

Actividad 3 Metodo Monte Carlo

Uriel G. Ernesto A. Oscar H.

8 de septiembre de 2022

Resumen

En esta actividad se realizará una simulación donde obtendremos el valor de Pi, dado que no es un valor exacto y este mismo sigue en discusión, buscaremos un resultado más parecido al que se nos mencionó.

1. Introducción

Los números imaginarios han tenido una gran influencia para el desarrollo y cálculo de diversos problemas, este nos ayudado a resolver problemas numéricos más complejos en ciencia, ingeniería, finanzas y estadísticas. En este caso lo usaremos para el cálculo de Pi, un número muy debatido, en casos prácticos lo usamos como "3.1416" pero en realidad es un numero más largo y complejo, para ello el uso del método Monte Carlo, en el cual se simulan distinta cantidad de casos para llegar a la respuesta más exacta .

2. Simulación Monte Carlo

La simulación Monte Carlo es básicamente un muestreo experimental cuyo propósito es estimar las distribuciones de las variables de salida que depende de variables probabilísticas de entrada. Los investigadores acuñaron este término por su similaridad al muestreo aleatorio en los juegos de ruleta en los casinos de Monte Carlo. Así, por ejemplo, el modelo de Monte Carlo puede simular los resultados que puede asumir el VAN de un proyecto. Pero lo más relevante es que la simulación permite experimentar para observar los resultados que va mostrando dicho VAN[1].

3. Números aleatorios

En el corazón de los Métodos de Monte-Carlo encontramos un generador de números aleatorios, es decir un procedimiento que produce un flujo infinito de variables aleatorias, que generalmente se encuentran en el intervalo $(0, 1)$; los cuales son independientes y están uniformemente distribuidos de acuerdo a una distribución de probabilidad. La mayoría de los lenguajes de programación hoy en día contienen un generador de números aleatorios por defecto al cual simplemente debemos ingresarle un valor inicial, generalmente llamado seed o semilla, y que luego en cada invocación nos va a devolver un secuencia uniforme de variables aleatorias independientes en el intervalo $(0, 1)$ [2].

4. Números pseudoaleatorios

El concepto de una secuencia infinita de variables aleatorias es una abstracción matemática que puede ser imposible de implementar en una computadora. En la práctica, lo mejor que se puede hacer es producir una secuencia de números pseudoaleatorios con propiedades estadísticas que son indistinguibles de las de una verdadera secuencia de variables aleatorias. Aunque actualmente métodos de generación física basados en la radiación de fondo o la mecánica cuántica parecen ofrecer una fuente estable de números verdaderamente aleatorios , la gran mayoría de

los generadores de números aleatorios que se utilizan hoy en día están basados en algoritmos simples que pueden ser fácilmente implementados en una computadora



Figura 1: Ejemplo del código de la simulación Monte Carlo

5. Código a ejecutar

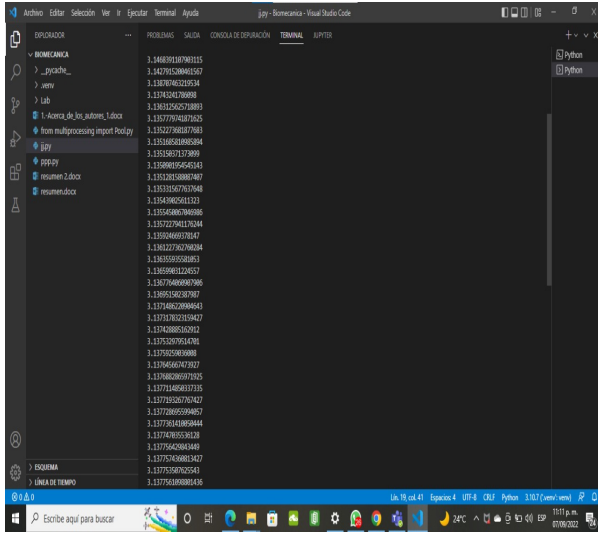
```
Archivo  Editar  Selección  Ver  Ir  Ejecutar  Terminal  Ayuda  jupyter - Biomechanica - Visual Studio Code

EXPLORADOR
  BIOMECHANICA
    > __pycache__
    > .venv
    > Lab
    1-Acerca_de_los_autores_1.docx
    from multiprocessing import Pool.py
    jupyter
    ppp.py
    resumen 2.docx
    resumen.docx

jupyter
1  from multiprocessing import Pool
2  from random import randint
3  import statistics
4  width = 10000
5  height = width
6  radio = width
7
8  npuntos = 0
9  ndentro = 0
10 radio2 = radio*radio
11 replicas = 100
12 promediopi = []
13
14
15 if __name__ == '__main__':
16     with Pool(4) as p:
17         for j in range(replicas):
18             for i in range(1,10000):
19                 x = randint(0,width)
20                 y = randint(0,width)
21                 npuntos += 1
22                 if x*x + y*y <= radio2:
23                     ndentro += 1
24                 pi = ndentro * 4 / npuntos
25                 promediopi.append(pi)
26                 print(statistics.mean(promediopi))
27
28     print(statistics.mean(promediopi))
29
30
```

Figura 2: Código de la simulación Monte Carlo

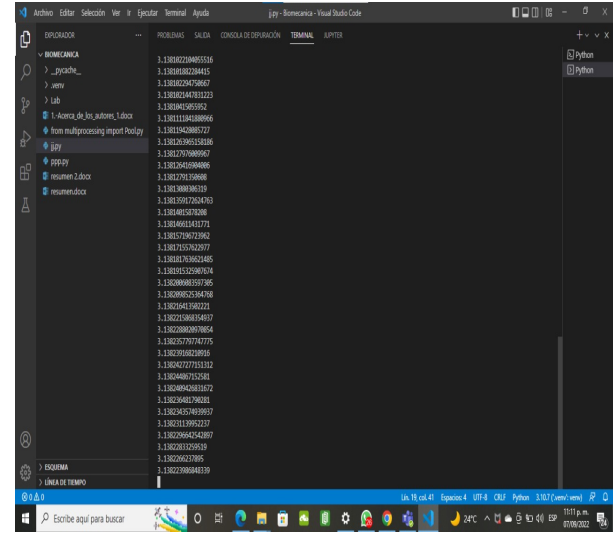
6. Resultados



Visual Studio Code interface showing the terminal output of a Python script. The file explorer on the left shows a project named 'BOMMECANCA' with files like '_pycache_', '.venv', 'Lab', '1-Acora_de los autores_1.docx', 'from multiprocessing import Pool.py', 'pp.py', 'resumen_1.docx', and 'resumen.docx'. The terminal displays a list of 40 numerical results, each on a new line, ranging from 3.348635287868115 to 3.337558765894339.

PROBLEMAS	SALIDA
1	3.348635287868115
2	3.342795238461557
3	3.342795238461557
4	3.338252625728893
5	3.338252625728893
6	3.335779174817525
7	3.335779174817525
8	3.332585238461557
9	3.332585238461557
10	3.3315567376948
11	3.3315567376948
12	3.3315567376948
13	3.3315567376948
14	3.3315567376948
15	3.3315567376948
16	3.3315567376948
17	3.3315567376948
18	3.3315567376948
19	3.3315567376948
20	3.3315567376948
21	3.3315567376948
22	3.3315567376948
23	3.3315567376948
24	3.3315567376948
25	3.3315567376948
26	3.3315567376948
27	3.3315567376948
28	3.3315567376948
29	3.3315567376948
30	3.3315567376948
31	3.3315567376948
32	3.3315567376948
33	3.3315567376948
34	3.3315567376948
35	3.3315567376948
36	3.3315567376948
37	3.3315567376948
38	3.3315567376948
39	3.3315567376948
40	3.3315567376948

(a) Primeros resultados

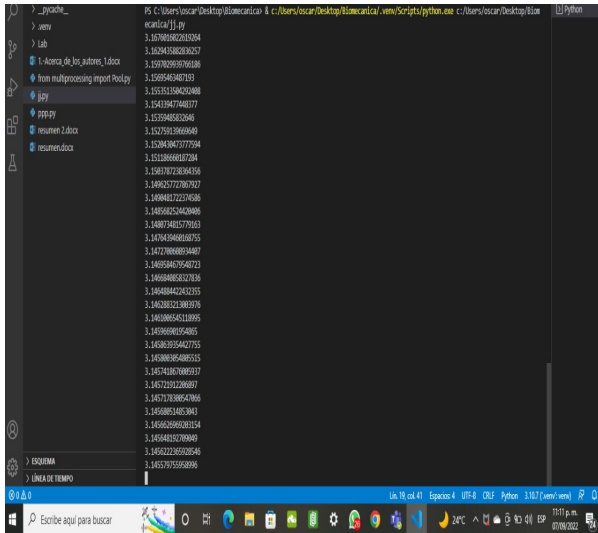


Visual Studio Code interface showing the terminal output of a Python script. The file explorer on the left shows a project named 'BOMMECANCA' with files like '_pycache_', '.venv', 'Lab', '1-Acora_de los autores_1.docx', 'from multiprocessing import Pool.py', 'pp.py', 'resumen_1.docx', and 'resumen.docx'. The terminal displays a list of 40 numerical results, each on a new line, ranging from 3.338222584615557 to 3.338222584615557.

PROBLEMAS	SALIDA
1	3.338222584615557
2	3.338222584615557
3	3.338222584615557
4	3.338222584615557
5	3.338222584615557
6	3.338222584615557
7	3.338222584615557
8	3.338222584615557
9	3.338222584615557
10	3.338222584615557
11	3.338222584615557
12	3.338222584615557
13	3.338222584615557
14	3.338222584615557
15	3.338222584615557
16	3.338222584615557
17	3.338222584615557
18	3.338222584615557
19	3.338222584615557
20	3.338222584615557
21	3.338222584615557
22	3.338222584615557
23	3.338222584615557
24	3.338222584615557
25	3.338222584615557
26	3.338222584615557
27	3.338222584615557
28	3.338222584615557
29	3.338222584615557
30	3.338222584615557
31	3.338222584615557
32	3.338222584615557
33	3.338222584615557
34	3.338222584615557
35	3.338222584615557
36	3.338222584615557
37	3.338222584615557
38	3.338222584615557
39	3.338222584615557
40	3.338222584615557

(b) Segunda lista resultados

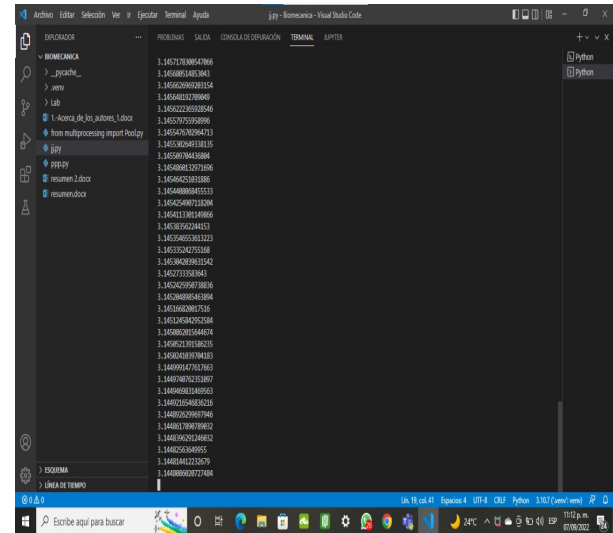
Figura 3: Lista de resultados



Visual Studio Code interface showing the terminal output of a Python script. The file explorer on the left shows a project named 'BOMMECANCA' with files like '_pycache_', '.venv', 'Lab', '1-Acora_de los autores_1.docx', 'from multiprocessing import Pool.py', 'pp.py', 'resumen_1.docx', and 'resumen.docx'. The terminal displays a list of 40 numerical results, each on a new line, ranging from 3.35138686157284 to 3.345757555894339.

PROBLEMAS	SALIDA
1	3.35138686157284
2	3.35138686157284
3	3.35138686157284
4	3.35138686157284
5	3.35138686157284
6	3.35138686157284
7	3.35138686157284
8	3.35138686157284
9	3.35138686157284
10	3.35138686157284
11	3.35138686157284
12	3.35138686157284
13	3.35138686157284
14	3.35138686157284
15	3.35138686157284
16	3.35138686157284
17	3.35138686157284
18	3.35138686157284
19	3.35138686157284
20	3.35138686157284
21	3.35138686157284
22	3.35138686157284
23	3.35138686157284
24	3.35138686157284
25	3.35138686157284
26	3.35138686157284
27	3.35138686157284
28	3.35138686157284
29	3.35138686157284
30	3.35138686157284
31	3.35138686157284
32	3.35138686157284
33	3.35138686157284
34	3.35138686157284
35	3.35138686157284
36	3.35138686157284
37	3.35138686157284
38	3.35138686157284
39	3.35138686157284
40	3.35138686157284

(a) Tercera lista resultados



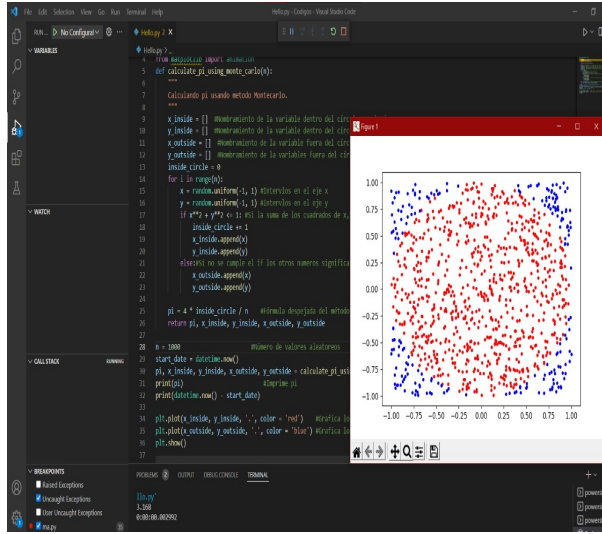
Visual Studio Code interface showing the terminal output of a Python script. The file explorer on the left shows a project named 'BOMMECANCA' with files like '_pycache_', '.venv', 'Lab', '1-Acora_de los autores_1.docx', 'from multiprocessing import Pool.py', 'pp.py', 'resumen_1.docx', and 'resumen.docx'. The terminal displays a list of 40 numerical results, each on a new line, ranging from 3.345757555894339 to 3.345757555894339.

PROBLEMAS	SALIDA
1	3.345757555894339
2	3.345757555894339
3	3.345757555894339
4	3.345757555894339
5	3.345757555894339
6	3.345757555894339
7	3.345757555894339
8	3.345757555894339
9	3.345757555894339
10	3.345757555894339
11	3.345757555894339
12	3.345757555894339
13	3.345757555894339
14	3.345757555894339
15	3.345757555894339
16	3.345757555894339
17	3.345757555894339
18	3.345757555894339
19	3.345757555894339
20	3.345757555894339
21	3.345757555894339
22	3.345757555894339
23	3.345757555894339
24	3.345757555894339
25	3.345757555894339
26	3.345757555894339
27	3.345757555894339
28	3.345757555894339
29	3.345757555894339
30	3.345757555894339
31	3.345757555894339
32	3.345757555894339
33	3.345757555894339
34	3.345757555894339
35	3.345757555894339
36	3.345757555894339
37	3.345757555894339
38	3.345757555894339
39	3.345757555894339
40	3.345757555894339

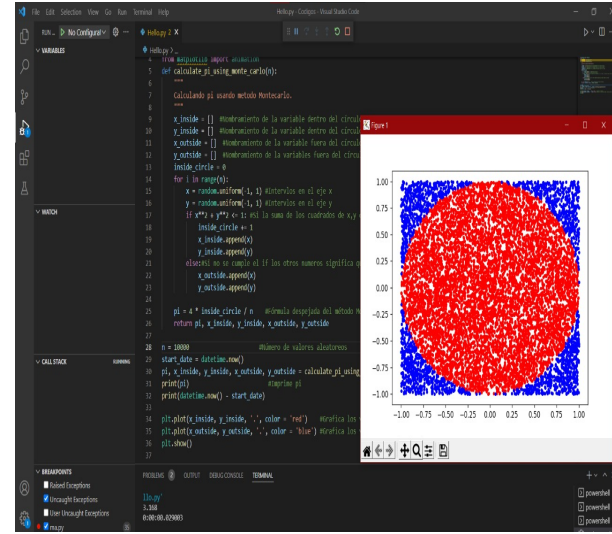
(b) Cuarta lista resultados

Figura 4: Lista de resultados

7. Graficas

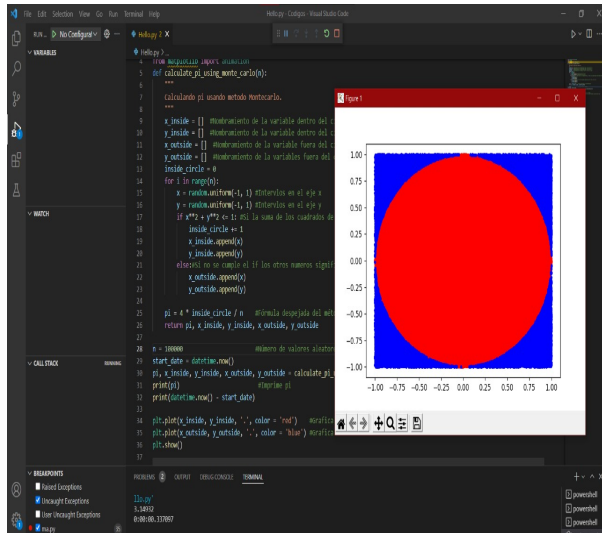


(a) Resultado con 1,000 números aleatorios

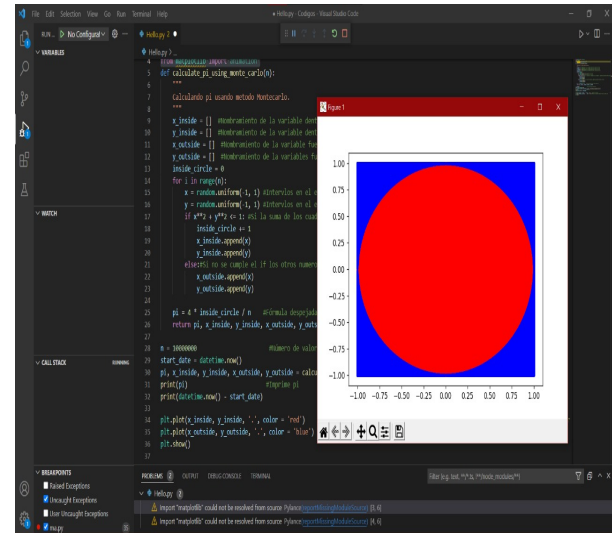


(b) Resultado con 10,000 números aleatorios

Figura 5: Graficas resultantes



(a) Resultado con 100,000 números aleatorios



(b) Resultado con 1,000,000 números aleatorios

Figura 6: Graficas resultantes

8. Conclusiones

Al realizar esta actividad nos dimos cuenta de lo importante que pueden llegar a ser la generación de números aleatorios, en este caso lo vimos con el método Monte Carlo, que entre más números generara más exacta fue nuestra respuesta, con esto lo podemos llegar aplicar en algún caso práctico de la vida diaria.

Referencias

- [1] Danysoft. Cómo instalar python, bibliotecas pandas y matplotlib, power bi, spyder y jupyter notebooks. <https://www.youtube.com/watch?v=JbsiKRDnIns&t=744s>, 2021.
- [2] R. E. Lopez. Introducción a los métodos de monte-carlo con python., enero 2017.