# Set7B

December 25, 2023

[ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

[ ]:
```python
def Pi(N):
    """estimate pi number

    Args:
        N (int): number of montecarlo steps

    Returns:
        float: pi number
    """
    s = 0
    for _ in range(N):
        x, y = np.random.random(2)
        if x**2 + y**2 < 1 :
            s +=1
    return (4*s)/N
```

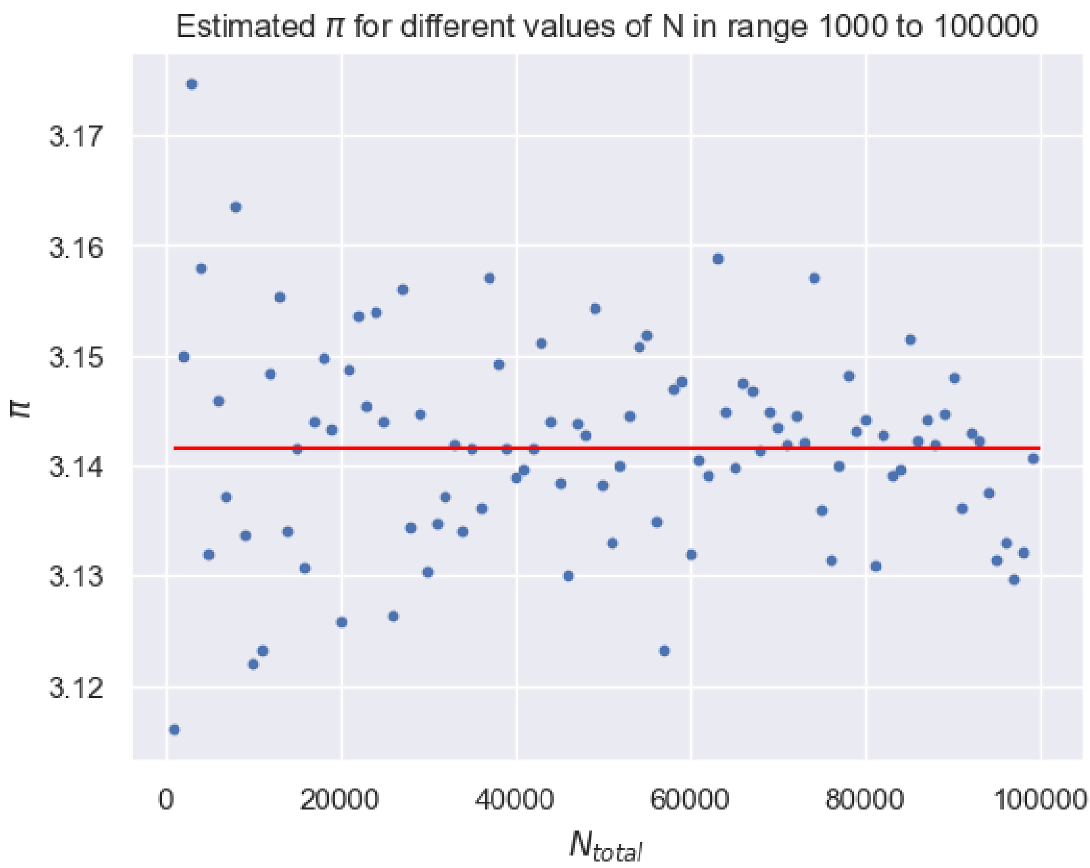
[ ]:
```python
N_min = 1000
N_max = 100000
step = 1000

N_total_array = range(N_min, N_max, step)
```

[ ]:
```python
P = []
for n in range(N_min, N_max, step):
    P.append(Pi(n))
```

[ ]:
```python
plt.scatter(range(N_min, N_max, step), P, s=10)
plt.hlines(np.pi, xmin=N_min, xmax=N_max, color="red")
plt.title(rf'Estimated $\pi$ for different values of N in range {N_min} to␣
 ↪{N_max}')
plt.xlabel(r'$N_{total}$')
plt.ylabel(r'$\pi$')
```

[ ]: Text(0, 0.5, '$\\pi$')




```
[ ]: @njit
def p_v(v, bm=2):
    return np.exp(-(bm/2)*(v**2))
```

```
[ ]: @njit
def integral_calc(lower_bound, upper_bound, N, bm=2):
    v_x, v_y, v_z = (upper_bound - lower_bound)*np.random.random_sample((3, N))␣
 ↪+ lower_bound

    integral_v_x = ((upper_bound - lower_bound)/N)*np.sum(p_v(v_x))
    integral_v_y = ((upper_bound - lower_bound)/N)*np.sum(p_v(v_y))
    integral_v_z = ((upper_bound - lower_bound)/N)*np.sum((v_z**2)*p_v(v_z))

    integral = ((bm/(2*np.pi))**1.5) * integral_v_x * integral_v_y *␣
 ↪integral_v_z

    return integral
```

[ ]:
```
lower_bound = -1000
upper_bound = 1000
N = 1000000
bm = 2


numerical_integral = integral_calc(lower_bound, upper_bound, N)
```

[ ]:
```
analytical_integral = 1 / (2*np.pi**3) / (((bm/(2*np.pi))**1.5)**2)
```

[ ]:
```
print(f'Analytical integration result is: {analytical_integral}')
print(f'Numerical integration result is: {round(numerical_integral, 3)}')
```

```
Analytical integration result is: 0.5
Numerical integration result is: 0.484
```