

Tema: Matplotlib y Seaborn

Contenido

El propósito de esta practica es aprender los fundamentos de la librería Matplotlib y Seaborn para ser aplicado posteriormente en la solución de problemas de aprendizaje automático y ciencias de datos.

Objetivo Especifico

- a) Instalar Matplotlib y Seaborn y utilizar entornos de desarrollo IDE como Pycharm o utilizar un editor de texto como Geany.
- b) Aprender la sintaxis de Matplotlib y Seaborn.

Material y Equipo

- a) Virtual Box
- b) Linux Mint 21.3

Introduccion Teorica

Matplotlib es una librería o API para crear gráficos 2D de matrices en Python. Aunque Matplotlib está escrito principalmente en Python puro, hace un uso intensivo de NumPy y otros códigos de extensión para proporcionar un buen rendimiento incluso para matrices grandes.

Matplotlib está diseñado con la filosofía de que debería poder crear gráficos simples con solo unos pocos comandos, o solo uno.

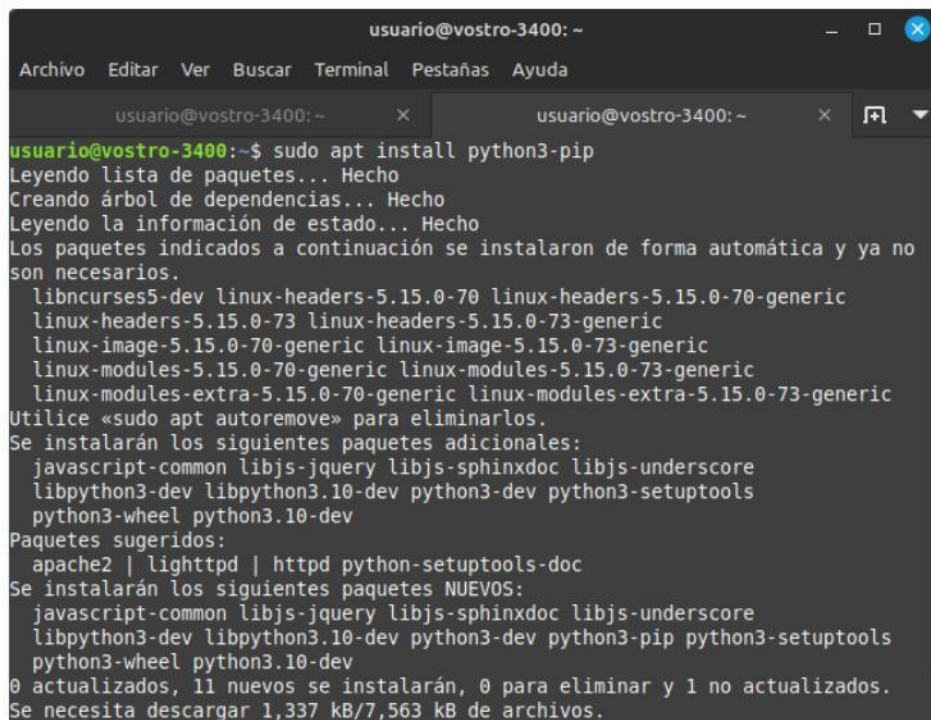
NumPy es una extensión del lenguaje de programación Python que agrega soporte para matrices multidimensionales grandes, junto con una amplia librería API de funciones matemáticas de alto nivel para operar en estas matrices.

Procedimiento

Instalación de Pip

Instalar Pip, en caso de no estar instalado.

```
$ sudo install python3-pip
```

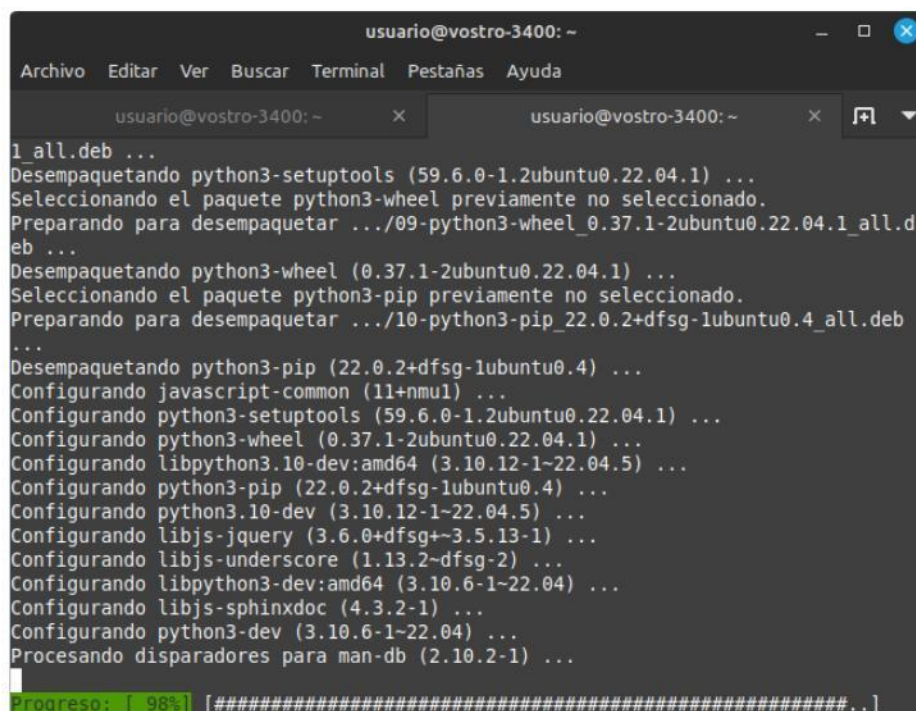


```

usuario@vostro-3400: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda

usuario@vostro-3400: ~
usuario@vostro-3400: ~
usuario@vostro-3400:~$ sudo apt install python3-pip
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Los paquetes indicados a continuación se instalaron de forma automática y ya no
son necesarios.
  libncurses5-dev linux-headers-5.15.0-70 linux-headers-5.15.0-70-generic
  linux-headers-5.15.0-73 linux-headers-5.15.0-73-generic
  linux-image-5.15.0-70-generic linux-image-5.15.0-73-generic
  linux-modules-5.15.0-70-generic linux-modules-5.15.0-73-generic
  linux-modules-extra-5.15.0-70-generic linux-modules-extra-5.15.0-73-generic
Utilice «sudo apt autoremove» para eliminarlos.
Se instalarán los siguientes paquetes adicionales:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore
  libpython3-dev libpython3.10-dev python3-dev python3-setuptools
  python3-wheel python3.10-dev
Paquetes sugeridos:
  apache2 | lighttpd | httpd python3-setuptools-doc
Se instalarán los siguientes paquetes NUEVOS:
  javascript-common libjs-jquery libjs-sphinxdoc libjs-underscore
  libpython3-dev libpython3.10-dev python3-dev python3-pip python3-setuptools
  python3-wheel python3.10-dev
0 actualizados, 11 nuevos se instalarán, 0 para eliminar y 1 no actualizados.
Se necesita descargar 1,337 kB/7,563 kB de archivos.

```



```

usuario@vostro-3400: ~
Archivo Editar Ver Buscar Terminal Pestañas Ayuda

usuario@vostro-3400: ~
usuario@vostro-3400: ~
1 all.deb ...
Desempaquetando python3-setuptools (59.6.0-1.2ubuntu0.22.04.1) ...
Seleccionando el paquete python3-wheel previamente no seleccionado.
Preparando para desempaquetar .../09-python3-wheel_0.37.1-2ubuntu0.22.04.1_all.d
eb ...
Desempaquetando python3-wheel (0.37.1-2ubuntu0.22.04.1) ...
Seleccionando el paquete python3-pip previamente no seleccionado.
Preparando para desempaquetar .../10-python3-pip_22.0.2+dfsg-1ubuntu0.4_all.deb
...
Desempaquetando python3-pip (22.0.2+dfsg-1ubuntu0.4) ...
Configurando javascript-common (11+nmu1) ...
Configurando python3-setuptools (59.6.0-1.2ubuntu0.22.04.1) ...
Configurando python3-wheel (0.37.1-2ubuntu0.22.04.1) ...
Configurando libpython3.10-dev:amd64 (3.10.12-1~22.04.5) ...
Configurando python3-pip (22.0.2+dfsg-1ubuntu0.4) ...
Configurando python3.10-dev (3.10.12-1~22.04.5) ...
Configurando libjs-jquery (3.6.0+dfsg+~3.5.13-1) ...
Configurando libjs-underscore (1.13.2~dfsg-2) ...
Configurando libpython3-dev:amd64 (3.10.6-1~22.04) ...
Configurando libjs-sphinxdoc (4.3.2-1) ...
Configurando python3-dev (3.10.6-1~22.04) ...
Procesando disparadores para man-db (2.10.2-1) ...
Progreso: [ 98%] [#####..]

```

Verificar que Pip esta correctamente instalado

3 ASI104 - Guía 6

\$ pip3 --version

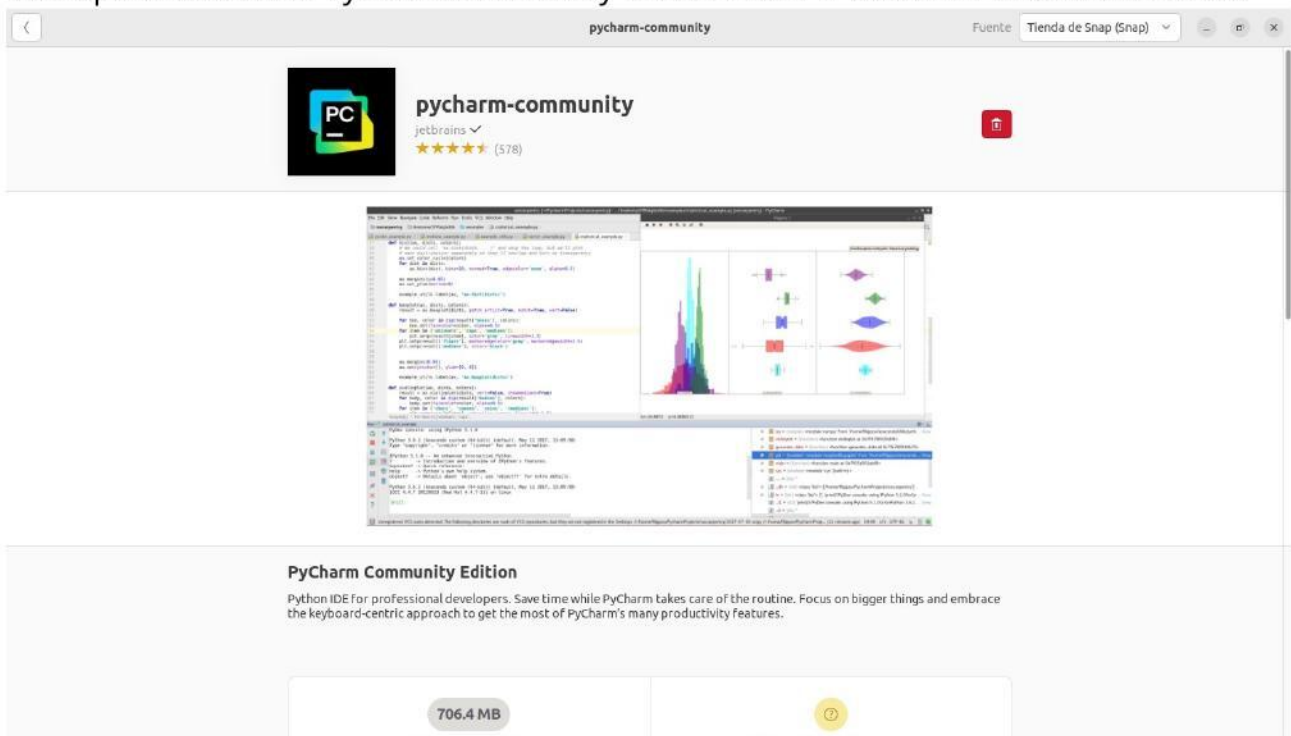
```
usuario@vostro-3400: ~  
Archivo  Editar  Ver  Buscar  Terminal  Pestañas  Ayuda  
usuario@vostro-3400: ~ x usuario@vostro-3400: ~ x [+]  
usuario@vostro-3400:~$ pip3 --version  
pip 22.0.2 from /usr/lib/python3/dist-packages/pip (python 3.10)  
usuario@vostro-3400:~$
```

Instalación de Matplotlib

Proceder a la instalación de Matplotlib

```
#sudo pip install matplotlib
```

Otra opción es instalar Pycharm Community Edition desde el Gestor de Software de Linux:

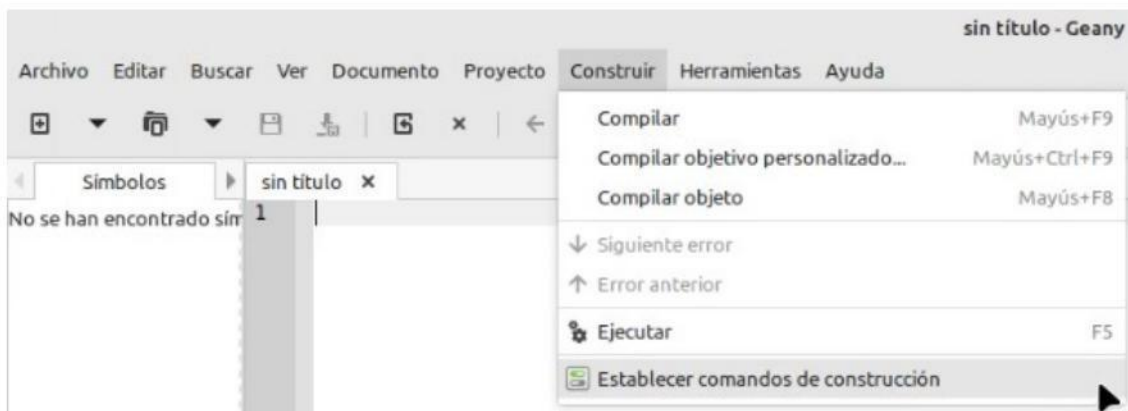


Otra opción más ligera es instalar Geany como IDE:

```
#sudo apt-get install geany
```

Instalado geany se debe configurar las opciones de compilación para que Geany pueda interpretar y ejecutar un programa en Python.

Desde el menú principal de Geany seleccionar la opción "Construir", después seleccionar "Establecer comandos de construcción"



5 ASI104 - Guía 6

En la ventana de “Establecer los comandos de construcción”, asegurarse que las opciones “Comandos de Python” y “Comandos de ejecución” estén configurados con la cadena que se indica en el campo “comando”.

Establecer los comandos de construcción

#	Etiqueta	Comando	Directorio de trabajo	Reiniciar
Comandos de Python				
1.	Compile	python3 -m py_compile "%f"		
2.				
3.	Lint	pep8 --max-line-length=80 '		
Expresión regular de error:		(.+) :([0-9]+) :([0-9]+)		
Comandos independientes				
1.	Compilar	make		
2.	Compilar objetivo personalizado...	make		
3.	Compilar objeto	make %e.o		
4.				
Expresión regular de error:				
<i>Nota: El elemento 2 abre un diálogo y añade la respuesta al comando.</i>				
Comandos de ejecución				
1.	Execute	python3 "%F"		
2.				
<i>%d, %e, %f, %p y %l se sustituirán en los campos de comandos y directorios, consulte el manual para más información.</i>				
			Cancelar	Aceptar

Fundamentos de Matplotlib

Gráficos de Líneas

Un gráfico de líneas o diagrama de líneas es un gráfico en el que los puntos del gráfico (a menudo denominados marcadores) están conectados por líneas para mostrar cómo algo cambia de valor a medida que cambia un conjunto de valores (normalmente el eje x); por ejemplo, a lo largo de una serie de intervalos de tiempo también conocidos como series de tiempo. Los gráficos de líneas de series de tiempo suelen dibujarse en orden cronológico.

El siguiente gráfico es un ejemplo de gráfico de líneas que representa el tiempo en la parte inferior (eje x) en comparación con la velocidad (representada por el eje y).

```
ejemplo01.py x
1 import matplotlib.pyplot as pyplot
2 x = [0, 1, 2, 3, 4, 5, 6]
3 y = [0, 2, 6, 14, 30, 43, 75]
4 pyplot.ylabel( ylabel: 'Velocidad', fontsize=12)
5 pyplot.xlabel( xlabel: 'Tiempo', fontsize=12)
6
7 pyplot.title("Velocidad vs Tiempo")
8 pyplot.plot( *args: x, y, 'bo-')
9 pyplot.show()
```

Salida:

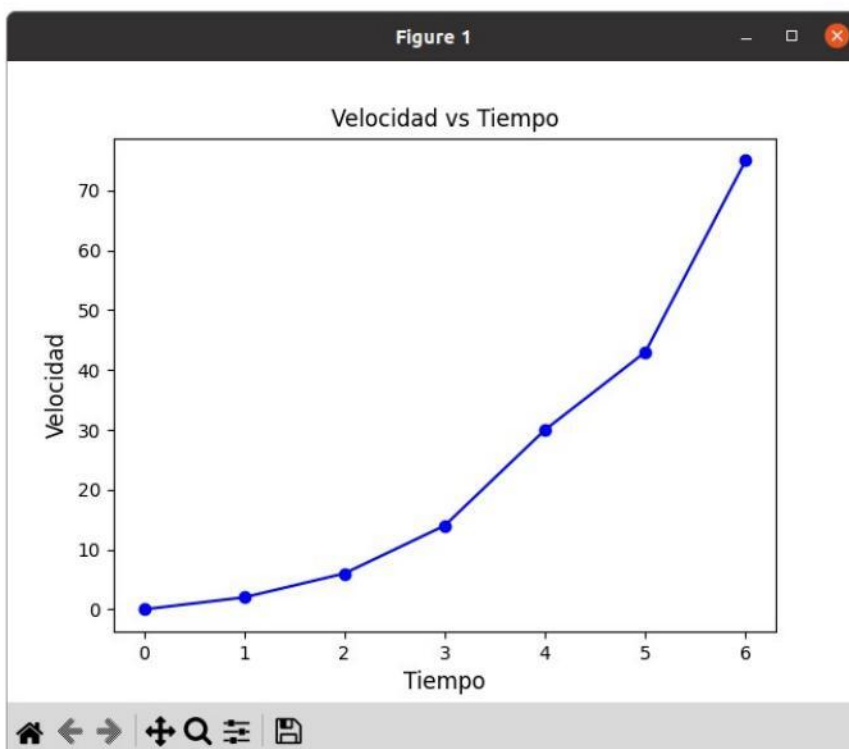


Gráfico de dispersión

Un gráfico de dispersión es un tipo de gráfico en el que los valores individuales se indican mediante coordenadas cartesianas (x,y).

Cada valor se indica mediante una marca, como un círculo o un triángulo en el gráfico. Se pueden utilizar para representar valores obtenidos para dos variables diferentes, una representada en el eje x y la otra en el eje y.

A continuación se muestra un ejemplo de un gráfico de dispersión con tres conjuntos de valores de dispersión.

```
ejemplo01.py x
1 import matplotlib.pyplot as pyplot
2 equitacion = ((17, 18, 21, 22, 19, 21, 25, 22, 25, 24),(3, 6, 3.5, 4, 5, 6.3, 4.5, 5, 4.5, 4))
3 natacion = ((17, 18, 20, 19, 22, 21, 23, 19, 21, 24),(8, 9, 7, 10, 7.5, 9, 8, 7, 8.5, 9))
4 navegacion = ((31, 28, 29, 36, 27, 32, 34, 35, 33, 39),(4, 6.3, 6, 3, 5, 7.5, 2, 5, 7, 4))
5 pyplot.scatter(x=equitacion[0], y=equitacion[1], c='red', marker='o', label='equitacion')
6 pyplot.scatter(x=natacion[0], y=natacion[1], c='green', marker='^', label='natacion')
7 pyplot.scatter(x=navegacion[0], y=navegacion[1], c='blue', marker='*', label='navegacion')
8 pyplot.xlabel('Edad')
9 pyplot.ylabel('Horas')
10 pyplot.title('Actividades')
11 pyplot.legend()
12 pyplot.show()
```

Salida:

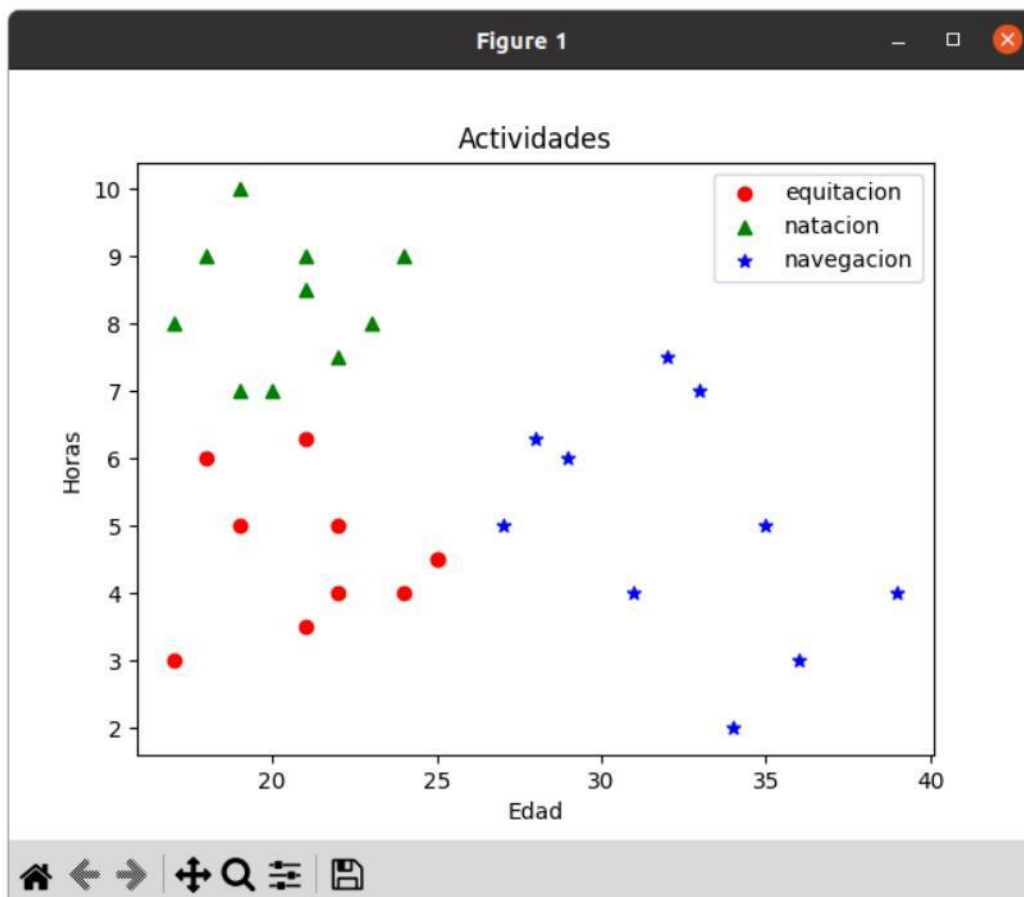


Gráfico de dispersión (continuación)

Se crea un gráfico de dispersión utilizando todos los marcadores disponibles para un gráfico de tipo scatter.

```
script.py x
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  markers = ['.', ',', 'o', 'v', '^', '<', '>', '1', '2', '3', '4', 's', 'p', '*',
5            'h', 'H', '+', 'x', 'D', 'd', '|', '_']
6
7  plt.style.use('ggplot')
8
9  plt.figure(figsize=(12, 8))
10
11  for i, marcador in enumerate(markers):
12      x = np.random.rand(10)
13      y = np.random.rand(10)
14      plt.scatter(x, y, marker=marcador, label=f'Marcador: {marcador}', s=100)
15
16  plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
17  plt.title('Visualización de todos los marcadores disponibles en matplotlib')
18  plt.xlabel('Eje X')
19  plt.ylabel('Eje Y')
20  plt.tight_layout()
21  plt.show()
```

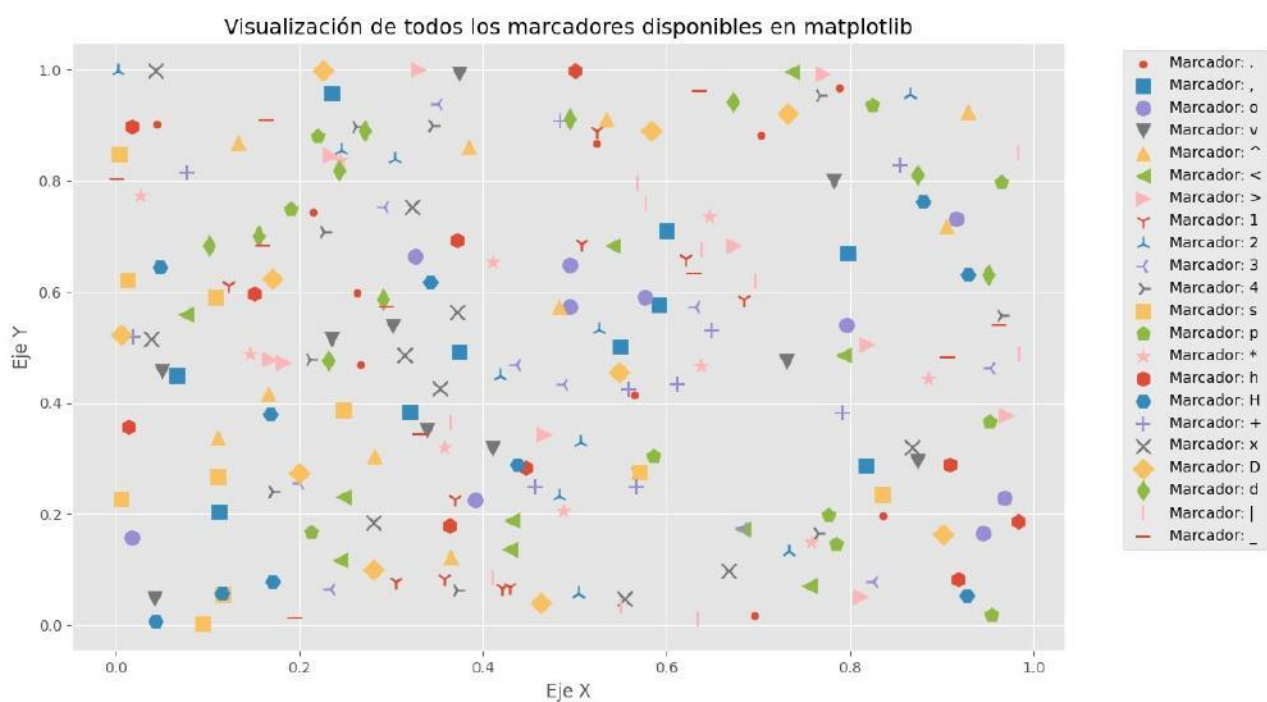


Gráfico de dispersión (continuación)

Los diagramas de dispersión se utilizan cuando es necesario mostrar la relación entre dos variables. Los diagramas de dispersión a veces se denominan diagramas de correlación porque muestran cómo se correlacionan dos variables.

En muchos casos, se puede discernir una tendencia entre los puntos trazados en un gráfico de dispersión, aunque puede haber valores atípicos. Para ayudar a visualizar la tendencia, puede ser útil dibujar una línea de tendencia junto con el gráfico de dispersión. La línea de tendencia ayuda a hacer más clara la relación de los diagramas de dispersión con la tendencia general.

El siguiente gráfico representa un conjunto de valores como un gráfico de dispersión y dibuja la línea de tendencia de este gráfico en mención. Como se puede ver, algunos valores están más cerca de la línea de tendencia que otros.

```
ejemplo01.py x
1 import numpy as np
2 import matplotlib.pyplot as pyplot
3 x = (5, 5.5, 6, 6.5, 7, 8, 9, 10)
4 y = (120, 115, 100, 112, 80, 85, 69, 65)
5 pyplot.scatter(x, y)
6 z = np.polyfit(x, y, deg: 1)
7 p = np.poly1d(z)
8 pyplot.plot(*args: x, p(x), 'r')
9 pyplot.show()
10
```

Salida:

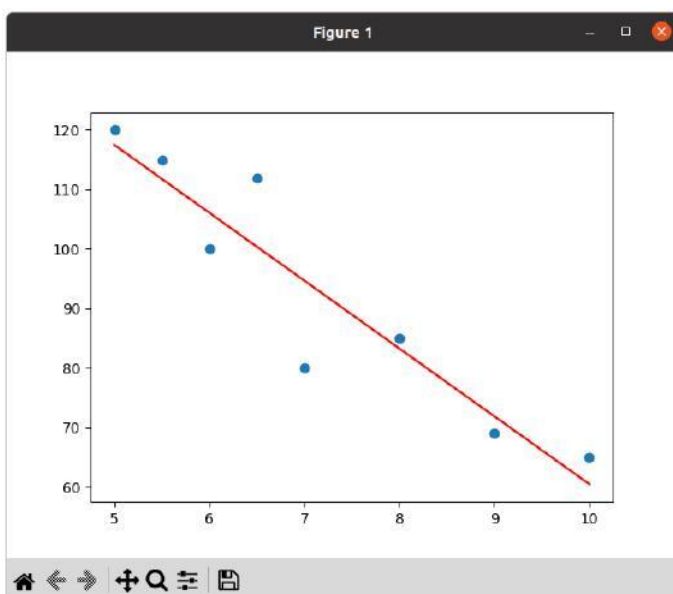


Gráfico de pastel

Un gráfico de pastel o circular es un tipo de gráfico en el que un círculo se divide en sectores (o porciones) que representan cada uno una proporción del total. Una porción del círculo representa la contribución de una categoría al total general. Como tal, el gráfico se asemeja a un pastel que se ha cortado en porciones de diferentes tamaños.

```
ejemplo01.py x
1 import matplotlib.pyplot as pyplot
2 etiquetas= ('Python', 'Java', 'Scala', 'C#')
3 tamaño = [45, 30, 15, 10]
4 pyplot.pie(tamaño,
5           labels=etiquetas,
6           autopct='%1.f%%',
7           counterclock=False,
8           startangle=90)
9 pyplot.show()
10
```

Salida:

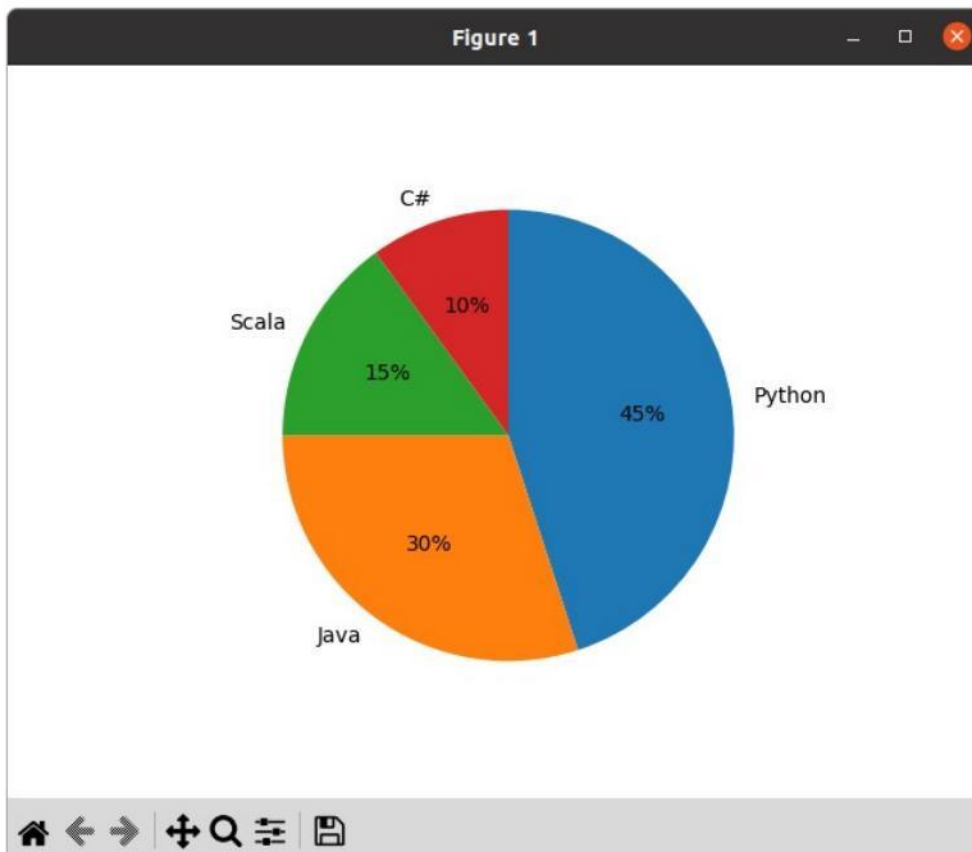


Gráfico de pastel con segmentos expandidos

Puede resultar útil enfatizar un segmento particular del gráfico circular, separándolo del resto del gráfico circular. Esto se puede hacer utilizando el parámetro `explode` de la función `pie()` que toma una secuencia de valores que indican cuánto se debe separar un segmento.

El impacto visual del gráfico circular también se puede mejorar en este caso al agregar una sombra a los segmentos utilizando el parámetro booleano `shadow`. El efecto de esto se muestra a continuación:

```
ejemplo01.py x
1 import matplotlib.pyplot as pyplot
2 etiquetas = ('Python', 'Java', 'Scala', 'C#')
3 tamaño = [45, 30, 15, 10]
4 separacion = (0.1, 0, 0, 0)
5 pyplot.pie(tamaño,
6 explode=separacion,
7 labels=etiquetas,
8 autopct='%1.1f%%',
9 shadow=True,
10 counterclock=False,
11 startangle=90)
12 pyplot.show()
13
```

Salida:

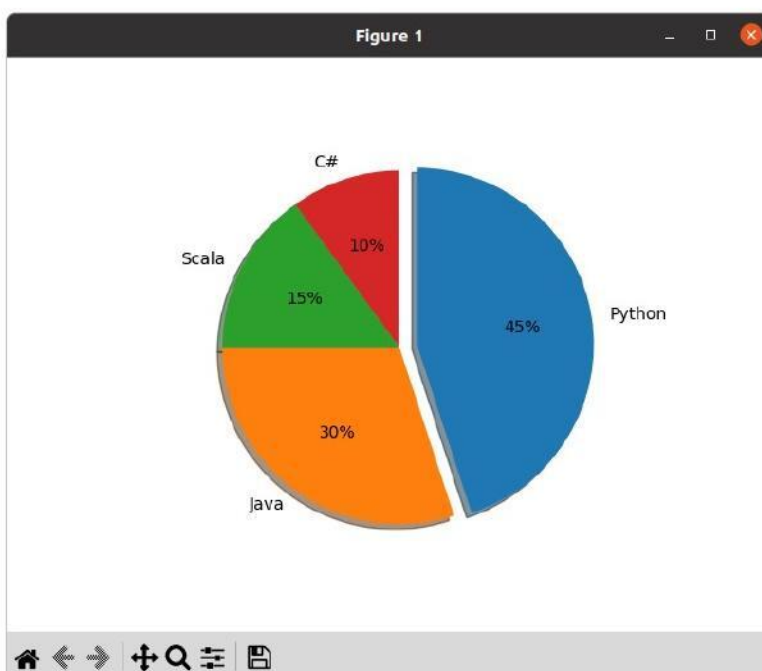
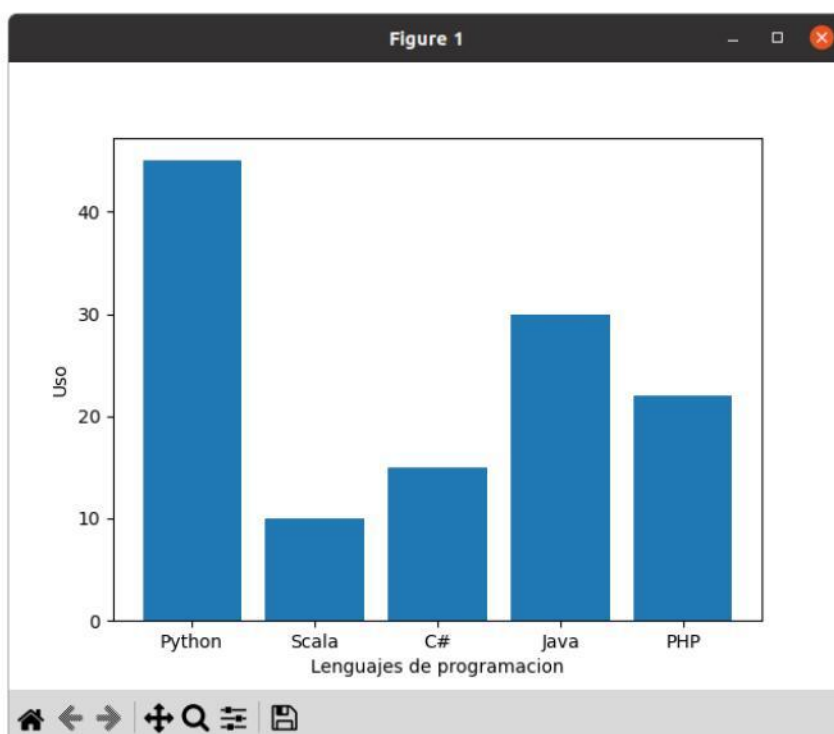


Gráfico de barras

Un gráfico de barras es un tipo de gráfico que se utiliza para presentar diferentes categorías discretas de datos. Los datos se presentan normalmente de forma vertical, aunque en algunos casos se pueden utilizar gráficos de barras horizontales. Cada categoría está representada por una barra cuya altura (o longitud) representa los datos de esa categoría. Debido a que es fácil interpretar los gráficos de barras y cómo se relaciona cada categoría con otra, son uno de los tipos de gráfico más utilizados. También existen varias variaciones comunes, como los gráficos de barras agrupadas y los gráficos de barras apiladas. El siguiente es un ejemplo de un gráfico de barras típico. Se presentan cinco categorías de lenguajes de programación a lo largo del eje x, mientras que el eje y indica el porcentaje de uso. Cada barra representa el porcentaje de uso asociado con cada lenguaje de programación.

```
ejemplo01.py x
1 import matplotlib.pyplot as pyplot
2 etiquetas = ('Python', 'Scala', 'C#', 'Java', 'PHP')
3 indices = (1, 2, 3, 4, 5)
4 tamanios = [45, 10, 15, 30, 22]
5 pyplot.bar(indices, tamanios, tick_label=etiquetas)
6 pyplot.ylabel('Uso')
7 pyplot.xlabel('Lenguajes de programacion')
8 pyplot.show()
9
```

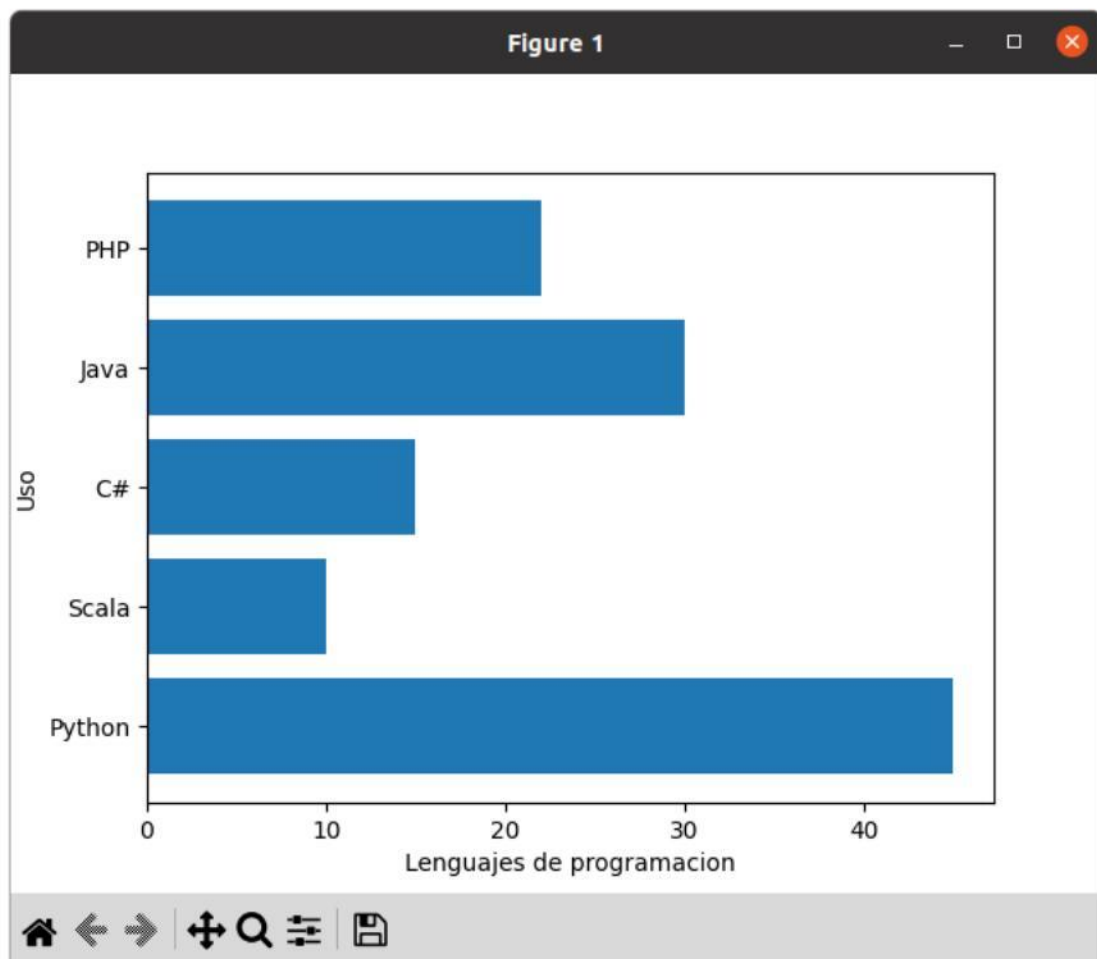
Salida:



Versión del gráfico de barras anterior, pero con orientación horizontal utilizando el método `barh()` de `matplotlib`.

```
ejemplo01.py x
1 import matplotlib.pyplot as pyplot
2 etiquetas = ('Python', 'Scala', 'C#', 'Java', 'PHP')
3 indices = (1, 2, 3, 4, 5)
4 tamanios = [45, 10, 15, 30, 22]
5 pyplot.barh(indices, tamanios, tick_label=etiquetas)
6 pyplot.ylabel('Uso')
7 pyplot.xlabel('Lenguajes de programacion')
8 pyplot.show()
9
```

Salida:



Versión del gráfico de barras anterior, pero estableciendo un color diferente para cada barra del gráfico.

```
ejemplo01.py x
1 import matplotlib.pyplot as pyplot
2 etiquetas = ('Python', 'Scala', 'C#', 'Java', 'PHP')
3 indices = (1, 2, 3, 4, 5)
4 tamanios = [45, 10, 15, 30, 22]
5 pyplot.bar(indices, tamanios, tick_label=etiquetas, color=('red', 'green', 'blue', 'yellow', 'orange'))
6 pyplot.ylabel('Uso')
7 pyplot.xlabel('Lenguajes de programacion')
8 pyplot.show()
9
```

Salida:

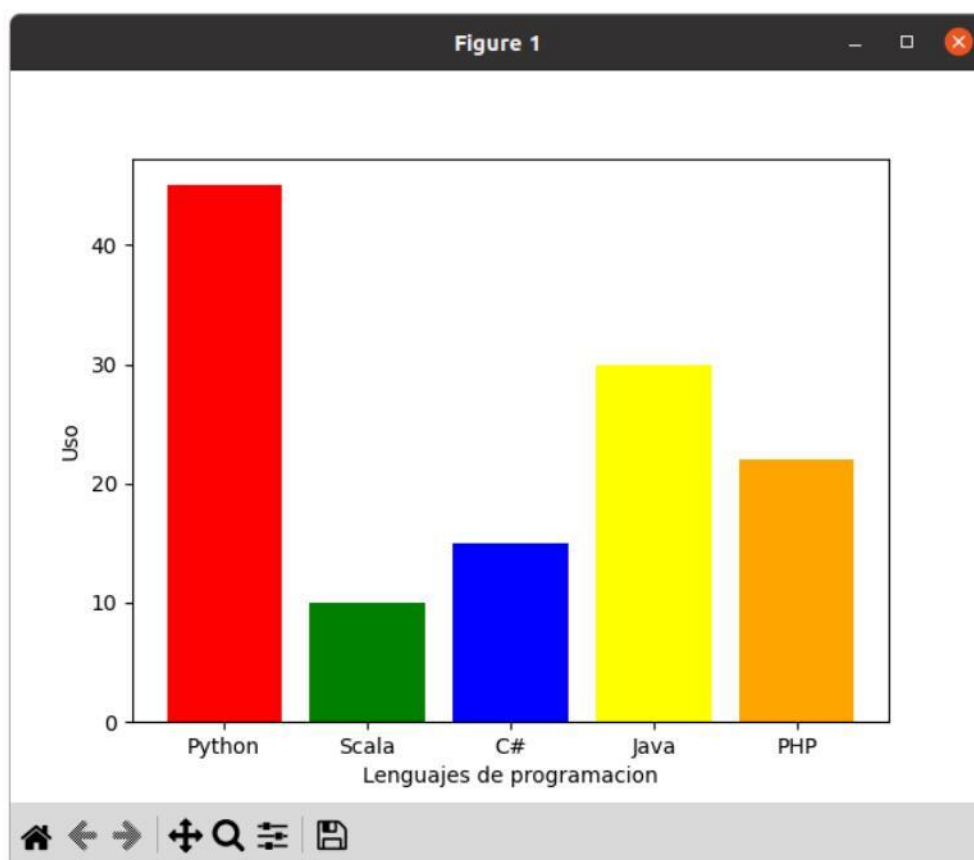
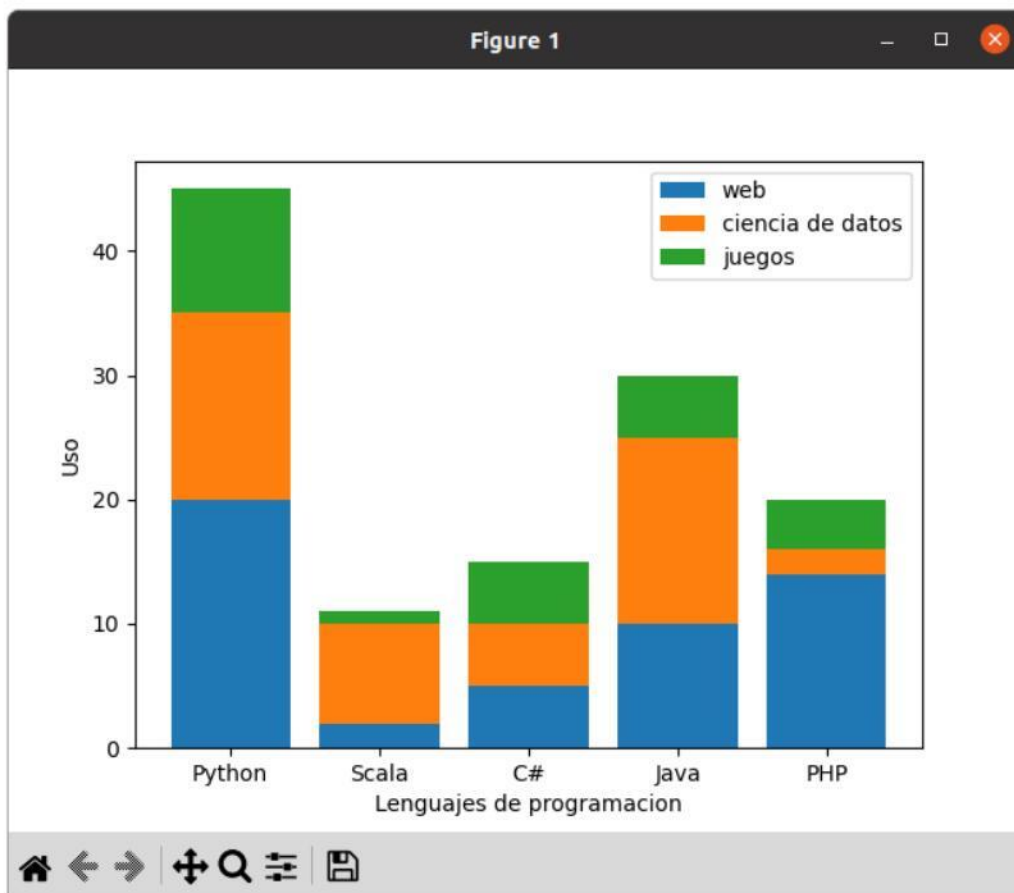


Gráfico de barras apiladas

```
ejemplo01.py x
1 import matplotlib.pyplot as pyplot
2 etiquetas = ('Python', 'Scala', 'C#', 'Java', 'PHP')
3 indices = (1, 2, 3, 4, 5)
4 web = [20, 2, 5, 10, 14]
5 ciencia_datos = [15, 8, 5, 15, 2]
6 juegos = [10, 1, 5, 5, 4]
7 pyplot.bar(indices, web, tick_label=etiquetas, label='web')
8 pyplot.bar(indices, ciencia_datos, tick_label=etiquetas, label='ciencia de datos', bottom=web)
9 web_y_juegos = [web[i] + ciencia_datos[i]
10 for i in range(0, len(web))]
11 pyplot.bar(indices, juegos, tick_label=etiquetas, label='juegos', bottom=web_y_juegos)
12 pyplot.ylabel('Uso')
13 pyplot.xlabel('Lenguajes de programacion')
14 pyplot.legend()
15 pyplot.show()
16
```

Salida:



Gráficos de barras agrupadas

```
ejemplo01.py x
1 import matplotlib.pyplot as pyplot
2 ancho_de_barra = 0.35
3
4 equipoa = (60, 75, 56, 62, 58)
5 equipob = (55, 68, 80, 73, 55)
6
7 indice_teama = (1, 2, 3, 4, 5)
8 indice_teamb = [i + ancho_de_barra for i in indice_teama]
9
10 marcas = [i + ancho_de_barra / 2 for i in indice_teama]
11 etiquetas_de_marcas = ('Laboratorio1', 'Laboratorio2', 'Laboratorio3', 'Laboratorio4', 'Laboratorio5')
12
13 pyplot.bar(indice_teama, equipoa, ancho_de_barra, color='b', label='Equipo A')
14 pyplot.bar(indice_teamb, equipob, ancho_de_barra, color='g', label='Equipo B')
15
16 pyplot.xlabel('Laboratorios')
17 pyplot.ylabel('Resultados')
18 pyplot.title('Resultados por Laboratorios')
19 pyplot.xticks(marcas, etiquetas_de_marcas)
20
21 pyplot.legend()
22 pyplot.show()
23
```

Salida:

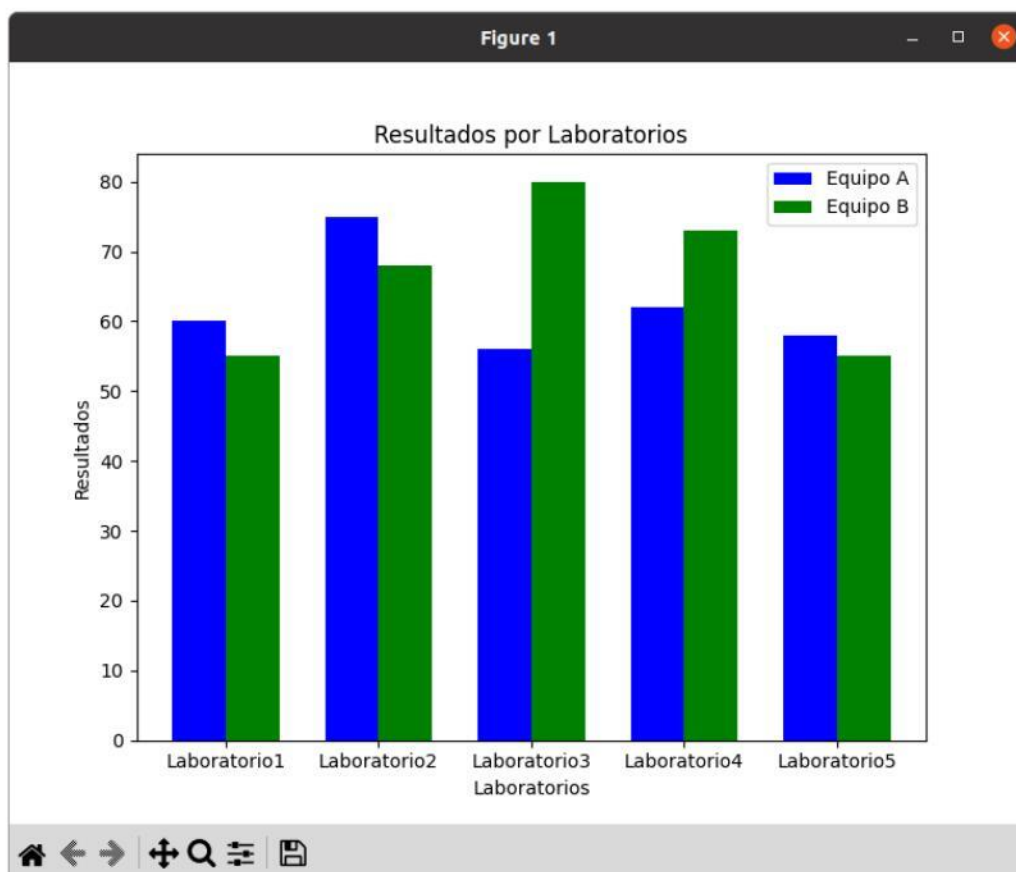


Gráfico Boxplot

El diagrama de caja es otra herramienta para describir el comportamiento de los datos y es de suma utilidad en estadística aplicada.

El diagrama de caja se basa en el concepto de cuartiles y divide los datos ordenados en cuatro grupos, que contienen, cada uno, 25% de las mediciones. De esta forma es posible visualizar dónde termina de acumularse 25% de los datos menores, y a partir de dónde se localiza 25% de los datos mayores. Entre estos dos cuartiles se ubica 50% de los datos que están al centro.

```

ejemplo01.py x
1  import numpy as np
2  import matplotlib.pyplot as plt
3  v = np.array([186,196,206,216,226,236,246,256,266,276,286,296,306,316,326,
4               336,346,356,366,376,386,200,186,193,193,196,191,200,190,191])
5  plt.boxplot(v)
6
7  plt.xlabel('Grupo de Datos')
8  plt.ylabel('Valores')
9  plt.title('Grafico boxplot')
10
11 plt.grid()
12 plt.show()

```

Salida:

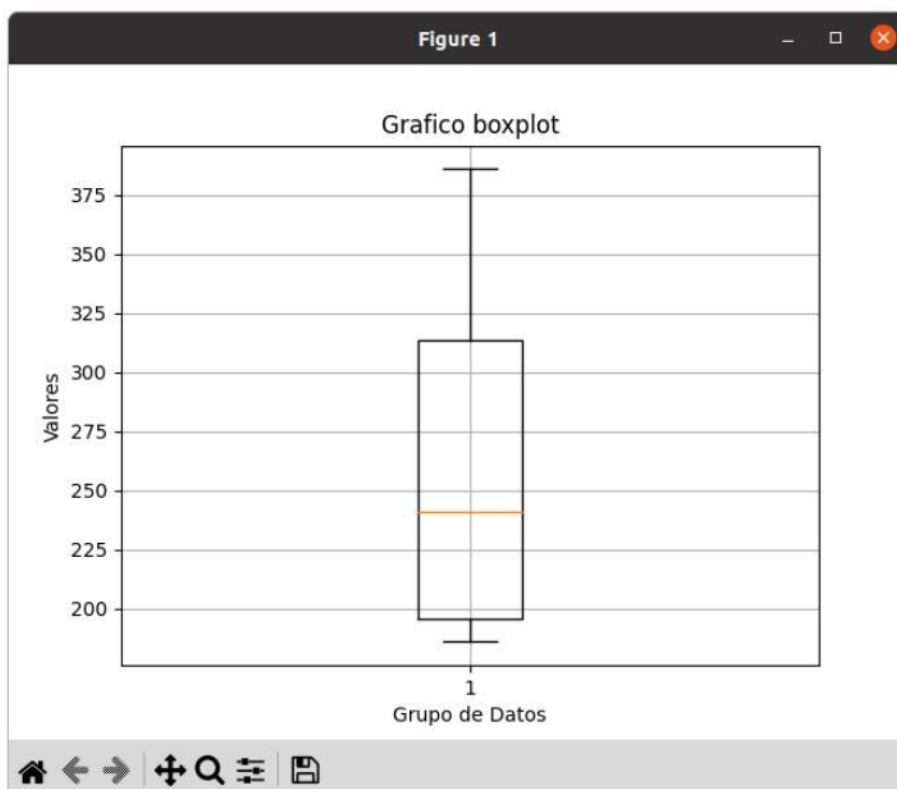


Gráfico Boxplot

Se muestra el gráfico anterior, pero se ha agregado información de estadísticos de los datos.

```
script.py x
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 v = np.array([186,196,206,216,226,236,246,256,266,276,286,296,306,316,326,
5 336,346,356,366,376,386,200,186,193,193,196,191,200,190,191])
6
7 plt.boxplot(v)
8
9 q1 = np.percentile(v, q: 25)
10 q2 = np.percentile(v, q: 50)
11 q3 = np.percentile(v, q: 75)
12 mean = np.mean(v)
13
14 plt.axhline(mean, color='red', linestyle='--', label=f'Media: {mean:.1f}')
15 plt.axhline(q1, color='blue', linestyle=':', label=f'Q1: {q1:.1f}')
16 plt.axhline(q2, color='green', linestyle='-.', label=f'Q2 (Mediana): {q2:.1f}')
17 plt.axhline(q3, color='purple', linestyle=':', label=f'Q3: {q3:.1f}')
18
19 plt.xlabel('Grupo de Datos')
20 plt.ylabel('Valores')
21 plt.title('Grafico boxplot')
22 plt.grid()
23 plt.legend()
24 plt.show()
```

Salida:

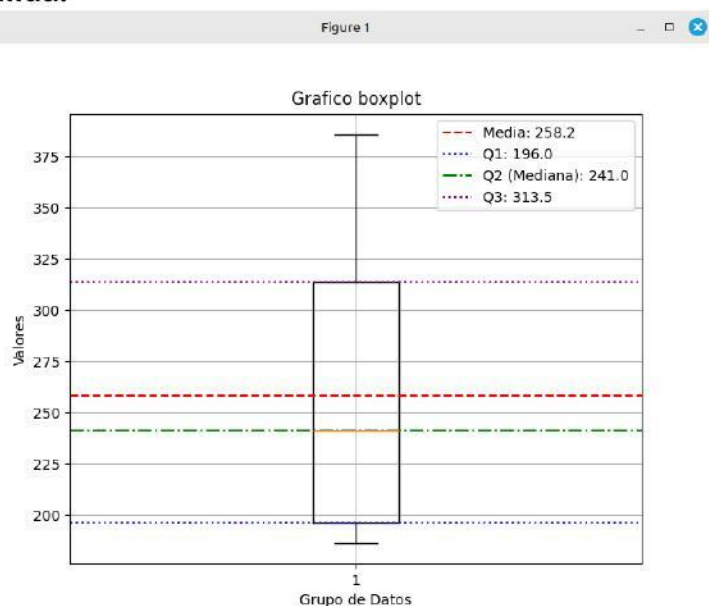
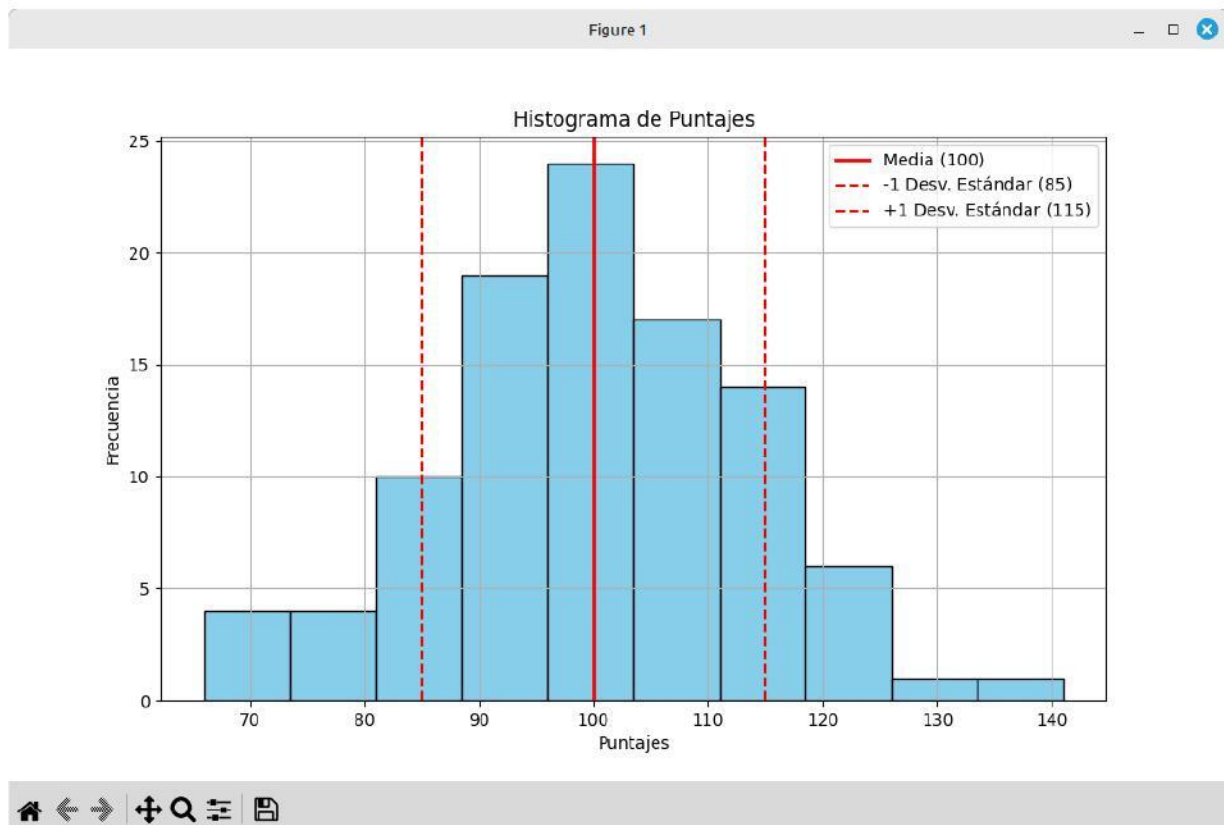


Gráfico de frecuencias

El siguiente ejemplo muestra un gráfico de frecuencias para un conjunto de puntajes

```
script.py x
1 import matplotlib.pyplot as plt
2
3 puntajes = [126, 89, 90, 101, 102, 74, 93, 101, 66, 120, 108,
4 97, 98, 105, 119, 92, 113, 81, 104, 108, 83, 102, 105, 111, 102,
5 107, 103, 89, 89, 110, 71, 110, 120, 85, 111, 83, 122, 120, 102,
6 84, 118, 100, 100, 114, 81, 109, 69, 97, 95, 106, 116, 109, 114,
7 98, 90, 92, 98, 91, 81, 85, 86, 102, 93, 112, 76, 89, 110,
8 75, 100, 90, 96, 94, 107, 108, 95, 96, 96, 114, 93, 95, 117,
9 141, 115, 95, 86, 100, 121, 103, 66, 99, 96, 111, 110, 105, 110,
10 91, 112, 102, 112, 75]
11
12 mean = 100
13 std_dev = 15
14
15 plt.figure(figsize=(10, 6))
16 plt.hist(puntajes, bins=10, color='skyblue', edgecolor='black')
17 plt.axvline(mean, color='red', linestyle='solid', linewidth=2, label='Media (100)')
18 plt.axvline(mean - std_dev, color='red', linestyle='dashed', linewidth=1.5, label='-1 Desv. Estándar (85)')
19 plt.axvline(mean + std_dev, color='red', linestyle='dashed', linewidth=1.5, label='+1 Desv. Estándar (115)')
20
21 plt.title('Histograma de Puntajes')
22 plt.xlabel('Puntajes')
23 plt.ylabel('Frecuencia')
24 plt.legend()
25 plt.grid(True)
26 plt.show()
```



Lectura de datos desde un archivo para la generación de un gráfico

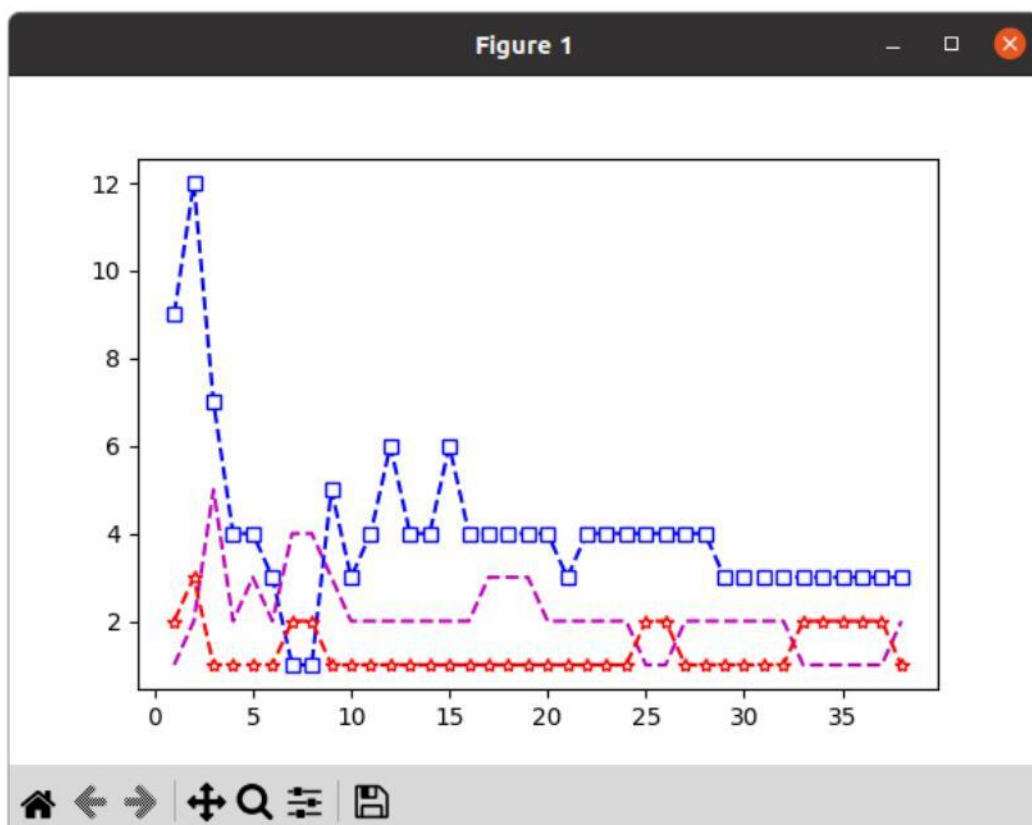
Utilizando el método `genfromtxt()` de `numpy` se leerán los datos de un archivo `.csv` para proceder a generar el gráfico correspondiente de los datos leídos del archivo.

```

script.py x
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  datos = np.genfromtxt( fname: 'datos_liga.csv', delimiter=',', skip_header=1)
5  print(datos)
6
7  jornada = datos[:, 0]
8  print("jornada = ", jornada)
9  print("datos[:,1] = ", datos[:,1])
10 print("datos[:,2] = ", datos[:,2])
11 print("datos[:,3] = ", datos[:,3])
12 plt.plot( *args: jornada, datos[:,1], 'bs',
13           jornada, datos[:,2], 'm',
14           jornada, datos[:,3], 'r*',
15           linestyle = '--', markerfacecolor = 'w')
16 plt.show()

```

Salida:



```
/home/usuario/PyCharmMiscProject/
[[ 1.  9.  1.  2.]
 [ 2. 12.  2.  3.]
 [ 3.  7.  5.  1.]
 [ 4.  4.  2.  1.]
 [ 5.  4.  3.  1.]
 [ 6.  3.  2.  1.]
 [ 7.  1.  4.  2.]
 [ 8.  1.  4.  2.]
 [ 9.  5.  3.  1.]
[10.  3.  2.  1.]
[11.  4.  2.  1.]
[12.  6.  2.  1.]
[13.  4.  2.  1.]
[14.  4.  2.  1.]
[15.  6.  2.  1.]
[16.  4.  2.  1.]
[17.  4.  3.  1.]
[18.  4.  3.  1.]
[19.  4.  3.  1.]
[20.  4.  2.  1.]
[21.  3.  2.  1.]
[22.  4.  2.  1.]
[23.  4.  2.  1.]
[24.  4.  2.  1.]
```

```
Run script x
jornada = [ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14. 15. 16. 17. 18.
           19. 20. 21. 22. 23. 24. 25. 26. 27. 28. 29. 30. 31. 32. 33. 34. 35. 36.
           37. 38.]
datos[:,1] = [ 9. 12.  7.  4.  4.  3.  1.  1.  5.  3.  4.  6.  4.  4.  6.  4.  4.  4.
              4.  4.  3.  4.  4.  4.  4.  4.  4.  4.  3.  3.  3.  3.  3.  3.  3.
              3.  3.]
datos[:,2] = [1.  2.  5.  2.  3.  2.  4.  4.  3.  2.  2.  2.  2.  2.  2.  3.  3.  3.  2.  2.  2.  2.
              1.  1.  2.  2.  2.  2.  2.  1.  1.  1.  1.  1.  2.]
datos[:,3] = [2.  3.  1.  1.  1.  1.  2.  2.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.
              2.  2.  1.  1.  1.  1.  1.  1.  2.  2.  2.  2.  2.  1.]
```

Fundamentos de Seaborn

Seaborn es una biblioteca de visualización de datos basada en Matplotlib que proporciona una interfaz de alto nivel para crear gráficos estadísticos atractivos y fáciles de interpretar. Diseñada para trabajar de forma eficiente con estructuras de datos como DataFrames de pandas, Seaborn simplifica la creación de gráficos complejos con pocas líneas de código. El API Seaborn fue creado para facilitar el análisis exploratorio de datos, permitiendo generar visualizaciones informativas con un diseño profesional sin necesidad de configurar manualmente cada aspecto del gráfico, entre los aspectos principales están los siguientes:

- a) Visualizar distribuciones de datos (como histogramas, KDE plots, boxplots).
- b) Explorar relaciones entre variables (scatter plots, line plots, heatmaps).
- c) Personalizar estilos y paletas de colores para mejorar la estética de los gráficos.

Matplotlib es la base sobre la que Seaborn construye, Matplotlib ofrece mayor flexibilidad pero también mayor complejidad. Seaborn, por su parte, está orientada a facilitar el trabajo del analista de datos, automatizando tareas comunes y mejorando la presentación visual de los resultados.

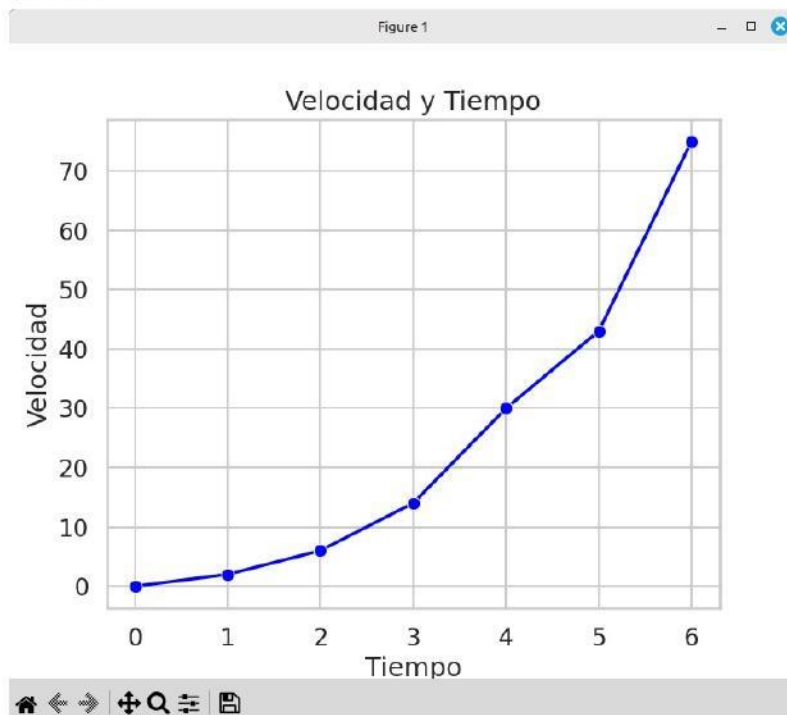
Instalación de Seaborn

Proceder a la instalación de Matplotlib

```
#pip install seaborn
```

```
script.py x
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 x = [0, 1, 2, 3, 4, 5, 6]
5 y = [0, 2, 6, 14, 30, 43, 75]
6
7 sns.set_theme(
8     style="whitegrid",
9     palette="deep",
10    context="talk",
11    font_scale=1
12 )
13
14 sns.lineplot(x=x, y=y, marker='o', color='blue')
15
16 plt.xlabel("Tiempo")
17 plt.ylabel("Velocidad")
18 plt.title("Velocidad y Tiempo")
19
20 plt.show()
```

Salida:

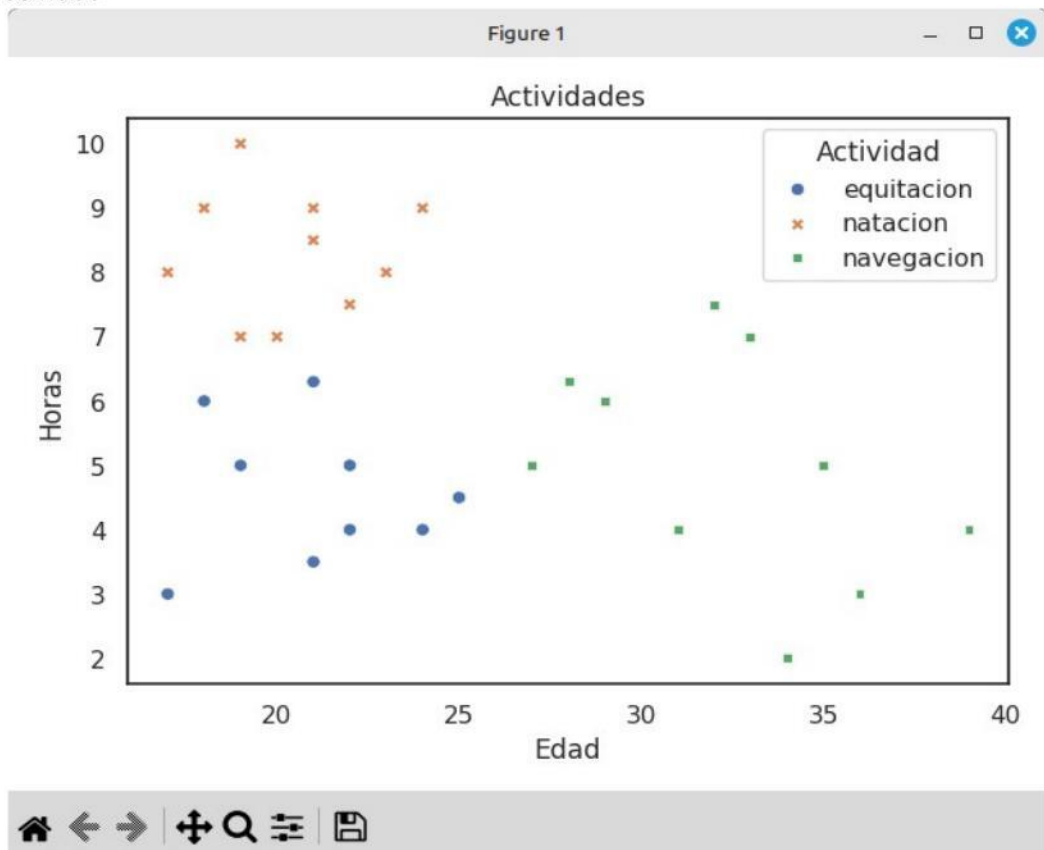



```

script.py x
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import pandas as pd
4
5 data = pd.DataFrame({
6     'Edad': [17, 18, 21, 22, 19, 21, 25, 22, 25, 24,
7             17, 18, 20, 19, 22, 21, 23, 19, 21, 24,
8             31, 28, 29, 36, 27, 32, 34, 35, 33, 39],
9     'Horas': [3, 6, 3.5, 4, 5, 6.3, 4.5, 5, 4.5, 4,
10             8, 9, 7, 10, 7.5, 9, 8, 7, 8.5, 9,
11             4, 6.3, 6, 3, 5, 7.5, 2, 5, 7, 4],
12     'Actividad': ['equitacion']*10 + ['natacion']*10 + ['navegacion']*10
13 })
14 print(data)
15 sns.set_theme(style="white")
16 sns.scatterplot(data=data, x='Edad', y='Horas', hue='Actividad', style='Actividad')
17
18 plt.xlabel('Edad')
19 plt.ylabel('Horas')
20 plt.title('Actividades')
21 plt.tight_layout()
22 plt.show()

```

Salida:



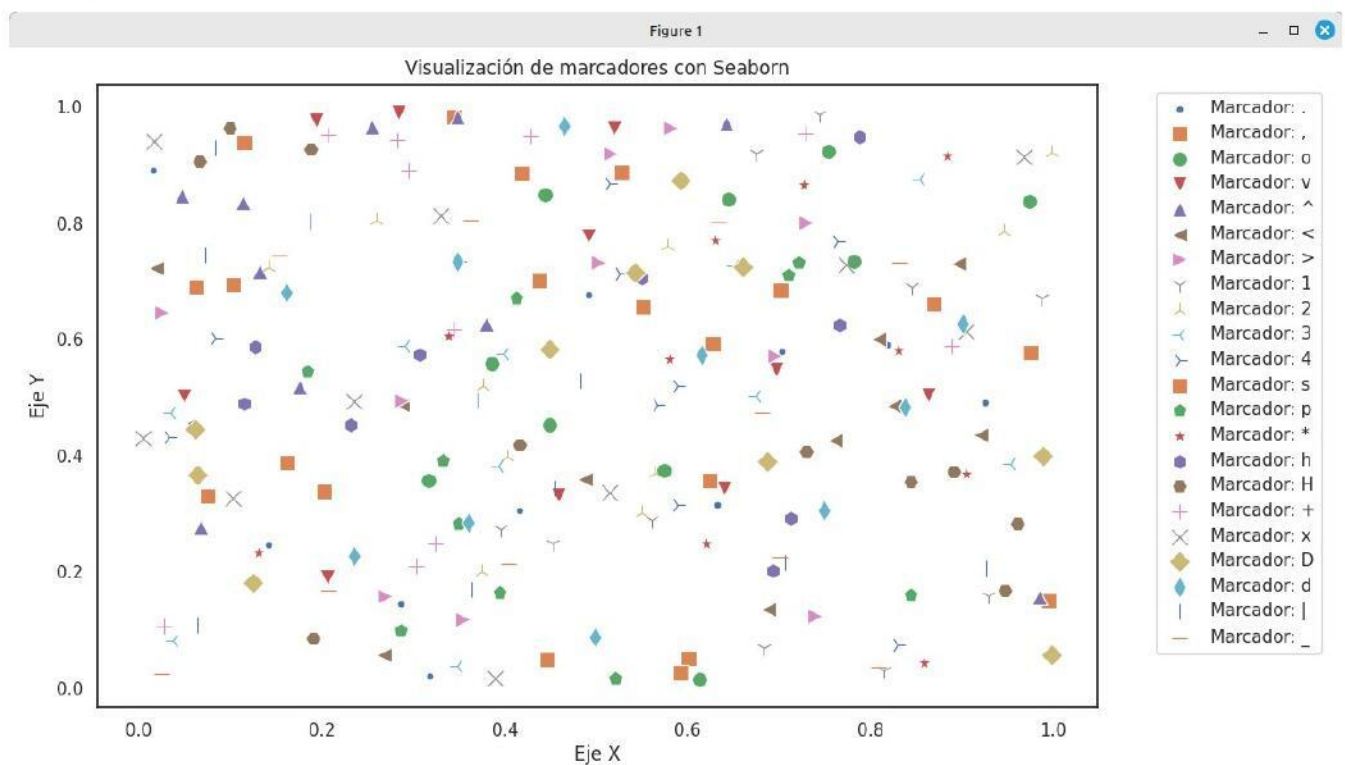
```
script x
/home/usuario/PyCharmMiscProject/
    Edad  Horas  Actividad
0      17    3.0  equitacion
1      18    6.0  equitacion
2      21    3.5  equitacion
3      22    4.0  equitacion
4      19    5.0  equitacion
5      21    6.3  equitacion
6      25    4.5  equitacion
7      22    5.0  equitacion
8      25    4.5  equitacion
9      24    4.0  equitacion
10     17    8.0   natacion
11     18    9.0   natacion
12     20    7.0   natacion
13     19   10.0   natacion
14     22    7.5   natacion
15     21    9.0   natacion
16     23    8.0   natacion
17     19    7.0   natacion
18     21    8.5   natacion
19     24    9.0   natacion
20     31    4.0  navegacion
21     28    6.3  navegacion
22     29    6.0  navegacion
```

script.py x

```

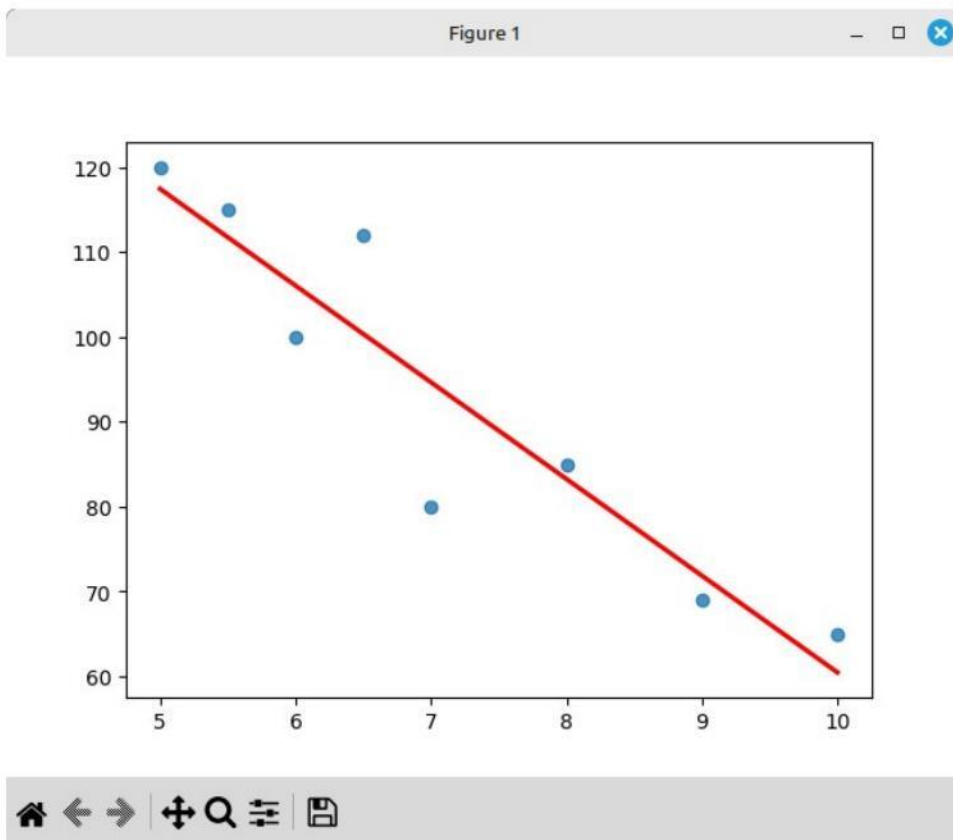
1  import numpy as np
2  import seaborn as sns
3  import matplotlib.pyplot as plt
4
5  sns.set_theme(style="white")
6
7  markers = ['.', ',', 'o', 'v', '^', '<', '>', '1', '2', '3', '4', 's', 'p', '*', 'h',
8            'H', '+', 'x', 'D', 'd', '|', '_']
9
10 plt.figure(figsize=(12, 8))
11
12 for marcador in markers:
13     x = np.random.rand(10)
14     y = np.random.rand(10)
15     sns.scatterplot(x=x, y=y, marker=marcador, s=100, label=f'Marcador: {marcador}')
16
17 plt.xlabel('Eje X')
18 plt.ylabel('Eje Y')
19 plt.title('Visualización de marcadores con Seaborn')
20 plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
21 plt.tight_layout()
22 plt.show()
23

```

Salida:

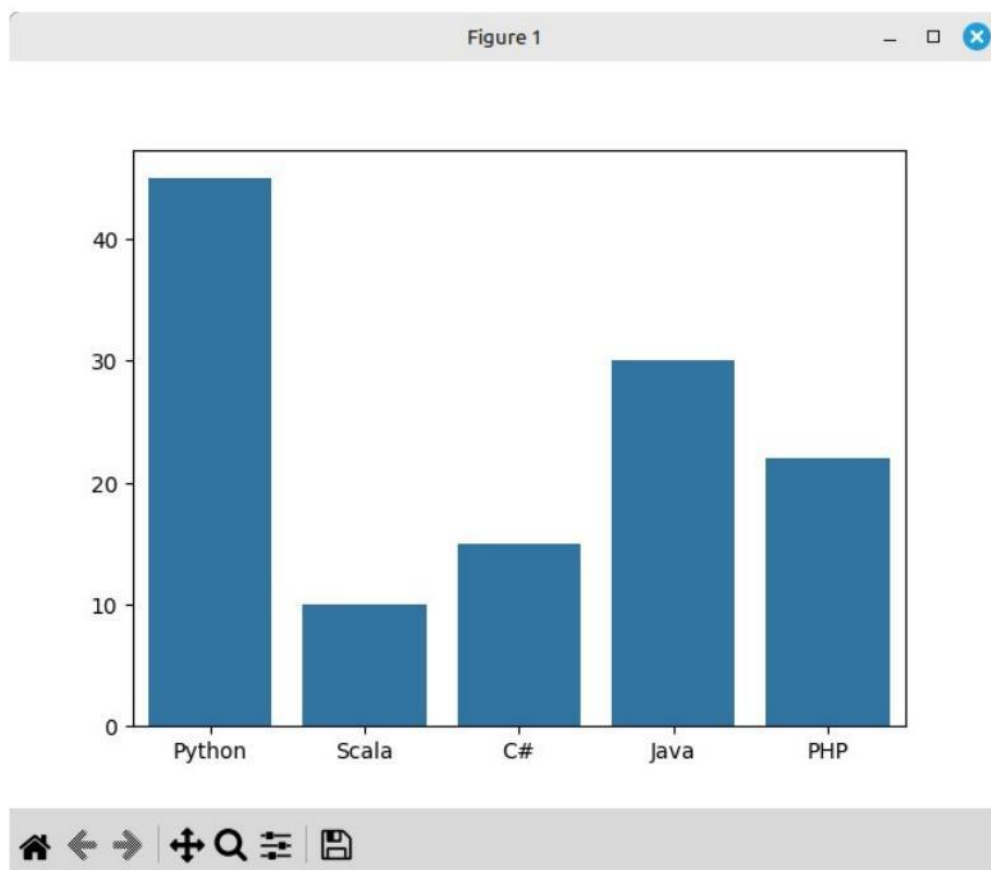
```
script.py x
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 x = [5, 5.5, 6, 6.5, 7, 8, 9, 10]
6 y = [120, 115, 100, 112, 80, 85, 69, 65]
7
8 sns.regplot(x=x, y=y, scatter=True, ci=None, line_kws={'color': 'red'})
9 plt.show()
10
```

Salida:



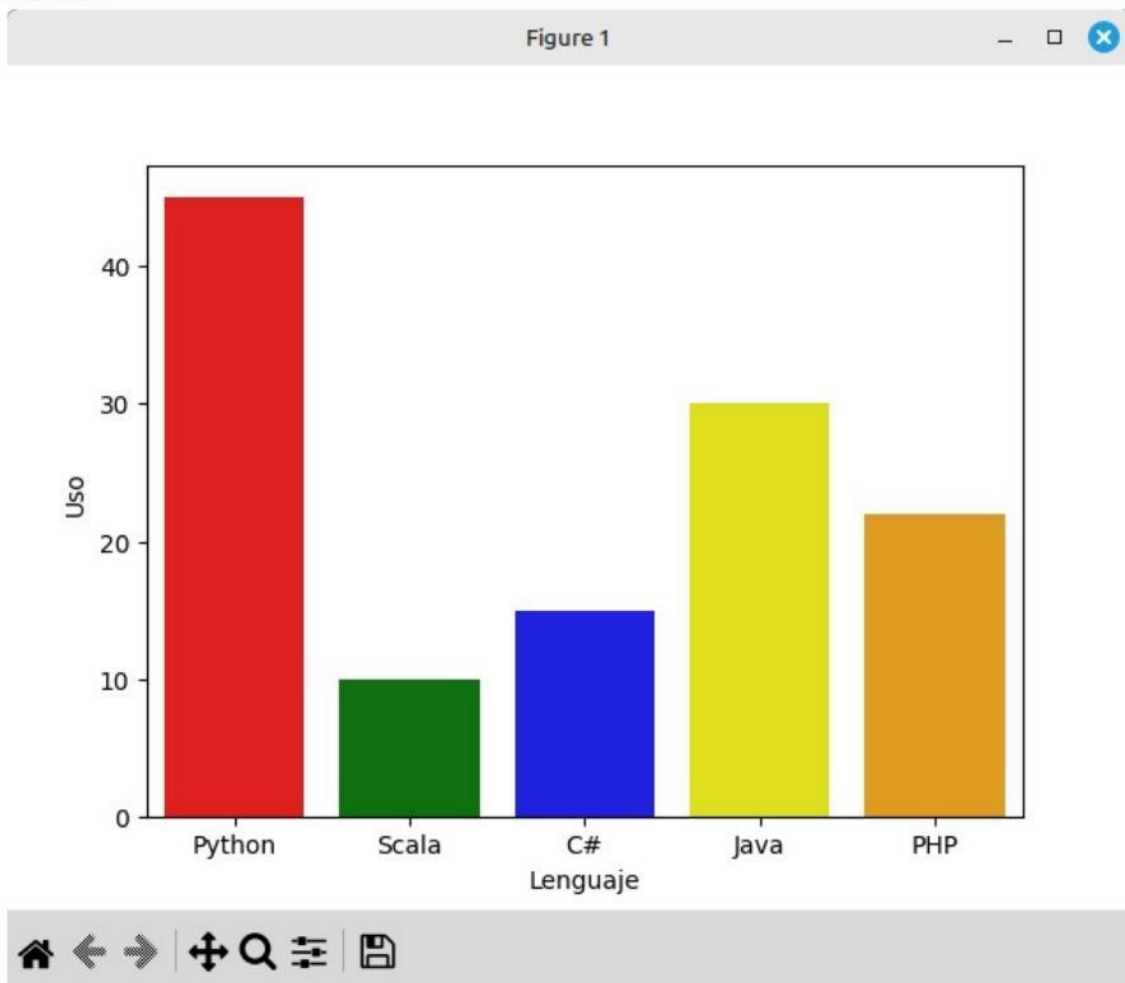
```
script.py x
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import pandas as pd
4
5 etiquetas = ['Python', 'Scala', 'C#', 'Java', 'PHP']
6 tamanios = [45, 10, 15, 30, 22]
7 datos = pd.DataFrame({'Lenguaje': etiquetas, 'Uso': tamanios})
8 print(datos)
9
10 sns.barplot(data=datos, x='Lenguaje', y='Uso')
11 plt.show()
```

Salida:



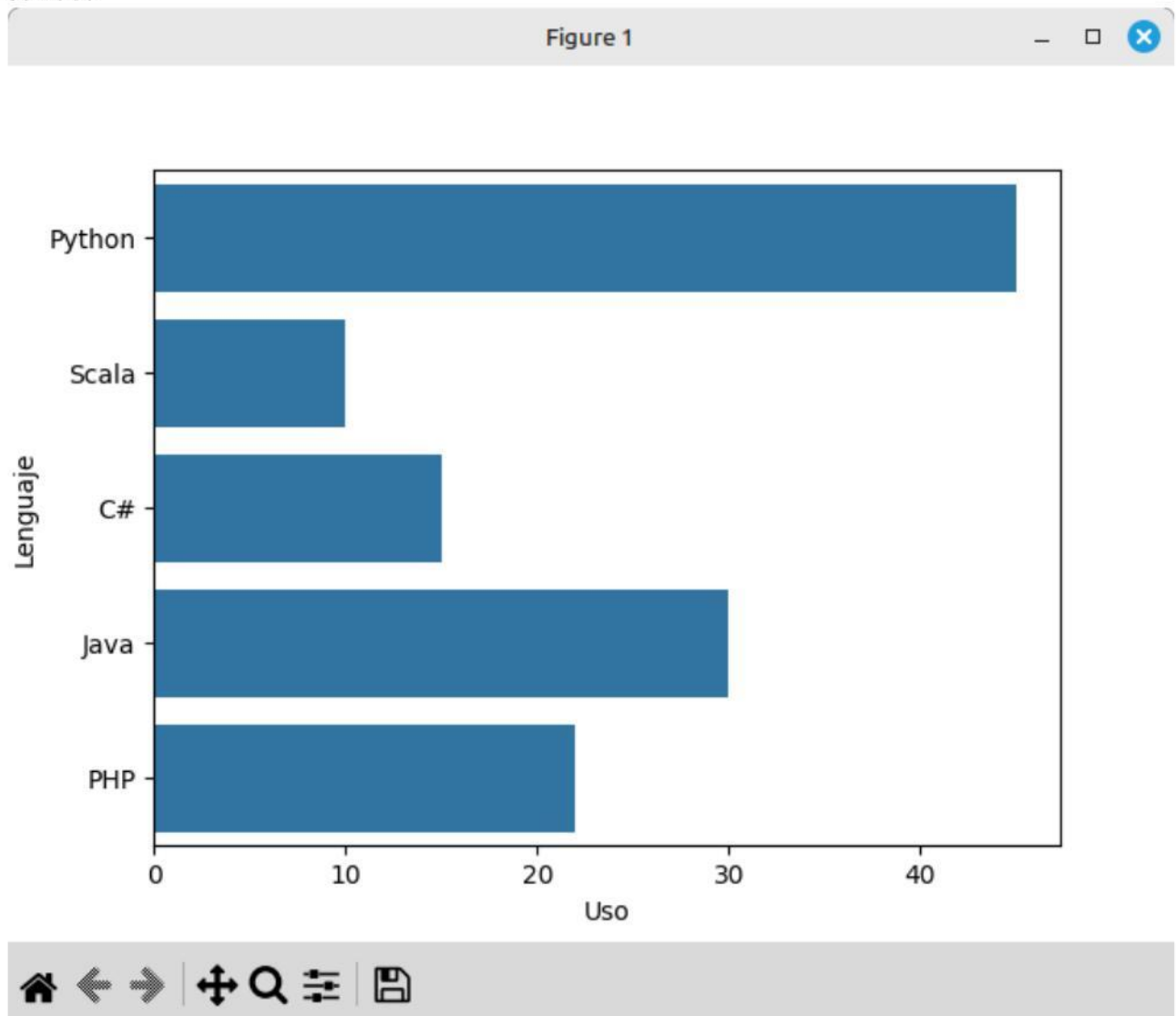

```
script.py x
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import pandas as pd
4
5 etiquetas = ['Python', 'Scala', 'C#', 'Java', 'PHP']
6 tamanios = [45, 10, 15, 30, 22]
7 colores = ['red', 'green', 'blue', 'yellow', 'orange']
8 datos = pd.DataFrame({'Lenguaje': etiquetas, 'Uso': tamanios})
9 print(datos)
10
11 sns.barplot(data=datos, x='Lenguaje', y='Uso', hue='Lenguaje',
12             palette=dict(zip(etiquetas, colores)), legend=False)
13 plt.show()
```

Salida:



```
script.py x
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import pandas as pd
4
5 etiquetas = ['Python', 'Scala', 'C#', 'Java', 'PHP']
6 tamanios = [45, 10, 15, 30, 22]
7 datos = pd.DataFrame({'Lenguaje': etiquetas, 'Uso': tamanios})
8
9 sns.barplot(data=datos, y='Lenguaje', x='Uso')
10 plt.show()
```

Salida:

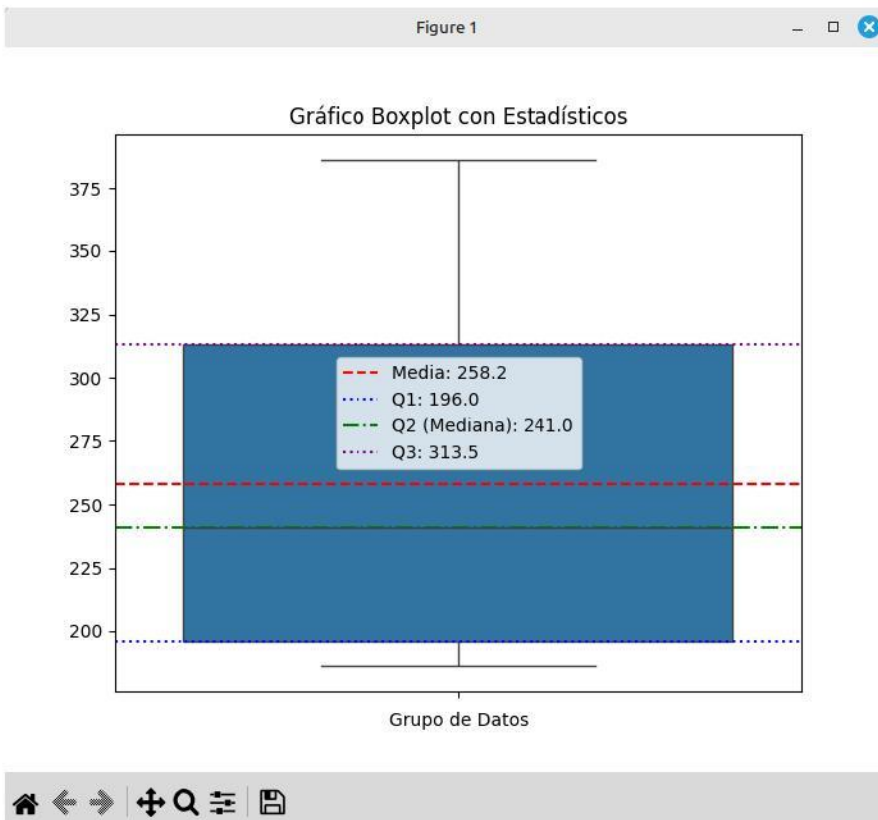


```

script.py x
1  import seaborn as sns
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  v = [186,196,206,216,226,236,246,256,266,276,286,296,306,316,326,
6      336,346,356,366,376,386,200,186,193,193,196,191,200,190,191]
7
8  q1 = np.percentile(v, q= 25)
9  q2 = np.percentile(v, q= 50)
10 q3 = np.percentile(v, q= 75)
11 mean = np.mean(v)
12
13 sns.boxplot(y=v)
14 plt.xlabel('Grupo de Datos')
15 plt.title('Gráfico Boxplot con Estadísticos')
16
17 plt.axhline(mean, color='red', linestyle='--', label=f'Media: {mean:.1f}')
18 plt.axhline(q1, color='blue', linestyle=':', label=f'Q1: {q1:.1f}')
19 plt.axhline(q2, color='green', linestyle='-.', label=f'Q2 (Mediana): {q2:.1f}')
20 plt.axhline(q3, color='purple', linestyle=':', label=f'Q3: {q3:.1f}')
21
22 plt.legend()
23 plt.show()

```

Salida:

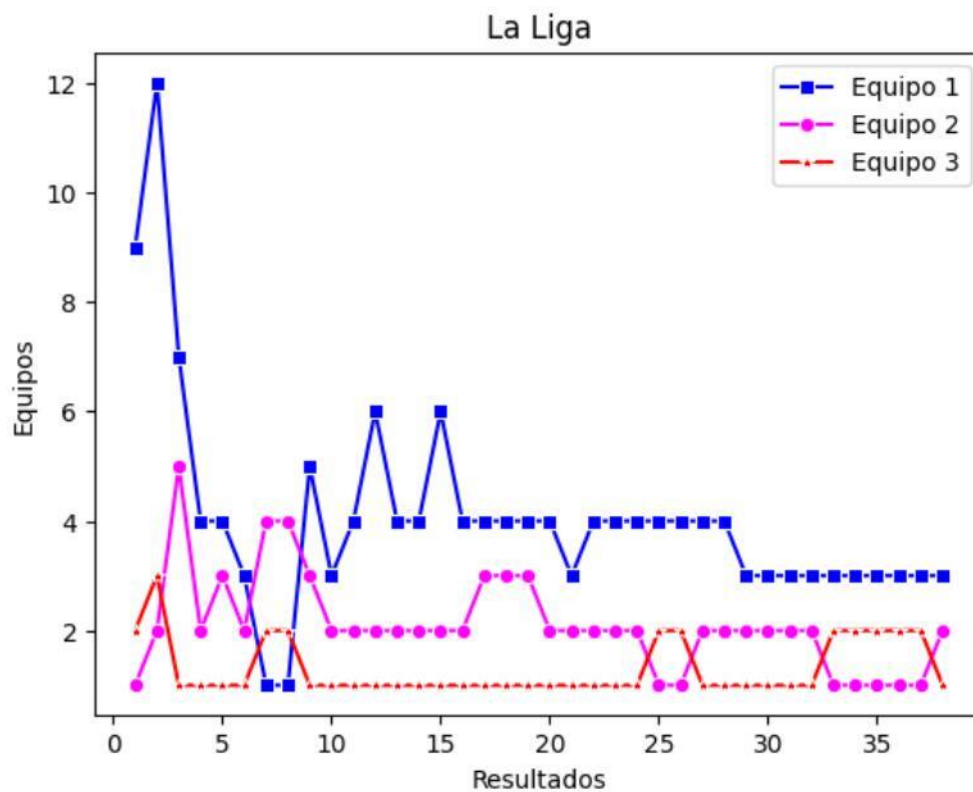


script.py

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3 import pandas as pd
4
5 datos = pd.read_csv('datos_liga.csv')
6 jornada = datos.iloc[:, 0]
7
8 sns.lineplot(x=jornada, y=datos.iloc[:, 1], marker='s', color='blue', label='Equipo 1')
9 sns.lineplot(x=jornada, y=datos.iloc[:, 2], marker='o', color='magenta', label='Equipo 2')
10 sns.lineplot(x=jornada, y=datos.iloc[:, 3], marker='*', color='red', label='Equipo 3')
11
12 plt.xlabel('Resultados')
13 plt.ylabel('Equipos')
14 plt.title('La Liga')
15 plt.legend()
16 plt.show()
```

Salida:

Figure 1



```

1 import matplotlib.pyplot as plt
2
3 puntajes = [126, 89, 90, 101, 102, 74, 93, 101, 66, 120, 108,
4 97, 98, 105, 119, 92, 113, 81, 104, 108, 83, 102, 105, 111, 102,
5 107, 103, 89, 89, 110, 71, 110, 120, 85, 111, 83, 122, 120, 102,
6 84, 118, 100, 100, 114, 81, 109, 69, 97, 95, 106, 116, 109, 114,
7 98, 90, 92, 98, 91, 81, 85, 86, 102, 93, 112, 76, 89, 110,
8 75, 100, 90, 96, 94, 107, 108, 95, 96, 96, 114, 93, 95, 117,
9 141, 115, 95, 86, 100, 121, 103, 66, 99, 96, 111, 110, 105, 110,
10 91, 112, 102, 112, 75]
11
12 mean = 100
13 std_dev = 15
14
15 plt.figure(figsize=(10, 6))
16 plt.hist(puntajes, bins=10, color='skyblue', edgecolor='black')
17 plt.axvline(mean, color='red', linestyle='solid', linewidth=2, label='Media (100)')
18 plt.axvline(mean - std_dev, color='red', linestyle='dashed', linewidth=1.5, label='-1 Desv. Estándar (85)')
19 plt.axvline(mean + std_dev, color='red', linestyle='dashed', linewidth=1.5, label='+1 Desv. Estándar (115)')
20
21 plt.title('Histograma de Puntajes')
22 plt.xlabel('Puntajes')
23 plt.ylabel('Frecuencia')
24 plt.legend()
25 plt.grid(True)
26 plt.show()

```

Salida:

