

**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**EVOLUTIONARY COMPUTING**

**PRÁCTICA 1: PYTHON Y MATPLOTLIB**

**ALUMNO: ORTEGA VICTORIANO IVAN**

**PROF.: JOSE LUIS ROSAS TRIGUEROS**

**FECHA DE REALIZACIÓN DE LA PRÁCTICA: 20/02/2018**

**FECHA DE ENTREGA DEL REPORTE: 27/02/2018**

## MARCO TEÓRICO.

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. [1]

Guido van Rossum es el autor de Python. A finales de los 80's, Van Rossum empezó a trabajar en Python en el *National Research Institute for Mathematics and Computer Science* en los Países Bajos. Desde entonces, Python se ha vuelto muy popular entre los desarrolladores, quienes son atraídos por su sintaxis limpia y su reputación por productividad. [2]

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. [3]

El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, <http://www.python.org/>, y puede distribuirse libremente. El mismo sitio contiene también distribuciones y enlaces de muchos módulos libres de Python de terceros, programas y herramientas, y documentación adicional.

El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables. [3]

Por otro lado, Matplotlib es una biblioteca de graficación 2D para Python que produce figuras de calidad de publicación en una variedad de formatos impresos y entornos interactivos en todas las plataformas. Matplotlib se puede utilizar en scripts de Python, el Shell de Python, IPython, Jupyter Notebook, servidores de aplicaciones web, entre otros. [4]

Matplotlib trata de hacer sencillas las cosas fáciles y posibles las cosas difíciles. Se pueden generar gráficas, histogramas, espectros de potencia, gráficas de barras, diagramas de error, diagramas de dispersión, etc., con solo unas pocas líneas de código. [4]

Para graficas simples, el módulo pyplot proporciona una interfaz parecida a MATLAB, particularmente si se combina con IPython. Para usuarios avanzados, tiene un control total de estilos de línea, propiedades de fuente, propiedades de los ejes, etc., a través de una interfaz orientada a objetos o mediante un conjunto de funciones familiares para los usuarios de MATLAB. [4]

## MATERIAL Y EQUIPO.

- Equipo de cómputo (PC o laptop).
- Python2 o Python3
- Matplotlib

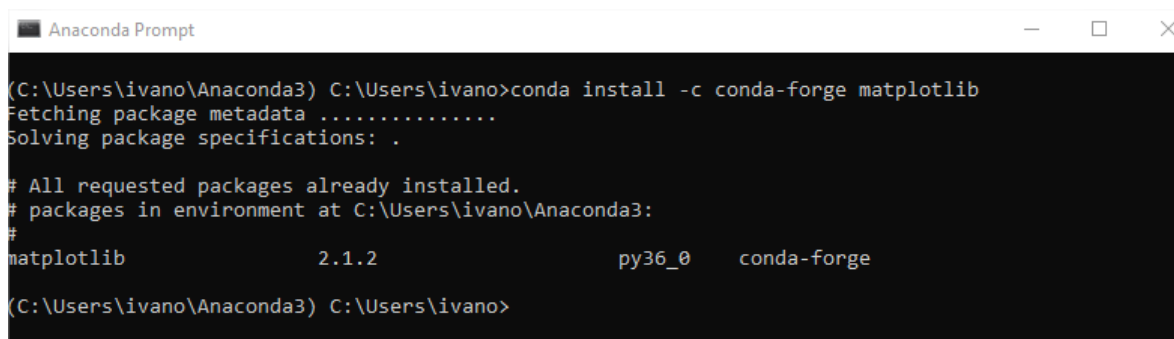
## DESARROLLO DE LA PRÁCTICA.

### PARTE 1: Cuestionario

1. ¿Quién es el autor de Python?
  - Guido van Rossum
2. ¿Por qué el lenguaje se llama Python?
  - Porque Guido era fan de un grupo de cómicos conocidos como los Monty Python.
3. Explica el término “pythonic”.
  - El término hace referencia al código que sigue los principios de Python de legibilidad y transparencia. Algunos de ellos son:
    - Simple el mejor que complejo.
    - Disperso es mejor que denso.
    - Bello es mejor que feo,
    - Entre otras.
4. Explica la diferencia entre “list”, “tuple” y “dictionary”.
  - Las tuplas son más eficientes que las listas. En las listas y tuplas tenemos control sobre el orden de los elementos, mientras que los diccionarios mapean llaves a valores (similar a un hash).

## PARTE 2: Uso de Matplotlib

Primero antes de utilizar esta biblioteca de graficación, es necesario contar con alguna de las versiones de Python (Python2 o Python3) instalada en nuestros equipos. Ya que hicimos eso, existen distintas maneras de hacerlo. En mi caso basto con ejecutar el siguiente comando en el bash: *sudo apt-get install Python-matplotlib*, pero como mencioné, hay más formas, como usando gestores de paquetes como Conda, pip, etc. De hecho, para que pudiera instalarlo en Windows utilice conda para hacerlo, abriendo el prompt de Anaconda y ejecutando el comando: *conda install -c conda-forge matplotlib*, como se muestra en la figura 1, ya que los otros medios resultaban aún más complicados.

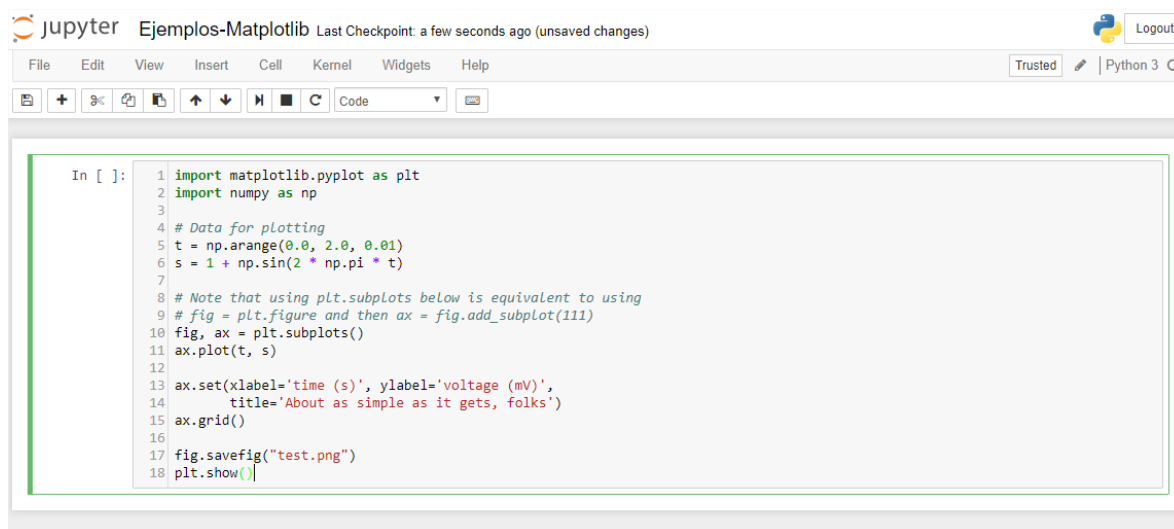


```
Anaconda Prompt
(C:\Users\ivano\Anaconda3) C:\Users\ivano>conda install -c conda-forge matplotlib
Fetching package metadata .....
Solving package specifications: .

# All requested packages already installed.
# packages in environment at C:\Users\ivano\Anaconda3:
#
matplotlib                2.1.2                py36_0    conda-forge
(C:\Users\ivano\Anaconda3) C:\Users\ivano>
```

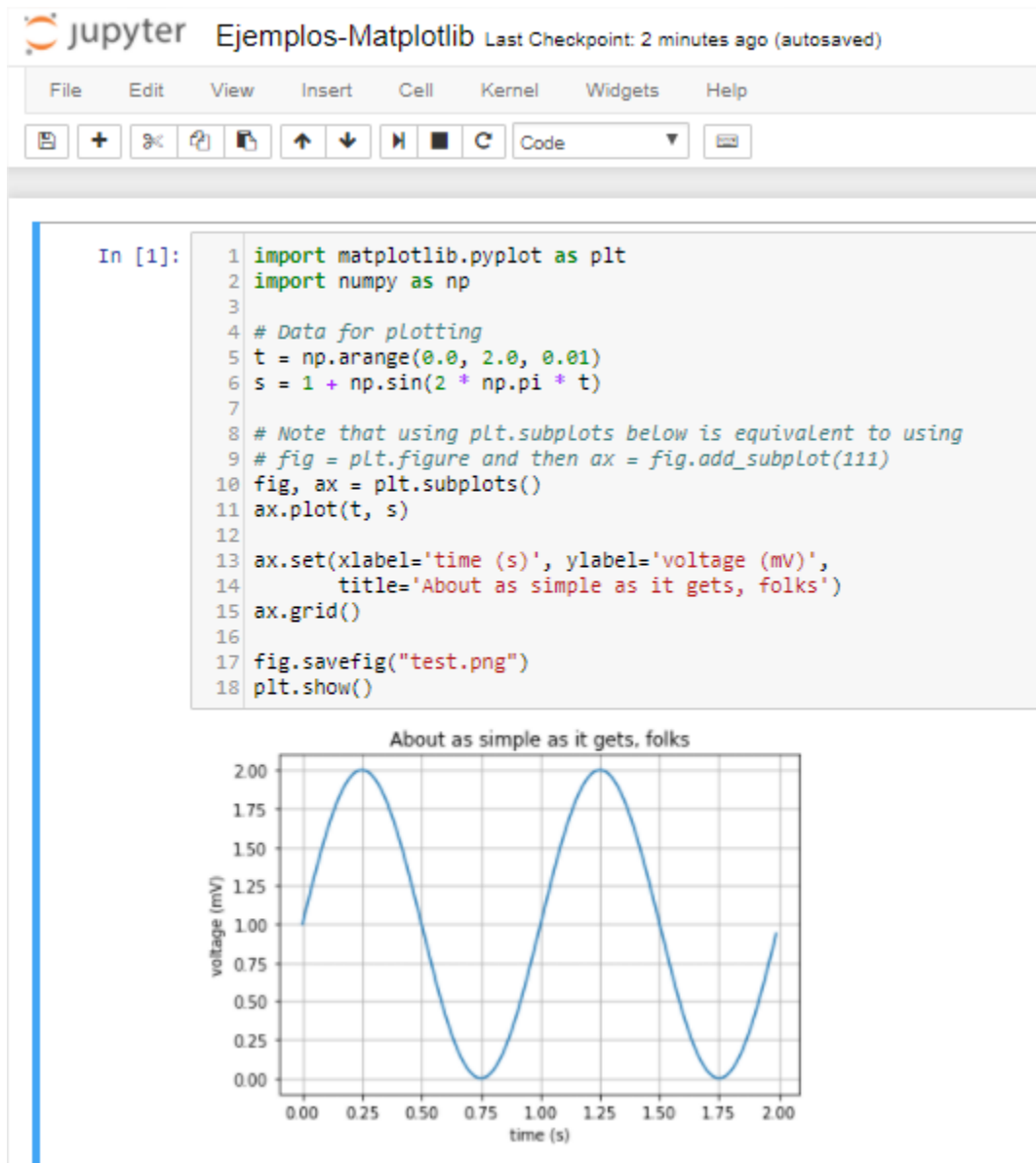
**Figura 1.** Instalación de matplotlib desde el prompt de Anaconda (Se puede ver que para este caso, la biblioteca ya ha sido instalada).

Ya que se realizó lo anterior, procedí a abrir un nuevo cuaderno en Jupyter Notebook, que viene en la instalación de Anaconda Navigator, copié uno de los códigos de prueba en la página de matplotlib y verifiqué que funcionara correctamente, el resultado se puede apreciar en las figuras 2 y 3:



```
Jupyter Ejemplos-Matplotlib Last Checkpoint: a few seconds ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [ ]: 1 import matplotlib.pyplot as plt
        2 import numpy as np
        3
        4 # Data for plotting
        5 t = np.arange(0.0, 2.0, 0.01)
        6 s = 1 + np.sin(2 * np.pi * t)
        7
        8 # Note that using plt.subplots below is equivalent to using
        9 # fig = plt.figure and then ax = fig.add_subplot(111)
       10 fig, ax = plt.subplots()
       11 ax.plot(t, s)
       12
       13 ax.set(xlabel='time (s)', ylabel='voltage (mV)',
       14       title='About as simple as it gets, folks')
       15 ax.grid()
       16
       17 fig.savefig("test.png")
       18 plt.show()
```

**Figura 2.** Se pega el código en una celda del cuaderno.



**Figura 3.** Resultado de la ejecución del ejemplo de matplotlib desde Jupyter Notebook.

Ya que vemos que funciona adecuadamente, se nos dejó graficar las funciones de Rastrigin y Ackley para una y dos variables.

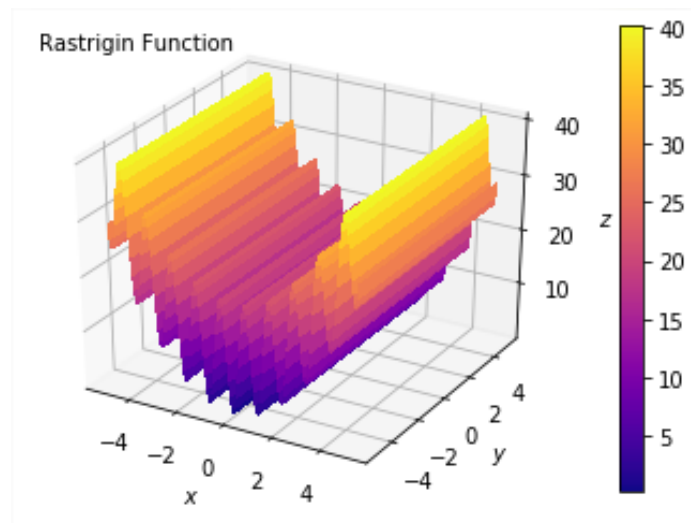
Los códigos utilizados para estas gráficas se enviarán junto con este reporte, además de que estarán disponibles en mi repositorio de Github: <https://github.com/IvanovskyOrtega/Evolutionary-Computing/tree/master/Practica-1>

Tanto la versión de una y dos variables de la gráfica, se realizaron en un plano tridimensional, la única variación, se vería reflejada en el eje Y, los valores no cambian en la versión de una variable, mientras que, en la versión de dos variables, se generan curvas por la variación de los valores de Y.

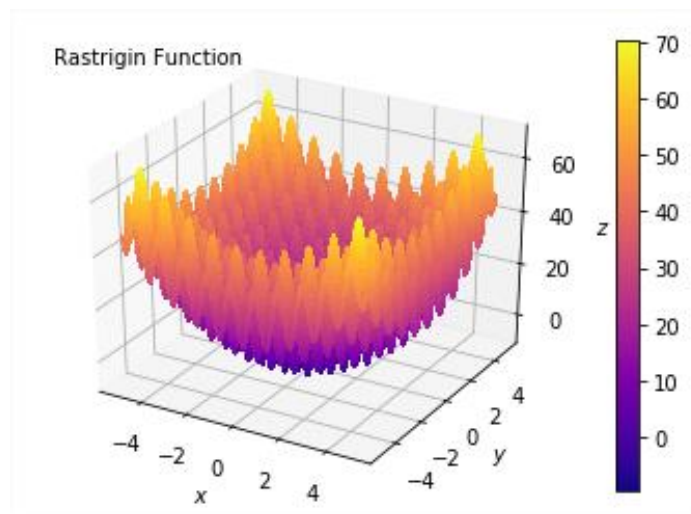
Ambas funciones son utilizadas para evaluar las características de los algoritmos de optimización, tales como:

- Tasa de convergencia
- Precisión.
- Robustez.
- Rendimiento general.

La versión de la función de Rastrigin de una variable se muestra en la figura 4 y su versión de dos variables en la figura 5.

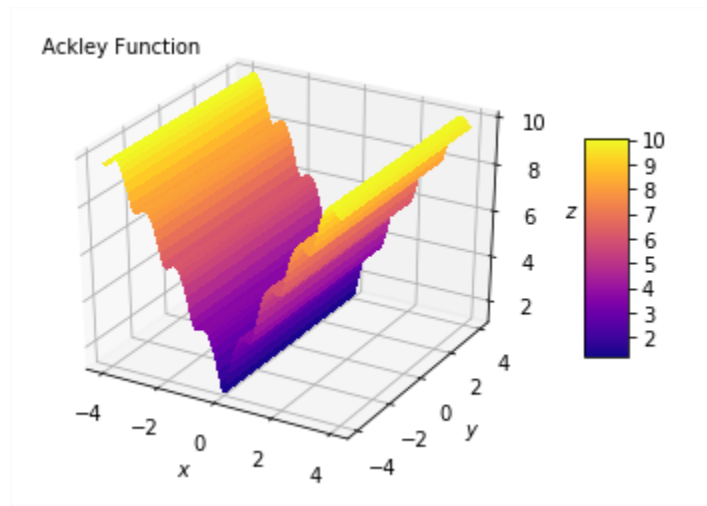


**Figura 4.** Función de Rastrigin (versión de una variable graficada en un plano tridimensional, como se mencionó anteriormente, los valores en Y no son modificados).

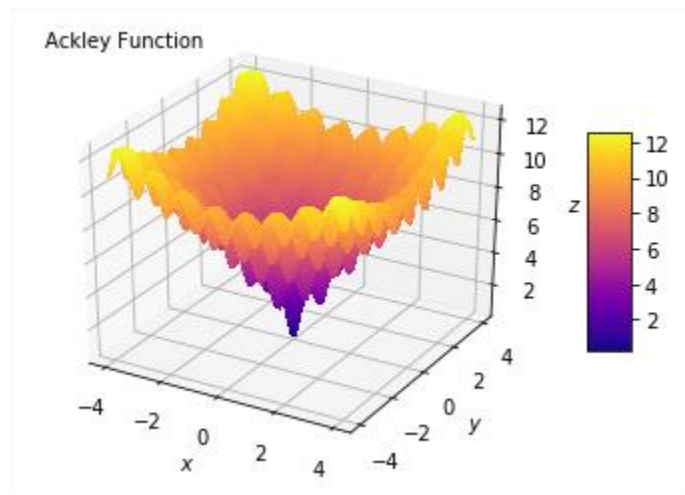


**Figura 5.** Función de Rastrigin (versión de dos variables).

La versión de la función de Ackley de una variable se muestra en la figura 6 y su versión de dos variables en la figura 7.



**Figura 6.** Función de Ackley (versión de una variable graficada en un plano tridimensional, como se mencionó anteriormente, los valores en Y no son modificados).



**Figura 7.** Función de Ackley (versión de dos variables).

## CONCLUSIONES.

Esta práctica me pareció muy útil, ya que es la primera vez que trabajo con Python para algo de graficación. Realmente tiene muy poco que empecé a aprenderlo, sin embargo, creo me fue suficiente para poder comprender el código. Aunque ya tenía un poco de experiencia con Matlab, utilizar Matplotlib me pareció más complicado en cuanto a los nombres y parámetros de las funciones. Me costó trabajo comprender que hacía cada función y como podía aplicarlo a mi código. Algo que mencioné en el desarrollo respecto a la instalación de Matplotlib, fue que usé Anaconda Navigator, realmente lo recomiendo y más aún si se trabaja en el sistema operativo Windows, ya que en el resulta un poco más difícil instalar muchas de las bibliotecas que están disponibles para Python, y el gestor de paquetes Conda, hace esto aún más sencillo. Y en cuanto a la ejecución de scripts y código en Python, Jupyter Notebook es una buena opción, este último igual viene integrado con Anaconda Navigator.

## Referencias

- [1] «Wikipedia: Python,» [En línea]. Available: <https://es.wikipedia.org/wiki/Python>.
- [2] B. Venners, «The Making Of Python,» 13 Enero 2003. [En línea]. Available: <https://www.artima.com/intv/pythonP.html>.
- [3] G. v. Rossum, «El tutorial de Python,» [En línea]. Available: <http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf>.
- [4] «Introduction to Matplotlib,» [En línea]. Available: <https://matplotlib.org/>.