Laboratorijska vježba 4 - Monte Carlo simulacija

Zadatak – Izračunavanje vrijednosti broja Pi

Na osnovu opisa postupka Monte Karlo simulacije (str. 34 u skripti), napisati program u proizvoljnom programskom jeziku za izračunavanje broja Pi. Omogućiti da korisnik na početku može da izabere jednu od dvije opcije – broj slučajno generisanih vrijednosti ili broj decimala koje treba da se poklope sa pravom vrijednošću.

In [28]:
```
# Potrebne biblioteke
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import random # za generisanje slučajnih vrijednosti
from decimal import * # za definisanje preciznosti (broja decimala koje se trebaju poklopiti)
```

Definišimo klasu Monte_Carlo koja sadrži potrebne metode za Monte Carlo simulaciju:

In [29]:
```
class Monte_Carlo:
    # Atributi klase
    x=[] # niz vrijednosti prve slučajne promjenljive koja prati uniformnu raspodjelu između 0 i 1
    y=[] # niz vrijednosti druge slučajne promjenljive koja prati uniformnu raspodjelu između 0 i 1
    inside_circle = 0 # broj tačaka unutar kružnice

    # Metode klase
    def monte_carlo(self,N):
        """
        monte_carlo metoda racuna i vraca vrijednost broja PI

        :param self: objekat klase Monte_Carlo
        :param N: broj slucajno generisanih vrijednosti
        :return PI: izracunata vrijednost broja PI
        """
        for i in range(0,N):
            x_i=random.uniform(0.0, 1.00000) # uniformna raspodjela između 0 i 1
            y_i=random.uniform(0.0, 1.00000) # uniformna raspodjela između 0 i 1
            self.x.append(x_i)
            self.y.append(y_i)
            if (x_i**2+y_i**2)<=1: # provjera da li je tačka (x_i,y_i) unutar jedinične kružnice
                self.inside_circle+=1
        PI = (self.inside_circle/N)*4 # formula za izračunavanje vrijednosti PI
        return PI

    def get_PI_generated_values(self):
        """
        get_PI_generated_values omogucava unos broja
        slucajno generisanih vrijednosti od strane korisnika,
        poziva monte_carlo metodu za izracunavanje vrijednosti broja PI
        i ispisuje raspored slucajno generisanih vrijednosti

        :param self: objekat klase Monte_Carlo
        """
        # Unos broja slučajno generisanih vrijednosti od strane korisnika
        num_of_generated_values = int(input("UNESITE BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI, A ZATIM PRITISNITE
        # Poziv monte_carlo metode
        PI = self.monte_carlo(num_of_generated_values)
        # Iscrtavanje jedinične kružnice i slučajno generisanih vrijednosti
        figure, axes = plt.subplots()
        x_fill = np.arange(0.0, 1, 0.01)
        axes.fill_between(x_fill, y1=0, y2=1) # sivančeno kvadratna površine 1 (četvrtina kružnice)
        plt.scatter(self.x, self.y, c='red') # skiciranje slučajno generisanih vrijednosti
        axes.set_aspect(1)
        axes.add_artist(plt.Circle((0,0),1, color = 'k', fill = False)) # jedinična kružnica
        plt.title("Raspored generisanih vrijednosti")
        plt.xlim(left=-1)
        plt.xlim(right=1)
        plt.ylim(bottom=-1)
        plt.ylim(top=1)
        print("================================================================================")
        print("        REZULTAT MONTE CARLO SIMULACIJE - ZA BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI     ")
        print("================================================================================")
        print('Broj slučajno generisanih vrijednosti = '+str(num_of_generated_values))
        print('Broj tačaka unutar kružnice = '+str(self.inside_circle))
        print('Broj tačaka izvan kružnice = '+str(num_of_generated_values - self.inside_circle))
        print('PI = '+str(PI))
        plt.show()

    def get_PI_decimal_values(self):
        """
        get_PI_decimal_values omogucava unos broja decimala od strane korisnika
        i taj broj se postavlja kao potrebna preciznost,
        zatim se u while petlji poziva metoda monte_carlo dok izračunata i stvarna
        vrijednost broja PI ne ne budu imale unijeti broj preklopljenih decimala

        :param self: objekat klase Monte_Carlo
        """
        # Unos broja decimala koje treba da se poklope sa stvarnom vrijednošću
        num_of_decimal_values = int(input("UNESITE BROJ DECIMALNIH VRIJEDNOSTI, A ZATIM PRITISNITE ENTER: \n"))
        num_exp = 10**num_of_decimal_values # za dobijanje broja slučajno generisanih vrijednosti
        #num_of_generated = num_exp**2
        accuracy = 1 / num_exp # dozvoljena greška
        while True:
            PI = self.monte_carlo(10000*num_exp)
            getcontext().prec = num_of_decimal_values
            diff = abs(Decimal(math.pi)-Decimal(PI))
            diff_str = str(diff)
            if (diff <= accuracy and diff_str[num_of_decimal_values+1] == '0'):
                print("================================================================================")
                print("            REZULTAT MONTE CARLO SIMULACIJE - ZA BROJ DECIMALNIH VRIJEDNOSTI           ")
                print("================================================================================")
                print('Izračunata vrijednost broja PI = '+str(PI))
                print('Stvarna vrijednost broja PI = '+str(math.pi))
                break
            else:
                continue
```

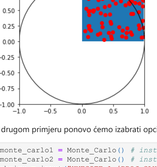
MONTE CARLO SIMULACIJA - ZA BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI

Sljedećim odlomkom koda korisniku se nudi *meni* unosa - ukoliko korisnik sa tastature unese **A** pozvaće se metoda *get_PI_generated_values()* nad objektom klase *Monte_Carlo*, a ukoliko unese **B** pozvaće se *get_PI_decimal_values()* nad objektom klase *Monte_Carlo*.

U prvom primjeru izabraćemo opciju **A**, a zatim unijeti **100** kao broj slučajno generisanih vrijednosti.

In [30]:
```
monte_carlo1 = Monte_Carlo() # instanciranje objekta 1 klase Monte Carlo
monte_carlo2 = Monte_Carlo() # instanciranje objekta 2 klase Monte Carlo
choice = input("UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER
if choice == 'A':
    monte_carlo1.get_PI_generated_values()
elif choice == 'B':
    monte_carlo2.get_PI_decimal_values()
else:
    print("Neispravan unos!")
```

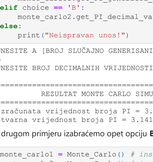
```
UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER:
A
UNESITE BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI, A ZATIM PRITISNITE ENTER:
100
================================================================================
        REZULTAT MONTE CARLO SIMULACIJE - ZA BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI
================================================================================
Broj slučajno generisanih vrijednosti = 100
Broj tačaka unutar kružnice = 71
Broj tačaka izvan kružnice = 29
PI = 2.84
```



U drugom primjeru ponovo ćemo izabrati opciju **A**, a zatim unijeti **100000** kao broj slučajno generisanih vrijednosti.

In [31]:
```
monte_carlo1 = Monte_Carlo() # instanciranje objekta 1 klase Monte Carlo
monte_carlo2 = Monte_Carlo() # instanciranje objekta 2 klase Monte Carlo
choice = input("UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER
if choice == 'A':
    monte_carlo1.get_PI_generated_values()
elif choice == 'B':
    monte_carlo2.get_PI_decimal_values()
else:
    print("Neispravan unos!")
```

```
UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER:
A
UNESITE BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI, A ZATIM PRITISNITE ENTER:
100000
================================================================================
        REZULTAT MONTE CARLO SIMULACIJE - ZA BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI
================================================================================
Broj slučajno generisanih vrijednosti = 100000
Broj tačaka unutar kružnice = 78520
Broj tačaka izvan kružnice = 21480
PI = 3.1408
```



Zaključak: Na osnovu prethodna dva primjera poziva metode *get_PI_generated_values()* zaključujemo da povećavanjem broja slučajno generisanih vrijednosti povećavamo vjerovatnoću da izračunata vrijednost broja PI bude što bliža stvarnoj vrijednosti broja PI i to je očekivano ponašanje. Takođe, možemo primijetiti da je u drugom slučaju označeni jedinični kvadrat i potpunosti *crvene boje*, jer je "preplavljen" izgenerisanim vrijednostima usljed velikog broja vrijednosti *(100000)*.

MONTE CARLO SIMULACIJA - ZA BROJ DECIMALNIH VRIJEDNOSTI

U prvom primjeru izabraćemo opciju **B**, a zatim unijeti **1** kao broj decimala koje se trebaju poklopiti sa stvarnim brojem decimala broja PI.

In [34]:
```
monte_carlo1 = Monte_Carlo() # instanciranje objekta 1 klase Monte Carlo
monte_carlo2 = Monte_Carlo() # instanciranje objekta 2 klase Monte Carlo
choice = input("UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER
if choice == 'A':
    monte_carlo1.get_PI_generated_values()
elif choice == 'B':
    monte_carlo2.get_PI_decimal_values()
else:
    print("Neispravan unos!")
```

```
UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER:
B
UNESITE BROJ DECIMALNIH VRIJEDNOSTI, A ZATIM PRITISNITE ENTER:
1
================================================================================
            REZULTAT MONTE CARLO SIMULACIJE - ZA BROJ DECIMALNIH VRIJEDNOSTI
================================================================================
Izračunata vrijednost broja PI = 3.147
Stvarna vrijednost broja PI = 3.141592653589793
```

U drugom primjeru izabraćemo opet opciju **B**, a zatim unijeti **2** kao broj decimala koje se trebaju poklopiti sa stvarnim decimalama broja PI.

In [35]:
```
monte_carlo1 = Monte_Carlo() # instanciranje objekta 1 klase Monte Carlo
monte_carlo2 = Monte_Carlo() # instanciranje objekta 2 klase Monte Carlo
choice = input("UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER
if choice == 'A':
    monte_carlo1.get_PI_generated_values()
elif choice == 'B':
    monte_carlo2.get_PI_decimal_values()
else:
    print("Neispravan unos!")
```

```
UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER:
B
UNESITE BROJ DECIMALNIH VRIJEDNOSTI, A ZATIM PRITISNITE ENTER:
2
================================================================================
            REZULTAT MONTE CARLO SIMULACIJE - ZA BROJ DECIMALNIH VRIJEDNOSTI
================================================================================
Izračunata vrijednost broja PI = 3.141948
Stvarna vrijednost broja PI = 3.141592653589793
```

I u posljednjem slučaju unijećemo **3** kao broj decimala koje se trebaju poklopiti sa stvarnim decimalama broja PI.

In [36]:
```
monte_carlo1 = Monte_Carlo() # instanciranje objekta 1 klase Monte Carlo
monte_carlo2 = Monte_Carlo() # instanciranje objekta 2 klase Monte Carlo
choice = input("UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER
if choice == 'A':
    monte_carlo1.get_PI_generated_values()
elif choice == 'B':
    monte_carlo2.get_PI_decimal_values()
else:
    print("Neispravan unos!")
```

```
UNESITE A [BROJ SLUČAJNO GENERISANIH VRIJEDNOSTI] ILI B [BROJ DECIMALA]. ZATIM PRITISNITE ENTER:
B
UNESITE BROJ DECIMALNIH VRIJEDNOSTI, A ZATIM PRITISNITE ENTER:
3
================================================================================
            REZULTAT MONTE CARLO SIMULACIJE - ZA BROJ DECIMALNIH VRIJEDNOSTI
================================================================================
Izračunata vrijednost broja PI = 3.141396
Stvarna vrijednost broja PI = 3.141592653589793
```

Zaključak: Na osnovu prethodna 3 primjera gdje se izračunavanje vrijednosti broja PI vršilo pozivom metode *get_PI_decimal_values()*, a gdje je broj slučajno generisanih vrijednosti zavisio od unesenog broja decimala (kako je implementirano u samoj metodi) možemo zaključiti da smo postigli željene rezultate. U sva 3 slučaja broj preklopljenih decimala izračunate i stvarne vrijednosti broja PI je jednak unesenom broju.

Napomena: U slučaju izračunavanja broja PI korištenjem Monte Carlo simulacije za slučaj željenog broja preklopljenih decimala treba biti racionalan sa brojem decimala, jer je za izračunavanje broja PI na 3 tačne decimale potrebno 10 000 000 slučajno generisanih vrijednosti, te u tom slučaju izračunavanje traje izvjesno vrijeme.