Facultad de Matemática, Astronomía, Física y Computación, U.N.C.

Computación

Aula Virtual: https://famaf.aulavirtual.unc.edu.ar/course/view.php?id=747

Resguardo tutoriales: https://www.famaf.unc.edu.ar/~moreschi/docencia/Computacion/

Tutorial Problema 4 de la Guía N° 2

Problema 4: Escriba un código en PYTHON que compute la suma:

$$s = \sum_{k=1}^{M} \frac{1}{k^n}$$

para $n = 2$ y $n = 4$.

¿Convergen estas series para valores cada vez más grandes de M? ¿Cuáles serían esos valores?

El código debe realizar la suma en ambos sentidos y debe usar el valor $M = 2000000$ para $n = 2$ y $M = 200000$ para $n = 4$.

Tutorial:

En los próximos programas aprovechamos para estudiar el efecto que tiene sumar de cantidades más grandes a más pequeñas, versus hacer la suma de cantidades más pequeñas a más grandes.

Vean que hemos usado otras formas de formatear los resultados y también hemos comparado el uso de while, en lazos, con las construcciones usando for. Para ello hemos cargado una librería que nos permite tomar tiempos.

- Guarde en el archivo p4-g2-kcuadrado.py las siguientes instrucciones:

```
import numpy as np
from timeit import default_timer as timer

#---------------------------------------------------------------------------
start= timer()

M=2000000

s=0
k=1
while k <= M:
    s = s + 1/k**2
    k = k + 1
    #print("s = ",s)


r = 0
k = M
while k > 0:
    r = r + 1/k**2
    k = k - 1
    #print("r = ",r)

```

```
print()
print(" M = ",M)
print()
print("el primer  método da s = %-39.32e,        directo" %(s) )
print("el segundo método da r = %-39.32e,        inverso" %(r) )

print("               pi**2/6 = %-39.32e,        exacto" %(np.pi**2/6.) )
print()

end = timer()

print("Tiempo de ejecución usando 'while':", end-start)

#-------------------------------------------------------------------------
start= timer()

M=2000000

s=0
for k in range(1,M+1):
    s = s + 1/k**2

r=0
for k in range(M,0,-1):
    r = r + 1/k**2

print()
print(" M = ",M)
print()
print("el primer  método da s = %-39.32e,        directo" %(s) )
print("el segundo método da r = %-39.32e,        inverso" %(r) )

print("               pi**2/6 = %-39.32e,        exacto" %(np.pi**2/6.) )
print()

end = timer()

print("Tiempo de ejecución usando 'for':", end-start)
print()
```

- Desde la terminal ejecute:

  python3 p4-g2-kcuadrado.py

  e interprete el resultado.

- Guarde en el archivo p4-g2-kcuarta.py las siguientes instrucciones:

```
import numpy as np
from timeit import default_timer as timer

#-------------------------------------------------------------------------
start= timer()

M=200000

s=0
```

```
k=1
while k <= M:
    s = s + 1/k**4
    k = k + 1
    #print("s = ",s)


r = 0
k = M
while k > 0:
    r = r + 1/k**4
    k = k - 1
    #print("r = ",r)

print()
print(" M = ",M)
print()
print("el primer  método da s = %-39.32e,        directo" %(s) )
print("el segundo método da r = %-39.32e,        inverso" %(r) )

print("               pi**4/90 = %-39.32e,        exacto" %(np.pi**4/90.) )
print()

end = timer()

print("Tiempo de ejecución usando 'while':", end-start)

#-------------------------------------------------------------------------
start= timer()

M=200000

s=0
for k in range(1,M+1):
    s = s + 1/k**4

r=0
for k in range(M,0,-1):
    r = r + 1/k**4

print()
print(" M = ",M)
print()
print("el primer  método da s = %-39.32e,        directo" %(s) )
print("el segundo método da r = %-39.32e,        inverso" %(r) )

print("               pi**4/90 = %-39.32e,        exacto" %(np.pi**4/90.) )
print()

end = timer()

print("Tiempo de ejecución usando 'for':", end-start)
print()
```

- Desde la terminal ejecute:

`python3 p4-g2-kcuarta.py`

e interprete el resultado.