Librerías

Este cuaderno forma parte del curso de Iniciación a la programación con Python del programa de Open Course Ware (OCW) de la Universidad de La Laguna.

Librerías

Una librería es un conjunto de código predefinido con funciones que están diseñadas para realizar tareas específicas. Se importan en Python utilizando la palabra `import`. Se utiliza poniendo el nombre seguido de un punto. Por ejemplo:

```
import datetime #Librería que nos permite trabajar con fechas.

print("Fecha y hora actual:", datetime.datetime.now()) #Sale por
pantalla `Fecha y hora actual: `, seguido de la fecha y la hora en la
que se ejecute el código.

print("Hora actual:", datetime.datetime.now().time()) #Sale por
pantalla `Hora actual: `, seguido de la hora en la que se ejecute el
código.
```

También se puede utilizar la palabra `as` para importar una librería y llamarla como queramos. Por ejemplo:

```
import datetime as dt #Librería que nos permite trabajar con fechas.

print("Fecha y hora actual:", dt.datetime.now()) #Sale por pantalla
`Fecha y hora actual: `, seguido de la fecha y la hora en la que se
ejecute el código.

print("Hora actual:", dt.datetime.now().time()) #Sale por pantalla
`Hora actual: `, seguido de la hora en la que se ejecute el código.
```

Puedes probar este tema en la siguiente celda de código.

```
import random #Librería que ermite generar números aleatorios.

print("Número aleatorio:", random.randint(1, 10)) #Sale por pantalla
un número aleatorio entre 1 y 10, diferente cada vez que se ejecute.

print("Número decimal aleatorio:", random.random()) #Sale por pantalla
un número aletorio con decimales entre 0 y 1.
```

math

La librería *math* proporciona un conjuntos de funciones y constantes matemáticas para realizar operaciones avanzadas. Es una librería estándar de Python, por lo que no requiere instalación por separado. Por ejemplo:

```
import math

print("El ángulo 45 en radianes es:",math.radians(45)) #Sale por
pantalla `El ángulo 45 en radianes es: 0.7853981633974483`.

print("La raíz cuadrada de 146 es:",math.sqrt(146)) #Sale por pantalla
`La raíz cuadrada de 146 es: 12.083045973594572`.
```

Existen muchas más funciones y la documentación completa de esta librería puede consultarse en la página oficial: math - Mathematical functions.

Para la elaboración de este cuaderno se ha utilizado la última versión disponible de Python en el momento: 3.11.5.

Puedes probar este tema en la siguiente celda de código.

```
import math

print("El número pi es:", math.pi) #Sale por pantalla `El número pi
es: 3.141592653589793`.

n=8.723
print("Redondeo hacia arriba de",n,"es:",math.ceil(n)) #Sale por
pantalla `Redondeo hacia arriba de 8.723 es: 9`.
print("Redondeo hacia abajo de",n,"es:",math.floor(n)) #Sale por
pantalla `Redondeo hacia abajo de 8.723 es: 8`.
print("El logaritmo natural de",n,"es:",math.log(n)) #Sale por
pantalla `El logaritmo natural de 8.723 es: 2.1659632154510815`.
```

NumPy

NumPy, abreviación de Numerical Python, es una librería con la que es posible trabajar con arreglos (arrays) multidimensionales. Por ejemplo:

```
import numpy as np

array = np.array([1, 2, 3, 4, 5]) #Se crea un array.

print("Array original:", array) #Sale por pantalla `Array original: [1
2 3 4 5]`.
print("Doble del array:", array*2) #Sale por pantalla `Doble del
array: [ 2  4  6  8 10]`.
```

```
print("Suma del array más 10:", array+10) #Sale por pantalla `Suma del array más 10: [11 12 13 14 15]`.
```

Existen muchas más funciones y la documentación completa de esta librería puede consultarse en la página oficial: NumPy user guide.

Para la elaboración de este cuaderno se ha utilizado la última versión disponible en el momento: 1.26.

Puedes probar este tema en la siguiente celda de código.

```
import numpy as np

matriz = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) #Crea una matriz bidimensional.

print("Matriz:", matriz) #Sale por pantalla `Matriz: [[1 2 3]; [4 5 6]; [7 8 9]]`.
print("Suma total de la matriz:", np.sum(matriz)) #Sale por pantalla `Suma total de la matriz: 45`.
print("Suma por columnas:", np.sum(matriz, axis=0)) #Sale por pantalla `Suma por columnas: [12 15 18]`.
print("Suma por filas:", np.sum(matriz, axis=1)) #Sale por pantalla `Suma por filas: [ 6 15 24]`.
print("Matriz traspuesta:",matriz.T) #Sale por pantalla `Matriz traspuesta: [[1 4 7]; [2 5 8]; [3 6 9]]`.
```

Pandas

La librería Pandas se utiliza en entornos de análisis de datos, ya que ofrece estructuras de datos flexibles y funciones para manipular datos de manera eficiente. Es posible trabajar con diferentes tipos de datos como:

- Series: estructura unidimensional similar a las listas.
- Dataframe: estructura bidimensional donde los datos están organizados en filas y columnas.
- Paneles: estructura tridimensional. Obseletos desde la versión 0.20.0 de Pandas. No están disponibles desde la versión 0.25.0.

Por ejemplo:

```
import pandas as pd

nombres = ['Ana', 'Carlos', 'David'] #Lista con nombres.
edades = [25, 30, 28] #Lista con edades.
ciudades = ['Santa Cruz de Tenerife', 'Arona', 'La Guancha'] #Lista con ciudades.

df = pd.DataFrame({'Nombre': nombres, 'Edad': edades, 'Ciudad':
```

```
ciudades}) #Se crea un DataFrame.

print("Personas mayores de 27 años:\n", df[df['Edad'] > 27]) #Sale por
pantalla
#`Personas mayores de 27 años:
#    Nombre  Edad      Ciudad
#1  Carlos    30       Arona
#2   David    28  La Guancha`.
#'\n' es un salto de línea.
```

Existen muchas más funciones y la documentación completa de esta librería puede consultarse en la página oficial: User Guide - pandas 2.1.1.

Para la elaboración de este cuaderno se ha utilizado la última versión disponible en el momento: 2.1.1.

Puedes probar este tema en la siguiente celda de código.

```
import pandas as pd

coches = {
    'Marca': ['Volkswagen', 'Audi', 'Seat', 'Skoda'],
    'Modelo': ['Golf', 'A4', 'Ibiza', 'Octavia'],
    'Año': [2021, 2020, 2022, 2021],
} #Datos de diferentes coches.

df_coches = pd.DataFrame(coches) #Se crea el DataFrame de los coches.

propietarios = {
    'Marca': ['Volkswagen', 'Seat', 'Audi'],
    'Propietario': ['Juan', 'Maria', 'Pedro'],
    'Ciudad': ['Santa Cruz de Tenerife', 'Candelaria', 'El Sauzal'],
} #Datos de diferentes propietarios.

df_propietarios = pd.DataFrame(propietarios) #Se crea el DataFrame de
los propietarios.

df_completo = pd.merge(df_coches, df_propietarios, on='Marca',
how='inner') #Se combinan los dos DataFrame. Se debe seleccionar una
clave única, en este caso, la marca.

print("Datos completo:\n",df_completo) #Sale por pantalla el DataFrame
combinado completo.
#`Datos completo:
#        Marca Modelo   Año Propietario                  Ciudad
#0  Volkswagen   Golf  2021        Juan  Santa Cruz de Tenerife
#1        Audi     A4  2020       Pedro               El Sauzal
#2        Seat  Ibiza  2022       Maria             Candelaria`.

print("Personas con un coche de 2021 o más nuevo:\n",
```

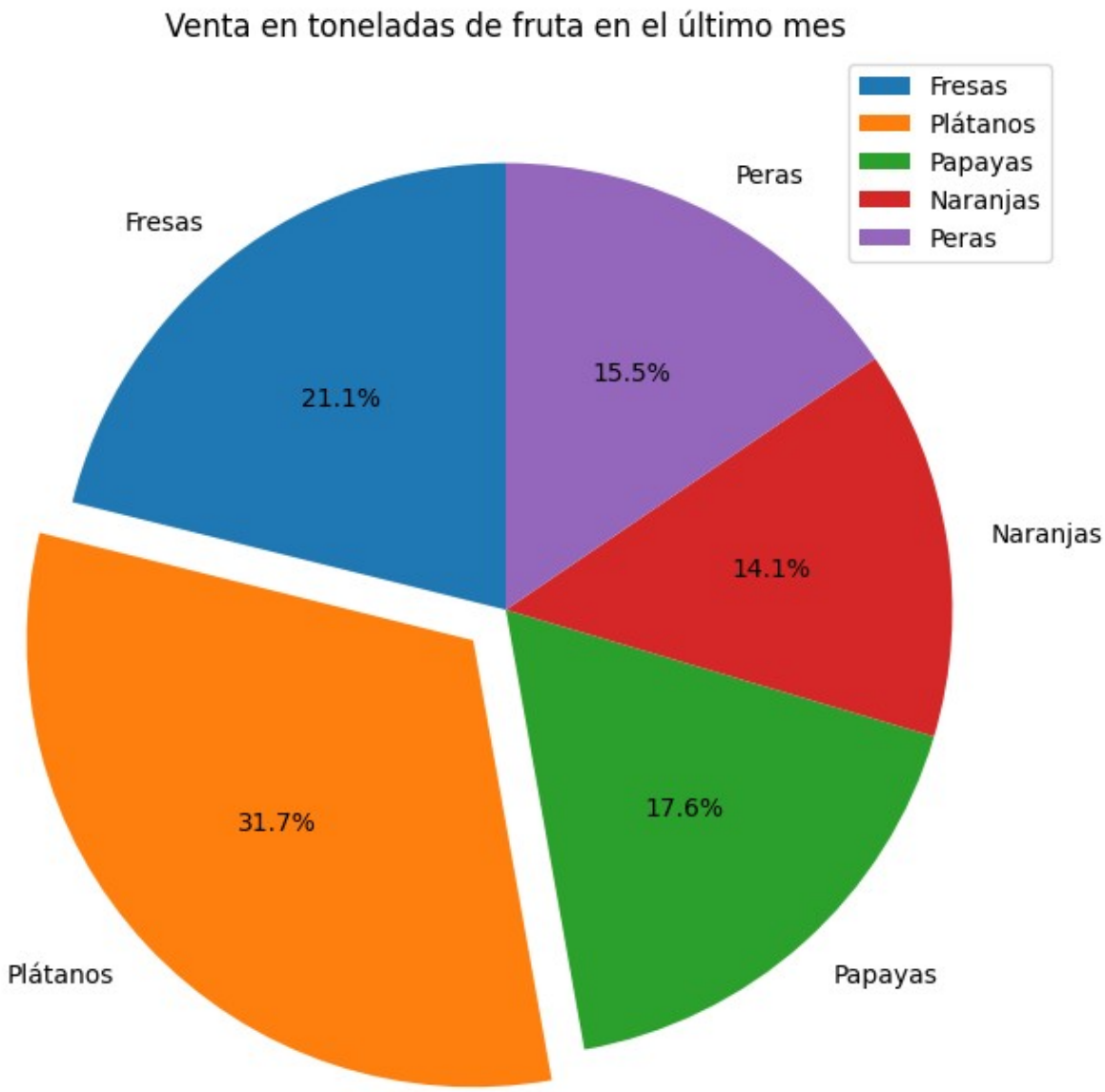
```
df_completo[df_completo['Año'] >= 2021]) #Sale por pantalla
#`Personas con un coche de 2021 o más nuevo:
#        Marca Modelo   Año Propietario                 Ciudad
#0  Volkswagen   Golf  2021        Juan  Santa Cruz de Tenerife
#2        Seat  Ibiza  2022       Maria              Candelaria`.
```

matplotlib

matplotlib es una librería con la que es posible visualizar datos, ya que permite crear gráficos de manera sencilla y muy potente. Por ejemplo:

```
import matplotlib.pyplot as plt

frutas = ['Fresas', 'Plátanos', 'Papayas', 'Naranjas', 'Peras']
#Nombre de frutas.
ventas = [30, 45, 25, 20, 22] #Toneladas de venta.
explode = (0, 0.1, 0, 0, 0) #Separamos la categoría de 'Plátanos'.

plt.figure(figsize=(8, 8))
plt.pie(ventas, labels=frutas, autopct='%1.1f%%', startangle=90,
explode=explode) #Se crea un gráfico de pastel, con un tamaño de 8x8,
con etiquetas de datos y de las frutas.

plt.title('Venta en toneladas de fruta en el último mes') #Título del
gráfico.
plt.legend(frutas, loc='best') #Se incluye leyenda.

plt.show() #Se muestra el gráfico.
```

El resultado es el siguiente:



Es posible crear gráficos de diferentes tipos, como barras, histogramas, pastel, dispersión, de caja, o de violín, entre otros. Existen muchos más, además de más funciones. La documentación completa de esta librería puede consultarse en la página oficial: Matplotlib 3.8.0 documentation.

Para la elaboración de este cuaderno se ha utilizado la última versión disponible en el momento: 3.8.0.

Puedes probar este tema en la siguiente celda de código.

```
import pandas as pd
import matplotlib.pyplot as plt

data = {
```

```
    'Categoría': ['Camisetas', 'Pantalones', 'Vestidos', 'Chaquetas',
'Zapatos'],
    'Cantidad': [100, 75, 50, 40, 60]
} #Datos de inventario de ropa

df_inventario = pd.DataFrame(data) #Se crea un DataFrame con los datos

plt.figure(figsize=(8, 5))
plt.barh(df_inventario['Categoría'], df_inventario['Cantidad'],
color='skyblue') #Se crea un gráfico de barras horizontal, de tamaño
8x5, con las barras de color azul cielo.

plt.xlabel('Cantidad en Stock') #Etiqueta del eje x.
plt.ylabel('Categorías de Ropa') #Etiqueta del eje y.
plt.title('Inventario de Ropa en el Almacén') #Título del gráfico.

plt.gca().invert_yaxis()  #Se invierte el eje y para que 'Camisetas'
sea la primera barra.
plt.show() #Se muestra el gráfico.
```

pip

pip es una herramienta con la que poder gestionar paquetes en Python, y el cual se utilizar para instalarlos, actualizarlos y administrarlos. Google Colab ya incluye algunos paquetes preinstalados, pero si es necesario instalar alguno en este o en otros entornos, es posible hacerlo ejecutando el comando `!pip`. Este gestor se encuentra incluido en Python desde la version 3.4. Por ejemplo:

```
!pip install ping3
import ping3 #Paquete no incluido entre los estándar que permite
realizar ping a un servidor.

web = "www.google.com" #Web a la que queremos hacer ping.
print("Tiempo de respuesta de Google:", ping3.ping(web),"ms") #Sale
por pantalla `Tiempo de respuesta de Google: ...`.
```