Ejercicio 1

Área de un rectángulo

Para el primero solo tuve que copiar el código y entenderlo así que no hubo complicación.
Lo siguiente fue añadir texto en el notebook para organizarme y así poder hacer el siguiente.

Área de un Círculo

El programa consiste en calcular el área de un círculo, con los datos que ingrese el usuario.
El esquema a seguir será:
1.- Importar la biblioteca NumPy
2.- Definir la función para el área.
3.- Imprimir la descripción del programa.
4.- Pedir al usuario el radio.
5.- Imprimir el valor del área con dos decimales.

Área de una Elipse

El programa consiste en calcular el área de una elipse, con los datos que ingrese el usuario.
El esquema a seguir será:
1.- Importar la biblioteca NumPy
2.- Definir la función para el área.
3.- Imprimir la descripción del programa.
4.- Pedir al usuario el eje menor y el mayor.
5.- Imprimir el valor del área con dos decimales.

Volumen de una Esfera

El programa consiste en calcular el volumen de una esfera, con los datos que ingrese el usuario.
El esquema a seguir será:
1.- Importar la biblioteca NumPy
2.- Definir la función para el volumen.
3.- Imprimir la descripción del programa.
4.- Pedir al usuario el radio.
5.- Imprimir el valor del volumen con dos decimales.

Volumen de un Cilindro circular

El programa consiste en calcular el volumen de un cilindro circular, con los datos que ingrese el usuario.
El esquema a seguir será:
1.- Importar la biblioteca NumPy
2.- Definir la función para el volumen.
3.- Imprimir la descripción del programa.
4.- Pedir al usuario el radio y la altura.
5.- Imprimir el valor del volumen con dos decimales.

Ejercicio 2

Desarrollar un programa en Python que calcule las raíces de una ecuación cuadrática.
El programa consistirá en que el usuario introducirá los valores de los coeficientes de una ecuación cuadrática del tipo
$$ax^2+bx+c$$

Con lo cual el programa utilizará la formula general
$$x=\frac{-b\pm \sqrt{b^2-4ac}}{2a} $$
Recordando que hay dos soluciones posibles, de momento solo reales.
El algoritmo será el siguiente:
1.- Definir la función de la fórmula chicharronera o general.
2.-Imprimir la descripción del programa.
3.-Pedir al usuario que introduzca las variables correspondientes.
4.- Calcular las dos soluciones de ser posible.
5.- Imprimir al usuario las dos soluciones.
Primera modificación:
Era necesario definir dos funciones, no solo una, para las soluciones.
Notas:
-No importé NumPy así que no pude usar la función de raíz cuadrada, así que elevé a la un medio.
-Este programa queda muy sencillo así que lo podemos actualizar para que también funcione para ecuaciones cuyo valor de a sea cero, así como informar al usuario cuando una raíz sea imaginaria utilizando el discriminante.
-Quise poner lo de los decimales pero NO JALÓ
-No le puse número a los enteros pero quiero agarrar esa costumbre
Segunda modificación:
Al parecer el programa si funcionó para encontrar soluciones imaginarias solo que con j en lugar de i. así que podemos dejar así el programa ya que funciona. Esto debido a que si el uruario quiere resolver una ecuación con a=0 deberá usar una calculadora de ecuaciones lineales y no cuadrática y ninguna calculadora lo resuelve.

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 la función np.sqrt(S).

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

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

$$\lim_{x \to \infty} x_n$$

Siendo este el método.
Para este programa será necesario hacer un loop que se repita hasta que se alcance el error menor a 0.01 que se menciona.
El algoritmo será el siguiente:
1.- Invocar a la biblioteca numpy.
2.- Pedirle al usuario el número S.

2.- Abrir un loop de while que se cumpla siempre y cuando el último valor obtenido menos el penúltimo sea menor que 0.01.
3.-En el loop también imprimimos el resultado actual y lo definimos según la formula.
4.- Por último afuera del loop imprimimos el número.
5.- También imprimimos el número calculado con numpy.
6. Hacemos la diferencia entre estos dos números para comparar.
Notas:
también imprimí el número de iteraciones utilizadas por el programa y el error respecto al valor obtenido con numpy. También se agregó una leyenda que advierte de error si se introduce un número negativo haciendo un gran IF para valores de S negativos, si esto no se cumple se procede al programa original.

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.

$$ln(1+x) = \sum_{n=1}^{\infty} (-1)^{n+1} \frac{x^n}{n} = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots$$

para $-1 \leq x \leq 1$

Siendo este el método.
El algoritmo será el siguiente
1.- Invocar la subbiblioteca pytlot de Matplotlib.
2.- Invocar la biblioteca numpy
3.- definir una x para que se creen 100 punos desde -1.5 hasta 1.5
4.-definir y1 como taylor grado 4
5.-definir y2 como taylor grado 7
6.-definir y3 como taylor grado 11
7.-definir y4 como taylor grado 16
8.-definir y5 como la funcion del logaritmo
9.-graficamos varias funciones cuidando las etiquetas.
10.agregamos una malla
11.fijamos los valores de y desde -4 a 2.
12.ubicamos las etiquetas de las funciones.
13.agregar el plt.show
Notas:
no sé que significa lo del paso 13. Salta el siguiente error:
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:20: RuntimeWarning: invalid value encountered in log, y por último no pude colocar las leyendas a la izquierda con ninguna de las

opciones (esperaba que upper left lo solucionaría en el box de las leyendas). Así como tampoco pude graficar el eje de las y de 0.5 en 0.5, y no de 1 en 1