Introducción al lenguaje Python

Martínez Cerda Mario Antonio

22 de enero de 2021

1 Introducción.

El siguiente documento contiene algunas de las funciones básicas de Python aprendidas en la clase de Física Computacional 1. Donde se abordarán de igual forma algunos ejercicios fundamentales que ayudan con el desarrollo y la práctica de este lenguaje de programación. El objetivo de esta actividad es (como ya mencionaba) prácticar la programación en Python y ver un poco del potencial que puede llegar a tener este lenguaje, que es uno de los más importantes actualmente y que además, sirve mucho para el ámbito científico y en particular, de la Física que es a donde va orientado el curso.

2 Desarrollo.

2.1 ¿Qué es y para qué sirve Python?

Python es un lenguaje de programación interpretado, multiparadigma y multiplataforma usado, principalmente en Big Data, AI, Data Science, frameworks de pruebas y desarrollo web. Esto lo convierte en un lenguaje de propósito general de gran nivel debido a su extensa bilioteca, cuya colección ofrece una amplia gama de instalaciones.

Python se gestó durante las vacaciones de Navidad de 1989, cuando el desarrollador holandés Guido van Rossum decidipo escribir un intérprete para el nuevo lenguaje de scripting que venía trabajando.

2.2 Características de Python.

Se caracteriza por ser simple, rápido y tener una curva de aprendizaje amigable y corta. Está desarrollado bajo una licencia de código abierto, por lo que es de libre uso y distribución.

Python interpreta el código del programador, lo traduce y ejecuta a la vez. es un lenguaje de programación que admite el uso de varios paradigmas de programación, por lo que, no exige a los programadores un estilo únido de

programación. Estos paradigmas son: La programación orientada a objetos, programación imperativa y programación funcional. Además, el lenguaje de Python puede ejecutarse en diferentes sitemas operativos, como Unix, Linux, Windows y macOS.

3 Actividad.

Para la actividad realizada en el lenguaje de programación, se propusieron cuatro ejercicios sencillos, donde veremos algunas de las características básicas de Python, como:
Aritmética básica.
-Operadores para enteros.
-Operadores para números flotantes (reales).
-Expresiones Booleanaas (lógica).
-Operadores lógicos.
Controles de Flujo.
-Condicional if.
-Loop: for.
-Loop: while.
-Funciones continue y break.
Biblioteca NumPy
-Tipos de datos en NumPy.
-Listas.
-Arreglos (vectores n-dimensionales).
-Operaciones con arreglos.
Gráficas de funciones en 2D.

3.1 Ejercicio 1.

Se proporciona el siguiente programa para calcular el área de un rectángulo:

```
# Se utiliza el símbolo # para hacer comentarios.
# Se define una función de 2 variables para calcular el área.
def area(x, y):
    return x*y

print("Proporciona el largo y altura del rectángulo: ", end="")
l = float(input())
w = float(input())
a = area(l, w)
# Imprime el valor del área en formato libre.
print("\nArea = ", a)
# Imprime el valor del área con 2 decimales.
print("\nArea = {:.2f}".format(a))
```

Usando este ejemplo, modifícalo para calcular:
-Área de: círculo y una elipse.
-Volumen de: una esfera y el de un cilíndro circular.

Antes de cada celda donde se calcule un área o volumen, inserte una descripción de lo que se hace, utilizando notación Latex para las ecuaciones o símbolos.

3.2 Ejercicio 2.

Desarrollar un programa en Python que calcule las raíces de una ecuación cuadrática.

3.3 Ejercicio 3.

Implementar el método Babilonio (o Método de Herón), para calcular la raíz cuadrada de un número S, cuando la sucesión converja con un error menor a 0.01. Compare con el valor obtenido por el método Babilonio con el valor de las función np.sqrt(s).

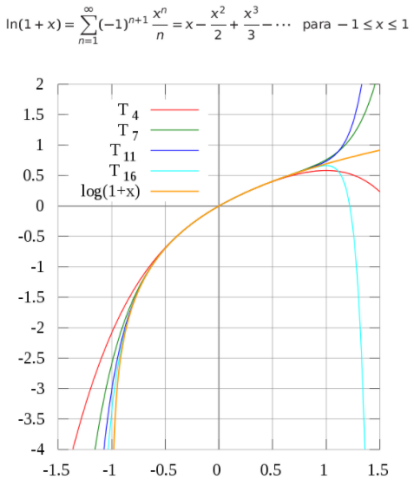
$$x_0 \approx \sqrt{S}$$

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right)$$

$$\sqrt{S} = \lim_{n \to \infty} x_n$$

3.4 Ejercicio 4.

Reproduce la figura que aparece inmediatamente abajo en el artículo de Wikipedia sobre Series de Taylor que muestra la aproximación de la función ln(1+x) alrededor de x=0.



4 Evidencias.

En este apartado se agregan las evidencias hechas en Google Colaboratory, encontradas en el repositorio personal en Github.

4.1 Resultado 1.

1.-Área de un círculo.

```
[ ]  #Primeramente agregaremos la biblioteca numpy.
     import numpy as np
     print("Este programa calculará el área del círculo de radio r.") #Imprime en pantalla que es lo que realizará el programa.
     #Definimos las función de 2 variables apra calcular el área.
     def area(x):
         return (np.pi)*(x**2)
     #Esta sección se encarga de pedirle al usuario que agregue los datos necesarios y los lee.
     print("Ingresa el radio del círculo: ")
     r= float(input())
     a= area(r)
     #Imprimimos el valor del área en formato libre.
     print("El área es=",a)
```

```
Este programa calculará el área del círculo de radio r.
Ingresa el radio del círculo:
4
El área es= 50.26548245743669
```

2.-Área de una elipse.

```
print("Este programa calculará el área de una elipse.")
def area(x,y):
    return (np.pi)*c*b
print("Ingresa el valor del semieje mayor:")
c= float(input())
print("Ingresa el valor del semieje menor:")
b= float(input())
a= area(c,b)
print("El área es=",a)
```

```
Este programa calculará el área de una elipse.
Ingresa el valor del semieje mayor:
3
Ingresa el valor del semieje menor:
2
El área es= 18.84955592153876
```

3.-Volumen de una esfera.

```
[ ]  print ("Este programa calculará el volumen de una esfera.")
     def volumen(x):
         return (4/3)*(np.pi)*(r**3)
     print("Ingresa el radio de la esfera:")
     r= float(input())
     v= volumen(r)
     print("El volumen de la esfera es=",v)
```

```
Este programa calculará el volumen de una esfera.
Ingresa el radio de la esfera:
6
El volumen de la esfera es= 904.7786842338603
```

4.-Volumen de un cilindro circular.

```
[ ]  print ("Este programa calculará el volumen de un cilindro circular recto.")
     def volumen(x,y):
         return (np.pi)*(r**2)*h
     print("Ingresa el valor del radio de una de las bases del cilindro:")
     r= float(input())
     print("Ingresa el valor de la altura del cilindro:")
     h= float(input())
     v= volumen(r,h)
     print("El volumen del cilindro es=",v)
```

```
Este programa calculará el volumen de un cilindro circular recto.
Ingresa el valor del radio de una de las bases del cilindro:
3
Ingresa el valor de la altura del cilindro:
4
El volumen del cilindro es= 113.09733552923255
```

4.2 Resultado 2.

```python
from math import sqrt
#Imprimimos en pantalla de qué se encarga el programa.
print("El programa calculará las raíces de una ecuación cuadrática.")
#Definimos el discriminante, dependiente de los coeficientes de a ecuación cuadrática.
def discriminante(a,b,c):
    dis=(b**2)-(4*a*c)
    return dis
#Definimos las raíces, que son dependientes de los coeficientes a, b y del discriminante antes definido.
#Agregando las dos raíces posibles (+ y -)
def raices(a,b,disc):
    raiz1=(-b+sqrt(disc))/(2*a)
    raiz2=(-b-sqrt(disc))/(2*a)
    return raiz1,raiz2
#Pedimos al usuario, que ingrese los valores de cada uno de los coeficientes y hacemos que el programa lo lea.
print("Ingrese el coeficiente a:")
a= float(input())
print("Ingrese el coeficiente b:")
b= float(input())
print("Ingrese el coeficiente c:")
c= float(input())
#Para efectos prácticos (y por que aún no lo sé hacer), agregaremos un condicionante.
#Este condicionante definirá en base al discriminante, si las raíces son positivas o no.
#Si el valor del discriminante es menor a 0, entonces no existen raíces positivas.
disc=discriminante(a,b,c)
if disc < 0:
   print("No hay raíces positivas")
else:
   print("Las raíces son:", raices(a,b,disc))
```

```
El programa calculará las raíces de una ecuación cuadrática.
Ingrese el coeficiente a:
3
Ingrese el coeficiente b:
4
Ingrese el coeficiente c:
-5
Las raíces son: (0.7862996478468913, -2.119632981180225)
```

4.3 Resultado 3.

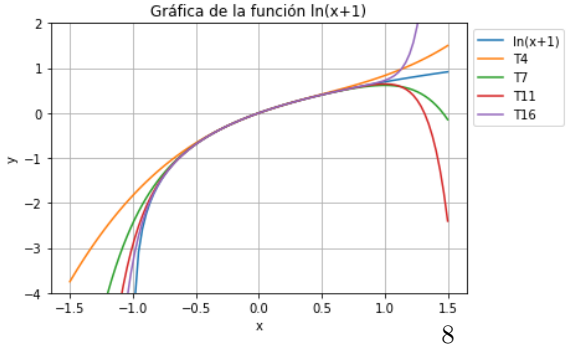
```
#Importamos la biblioteca numpy para agregar la función sqrt.
import numpy as np
#Especificamos para que sirve el siguiente programa.
print("El siguiente programa calcula la raíz cuadrada de cualquier número positivo")
#Solicitamos al usuario que indique el número al que quiera sacarle raíz.
print("Ingrese el número al que quieras sacarle raíz cuadrada:")
s = float(input())
#Agregamos el valor inicial en el que empezará el loop.
xn = 1
#Ingresamos el loop que se encargará de realizar el método de Herón.
for i in range(1,10,1): #Después de pruebas, es suficiente con 10 incrementos.
    raiz = (1/2)*(xn + s/xn)
    xn = raiz
print("La raíz cuadrada de",s,"es:",raiz)
```

```
El siguiente programa calcula la raíz cuadrada de cualquier número positivo
Ingrese el número al que quieras sacarle raíz cuadrada:
2
La raíz cuadrada de 2.0 es: 1.414213562373095
```

4.4 Resultado 4.

```
import matplotlib.pyplot as plt
import numpy as np
from sympy.functions import ln
import sympy as sp

print("Este programa calcula los valores de la serie de taylor para n términos.")
x = sp.Symbol('x')
f = ln(x+1)
def taylor(p,x0,n):
  i = 1
  p = 0
  while i < n:
    p = p + ((-1)**(i+1))*((x**i)/i)
    i += 1
  return p
print (taylor(f,0,4))

x = np.linspace(-1.5,1.5,100)
y = np.log(x+1)

plt.subplot(111)
plt.xlabel("x")
plt.ylabel("y")
plt.title("Gráfica de la función ln(x+1)")

plt.plot(x,y, label="ln(x+1)")
plt.plot(x,taylor(f,0,4), label="T4")
plt.plot(x,taylor(f,0,7), label="T7")
plt.plot(x,taylor(f,0,11), label="T11")
plt.plot(x,taylor(f,0,16), label="T16")

plt.grid(True)

plt.ylim(-4,2)

plt.legend(bbox_to_anchor=(1,1), loc="best")

plt.show()

plt.savefig('plot.png')
```

```
Este programa calcula los valores de la serie de taylor para n términos.
x**3/3 - x**2/2 + x
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:19: RuntimeWarning: invalid value encountered in log
```



5 Conclusión.

Hubieron diversos problemas durante la realización de las actividades hechas en Python, debido a que mucho de los problemas introducían loops (el cual no tengo practicado), se me dificultaron debido a eso. Sin embargo, debido a las guías y videos que se pueden encontrar por internet, fue de grata ayuda para poder resolver los problemas. También se ponen en práctica la gestión del tiempo para las actividades, debido a que cada una de ellas toma su tiempo y me ayudó a no estar presionando con cada programa. A parte de eso, la actividad fue sencilla, entendible y ayuda mucho a la práctica de Python.

6 Referencias.

1.-http://blog.espol.edu.ec/analisisnumerico/u1t1-polinomio-de-taylor-con-python/
2.-python.org
3.-http://computacional1.pbworks.com/w/page/142820286/Actividad%202%20(2021-1)