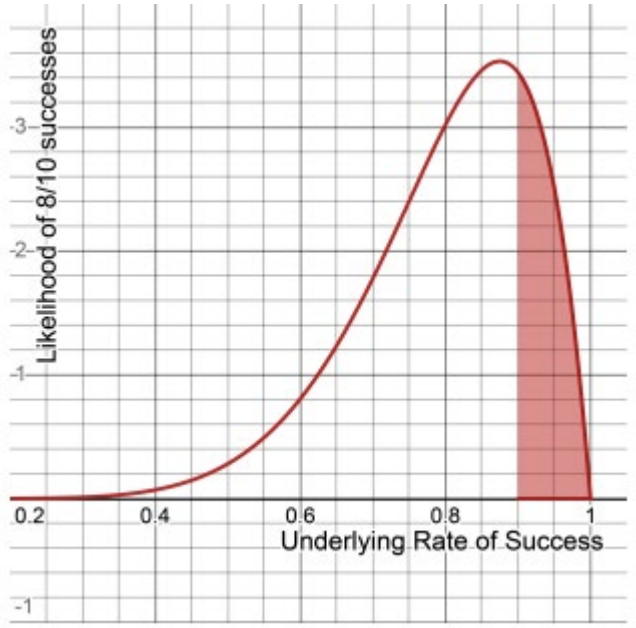
Beta raspodela

Beta distribucija nam omogućava da vidimo npr. verovatnoću 8/10 uspeha.



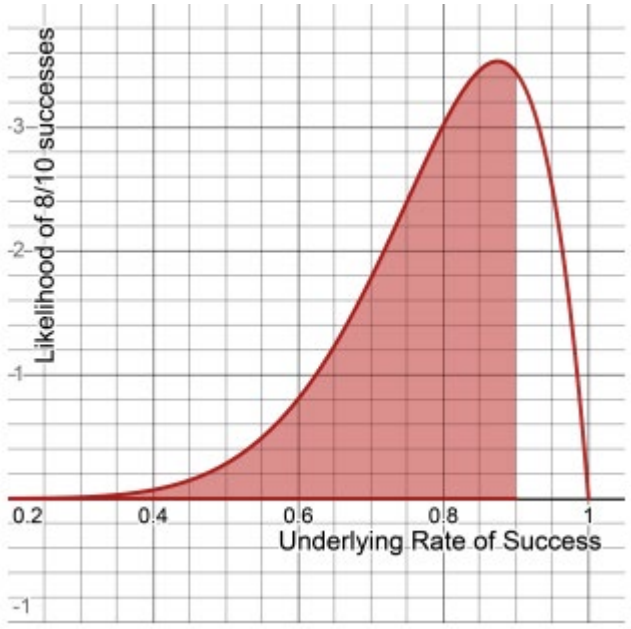
Grafik beta distribucije sa osam uspeha i dva neuspeha

Da bismo pronašli verovatnoću, moramo da pronađemo oblast unutar nekog opsega. Na primer, ako želimo da procenimo verovatnoću da će 8/10 uspeha dati 90% ili veću stopu uspeha, moramo da pronađemo oblast između 0,9 i 1,0, što je .225



Beta raspodela

Svaka kontinuirana distribucija verovatnoće ima funkciju kumulativne gustine (CDF, cumulative density function), koja izračunava površinu do date k-vrednosti. Recimo da želimo da izračunamo površinu do 90% (0,0 do 0,90) – osenčeni deo na slici ispod.




```
from scipy.stats import beta
a = 8
b = 2
p = beta.cdf(.90, a, b)
print(p)
```

0.7748409780000001

Prema našem proračunu, postoji 77,48% šanse da je osnovna verovatnoća uspeha 90% ili manje.

Beta raspodela

Kako izračunati da je verovatnoća uspeha 90% ili više?

```
from scipy.stats import beta
a = 8
b = 2
p = 1.0 - beta.cdf(.90, a, b)
print(p)
```

0.225159021999999993

To znači da od 8/10 uspešnih testova, postoji samo 22,5% šanse da je osnovna stopa uspeha 90% ili veća. Ali postoji oko 77,5% šanse da je manje od 90%. Šanse ovde nisu u našu korist da su naši testovi bili uspešni, ali bismo mogli da prokockamo tu šansu od 22,5% sa više testova.

```
from scipy.stats import beta
a = 30
b = 6
p = 1.0 - beta.cdf(.90, a, b)
print(p)
```

0.13163577484183708

Naša verovatnoća da ispunimo minimalnu stopu uspeha od 90% se smanjila, sa 22,5% na 13,16%

Populacija i uzorak

Co-funded by the European Union

Populacija je posebna grupa interesovanja koju želimo da proučavamo, kao što su „sve starije osobe od 65 godina u Srbiji“ ili „svi zlatni retriveri u Srbiji“. Kako definisati populaciju zavisi od toga šta ste zainteresovani da proučavate.

Uzorak je podskup populacije koji je idealno nasumičan i nepristrasan, koji koristimo da bi zaključili o atributima populacije. Često moramo da proučavamo uzorke jer anketiranje celokupne populacije nije uvek moguće.

Srednja vrednost i ponderisana srednja vrednost

Co-funded by the European Union

```
# Temperatura vazduha
uzorak = [15, 13, 12, 15, 17, 10, 12, 13]
mean = sum(uzorak) / len(uzorak)
print(mean)
```

$$\bar{x} = \frac{x_1 + x_2 + x_3 + \ldots + x_n}{n} = \sum \frac{x_i}{n}$$

srednja vrednost uzorka

$$\mu = \frac{x_1 + x_2 + x_3 + \ldots + x_n}{N} = \sum \frac{x_i}{N}$$

srednja vrednost populacije

13.375

$$\text{weighted mean} = \frac{(x_1 \cdot w_1) + (x_2 \cdot w_2) + (x_3 \cdot w_3) + \ldots (x_n \cdot w_n)}{w_1 + w_2 + w_3 + \ldots + w_n}$$

Ponderisana srednja vrednost može biti od pomoći kada želimo da neke vrednosti doprinesu srednjoj vrednosti više od drugih. Na primer, ponderisanje akademskih ispita da bi se dala konačna ocena. Ako su data tri ispita i završni ispit, i svakom od tri ispita dajemo 20% težine a završnom ispitu 40% težine konačne ocene.

```
uzorak = [90, 80, 63, 87]
tezine = [.20, .20, .20, .40]
weighted_mean = sum(u * t for u, t in zip(uzorak, tezine)) / sum(tezine)
print(weighted_mean)
```

81.4

Medijana

Co-funded by the European Union

Medijana daje središnji broj grupe brojeva, što znači da polovina brojeva ima vrednosti koje su veće od srednje, a polovina ima vrednosti koje su manje od nje. Na primer, srednja vrednost brojeva 2, 3, 3, 5, 7, i 10 je broj 4. Manje je osetljiva na granične vrednosti i seče podatke striktno po sredini na osnovu njihovog relativnog redosleda, a ne gde se nalaze tačno na brojevnoj pravoj. Kada se medijana veoma razlikuje od srednje vrednosti, to znači da postoji iskrivljen skup podataka sa odstupanjima (outliers).

```
uzorak = [15, 13, 12, 15, 17, 10, 12, 13]
def median(vrednosti):
        uredjen = sorted(vrednosti)
        print(uredjen)
        n = len(uredjen)
        mid = int(n / 2) - 1 if n % 2 == 0 else int(n/2)
        if n % 2 == 0:
                return (uredjen[mid] + uredjen[mid + 1]) / 2.0
        else:
                return uredjen[mid]
print(median(uzorak))
```

```
[10, 12, 12, 13, 13, 15, 15, 17]
13.0
```

Modus

Co-funded by the European Union

Modus je broj koji se najčešće javlja u grupi brojeva. Na primer, modus brojeva 2, 3, 3, 5, 7 i 10 je broj 3. Kada se nijedna vrednost ne pojavljuje više puta, nema modusa. Kada se dve vrednosti javljaju sa jednakom frekvencijom, tada se skup podataka smatra bimodalnim.

```
from collections import defaultdict
uzorak = [15, 13, 12, 15, 17, 10, 12, 13]

def mode(vrednosti):
        counts = defaultdict(lambda: 0)
        for s in vrednosti:
                counts[s] += 1
        max_count = max(counts.values())
        modes = [v for v in set(vrednosti) if counts[v] == max_count]
        return modes

print(mode(uzorak))
```

```
[12, 13, 15]
```

Varijansa i standardna devijacija

Co-funded by the European Union

$$s^2 = \frac{\Sigma (x_i - \bar{x})^2}{n-1}$$ varijansa

$$s = \sqrt{\frac{\Sigma (x_i - \bar{x})^2}{n-1}}$$ standardna devijacija

```
from math import sqrt
uzorak = [15, 13, 12, 15, 17, 10, 12, 13]

def variance(values, is_sample: bool = False):
        mean = sum(values) / len(values)
        _variance = sum((v - mean) ** 2 for v in values) /(len(values) - (1 if is_sample else 0))
        return _variance

def std_dev(values, is_sample: bool = False):
        return sqrt(variance(values, is_sample))
print("VARIANCE = {}".format(variance(uzorak, is_sample=True)))
print("STD DEV = {}".format(std_dev(uzorak, is_sample=True)))
```

```
VARIANCE = 4.839285714285714
STD DEV = 2.1998376563477846
```

Safety4EU projekat
L3. Analiza podataka, programiranje i interpretacija

Linerana algebra

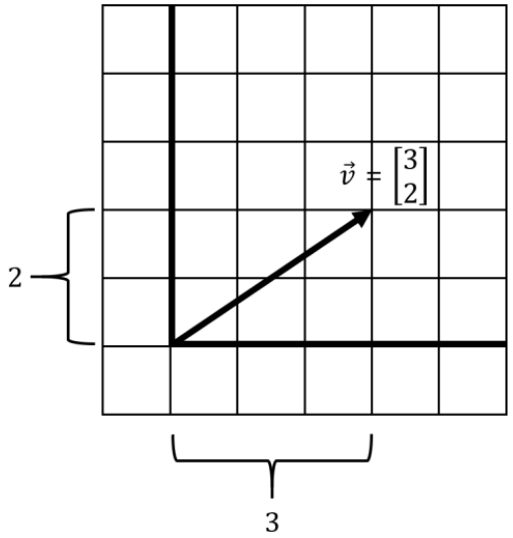
Milan Gocić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Vektor

Co-funded by the European Union

Vektor je strelica u prostoru sa određenim pravcem i dužinom, često predstavlja deo podatka. To je centralni element linearne algebre, uključujući matrice i linearne transformacije.



$$\vec{v} = \begin{bmatrix} x \\ y \end{bmatrix} \qquad \vec{v} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

```
v = [3, 2]
print(v)

[3, 2]
```

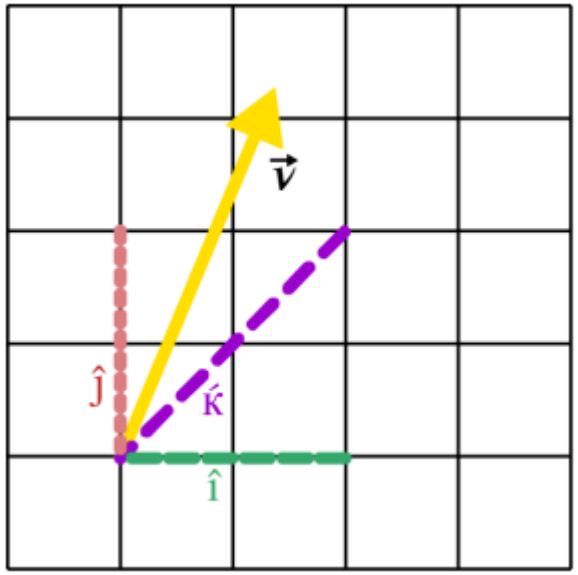
Da biste deklarisali vektor, možete koristiti funkciju NumPy array() i zatim joj proslediti kolekciju brojeva.

```
import numpy as np
v = np.array([3, 2])
print(v)

[3 2]
```

Višedimenzioni vektori

Co-funded by the European Union



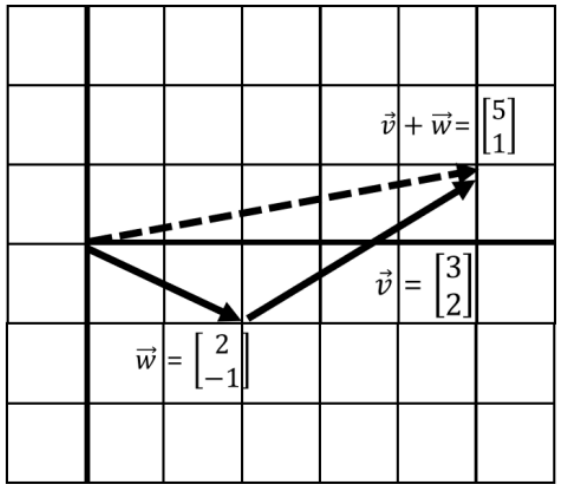
$$\vec{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix}$$

```
import numpy as np
v = np.array([4, 1, 2])
print(v)
```

$$\vec{v} = \begin{bmatrix} 6 \\ 1 \\ 5 \\ 8 \\ 3 \end{bmatrix}$$

```
import numpy as np
v = np.array([6, 1, 5, 8, 3])
print(v)
```

Sabiranje vektora



$$\vec{v} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$\vec{w} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$\vec{v} + \vec{w} = \begin{bmatrix} 3+2 \\ 2+-1 \end{bmatrix} = \begin{bmatrix} 5 \\ 1 \end{bmatrix}$$

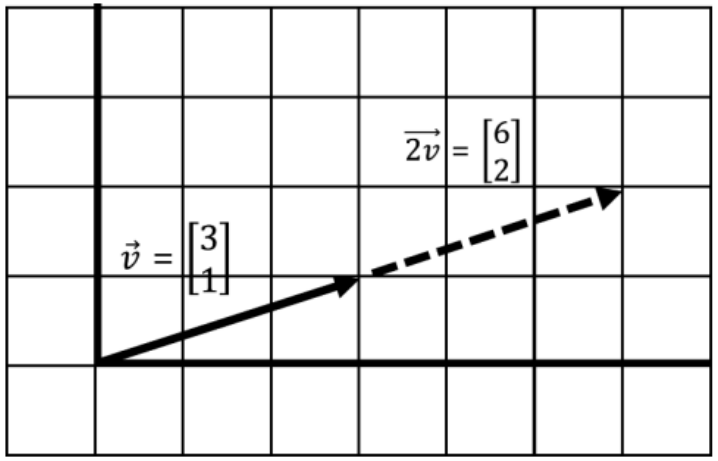
```
from numpy import array
v = array([3,2])
w = array([2,-1])

# sabiranje vektora
zbir = v + w

# prikaz sumarnog vektora
print(zbir)
```

```
[5 1]
```

Množenje skalarom



$$\vec{v} = \begin{bmatrix} 3 \\ 1 \end{bmatrix}$$

$$2\vec{v} = 2\begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \times 2 \\ 1 \times 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \end{bmatrix}$$

```
from numpy import array
v = array([3,1])

# množenje vektora skalarom 2
skalar = 2.0 * v

# prikaz novog vektora
print(skalar)
```

```
[6. 2.]
```

Rešavanje sistema jednačina

Co-funded by the European Union

$$4x + 2y + 4z = 44$$
$$5x + 3y + 7z = 56$$
$$9x + 3y + 6z = 72$$

$$A = \begin{bmatrix} 4 & 2 & 4 \\ 5 & 3 & 7 \\ 9 & 3 & 6 \end{bmatrix} \quad B = \begin{bmatrix} 44 \\ 56 \\ 72 \end{bmatrix} \quad X = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$AX = B$$

$$\begin{bmatrix} 4 & 2 & 4 \\ 5 & 3 & 7 \\ 9 & 3 & 6 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 44 \\ 56 \\ 72 \end{bmatrix}$$

$$AX = B$$
$$A^{-1}AX = A^{-1}B$$
$$X = A^{-1}B$$

dot() operator će izvršiti skaliranje i sabiranje između naše matrice i vektora

```
from numpy import array
from numpy.linalg import inv

# 4x + 2y + 4z = 44
# 5x + 3y + 7z = 56
# 9x + 3y + 6z = 72

A = array([
            [4, 2, 4],
            [5, 3, 7],
            [9, 3, 6]
])
B = array([ 44, 56, 72 ])

X = inv(A).dot(B)

print(X)
```

```
[ 2. 34. -8.]
```

Primer rešavanja sistema jednačina

Co-funded by the European Union

Naći rešenja sistema jednačina

$$2x - 3y + z = 8$$
$$3x - 2y + z = 5$$
$$x + y - z = 1$$

```
from numpy import array
from numpy.linalg import inv

A = array([
            [2, -3, 1],
            [3, -2, 1],
            [1, 1, -1]
])
B = array([8, 5, 1])

X = inv(A).dot(B)

print(X)
```

```
[ 0.6 -3.6 -4. ]
```

Sopstvene vrednosti i sopstveni vektori

Co-funded by the European Union

A ν = λ ν

λ – sopstvene vrednosti; ν – sopstveni vektori

Neka je zadata matrica A oblika $A = \begin{bmatrix} 1 & 2 \\ 3 & 2 \end{bmatrix}_{2x2}$.

Naći sopstvene vrednosti i sopstvene vektore matrice A.

```
from numpy import array, diag
from numpy.linalg import eig, inv

A = array([ [1, 2], [3, 2] ])

eigenvals, eigenvecs = eig(A)
print("EIGENVALUES")
print(eigenvals)
print("\nEIGENVECTORS")
print(eigenvecs)
```

```
EIGENVALUES
[-1.  4.]

EIGENVECTORS
[[-0.70710678 -0.5547002 ]
 [ 0.70710678 -0.83205029]]
```

Rekonstrukcija matrice iz sopstvenih vektora

Co-funded by the European Union

$$A = Q\Lambda Q^{-1}$$

Q – sopstveni vektori
Λ – sopstvene vrednosti u dijagonalnoj formi

```
from numpy import array, diag
from numpy.linalg import eig, inv

A = array([ [1, 2], [3, 2] ])

eigenvals, eigenvecs = eig(A)
print("EIGENVALUES")
print(eigenvals)
print("\nEIGENVECTORS")
print(eigenvecs)

Q = eigenvecs
R = inv(Q)
L = diag(eigenvals)
B = Q @ L @ R
print(B)
```

```
[[1. 2.]
 [3. 2.]]
```

Safety4EU projekat
L3. Analiza podataka, programiranje i interpretacija

Grafičko predstavljanje podataka

Milan Gocić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Matplotlib

Co-funded by the European Union

Matplotlib čine klase i funkcije pomoću kojih se podaci grafički predstavljaju nekim od standardnih grafikona poput linijskog, pravougaonog ili kružnog dijagrama.

Podaci se, po svojoj prirodi, dele na numeričke, kategorijske i ordinalne. **Numerički podaci** (fizičke veličine, prihodi, trajanja, ...) imaju kvantitativnu prirodu. Njihove vrednosti mogu se međusobno razlikovati i porediti korišćenjem matematičkih operacija. **Kategorijski podaci** (nacionalnost, boja, profesija, ...) imaju kvalitativnu prirodu i jedino se mogu medusobno razlikovati. **Ordinalni podaci** (rejtinzi, preferencije, ...) su po karakteristikama između prva dva tipa i mogu se razlikovati i porediti.

Pogledati: matplotlib.org/stable/users/index.html

Linijski dijagram

Co-funded by the European Union

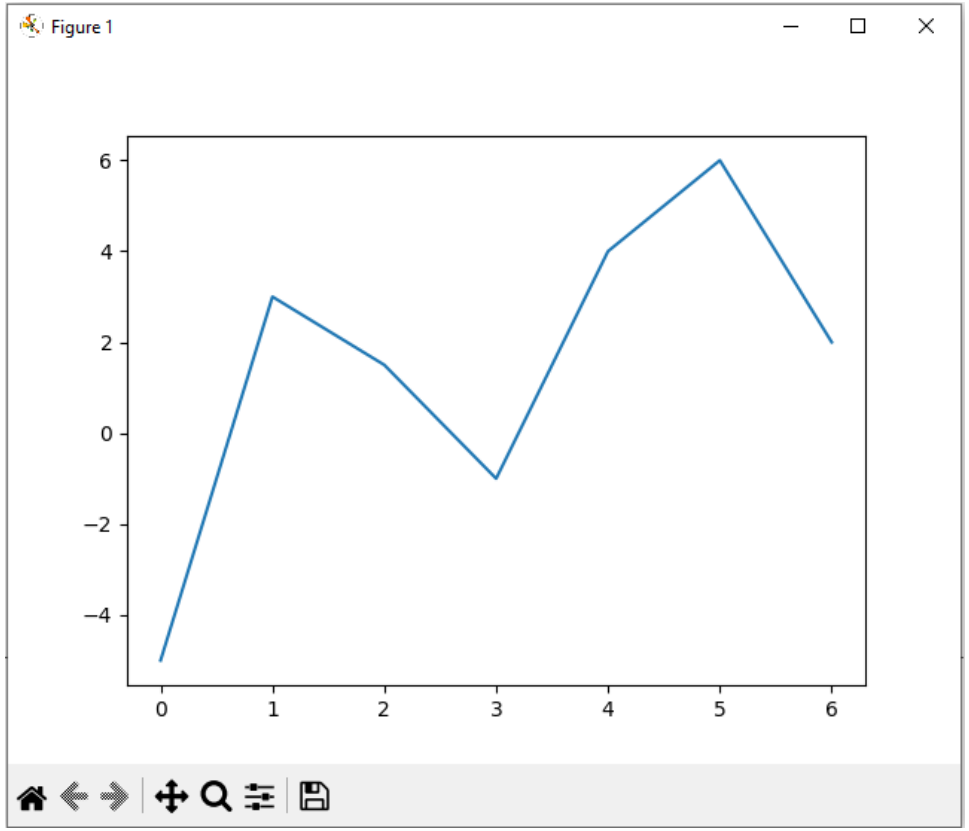
Linijski dijagram predstavlja promenu posmatranog numeričkog podatka u vremenu, ili funkcionalnu zavisnost između dve numeričke veličine. Čini ga niza tačaka u ravni zadatih x i y koordinatom, pri čemu su tačke spojene linijskim segmentima.

```
import matplotlib.pyplot as plt

# podaci za prikaz
x = [-5, 3, 1.5, -1, 4, 6, 2]

# kreira linijski dijagram
plt.plot(x)

# otvara grafički prozor
plt.show()
```



Funkcijae show() kreira prozor sa grafikonom i opcijama prikaza. Opcije omogućavaju kretanje kroz prikaz upotrebom miša (engl. Pan), uvećanje (umanjenje) prikaza (engl. Zoom) i snimanje prikaza u nekom od formata za čuvanje slika.

Funkcija plot

Co-funded by the European Union

Funkcija plot() prima x i y koordinate tačaka koje će biti povezane linijskim segmentima i tekstualni opis boje i tipa linije i markera za prikaz tačaka.

Boja		Marker		Tip linije	
`b`	plava	`o`	krug	`-`	kontinualna
`r`	crvena	`^`	trougao	`:`	tačkasta
`g`	zelena	`*`	zvezda	`--`	crtkasta
`y`	žuta	`s`	kvadrat	`-.`	crta-tačka
`k`	crna	`d`	dijamant		

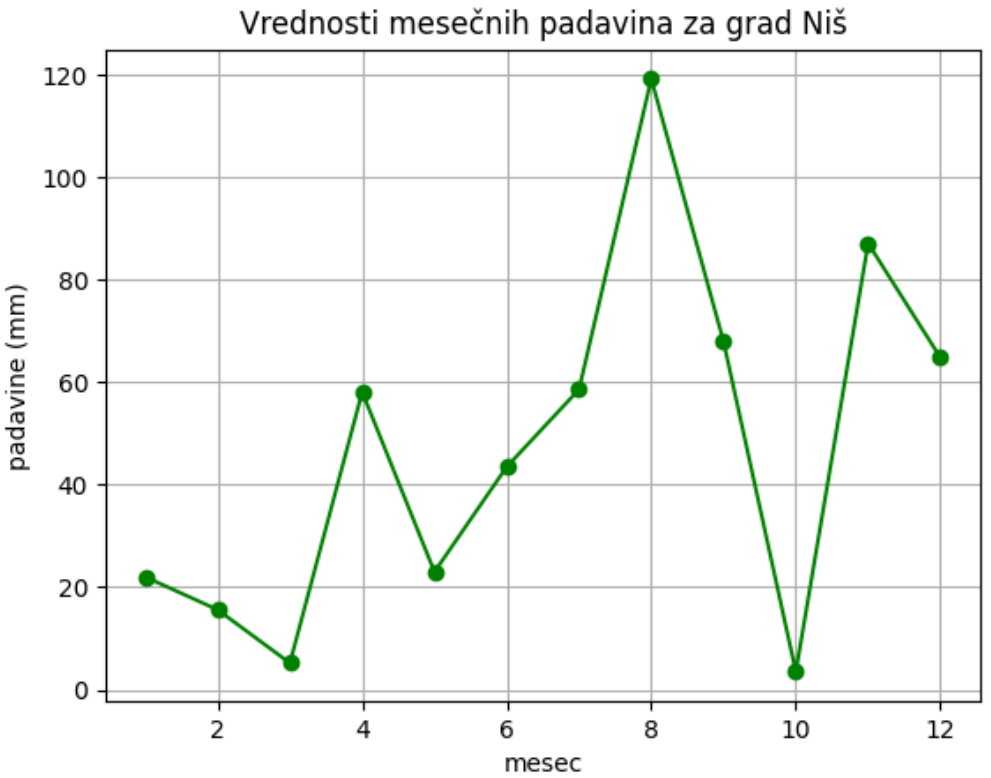
Prikaz linijskog dijagrama

Co-funded by the European Union

Za zadati niz vrednosti mesečnih padavina (mm) za grad Niš nacrtati linijski dijagram.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(1, 13)
y = [22, 15.6, 5.2, 58.1, 23, 43.5, 58.6, 119.2, 67.9, 3.5, 87.1, 64.8]

# kreira linijski dijagram
plt.plot(x, y, "go-")
# opis x ose
plt.xlabel("mesec")
# opis y ose
plt.ylabel("padavine (mm)")
# ime grafikona
plt.title("Vrednosti mesečnih padavina za grad Niš")
# mreža pomoćnih linija
plt.grid()
plt.show()
```



Višestruki linijski dijagram

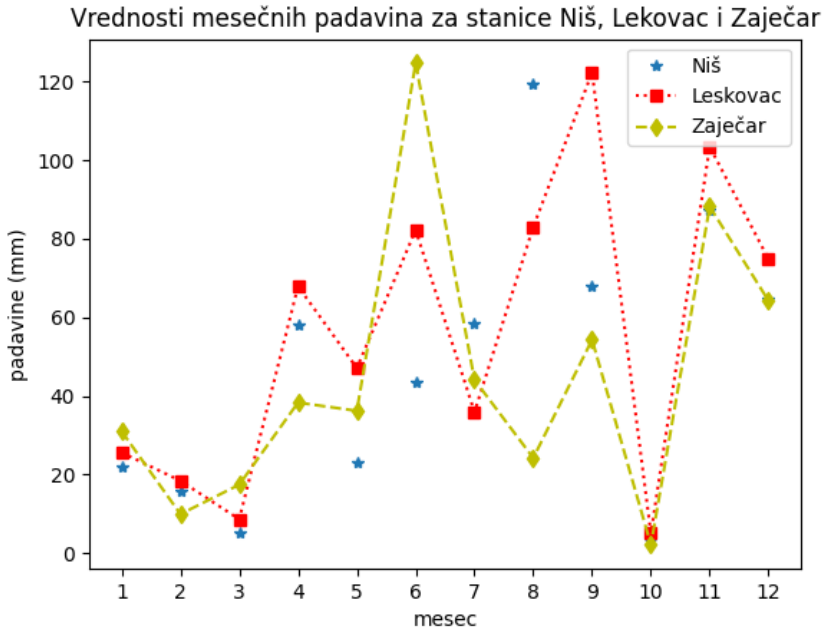
Co-funded by the European Union

Za zadate nizove vrednosti mesečnih padavina (mm) za Niš, Leskovac i Zaječar nacrtati linijske dijagrame.

```
import numpy as np
import matplotlib.pyplot as plt
meseci = np.arange(1, 13)
nis = [22, 15.6, 5.2, 58.1, 23, 43.5, 58.6, 119.2, 67.9, 3.5, 87.1, 64.8]
leskovac = [25.6, 18.4, 8.6, 67.9, 47.1, 82.2, 35.9, 83, 122.3, 5, 103.4, 74.8]
zajecar = [31, 9.9, 17.4, 38.3, 36.2, 124.7, 44.1, 24.2, 54.3, 2.1, 88.2, 64.2]

plt.plot(meseci, nis, "*", label = "Niš")
plt.plot(meseci, leskovac, "rs:", label = "Leskovac")
plt.plot(meseci, zajecar, "yd--", label = "Zaječar")

plt.xlabel("mesec")
plt.ylabel("padavine (mm)")
plt.title("Vrednosti mesečnih padavina za stanice Niš, Lekovac i Zaječar")
plt.xticks(meseci)
plt.legend()
plt.show()
```



Grafik funkcije

Co-funded by the European Union

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

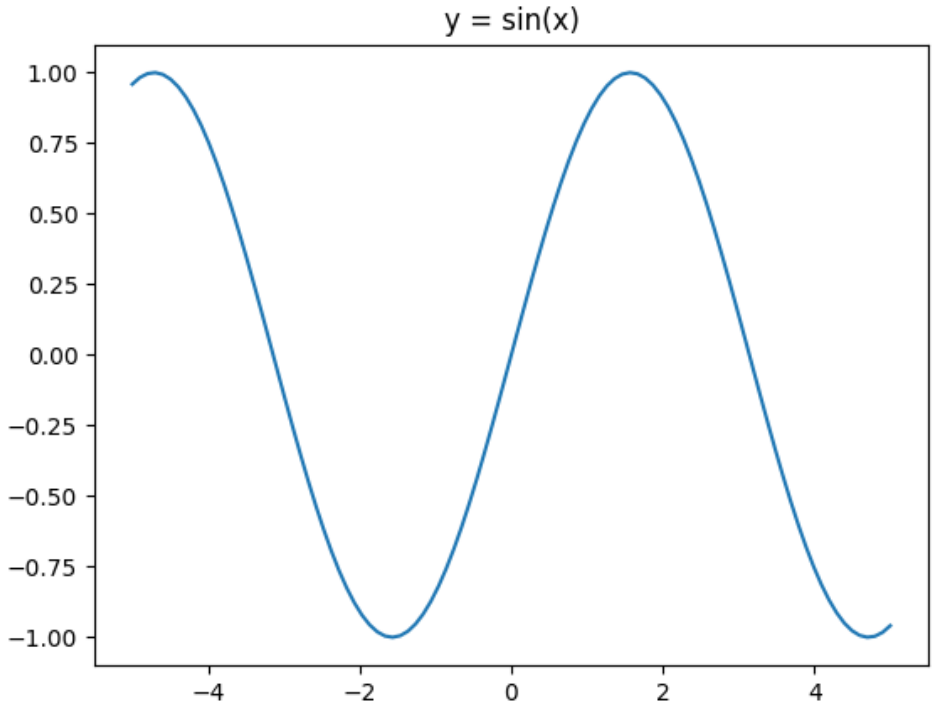
x = np.linspace(-5, 5, 100)

plt.title("y = sin(x)")

# postavlja veličinu fonta
plt.rc("font", size=12)

# naslov je podebljan
plt.rc("axes", titleweight="bold")

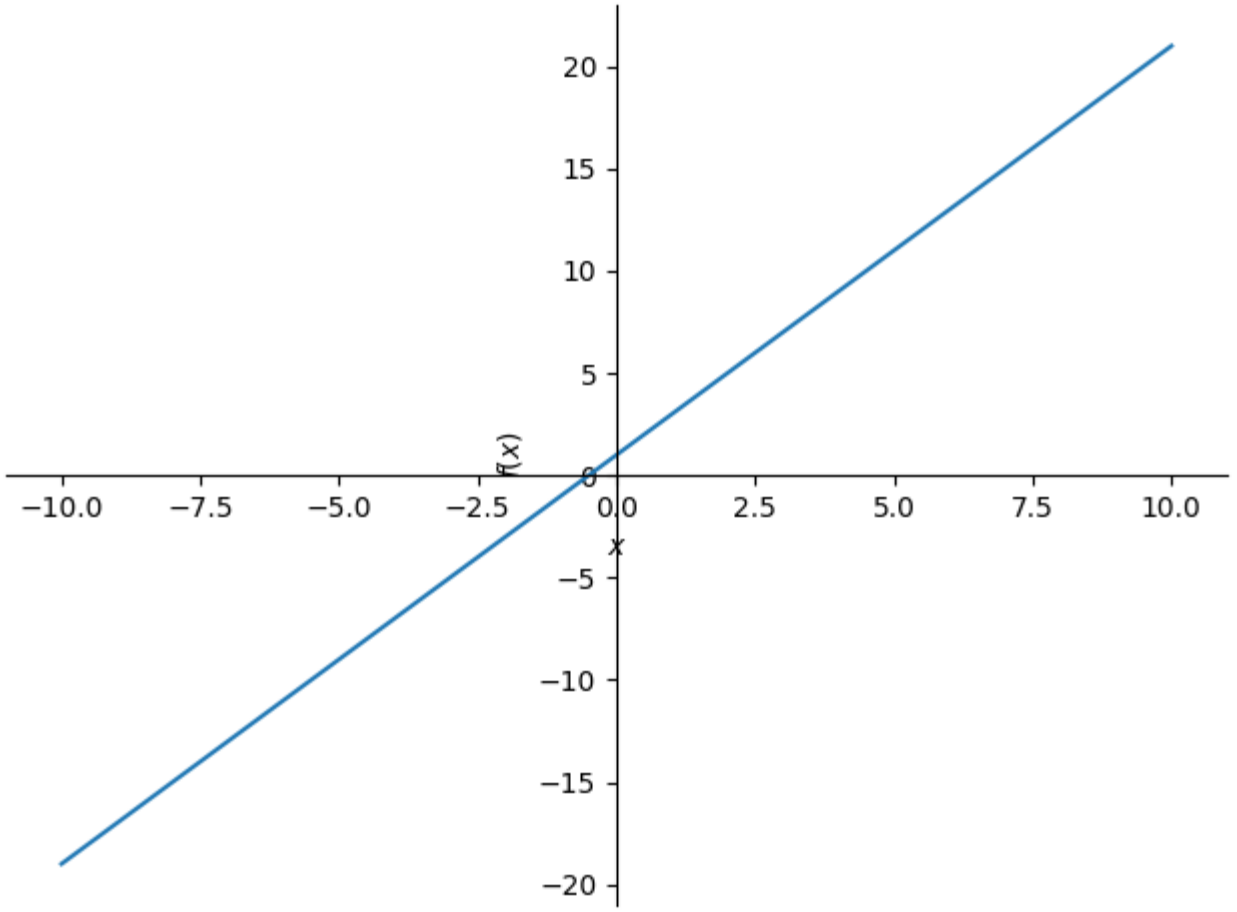
plt.plot(x, np.sin(x))
plt.show()
```



Iscrtavanje grafika funkcije

Co-funded by the European Union

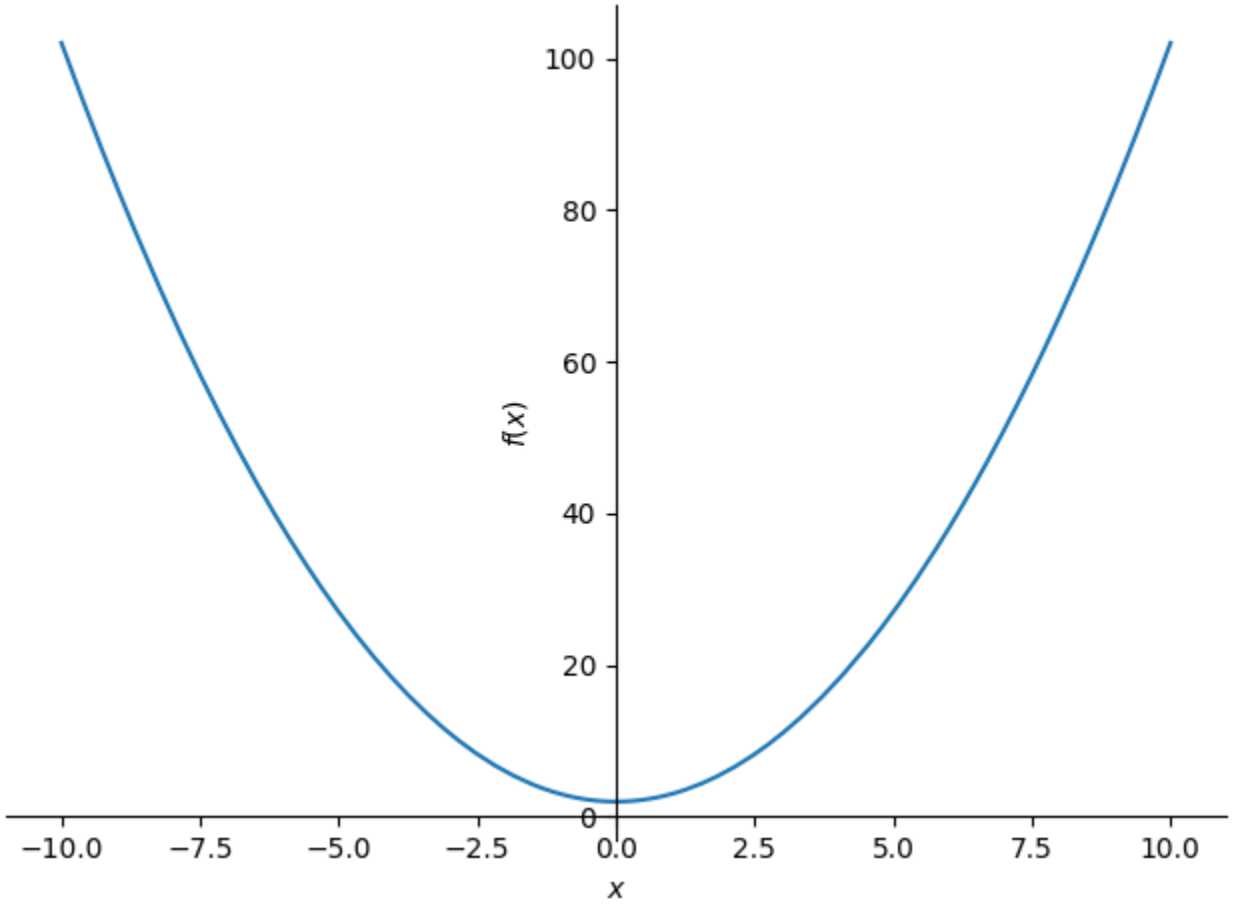
```
from sympy import *
x = symbols('x')
f = 2 * x + 1
plot(f)
```



Iscrtavanje grafika funkcije

Co-funded by the European Union

```
from sympy import *
x = symbols('x')
f = x**2 + 2
plot(f)
```



Pravougaoni dijagram

Co-funded by the European Union

Pravougaoni dijagram (engl. bar chart) najčešće prikazuje numeričke vrednosti pridružene pojedinim kategorijskim podacima. Njime se mogu porediti i numeričke vrednosti dodeljene različitim objektima unutar iste kategorijske podele.

Analiziraćemo četiri ispitna roka i broj studenata koji su polagali ispite na dva različita studijska programa.

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

kategorije = ["Januar", "April", "Jun", "Septembar"]
arh, gr = [100, 82, 67, 53], [82, 54, 62, 44]

plt.rc("font", size = 12)
plt.subplot(131)
plt.title("Arhitektura")
plt.barh(kategorije, arh, height = 0.15)
plt.subplot(132)
plt.title("Građevinarstvo")
plt.bar(kategorije, gr, color = "red", width = 0.25)
plt.subplot(133)
plt.title("Arhitektura/Građevinarstvo")
x = np.arange(4)
plt.bar(x-0.2, arh, width = 0.4, label = "Arhitektura")
plt.bar(x+0.2, gr, width = 0.4, label = "Građevinarstvo")
plt.xticks(x, kategorije)
plt.legend()
plt.show()
```

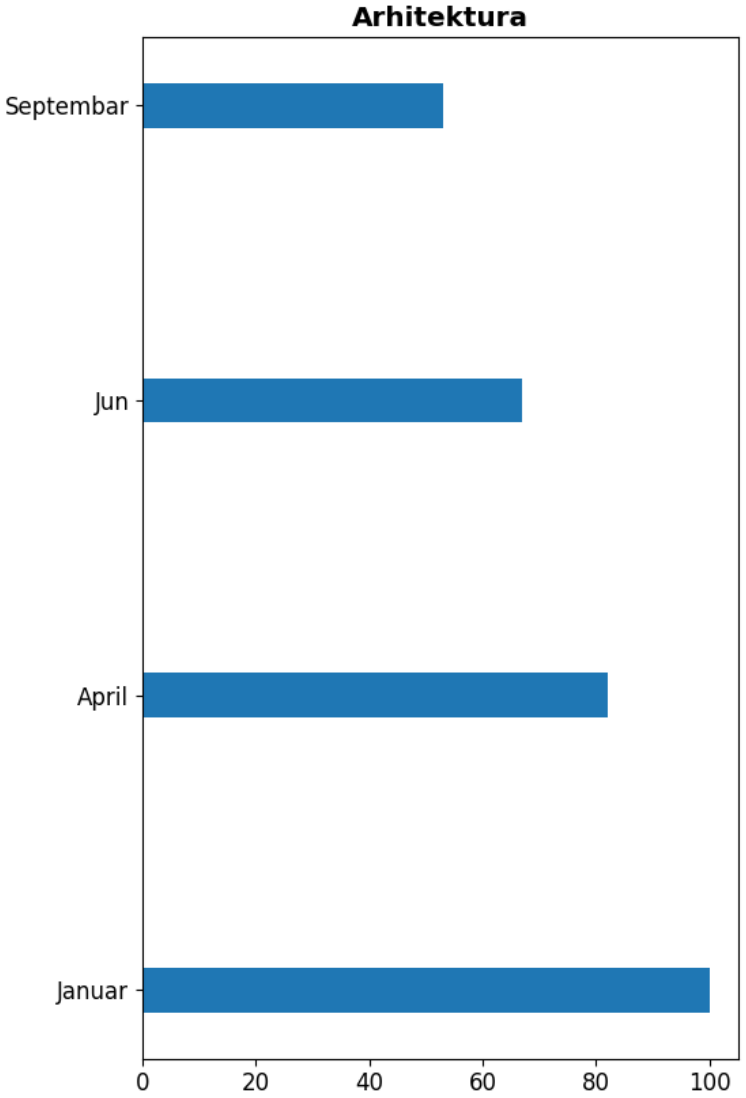
Primer ilustruje upotrebu tri dijagrama u okviru iste slike, organizovane u obliku matrice sa jednom vrstom i tri kolone. Za postavljanje tekućeg prikaza za iscrtavanje koristi se funkcija subplot().

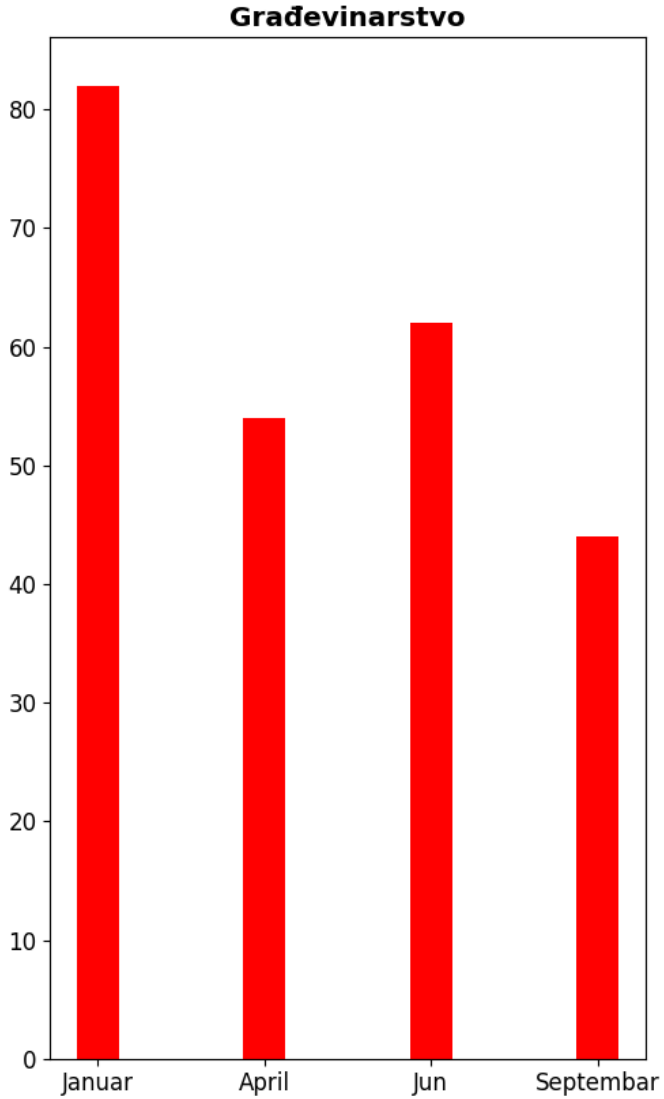
Funkcija barh() prihvata bar dva parametra, sekvencu sa kategorijama i sekvencu sa pridruženim numeričkim vrednostima. Parametar height definiše visinu horizontalnog pravougaonika.

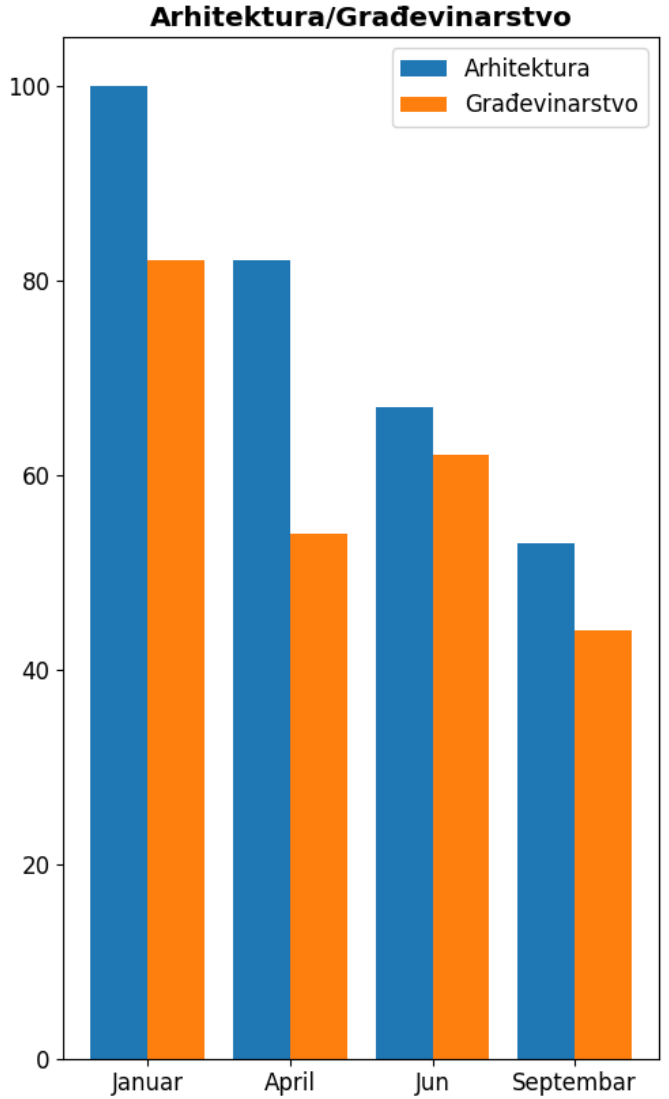
Funkcija bar() iscrtava vertikalni dijagram. Parametar width definiše širinu vertikalnog pravougaonika, a color upotrebljenu boju ispune.

Pravougaoni dijagram

Co-funded by the European Union







Kružni dijagram

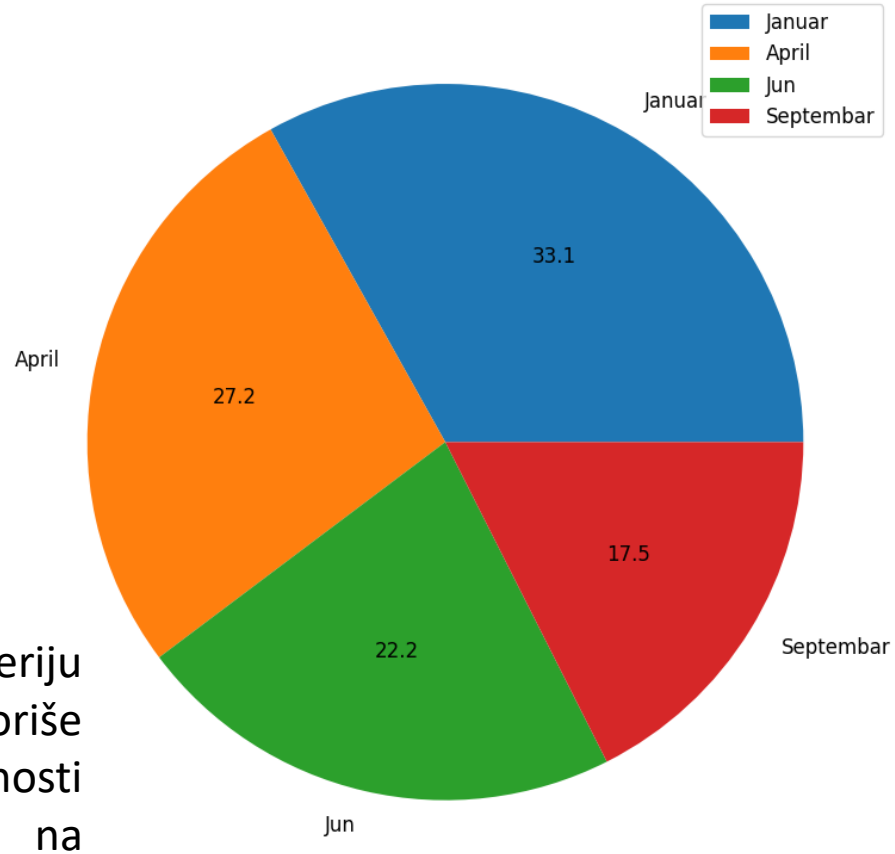
Co-funded by the European Union

Kružni dijagram (engl. pie chart) prikazuje procentualno učešće posmatranih kategorija u odnosu na neku numeričku veličinu.

```
import matplotlib.pyplot as plt

kategorije = ["Januar", "April", "Jun", "Septembar"]
arh = [100, 82, 67, 53]

plt.rc("font", size = 12)
plt.pie(arh, labels = kategorije, autopct = "%3.1f")
plt.legend()
plt.show()
```



Funkcija pie() prima tri argumenta: prvi predstavlja numeričku seriju čiji zbir predstavlja 100% površine kružnog dijagrama, drugi memoriše kategorijske labele, a treći se odnosi na format ispisa za vrednosti (procente) na kružnim odsečcima. Format %3.1f odnosi se na decimalni broj sa tri cifre od kojih je jedna iza decimalne tačke.

Histogram

Co-funded by the European Union

Za razliku od pravougaonog dijagrama koji prikazuje frekvencije kategorijskih podataka, histogram predstavlja grafikon frekvencija za kontinualne numeričke podatke. On uključuje opsege podataka grupisane po intervalima na x osi, sa pridruženim frekvencijama na y osi.

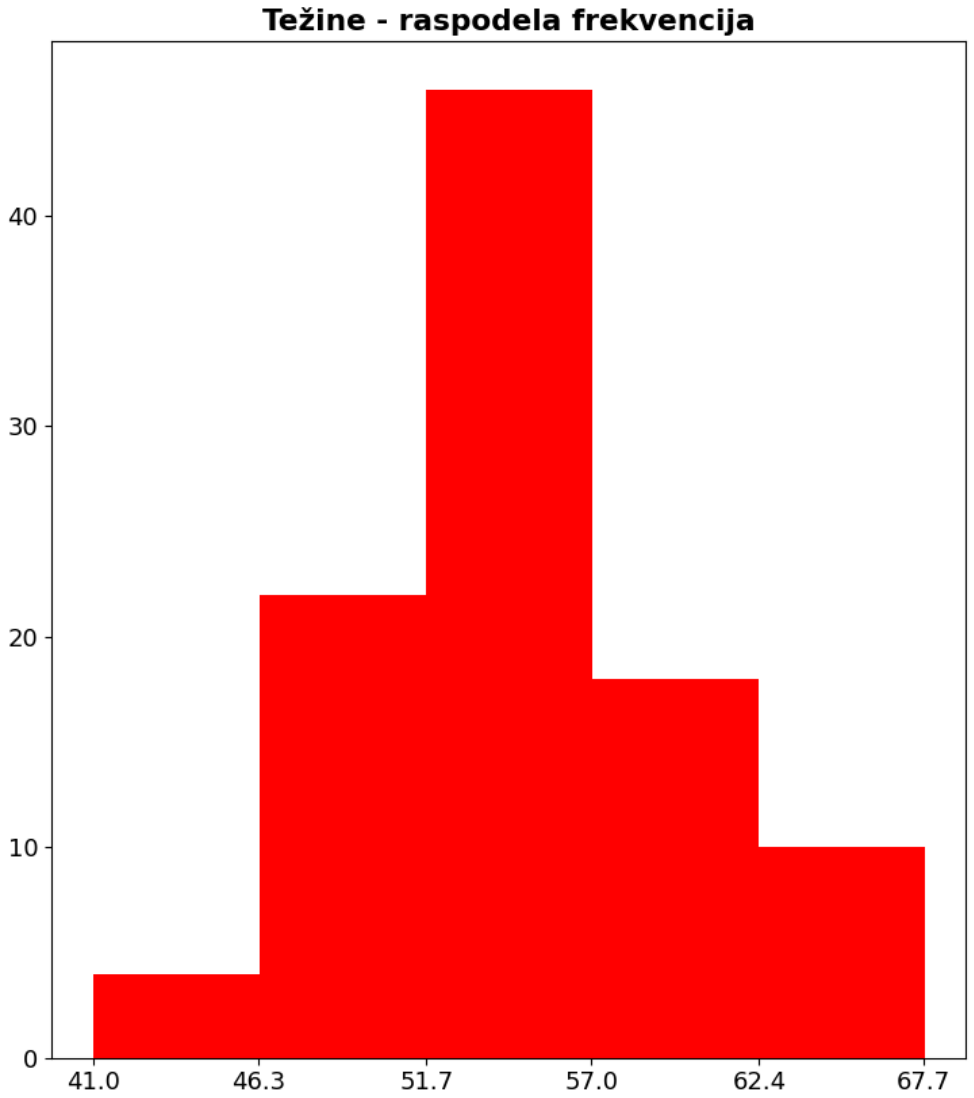
```
import matplotlib.pyplot as plt
import numpy as np

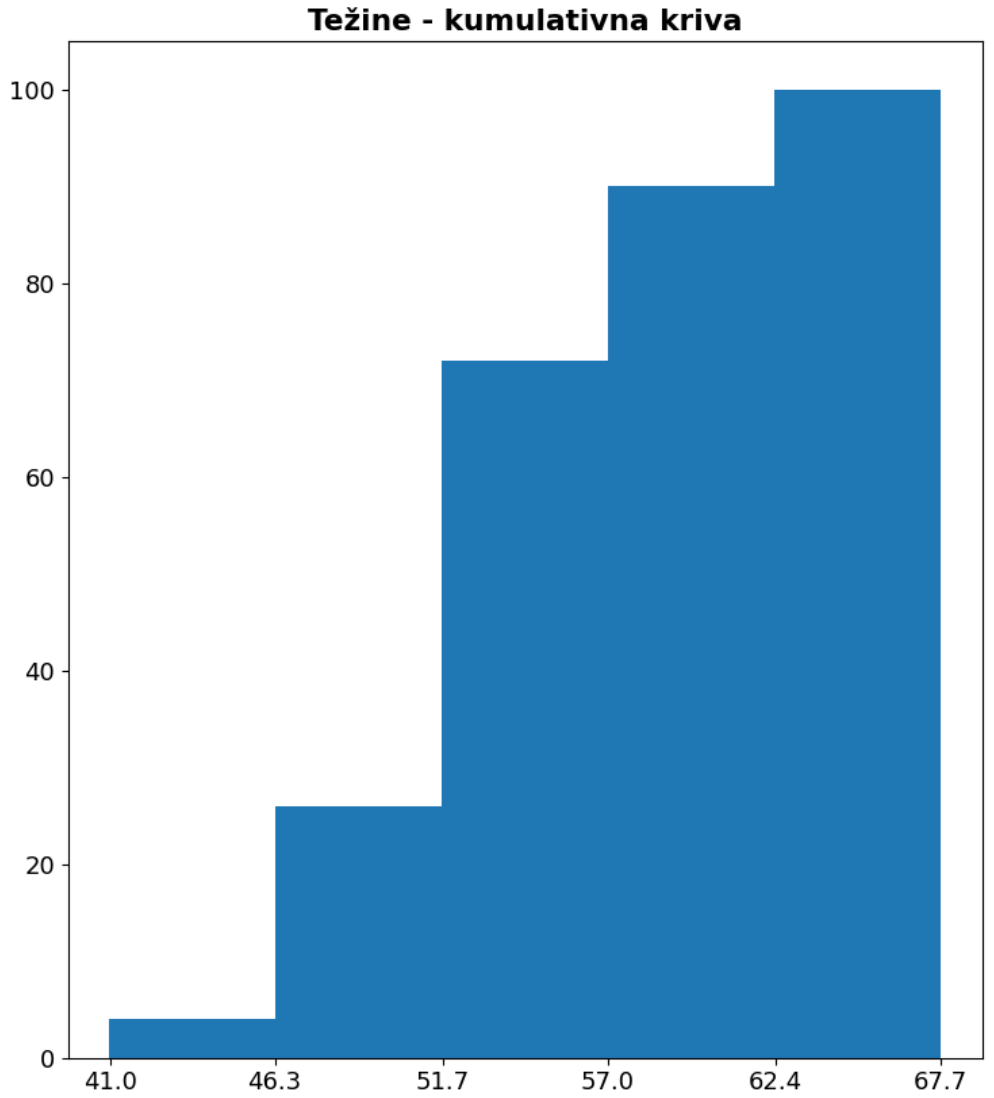
tezine = np.random.normal(55, 5, 100)
x = np.round(np.linspace(tezine.min(), tezine.max(), 6), 1)
plt.rc('font', size = 13)
plt.subplot(121)
plt.title("Težine - raspodela frekvencija")
plt.hist(tezine, bins = 5, color = "red")
plt.xticks(x)
plt.subplot(122)
plt.title("Težine - kumulativna kriva")
plt.hist(tezine, bins = 5, cumulative = True)
plt.xticks(x)
plt.show()
```

Prvo se formira niz od 100 težina koje podležu normalnoj raspodeli sa očekivanom težinom od 55 kg i standardnom devijacijom od 5 kg. Putem niza x definišu se granice 5 jednakih intervala širine (maxtezina – mintezina)/5.
Upotrebom funkcije hist() formiraju se dva histograma: prvi prikazuje raspodelu težina po 5 intervala (parametar bins), uz upotrebu crvene boje (parametar color); drugi histogram je kumulativan (parametar cumulative = True).

Histogram

Co-funded by the European Union





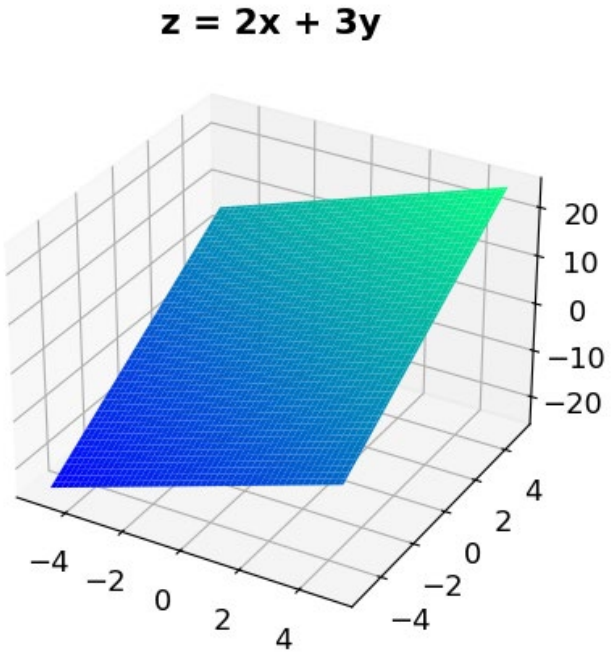
Površinski dijagram

Co-funded by the European Union

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

x, y = np.linspace(-5, 5, 100), np.linspace(-5, 5, 100)
X, Y = np.meshgrid(x, y)
Z = 2 * X + 3 * Y

tekuci = plt.subplot(projection="3d")
plt.title("z = 2x + 3y")
tekuci.plot_surface(X, Y, Z, cmap="winter")
plt.show()
```



Funkcija meshgrid() vraća dve matrice (X i Y) koje čuvaju koordinate tačaka iz pravougaone mreže – tačka iz i-te vrste i j-te kolone mreže ima koordinate X[i, j], Y[i, j]. Program za kreiranje površinskog dijagrama koristi metodu plot_surface() klase koja modeluje tekući prikaz u okviru slike.

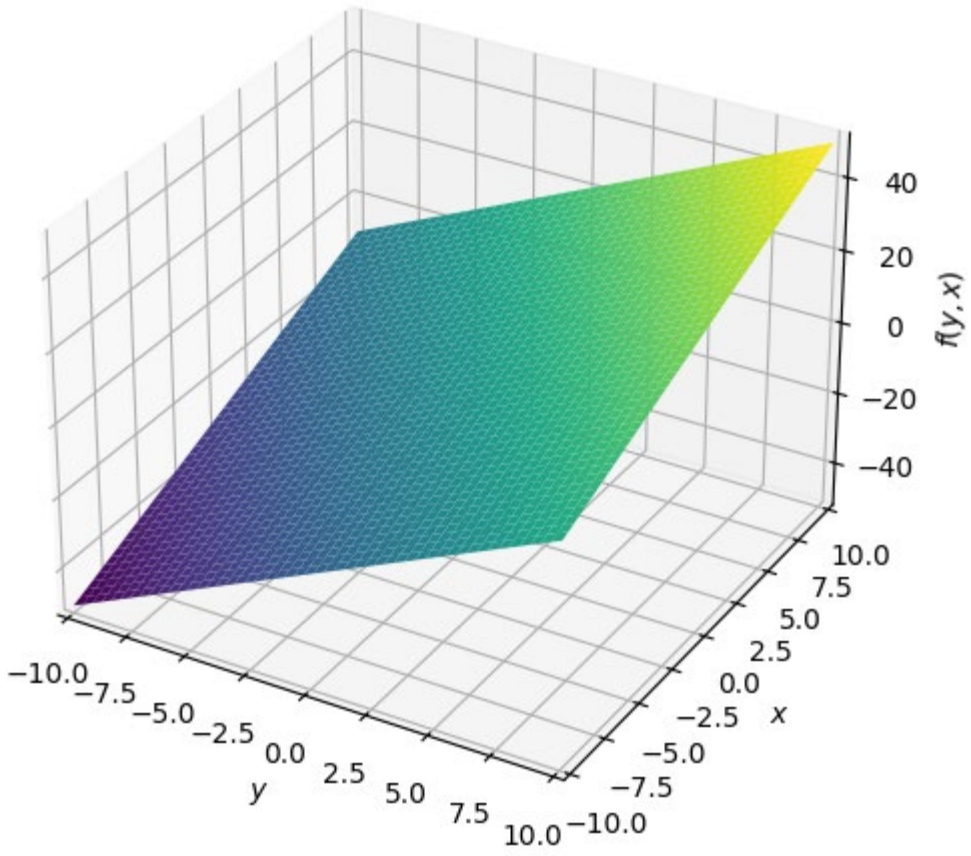
Opcioni parametar funkcije subplot() projection="3d" definiše trodimenzionalnu prirodu prikaza. Prilikom crtanja površinskog dijagrama koristi se mapa boja "winter". Mapa omogućava da se tačke na površi prikažu različitim bojama (ili nijansama iste boje) u zavisnosti od vrednosti funkcije z = f(x, y).

Iscrtavanje grafika funkcije više promenljiva

Co-funded by the European Union

f(x, y) = 2 x + 3 y

```
from sympy import *
from sympy.plotting import plot3d
x, y = symbols('x y')
f = 2 * x + 3 * y
plot3d(f)
```



Safety4EU projekat
L3. Analiza podataka, programiranje i interpretacija

Baze podataka

Milan Gocić

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Šta je baza podataka?

Co-funded by the European Union

Baza podataka predstavlja kolekciju međusobno povezanih podataka koji su organizovani u tabele.

Podaci u bazama podataka su organizovani u **dvodimenzionalne tabele**.

Tabela može da ima više **kolona**, gde svaka kolona predstavlja neku **osobinu** ili **atribut**.

Vrste tabele čine konkretni podaci, odnosno konkretne vrednosti osobina/atributa nekog objekta.

Osnovna namena baze podataka je da bude skladište za podatke.

Podaci mogu biti različitog tipa: tekstualni, numerički, slike, audio i video zapisi.

Postupak izbora i definisanja tabela za bazu podataka je deo procesa modeliranja odnosno izgradnje **modela podataka**.

Međusobna povezanost podataka je ono po čemu se baza podataka razlikuje u odnosu na fajl sisteme (datoteke) i programe za unakrsna izračunavanja kao što je Excel.

Baza podataka sadrži i tzv. **metapodatke**, odnosno podatke o samoj strukturi baze podataka.

Co-funded by the European Union

Sistem za upravljanje bazama podataka (DBMS)

DBMS obično nudi:

- **Jezik za opis podataka** (*eng. Data Definition Language - DDL*), koji omogućava korisnicima definisanje tipa i strukture podataka, kao i ograničenja nad podacima memorisanim u bazi podataka (CREATE TABLE naredba).
- **Jezik za manipulaciju podacima** (*eng. Data Manipulation Language - DML*), koji omogućava korisnicima umetanje, ažuriranje, brisanje i pretraživanje podataka iz baze podataka (SELECT, INSERT INTO, UPDATE naredbe).
- **Jezik za definisanje načina memorisanja podataka** (*eng. Storage Definition Language - SDL*), koji se koristi za specificiranje interne šeme baze podataka.
- **Kontrolisani pristup bazi podataka**, što uključuje različite funkcije i mehanizme za pristup podacima u bazi podataka

Funkcije DBMS-a

Co-funded by the European Union

DBMS treba da obezbedi sledeće funkcije za kontrolisani pristup podacima u bazi podataka:

- **Sigurnosni sistem**, koji onemogućava pristup bazi podataka neautorizovanim korisnicima (sigurnosni servisi), odnosno samo autorizovani korisnici mogu da koriste podatke u skladu sa definisanim privilegijama (autorizacioni servisi)
- **Integritetni sistem**, koji održava konzistentnost podataka u bazi podataka, odnosno da se sve promene dešavaju u skladu sa definisanim pravilima.
- **Sistem za kontrolu konkurencije**, koji dopušta deljivi pristup podacima iz baze podataka, tj. da se obezbedi korektno ažuriranje podataka kada više korisnika pokušava istovremeno da vrši ažuriranja.
- **Sistem za kontrolu oporavka baze podataka,** koji omogućava rekonstrukciju prethodnog konzistentnog stanja u slučaju neke hardverske ili softverske neispravnosti.
- **Katalog** kome korisnici mogu pristupati, koji sadrži opis podataka koji su memorisani u bazi podataka.
- **Podrška za transakcije**, koja obezbeđuje korektno izvršavanje niza transakcija koje mogu biti međusobno zavisne; transakcija je skup operacija upisa i čitanja iz baze podataka koji se tretira kao celina tj. ima svoj početak i kraj.
- **Razne korisničke funkcije**, kao što su import, eksport podataka, statističke analize, funkcije za nadgledanje.

Model podataka

Co-funded by the European Union

U procesu razvoja baze podataka formira se model realnog sistema.

Postoji mnogo različitih mogućnosti da se modelira sistem.

Deo modela sistema koji se odnosi na podatke naziva se **model podataka**.

Model podataka je u literaturi definisan kao Model Objekti-Veze (MOV) odnosno *Entity-Relationship* (E-R) model

Tipovi podataka

Co-funded by the European Union

Tipovi podataka specificiraju tip resursa koji se može čuvati u jednoj koloni tabele. Ta kolona predstavlja atribut koji opisuje sve objekte u tabeli, i svaka kolona može čuvati atribut različitog tipa.

Postoji pet tipova podataka:

- NULL – prazno polje, informacija je trenutno nepoznata
- INTEGER – označeni celi broj
- REAL – decimalna vrednosti
- TEXT – niz karaktera
- BLOB – binarni objekat koji može čuvati bilo koju vrstu podataka

Relacioni model podataka

Co-funded by the European Union

Najpopularniji i najrasprostranjeniji model podataka danas.

Relacija, kao osnovni koncept relacionog modela je zapravo matematička relacija, i ima jednostavnu reprezentaciju u obliku dvodimenzionalne tabele sa podacima.

Ovaj model daje osnovu za jezik visokog nivoa za pristup podacima.

U fazi projektovanja baze podataka treba najpre prepoznati objekte realnog sveta (entiteti) za koje treba čuvati podatke i prepoznati njihove atribute.

Izabrani entiteti će u bazi podataka biti predstavljeni tabelama.

Atribut

Co-funded by the European Union

Svaki objekat, odnosno entitet, poseduje neka **svojstva**.

Svojstva ili atributi objekta će biti **predstavljena kolonama** u odgovarajućoj tabeli.

Objekti međusobno mogu biti **povezani** različitim odnosima odnosno **relacijama**.

Atributi su zajedničke osobine koje poseduju svi entiteti jednog skupa entiteta.

Pošto se entiteti odnosno objekti realnog sveta predstavljaju **relacijom**, atributi predstavljaju svojstva te relacije.

Svaka relacija predstavlja skup **torki**, gde se svaka torka odnosi na konkretan entitet iz skupa entiteta.

U tabelarnom prikazu relacije, **vrste** tabele su podaci o konkretnim entitetima, odnosno torke

Domen atributa

Co-funded by the European Union

Domen atributa je skup vrednosti koje neki atribut može uzimati.

Svaki domen atributa se definiše: tipom podataka, dužinom podataka i opsegom vrednosti.

Relacija se u bazi podataka predstavlja **dvodimenzionalnom tabelom**, gde vrste odgovaraju pojedinim slogovima, a kolone atributima.

Atributi se mogu pojavljivati u bilo kom redosledu u tabeli. Redosled vrsta tabele takođe nije bitan.

Svaka tabela, kao i svaka kolona u tabeli imaju ime.

NULL vrednost atributa

Co-funded by the European Union

DBMS dozvoljava da atribut nema dodeljenu vrednost što podrazumeva korišćenje tzv. NULL vrednosti.

NULL vrednost može da ima dva značenja:

- Vrednost atributa za posmatrani entitet ne postoji ili još uvek nije poznata.
- Vrednost atributa za posmatrani nije primenjiva.

Svojstva relacije

Co-funded by the European Union

Relacija ima sledeća svojstva:

- Svaka **relacija** ima **ime** koje se razlikuje od imena svih ostalih relacija u šemi relacione baze podataka,
- Svaka **ćelija** tabele (određena vrstom i kolonom) kojom je relacija predstavljena sadrži samo jednu **atomičnu (prostu) vrednost**,
- Svi **atributi** jedne relacije imaju **različito ime**,
- Sve **vrednosti** jednog **atributa** su iz istog domena,
- Sve **torke** relacije su **različite**, tj. u relaciji ne postoje duple torke.

Ključevi relacije

Co-funded by the European Union

U relacionom modelu podataka atribut ili skup atributa čije vrednosti predstavljaju identifikator entiteta (torke u relaciji) nazivamo **ključem relacije**.

Takvi atributi se nazivaju **ključni atributi**.

Ako relacija ne poseduje atribut ili skup atributa koji je identifikuju, tada se uvodi specijalni identifikacioni atribut - **ključ surogat**, koji se obično označava sa ID.

Ključ

Pošto su sve torke relacije različite, u relaciji mora postojati atribut ili skup atributa (tzv. kompozitni ključ – ključ od više atributa), nazvani relacioni ključevi ili ključevi relacije, koji na jedinstven način identifikuje svaku torku relacije.

Primarni ključ

Ključ kandidat koji je odabran da na jedinstven način identifikuje torke unutar relacije.

Ključ surogat

Identifikator koji je dodat relaciji kao primarni ključ, zato što relacija nema odgovarajući atribut ili skup atributa koji može biti primarni ključ relacije.

Spoljni ključ / Strani ključ

Atribut ili skup atributa jedne relacije koji se uparuje sa ključem kandidatom neke druge ili iste relacije. **Važan za ostvarivanje međusobnih veza između tabela**.

Osobine primarnog ključa

Co-funded by the European Union

Da bi neki atribut mogao da bude primarni ključ, mora da zadovolji sledeće uslove:

- **jednoznačnost** – ne smeju da postoje dve pojave za istim vrednostima ključnih atributa
- **minimalnost** – da ne postoji podskup atributa ključa koji nije jednozačan (npr. složeni primarni ključ objekta Osobe ne može biti kombinacija atributa JMBG i Prezime, jer Prezime nema osobinu jednoznačnosti)
- **određenost** – postojanje vrednosti u trenutku stvaranja instance
- **stabilnost** – otpornost na promene tokom vremena
- **raspoloživost** – dostupnost svim korisnicima
- **neutralnost** – prema značenju vrednosti ključa (samogovoreće šifre)

Difoltna vrednost primarnog ključa je NOT NULL, odnosno ključ ne sme da bude prazno polje, jer onda neće moći da jedinstveno identifikuje dati objekat.

SQL jezik

Co-funded by the European Union

Da bismo mogli da napišemo bilo kakav program ili kreiramo veb sajt koji pristupa bazi podataka neophodno je da upoznamo **jezik** kojim se manipuliše podacima baze podataka.

U pitanju je jezik pod nazivom **SQL** (eng. Structured Query Language) odnosno **struktuirani jezik za zadavanje upita**.

Ponegde se SQL akronim interpetira i kao "Standard Query Language" odnosno **standardni jezik za zadavanje upita**.

SQL jezik je standardni jezik koji je predstavljen ranih 1970-tih i od tada je doživeo nekoliko standardizacija od kojih je poslednja bila 1999, a nakon nje još nekoliko revizija.

Dobra stvar kod standardizacije SQL jezika je što jednom naučen može biti primenjen na bazama podataka različitih proizvođača.

Da ne bi sve bilo idealno, u praksi možete računati da prethodna tvrdnja nije potpuno tačna, jer svaki proizvođač uvodi nove mogućnosti koje manje ili više odstupaju od standarda.

SQL jezik

Co-funded by the European Union

Structured Query Language (SQL) predstavlja jezik koji je projektovan za potrebe pretraživanja i upravljanja podacima u **sistemima za upravljanje relacionim bazama podataka** (Relational Database Management Systems - RDBMS), za kreiranje i modifikaciju šema relacione baze podataka i za kontrolu pristupa objektima baze podataka.

Osnovna karakteristika SQL-a je da se radi o deklarativnom jeziku za upravljanje i manipulaciju podacima u relacionim bazama podataka.

SQL omogućava pribavljanje, dodavanje, ažuriranje i brisanje podataka u relacionoj bazi podataka.

Većina proizvođača RDBMS-a proširuje SQL standard dodajući proceduralne elemente, kontrolne strukture, korisnički definisane tipove podataka i druge elemente.

SQL upiti

Co-funded by the European Union

SQL upiti se po svojoj funkciji mogu podeliti na sledeće grupe:

1. Data Definition Language (DDL) - jezik koji se koristi za definisanje strukture relacione baze podataka

2. Data Manipulation Language (DML) - jezik za pribavljanje i ažuriranje podataka u relacionoj bazi podataka.

3. Data Control Language (DCL) - jezik za kontrolu podataka pomoću koga se određuje koji korisnik šta sme da radi sa podacima.

Jezik za kontrolu pristupa podacima (DCL)

Co-funded by the European Union

Jezik za kontrolu pristupa podacima (DCL) koristi se da bi se upravljalo pristupu bazi podataka.

Sastoji se od dve naredbe: naredba GRANT koristi se da bi se korisniku dodelio pristup bazi podataka i naredba REVOKE, koja se koristi da bi se oduzela postojeća dozvola pristupa.

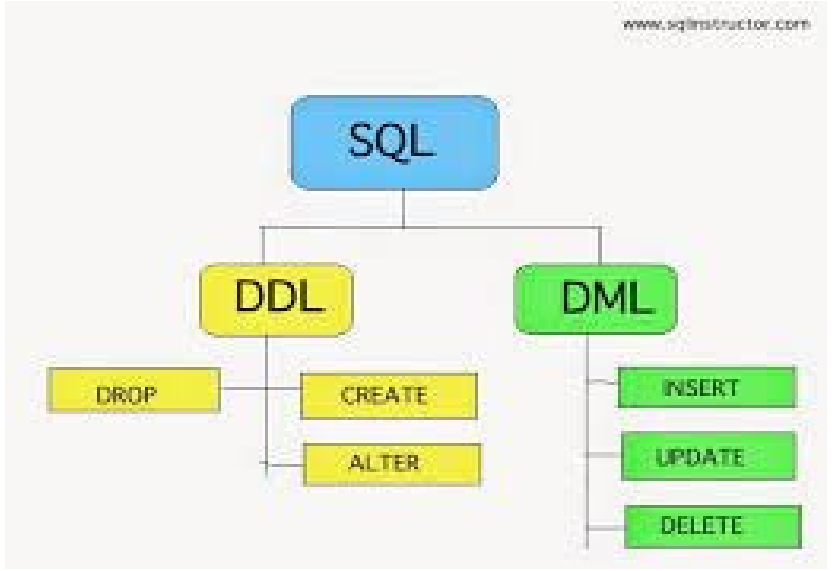
Ove dve naredbe predstavljaju srž modela obezbeđenja relacionih baza podataka.

Jezik za definisanje podataka (DDL)

Co-funded by the European Union

SQL DDL naredbe koriste se za kreiranje, izmenu i brisanje same relacione baze podataka kao i objekata koji čine relacionu bazu podataka.

Centralni objekat svake relacione baze podataka jeste tabela.



DDL naredbe

Co-funded by the European Union

Primeri uobičajeno korišćenih naredbi ovog jezika obuhvataju one koje se upotrebljavaju da bi se napravila nova tabela baze podataka (CREATE TABLE), menjaju definiciju tabele baze podataka (ALTER TABLE) i brišu tabelu baze podataka (DROP TABLE).

Kreiranje tabele

Co-funded by the European Union

Za kreiranje tabele koristi se SQL naredba CREATE TABLE.

CREATE TABLE <ime tabele>
(<lista_deklaracija_kolona>
[, <lista_deklaracija_ogranicenja_tabele]);

Primer kreiranja tabele RADNIK

Co-funded by the European Union

```
CREATE TABLE RADNIK
        (MatBr Int,
        Ime VarChar(15),
        SSlovo Char(1),
        Prezime VarChar(15),
        DatRodj Date,
        Pol Char(1),
        Plata Numeric(8, 2),
        Adresa VarChar(30),
        Rukovodilac Int,
        Sektor Int);
```

Brisanje tabele

Co-funded by the European Union

DROP TABLE <ime tabele>;

Koristi se za brisanje postojeće tabele u bazi podataka.

Primer:

DROP TABLE RADNIK;

ALTER TABLE iskaz

Co-funded by the European Union

Koristi se za dodavanje, brisanje ili modifikovanje kolona u postojećoj tabeli.

Za dodavanje kolone koristi se sledeća sintaksa

```
ALTER TABLE <ime tabele>
ADD <naziv kolone> <tip podatka>;
```

Primer dodavanja kolone:

```
ALTER TABLE RADNIK
ADD Zanimanje VarChar(15);
```

Brisanje kolone

Co-funded by the European Union

Za brisanje kolone iz tabele koristi se sledeća sintaksa

```
ALTER TABLE <ime tabele>
DROP COLUMN <naziv kolone>;
```

Primer brisanja kolone Zanimanje iz tabele RADNIK

```
ALTER TABLE RADNIK
DROP COLUMN Zanimanje;
```

Ograničenja

Co-funded by the European Union

Ograničenja koja možete da definišete za kolonu kod kreiranja tabela su:

- NULL ili NOT NULL - definiše da kolona može ili ne može imati NULL vrednosti.
- UNIQUE - definiše da kolona ima jedinstvene vrednosti (kandidati za ključeve).
- PRIMARY KEY - definiše da kolona predstavlja primarni ključ tabele (može da se primeni na samo jednu kolonu u tabeli).
- FOREIGN KEY - definiše koja kolona ili koje kolone čine spoljni ključ tabele.
- CHECK - definiše ograničenje za proveru vrednosti kolone (koristi se kod upisa ili ažuriranja vrednosti).
- DEFAULT - definiše podrazumevanu vrednost za kolonu (kolona uzima ovu vrednost, ako vrednost kolone nije navedena).
- REFERENCES - definiše da kolona predstavlja spoljni ključ tabele.

Spoljni ključ

Co-funded by the European Union

Kod deklaracije spoljnog ključa tabele treba navesti iza ključne reči REFERENCES ime referencirane tabele i opciono, u maloj zagradi, ime referencirane kolone u toj tabeli.

DBMS očekuje da je referencirana kolona primarni ključ (kolona deklarisana sa PRIMARY KEY) u referenciranoj tabeli.

Na taj način atribut iza koga stoji klauzula REFERENCES definisan je kao spoljni ključ u odnosu na primarni ključ tabele čije je ime navedeno iza klauzule REFERENCES.

Primer:

```
CREATE TABLE PROJEKAT
        (Broj Int PRIMARY KEY,
        Naziv VarChar(25) NOT NULL,
        Lokacija VarChar(15) DEFAULT("Niš"),
        Sektor Integer REFERENCES SEKTOR(Broj));
```

Jezik za manipulisanje podacima (DML)

Co-funded by the European Union

Jezik za manipulisanje podacima (DML) sadrži podskup SQL naredbi koje se najčešće koriste – i to one koje jednostavno manipulišu sadržajem baze podataka u nekom obliku.

Četiri najčešće korišćene naredbe ovog jezika su:

SELECT - izvlači informacije iz baze podataka;
INSERT - dodaje nove informacije bazi podataka;
UPDATE - modifikuje informacije trenutno uskladištene u bazi podataka, i DELETE - uklanja informacije iz baze podataka.

SELECT naredba

Co-funded by the European Union

Pretraživanje i pribavljanje podataka su najčešće operacije koje korisnici izvršavaju u relacionoj bazi podataka. Za pretraživanje i pribavljanje podataka SQL jezik obezbeđuje naredbu SELECT.

Naredba SELECT pribavlja podatke iz jedne tabele ili više povezanih tabela koje se nalaze u relacionoj bazi podataka.

U svom osnovnom obliku naredba SELECT ne može ni na koji način da izmeni podatke koji se nalaze u relacionoj bazi podataka.

Naredba SELECT je jedna od najkompleksnijih naredbi SQL programskog jezika. Uključuje veći broj ključnih reči klauzula:

```
SELECT kolona1, kolona2, ...
FROM ime tabele
WHERE uslov
ORDER BY kolona1, kolona2, ... ASC|DESC;
```

SELECT naredba

Co-funded by the European Union

SELECT - definiše listu kolona koje će biti uključene u rezultujuću tabelu.

FROM - definiše tabele iz kojih se pribavljaju podaci za potrebe generisanja rezultujuće tabele. Klauzula FROM može da uključi jednu ili više opcionih JOIN klauzula za povezivanje tabela na osnovu kriterijuma zadatih od strane korisnika.

WHERE - definiše predikat na osnovu koga se ograničava broj vrsta u rezultujućoj tabeli. Ova klauzula iz rezultata eliminiše sve vrste za koje specificirani predikat ne vraća vrednost TRUE.

GROUP BY - grupiše vrste koje u određenim kolonama imaju identične vrednosti.

HAVING - definiše predikat na osnovu koga se eliminišu vrste nakon što je klauzula GROUP BY primenjena na rezultujuću tabelu.

ORDER BY - koristi se za sortiranje rezultujuće tabele. Korisnici specificiraju kolone po kojima se vrši sortiranje kao i smer sortiranja.



Primer

Co-funded by the European Union

SELECT * **FROM** RADNIK;

Rezultat bi bio ekvivalentan da smo napisali upit kod koga su umesto * navedena imena svih kolona u tabeli.

SELECT MatBr, Ime, Sslovo, Prezime, DatRodj, Adresa, Pol, Plata, Rukovodilac, Sektor **FROM** RADNIK;

```
1 SELECT * FROM RADNIK;
2 SELECT MatBr, ime, Sslovo, Prezime, DatRodj, Adresa, Pol, Plata, Rukovodilac, Sektor FROM RADNIK;
```

MatBr	Ime	SSlovo	Prezime	DatRodj	Pol	Plata	Adresa	Rukovodilac	Sektor
123456789	Đorđe	B	Savić	09/01/1955	M	3000	NULL	333445555	5
3334455555	Voja	T	Vasić	08/12/1945	M	6000	NULL	888665555	5
999887777	Ana	J	Petrović	19/07/1958	Ž	3500	NULL	987654321	4
987654321	Jelena	S	Marković	20/06/1965	Ž	5300	NULL	888665555	4
666884444	Ratko	K	Veljković	15/09/1952	M	3800	NULL	333445555	5
456456456	Milica	A	Stojić	31/07/1962	Ž	2300	NULL	3334455555	5
987987987	Jovan	V	Janković	29/03/1959	M	2500	NULL	987654321	4
888665555	Ivan	E	Perić	09/01/1955	M	5500	NULL	NULL	5

Primer

Co-funded by the European Union

Ukoliko želimo da eliminišemo duplikate koristićemo ključnu reč DISTINCT.

SELECT DISTINCT Rukovodilac **FROM** RADNIK;

SELECT Rukovodilac
FROM RADNIK;

```
1 SELECT Rukovodilac FROM RADNIK;
2 SELECT DISTINCT Rukovodilac FROM RADNIK;
```

Rukovodilac
333445555
888665555
987654321
3334455555

WHERE klauzula

Co-funded by the European Union

Klauzula WHERE specificira uslov na osnovu koga se kreira rezultujuća tabela. U rezultujuću tabelu će biti uključene samo one vrste koje zadovoljavaju specificirani uslov.

U uslovu se mogu javiti:

- Relacioni operatori
- Logički operatori
- Operator **BETWEEN**
- Operator **IN**
- Operator **LIKE**
- Operator **IS NULL**

Relacioni operatori

Co-funded by the European Union

SQL podržava šest relacionih operatora.

1.	**=**	**Jednako**
2.	<>	Nije jednako (različito)
3.	<	Manje od
4.	>	Veće od
5.	<=	Manje ili jednako od
6.	>=	Veće ili jednako od

Primer

Co-funded by the European Union

U ovom primeru dat je SQL upit koji prikazuje podatke o radnicima koji se prezivaju Petrović.

```
SELECT *
FROM RADNIK
WHERE Prezime = "Petrović";
```

```
1 SELECT *
2 FROM RADNIK
3 WHERE Prezime = "Petrović";
```

MatBr	Ime	SSlovo	Prezime	DatRodj	Pol	Plata	Adresa	Rukovodilac	Sektor
999887777	Ana	J	Petrović	19/07/1958	Ž	3500	NULL	987654321	4

Primer

Co-funded by the European Union

Primer sadrži SQL upit koji prikazuje imena i prezimena radnika čija je plata jednaka ili veća od 3000.

```
SELECT Ime, Prezime
FROM RADNIK
WHERE PLATA >= 3000;
```

```
1 SELECT Ime, Prezime
2 FROM RADNIK
3 WHERE PLATA >= 3000;
```

Ime	Prezime
Đorđe	Savić
Voja	Vasić
Ana	Petrović
Jelena	Marković
Ratko	Veljković
Ivan	Perić

Logički operatori

Co-funded by the European Union

SQL omogućava korišćenje standardnih logičkih operatora **AND**, **OR** i **NOT**, ali i operatore **IN** i **BETWEEN** koji omogućavaju jednostavnije korišćenje prethodno navedenih operatora u nekim slučajevima.

Prioritet logičkih operatora je sledeći: **NOT** **AND** **OR**

AND sintaksa

```
SELECT kolona1, kolona2, ...
FROM ime tabele
WHERE uslov1 AND uslov2 AND uslov3 ...;
```

OR sintaksa

```
SELECT kolona1, kolona2, ...
FROM ime tabele
WHERE uslov1 OR uslov2 OR uslov3 ...;
```

Logički operatori

Co-funded by the European Union

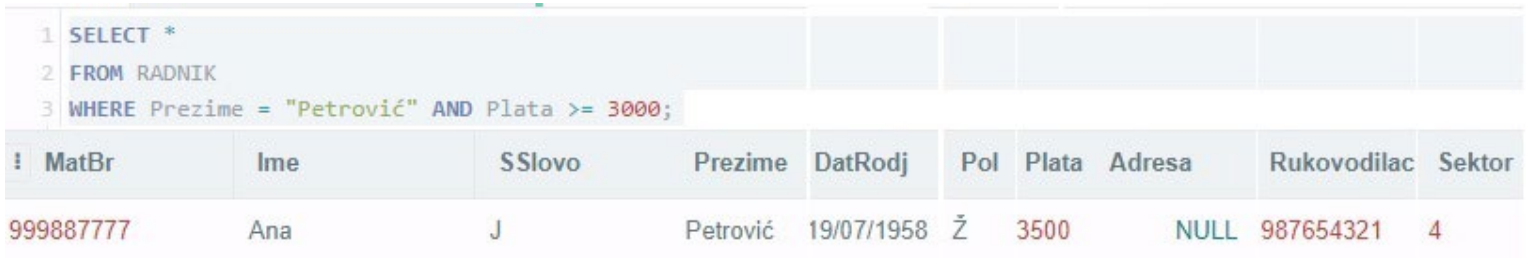
NOT sintaksa

```
SELECT kolona1, kolona2, ...
FROM ime tabele
WHERE NOT uslov;
```

Primer

Ukoliko želimo da prikažemo podatke o radnicima koji se prezivaju Petrović i čija je plata jednaka ili veća od 3000 možemo koristiti sledeći SQL upit:

```
SELECT *
FROM RADNIK
WHERE Prezime = "Petrović" AND Plata >= 3000;
```



```
1 SELECT *
2 FROM RADNIK
3 WHERE Prezime = "Petrović" AND Plata >= 3000;
```

MatBr	Ime	SSlovo	Prezime	DatRodj	Pol	Plata	Adresa	Rukovodilac	Sektor
999887777	Ana	J	Petrović	19/07/1958	Ž	3500	NULL	987654321	4

IN

Co-funded by the European Union

IN operator omogućava da se navedu više vrednosti u klauzuli WHERE.

IN operator je skraćenica za više uslova OR.

```
SELECT kolona1, kolona2, ...
FROM ime tabele
WHERE kolonaN IN (vrednost1, vrednost2, ...);
```

ili

```
SELECT kolona1, kolona2, ...
FROM ime tabele
WHERE kolonaN IN (SELECT iskaz);
```

Primer

Co-funded by the European Union

Korišćenjem operatora **IN**, **NOT IN** izdvajaju se samo radnici koji se prezivaju Petrović ili Janković.

```
SELECT *
FROM RADNIK
WHERE Prezime IN ("Petrović", "Janković");
```

```
1 SELECT *
2 FROM RADNIK
3 WHERE Prezime IN ("Petrović","Janković");
```

MatBr	Ime	SSlovo	Prezime	DatRodj	Pol	Plata	Adresa	Rukovodilac	Sektor
999887777	Ana	J	Petrović	19/07/1958	Ž	3500	NULL	987654321	4
987987987	Jovan	V	Janković	29/03/1959	M	2500	NULL	987654321	4

BETWEEN

Co-funded by the European Union

Operator BETWEEN bira vrednosti unutar datog opsega. Vrednosti mogu biti brojevi, tekst ili datumi.

Operator BETWEEN uključuje početne i krajnje vrednosti.

```
SELECT kolona1, kolona2, ...
FROM ime tabele
WHERE kolonaN BETWEEN vrednost1 AND vrednost2;
```

Primer

Za prikazivanje podataka o radnicima čija je plata u opsegu od 3000 do 4000 (uključujući i granice opsega) možemo iskoristiti operator **BETWEEN**.

```
SELECT *
FROM RADNIK
WHERE Plata BETWEEN
3000 AND 4000;
```

```
7 SELECT Ime, Prezime
8 FROM RADNIK
9 WHERE Plata BETWEEN 3000 AND 4000;
```

Ime	Prezime
Đorđe	Savić
Ana	Petrović
Ratko	Veljković

LIKE

Co-funded by the European Union

Naredba LIKE koristi se u WHERE klauzuli radi pretraživanja specificiranog šablona u koloni.

```
SELECT kolona1, kolona2, ...
FROM ime tabele
WHERE  kolonaN LIKE šablon;
```

Operator **LIKE(NOT LIKE)** omogućava poređenje vrednosti kolone sa zadatim šablonom.

Kod definisanja šablona koristi se procenat (%) i znak pitanja (?).

Procenat (%) predstavlja bilo koji mogući znak (broj, slovo, interpunkcijski znak) ili skup znakova.

Znak pitanja (?) zamenjuje samo jedan znak.

Primer

Co-funded by the European Union

Za nalaženje svih radnika koji imaju "J" na početku prezimena, može se koristiti šablon "J%".

```
SELECT Ime, Prezime
FROM RADNIK
WHERE Prezime LIKE "J%";
```

```
10 SELECT Ime, Prezime
11 FROM RADNIK
12 WHERE Prezime LIKE 'J%';
13
```

Ime	Prezime
Jovan	Janković

IS NULL

Co-funded by the European Union

Operator **IS NULL** koristi se za poređenje sa **NULL** vrednostima.

Polje sa **NULL** vrednošću je polje koje nema vrednost.

Treba voditi računa da se na **NULL** vrednosti ne može primeniti ni jedan relacioni operator.

Može se samo proveravati da li kolona ima **NULL** vrednost ili nema.

```
SELECT kolona1, kolona2, ...
FROM ime tabele
WHERE  kolonaN IS NULL;
```

Primer

Co-funded by the European Union

SQL upit koji izdvaja podatke o svim radnicima kojima nije definisan datum rođenja odnosno datum rođenja ima **NULL** vrednost.

```
SELECT Ime, Prezime
FROM RADNIK
WHERE DatRodj IS NULL;
```

```
16 SELECT Ime, Prezime
17 FROM RADNIK
18 WHERE DatRodj IS NULL;
```

Ime	Prezime
Miljan	Jeremić

ORDER BY

Co-funded by the European Union

Klauzula **ORDER BY** specificira redosled prikazivanja vrste rezultujuće tabele, sortiranjem po vrednosti nekih kolona u rastući redosled (**ASC**) (predefinisana vrednost) ili opadajući redosled (**DESC**).

Ukoliko klauzula **ORDER BY** nije navedena vrste u rezultujućoj tabeli su poređane po slučajnom principu i ne postoji nikakva garancija da će isti upit uvek generisati rezultujuću tabelu čije su vrste poređane na isti način.

```
SELECT kolona1, kolona2, ...
FROM ime tabele
ORDER BY kolona1, kolona2, ... ASC|DESC;
```

Primer

Co-funded by the European Union

SQL upit koji prikazuje podatke o radnicima i sortira ih prema prezimenu u rastućem redosledu.

```
SELECT *
FROM RADNIK
ORDER BY Prezime;
```

```
1 SELECT *
2 FROM RADNIK
3 ORDER BY Prezime;
```

MatBr	Ime	SSlovo	Prezime	DatRodj	Pol	Plata	Adresa	Rukovodilac	Sektor
987987987	Jovan	V	Janković	29/03/1959	M	2500	NULL	987654321	4
123456789	Miljan	G	Jeremić	NULL	M	3000	Kaplarova 37	333445555	5
987654321	Jelena	S	Marković	20/06/1965	Ž	5300	NULL	888665555	4
888665555	Ivan	E	Perić	09/01/1955	M	5500	NULL	NULL	5
999887777	Ana	J	Petrović	19/07/1958	Ž	3500	NULL	987654321	4
123456789	Đorđe	B	Savić	09/01/1955	M	3000	NULL	333445555	5
456456456	Milica	A	Stojić	31/07/1962	Ž	2300	NULL	3334455555	5
3334455555	Voja	T	Vasić	08/12/1945	M	6000	NULL	888665555	5
666884444	Ratko	K	Veljković	15/09/1952	M	3800	NULL	333445555	5

Aritmetički operatori

Co-funded by the European Union

SQL dozvoljava korišćenje aritmetičkih operatora u **SELECT** i **WHERE** klauzulama. Na taj način se kao rezultat pretraživanja mogu prikazati rezultati izračunavanja nekog matematičkog izraza.

Aritmetički operatori izvršavaju određene akcije sa operandima brojčanog tipa.

Operator	Svrha	Primer
+, -	Definiše predznak broja (unarna operacija)	SELECT * FROM cena WHERE min_cena < -10;
+, -	Sabiranje i oduzimanje brojeva (binarni operator)	SELECT iznos+1000 FROM prodaja;
*, /	Množenje i deljenje brojeva (binarni operator)	SELECT iznos/25.5 FROM prodaja;
MOD, %	Moduo - daje ostatak pri deljenju (binarni operator)	SELECT MOD(iznos,100) FROM prodaja;

Alijasi kolone

Co-funded by the European Union

Alijas je način preimenovanja imena kolona na izlazu.

Bez alijasa, kolone prikazane kao rezultat SQL rečenice imaće ista imena kao kolone u tabeli ili ime koje pokazuje aritmetičku operaciju.

Nekada je poželjno da na izlazu bude prikazano ime kolone koje je lakše razumeti, ime koje je više „user friendly“. Alijasi kolona dozvoljavaju preimenovanje kolone na izlazu.

Postoji nekoliko pravila kod korišćenja alijasa kolona.

Alijas kolone:

- menja naslov kolone,
- koristan je kod proračuna,
- navodi se odmah posle imena kolone,
- može imati opcionu ključnu reč AS između imena kolone i alijasa,
- zahteva znake navoda ako alijas sadrži blanko znake ili je osetljiv na velika i mala slova.

Alijas kolone

Co-funded by the European Union

Alijas kolone sintaksa

SELECT *kolona* AS *alijas kolone*
FROM *ime tabele*

Primer

SQL upit koji prikazuje imena i prezimena radnika kao i njihove plate uvećane za bonus od 5000.

SELECT Ime, Prezime, Plata + 5000 **AS** PlataSaBonusom
FROM RADNIK;

```
10 SELECT Ime, Prezime, Plata + 5000 AS PlataSaBonusom
11 FROM RADNIK;
```

Ime	Prezime	PlataSaBonusom
Đorđe	Savić	8000
Voja	Vasić	11000
Ana	Petrović	8500
Jelena	Marković	10300
Ratko	Veljković	8800
Milica	Stojić	7300
Jovan	Janković	7500
Ivan	Perić	10500
Miljan	Jeremić	8000

Funkcije agregacije

Co-funded by the European Union

Funkcije agregacije su dobile naziv po tome što vrše agregaciju rezultata upita.

Korišćenje ovih funkcija je jednostavno, pošto se navode u listi kolona **SELECT** klauzule koje se prikazuju, tako što im se kao argument u zagradama navodi atribut ili lista atributa nad kojima želimo da operišu.

Ako se u **SELECT** klauzuli pojavi funkcija agregacije onda se u njoj ne sme pojaviti ništa osim još neke druge funkcije agregacije.

Funkcije agregacije se ne smeju pojaviti u **WHERE** klauzuli direktno (već jedino u nekom podupitu).

Funkcije agregacije

Co-funded by the European Union

AVG(kolona) - izračunava srednju vrednost datog atributa

SUM(kolona) - izračunava sumu svih vrednosti atributa

MIN(kolona) - nalazi minimalnu vrednost atributa

MAX(kolona) - nalazi najveću vrednost atributa

COUNT(*) - nalazi broj vrsta u tabeli (grupi)

COUNT(kolona) - nalazi broj broj vrsta sa ne **NULL** vrednostima kolone

COUNT(DISTINCT kolona) - nalazi broj vrsta sa različitim vrednostima zadate kolone (nije podržano od strane MS Access-a)

STDDEV(kolona) - izračunava standardnu devijaciju

VARIANCE(kolona) - izračunava varijansu po zadatom atributu

Funkcije **STDDEV**, **VARIANCE**, **SUM** i **AVG** dopuštaju kao argumente samo atribute koji su numeričkog tipa.

Count

Co-funded by the European Union

Funkcija COUNT određuje broj vrsta u rezultujućoj tabeli.

```
SELECT COUNT(*) AS PlataSaBonusom
FROM RADNIK;
```

```
1 SELECT COUNT(*) AS PlataSaBonusom
2 FROM RADNIK;
```

PlataSaBonusom
9