

# Reporte del archivo RaízCuadrada.tex

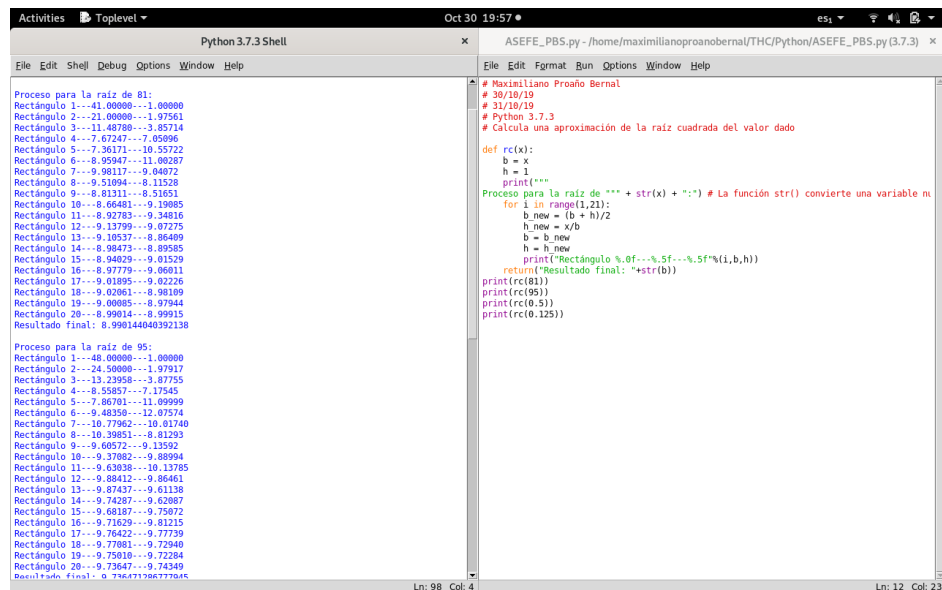
Maximiliano Proaño Bernal

October 30, 2019

En este documento hablaré sobre algunos detalles que surgieron mientras realizaba y al terminar el programa para calcular la raíz cuadrada.

Al inicio, no se tuvo mayor complicación con el entendimiento del problema y cómo atacarlo. Parecieron muy claras las herramientas que se debían utilizar en Python para poder calcular la raíz. Afortunadamente hubo pocos errores al programar y se pudieron resolver sin mayor complicación. A su vez, se utilizaron algunos conocimientos que adquirí, no en clase, sino interactuando por cuenta propia con las diversas herramientas que Python proporciona.

Una vez terminado el programa, se corrió para corroborar que funcionara como debía de hacerlo, lo cual hizo. Sin embargo, algo que noté y que me intrigó es que, comparando mi programa con el de mis compañeros, noté que el mío tardaba más en calcular la raíz, no es cuestión de tiempo, sino en cuestión de pasos necesarios.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Proceso para la raíz de 81:
Rectángulo 1--41.00000---1.00000
Rectángulo 2--21.00000---1.97561
Rectángulo 3--11.48780---3.85714
Rectángulo 4--7.67247---7.05096
Rectángulo 5--5.36171---10.55722
Rectángulo 6--4.05947---11.00287
Rectángulo 7--3.08117---9.04072
Rectángulo 8--2.51094---8.11528
Rectángulo 9--2.13111---8.51651
Rectángulo 10--1.86481---9.19085
Rectángulo 11--1.67283---9.34816
Rectángulo 12--1.51799---9.07275
Rectángulo 13--1.40337---8.96409
Rectángulo 14--1.30473---8.89585
Rectángulo 15--1.21929---9.01529
Rectángulo 16--1.14779---9.06011
Rectángulo 17--1.08595---8.82226
Rectángulo 18--1.02061---8.98109
Rectángulo 19--0.96065---8.97944
Rectángulo 20--0.90014---8.99015
Resultado final: 8.99014040392138

Proceso para la raíz de 95:
Rectángulo 1--40.00000---1.00000
Rectángulo 2--24.50000---1.97917
Rectángulo 3--13.23958---3.87755
Rectángulo 4--8.55857---7.17545
Rectángulo 5--5.78670---11.09999
Rectángulo 6--4.48359---12.07574
Rectángulo 7--3.77962---10.01740
Rectángulo 8--3.39051---8.81293
Rectángulo 9--3.06572---9.13592
Rectángulo 10--2.78082---9.88994
Rectángulo 11--2.53038---10.13785
Rectángulo 12--2.30842---9.86461
Rectángulo 13--2.10743---9.61138
Rectángulo 14--1.92487---9.62087
Rectángulo 15--1.76187---9.75072
Rectángulo 16--1.61529---9.81215
Rectángulo 17--1.48422---9.77739
Rectángulo 18--1.36701---9.72940
Rectángulo 19--1.26108---9.72284
Rectángulo 20--1.16547---9.74349
Resultado final: 9.74671726777045

# Maximiliano Proaño Bernal
# 30/10/19
# 31/10/19
# Python 3.7.3
# Calcula una aproximación de la raíz cuadrada del valor dado

def rc(x):
    b = x
    h = 1
    print('***
Proceso para la raíz de *** + str(x) + "":') # La función str() convierte una variable n
    for i in range(1,21):
        b_new = (b + h)/2
        h_new = x/b
        b = b_new
        h = h_new
        print("Rectángulo %.0f---%.5f---%.5f"%(i,b,h))
    return("Resultado final: "+str(b))

print(rc(81))
print(rc(95))
print(rc(0.51))
print(rc(0.125))

Ln: 98 Col: 4
Ln: 12 Col: 23
```

Figure 1:

En la imagen se aprecia que se utilizaron alrededor de 20 pasos para aproximar la raíz de 81 a 8.99. Esto me llamó mucho la atención, ya que aparentemente utilizamos métodos similares. Después de analizar el código, noté que definí variables extras innecesarias, las cuales sospecho que eran las responsables del retraso del programa, así que las corregí.

```

===== RESTART: /home/maximiliano/THC/Python/ASEFE_PBS.py =====
Proceso para la raíz de 81:
Rectángulo 1--41.00000---1.97561
Rectángulo 2--21.48788---3.76958
Rectángulo 3--12.62869---6.41397
Rectángulo 4--9.52133---8.50722
Rectángulo 5--9.01427---8.98575
Rectángulo 6--9.00000---9.00000
Rectángulo 7--9.00000---9.00000
Rectángulo 8--9.00000---9.00000
Resultado final: 9.0

Proceso para la raíz de 95:
Rectángulo 1--48.00000---1.97917
Rectángulo 2--24.98958---3.80158
Rectángulo 3--14.39558---6.59925
Rectángulo 4--10.49742---9.04085
Rectángulo 5--9.77363---9.72803
Rectángulo 6--9.74683---9.74676
Rectángulo 7--9.74679---9.74679
Rectángulo 8--9.74679---9.74679
Resultado final: 9.74679434888965

Proceso para la raíz de 0.5:
Rectángulo 1--0.75000---0.66667
Rectángulo 2--0.70833---0.70588
Rectángulo 3--0.70711---0.70711
Rectángulo 4--0.70711---0.70711
Rectángulo 5--0.70711---0.70711
Rectángulo 6--0.70711---0.70711
Rectángulo 7--0.70711---0.70711
Rectángulo 8--0.70711---0.70711
Resultado final: 0.7071067811865475

Proceso para la raíz de 0.125:
Rectángulo 1--0.36250---0.22222
Rectángulo 2--0.39236---0.31858
Rectángulo 3--0.35547---0.35164
Rectángulo 4--0.35356---0.35355
Rectángulo 5--0.35355---0.35355
Rectángulo 6--0.35355---0.35355
Rectángulo 7--0.35355---0.35355
Rectángulo 8--0.35355---0.35355
Resultado final: 0.3535539059327373
>>>

```

Figure 2:

Como se puede ver en la imagen, ahora solo toma alrededor de 10 pasos para aproximar de forma muy precisa a las respectivas raíces de cada número.

Para finalizar, al final del archivo RaízCuadrada.tex se le hacen dos preguntas al lector: 1. ¿Siempre se obtiene el valor exacto de la raíz? 2. ¿Hay un número fijo de rectángulos que se tengan que calcular antes de obtener la raíz?

La respuesta a la pregunta uno es: depende de cómo definamos "exacto". Si con exacto nos referimos a una escala de decimales, es decir, si nos fijamos en los primeros ocho o nueve decimales, la cual es una aproximación muy buena, la respuesta es sí, ya que siempre podemos agregar un paso extra para tener mayor precisión. Si con exacto nos referimos a perfecta exactitud, la respuesta es no, ya que el proceso de sacar promedio y dividir y todo lo que implica calcular la raíz, es un proceso infinito, lo cual es evidente que es imposible.

La respuesta a la pregunta 2 es, nuevamente: depende, pero ahora del número al que se le saca raíz. En la imagen se puede apreciar muy bien. Para algunos números fueron suficientes 3 pasos para dar una respuesta de más de 4 decimales de precisión, cosa que ocurría después de 3 o 4 pasos más para otros números. Para concluir, agrego que fue un gran ejercicio para los alumnos, ya que no solo se puso a prueba los conocimientos adquiridos en clase para usar las

diferentes herramientas de Python, sino que también nos obligó a razonar cuál sería la solución más eficiente al problema en cuestión, la cual es una habilidad casi necesaria al programar.

**1**