

O cálculo aproximado do PI pelo método de Monte Carlo é feito simulando pontos aleatórios em um quadrado de área 1 unidade e um círculo com o diâmetro de 1 unidade. Após isso é verificado a quantidade de pontos que estão dentro do círculo e utilizada a fórmula $4 \times (\text{pontos dentro do círculo} / \text{total de pontos})$ para aproximar o valor de PI.

O primeiro passo para realizar esse método foi importar as bibliotecas math, random, time e o método pool da biblioteca multiprocessing, como pode ser visto na imagem a seguir.

```
import math
import random
import time
from multiprocessing.pool import Pool
```

Para realizar a simulação dos pontos foi criada a função worker, ela recebe uma quantidade de pontos para gerar aleatoriamente através da função random.uniform que gera valores aleatórios dentro de um intervalo. Após isso é verificado quantos pontos destes gerados estão dentro do círculo, usando a função math.hypot, que faz a soma dos quadrados dos parâmetros, verificando se o valor é menor que 1, ou seja, está dentro do círculo. Por fim, essa função retorna a quantidade de pontos que estavam dentro do círculo.

```
def worker(pontos):
    ponto_circ = 0
    for _ in range(pontos):
        x, y = random.uniform(-1, 1), random.uniform(-1, 1)
        if math.hypot(x, y) <= 1:
            ponto_circ += 1
    return ponto_circ
```

Para realizar esse cálculo de forma paralela foi criada a função calculo_pi, que vai receber o número total de pontos a serem testados e vai dividir os número de pontos igualmente e criar vários processos com a mesma quantidade de pontos a serem testados. Para fazer essa divisão é utilizada uma variável chamada pontos_p_processo, que vai definir o número de pontos que cada processo irá testar, a partir disso é pego o número total de pontos informados na função e feita um arredondamento para utilizar uma quantidade múltipla de pontos da variável pontos_p_processo. Utilizando a função pool.map em um for, são criados processos paralelos com o mesmo número de pontos a serem testados, e com isso somados os números de pontos dentro do círculo retornados por eles.

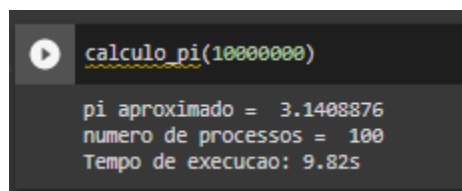
```
def calculo_pi(pontos):
    pool = Pool()
    start = time.time()
    workers = 0
    ponto_circ = 0
    pontos_p_processo = 10000
    for result in pool.map(worker, [pontos_p_processo for _ in range(pontos//pontos_p_processo)]):
        ponto_circ += result
        workers = workers+1
```

Com essas informações, é utilizada a fórmula para aproximar o PI e é informado o resultado, o número de processos criados e o tempo de execução.

```
pi = ((4*ponto_circ)/pontos)

print("pi aproximado = ", pi)
print("numero de processos = ", workers)
print("Tempo de execucao: {:.2f}s".format(time.time()-start));
```

Exemplo de resultado de uma execução com 10000000 pontos e 100000 pontos por processo, resultando em 100 processos iguais, aproximando o PI para 3.1408876 em 9.82 segundos.



The screenshot shows a Jupyter Notebook cell with a play button icon on the left. The code in the cell is `calculo_pi(10000000)`. Below the code, the output is displayed: `pi aproximado = 3.1408876`, `numero de processos = 100`, and `Tempo de execucao: 9.82s`.