
Universidad
Tecmilenio®

Universidad Tecmilenio

Visualización y programación en Python

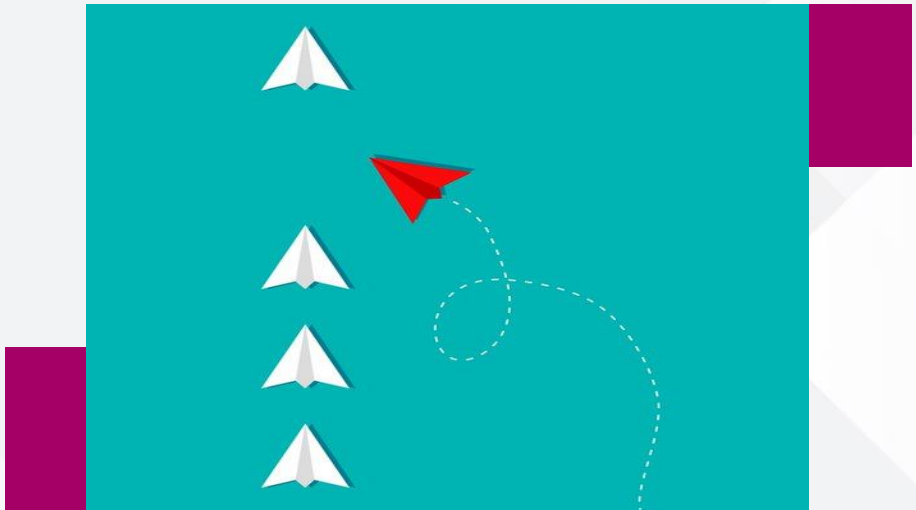
Tema 3.
Ciclos en Python

Introducción

El control del flujo en un programa Python es una de las claves para que cumpla con su propósito.

Una de las claves son los ciclos o bucles, que sirven para repetir ciertas instrucciones a ciertos datos, determinado el número de veces o hasta que una condición particular se cumpla.

Su uso permite hacer cálculos para varios datos o bien revisar los datos y transformarlos dentro de una lista o arreglo de datos y así con el ciclo recorrer toda la información para aplicar a todos los datos el mismo campo o cálculo.



Universidad Tecmilenio

Ciclo for, función range y función enumerate.

Uno de los ciclos más conocidos en la programación de sistemas es ***for***, se caracteriza por iterar (realizar) cierto número de elementos o valores en cierta secuencia.

Ésta es su sintaxis:
for <variable> in <iterable>:
<Código>

Figura 1. Diagrama del ciclo For

Explicación

Universidad Tecmilenio

Las iteraciones más comunes con el ciclo **for** son las que van de **0 a n**, en esos casos es donde podemos usar la **función range()** ya que genera una secuencia de números que pueden ir desde cero hasta el número que se pase como parámetro, excepto 1, esta es su sintaxis:

range ([inicio], fin, [paso])

Donde "**inicio**" quiere decir desde qué número se quiere iniciar, **fin** hasta qué número se quiere llegar y **paso** el numero a avanzar o sumar.

Tabla 2. Ejemplos del uso de range

range (6)	0, 1, 2, 3, 4, 5, 6
range (5, 20, 2)	5, 7, 9, 11, 13, 15, 17, 19
range (5, 0, -1)	5, 4, 3, 2, 1

Explicación

En este ejemplo estamos también usando la función len, que nos deja saber la longitud de una lista, es decir, el número de elementos que tiene.

Al usar **len y range**, no solo podemos obtener cada valor que hay dentro de la lista, sino también el número que tiene en ella.

```
#Se define una lista de elementos de texto
prestaciones = ["vacaciones", "seguros", "cursos"]
#En este ciclo for se recorre la lista
#se ejecuta la línea de código por cada elemento
for nelemento in range(len(prestaciones)):
  #Imprime en pantalla número de elemento
  #y el elemento que tiene en memoria.
  print(nelemento, prestaciones[nelemento])
```

Cómo hacer un comportamiento similar con la función enumerate, la cual tiene esta sintaxis:

```
enumerate(<iterable>, start = 0)
```

Explicación

Universidad Tecmilenio

Figura 2. Diagrama del ciclo while

Otro tipo de ciclo que podemos usar en Python es **while**, que en español significa "**mientras**", el número de iteraciones que se hará en while el ciclo se ejecutará mientras la condición o condiciones que coloquemos sean verdaderas.

while <condicion>:

Explicación

Universidad Tecmilenio

Operaciones en ciclos: break, continue y else

Existen maneras de irrumpir en los ciclos sacando del mismo o brincando a la siguiente iteración según sea necesario con la instrucción **break**. Lo que hace es salirse del ciclo o bucle en el que está en el momento, afectando solo ese bucle en particular y se puede usar tanto en **for** como en **while** A cada ciclo del ejemplo le corresponde su **break. Ejemplo:**

Tabla 6. Uso de break en ciclos for y while.

```
s = input("Escriba algo o de enter para terminar: ")
while s != "" :
    if s.isnumeric():
      print (s, " es un número")
      break
      #Este break sale del while si es un número
    else:
      vocales = 0
      otros = 0
      for letra in s:
        if letra in ("a","e","i","o","u"):
          vocales+=1
        elif letra==" ":
          break
          #Este break sale del for al encontrar un espacio.
        else:
          otros+=1
      print (s, " tiene ", vocales, "vocales en la primera palabra")
    s = input("Escriba algo o de enter para terminar: ")
```

Explicación

Universidad Tecmilenio

Otra instrucción que puede modificar el curso de un ciclo es **continue**, esta instrucción lo que logra es pasarse a la siguiente iteración desde el punto que se coloca, puede servir para saltarnos un valor u omitir el ciclo para ciertas condiciones, de la misma manera se requiere uno por ciclo y lo que define de qué ciclo es el bloque de código en el que está colocado.

Tabla 7. Uso de continue en ciclos for y while.

```
s = input("Escriba algo o de enter para terminar: ")
while s != "" :
    if s.isnumeric():
      print (s, " es un número")
      s = input("Escriba algo o de enter para terminar: ")
      #se tuvo que agregar esta línea para que no quede en ciclo infinito
.
      continue
      #Este continue vuelve al inicio del while si es un número.
    else:
      vocales = 0
      otros = 0
      for letra in s:
        if letra in ("a","e","i","o","u"):
          vocales+=1
        elif letra==" ":
          continue
          #Este continue vuelve al inicio del for al encontrar un espacio
        else:
          otros+=1
      print (s, " tiene ", vocales, "vocales.")
      #Ahora el conteo si es del total de vocales
    s = input("Escriba algo o de enter para terminar: ")
```

Explicación

Universidad Tecmilenio

Justamente con el ciclo while podemos usar también la sentencia else, de manera similar a con el if. Es decir, que en el ciclo podemos tener algo qué hacer antes de terminar con el ciclo como últimos comandos.

En este notebook de Google colab puedes ver todo el código desarrollado en el tema:

https://colab.research.google.com/drive/1LB82cDfrufbdpEc7yGuObpO5elJBAEpb?usp=sharing

Explicación

Universidad Tecmilenio

Ejemplo de código

Objetivo: Realizar un **ejemplo de código en lenguaje python** donde se le pidan dos números al usuario y el programa arroje un mensaje: si este es mayor, menor o igual que, al finalizar el análisis, el usuario podrá elegir si salir o repetir el proceso.

Instrucciones:

- Para realizar esta actividad es importante que consultes el contenido del tema.
- Con base en lo aprendido elabora un código donde el programa solicite el ingreso de dos números diferentes y los compare.
- Incluye los operadores lógicos vistos en las lecciones anteriores y los ciclos para el correcto funcionamiento del programa y realiza un diagrama de flujo que explique la lógica que seguirá tu código.
- Agrega comentarios donde se explique el proceso que va siguiendo tu código.
- Agrega capturas del proceso llevado a cabo y el funcionamiento final.

Actividad

¿Te imaginas el uso que puedes dar a estos ciclos para los proyectos que puedas realizar?

Universidad Tecmilenio

Visualización y programación en Python

Tema 4.
Uso de gráficos en Python.

Introducción

Universidad Tecmilenio

Es común que regularmente se necesite en la programación Python mostrar la información de manera gráfica de tal forma que sea más fácil de comprender y a la vez que sea representativa.

Existen gráficos o visualizaciones que favorecen la exploración y otros gráficos que favorecen la presentación de resultados.

Universidad Tecmilenio

Matplotlib es una de las librerías de Python más robustas en cuanto a gráficos:

VENTAJAS

- Permite personalizar casi cualquier factor que se considere importante.
- Es ampliamente utilizado.
- Otros paquetes se basan en él.

DESVENTAJAS

- No es sencillo de aprender, por su variedad de opciones.
- Pudiera generar muchas líneas de código para una sola gráfica.
- Si lo que se busca es tener un gráfico interactivo, ésta no es opción.

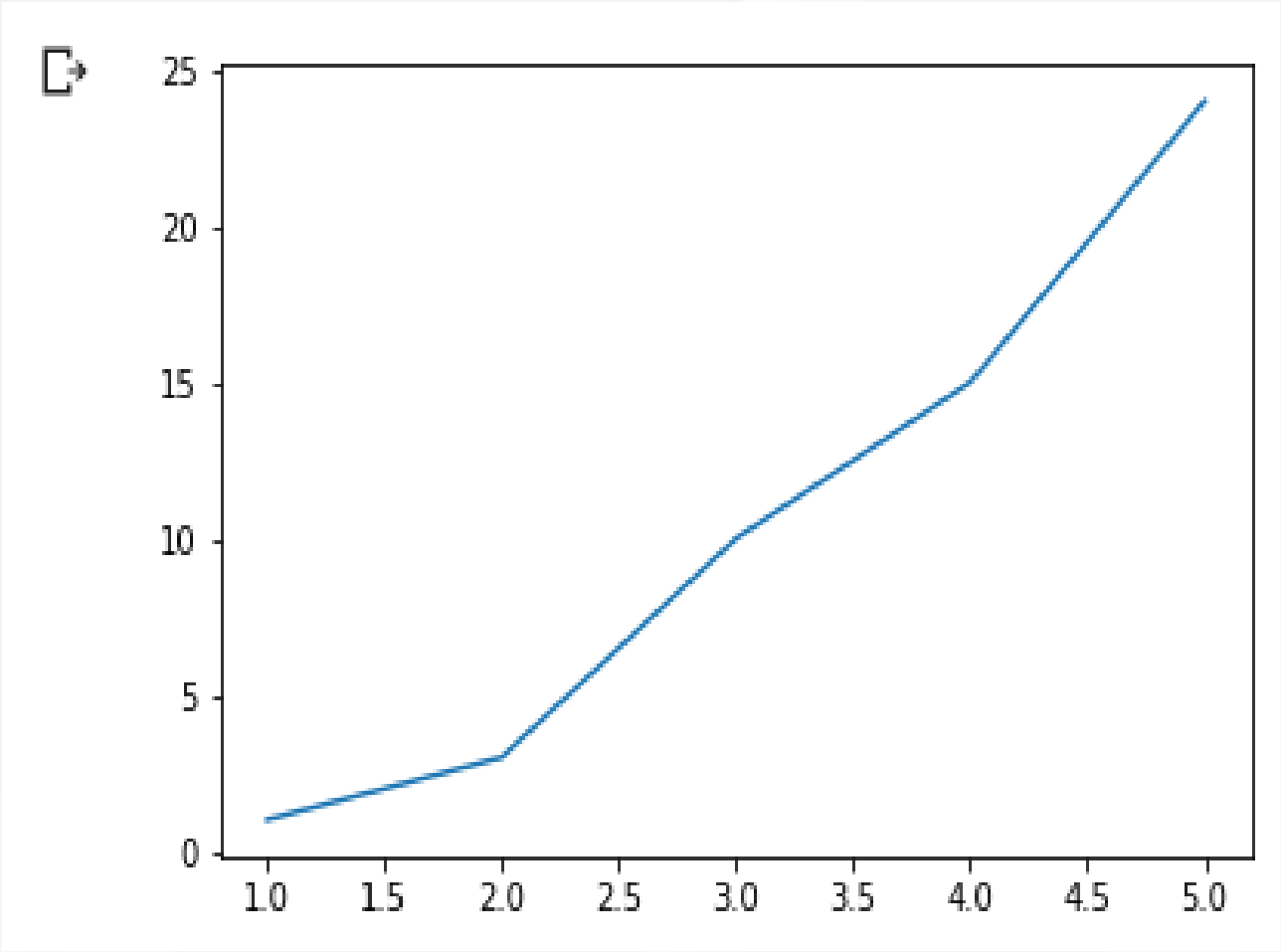
Explicación

Universidad Tecmilenio

Tabla 1. Uso del simple de la librería Matplotlib creando un gráfico

```
#Primer paso es importar las librerías necesarias.
import matplotlib as mpl
import matplotlib.pyplot as plt
#Segundo paso es tener los datos a graficar.
x = [1,2,3,4,5]
y = [1,3,10,15,24]
#Tercer paso dar los datos y graficar.
plt.plot(x,y)
plt.show()
```

Figura 1. Gráfica de líneas realizada con Matplotlib.



Fuente: Mulla, R. (2022). *ALL Python Data Visualization Libraries in 2022*. Recuperado de https://www.kaggle.com/code/robikscube/all-python-data-visualization-libraries-in-2022/notebook

Explicación

Universidad Tecmilenio

Otro estilo de los más usados de Matplotlib es el grafico de **dispersión**, scatter, que permite ver sobre todo cómo se agrupan y dispersan los datos visualmente.

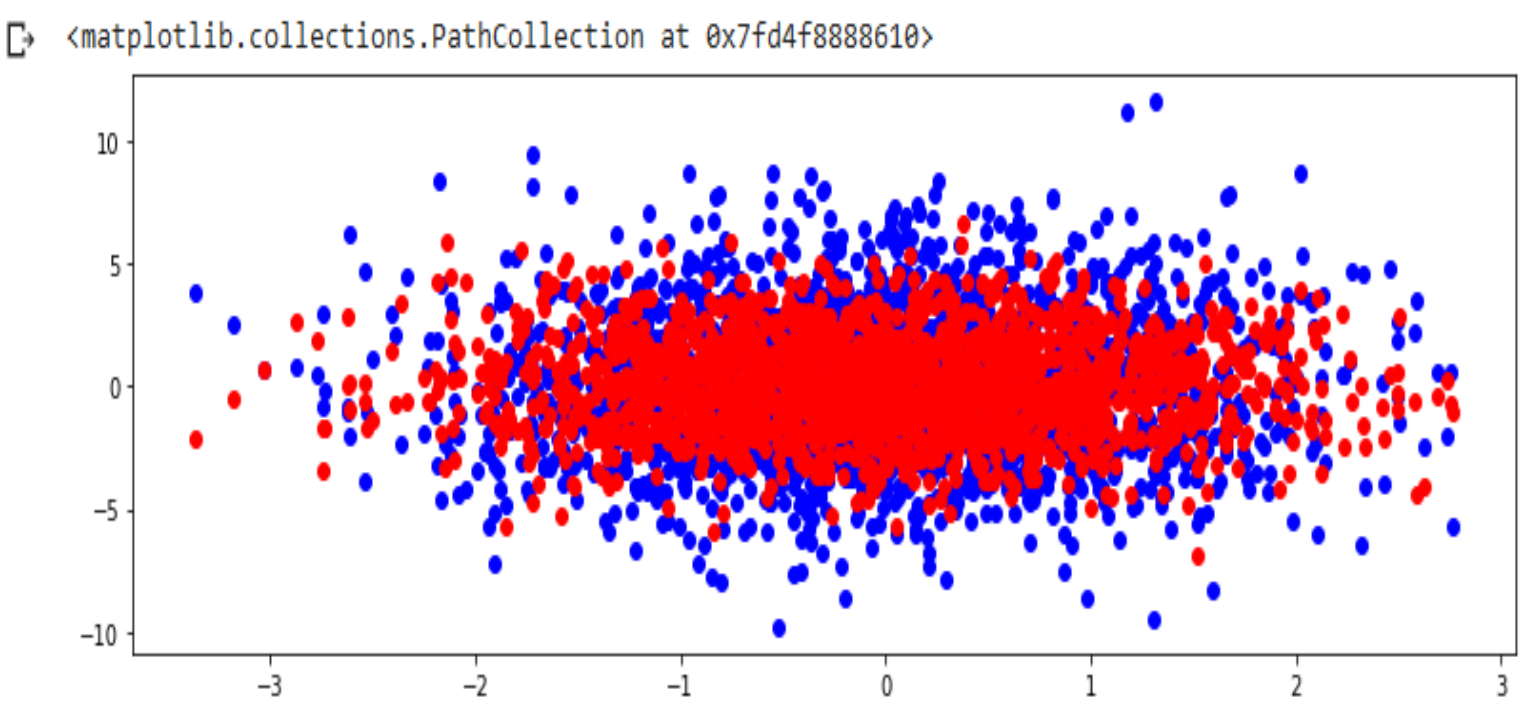
Tabla 2. Uso de la librería Matplotlib creando un gráfico de dispersión.

```
#Primer paso, es importar las librerías necesarias.
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
#Segundo paso es tener los datos a graficar.
n = 2048
X = np.random.normal(0,1,n)
Y = np.random.normal(0,3,n)
Z = np.random.normal(0,2,n)
#Tercer paso dar los datos y graficar.
f, ax = plt.subplots(figsize=(13, 4))
#Se elige un color para graficar cada serie.
plt.scatter(X,Y,c="blue")
plt.scatter(X,Z,c="red")
```

Explicación

Universidad Tecmilenio

Figura 2. Gráfica de dispersión realizada con Matplotlib.



Explicación

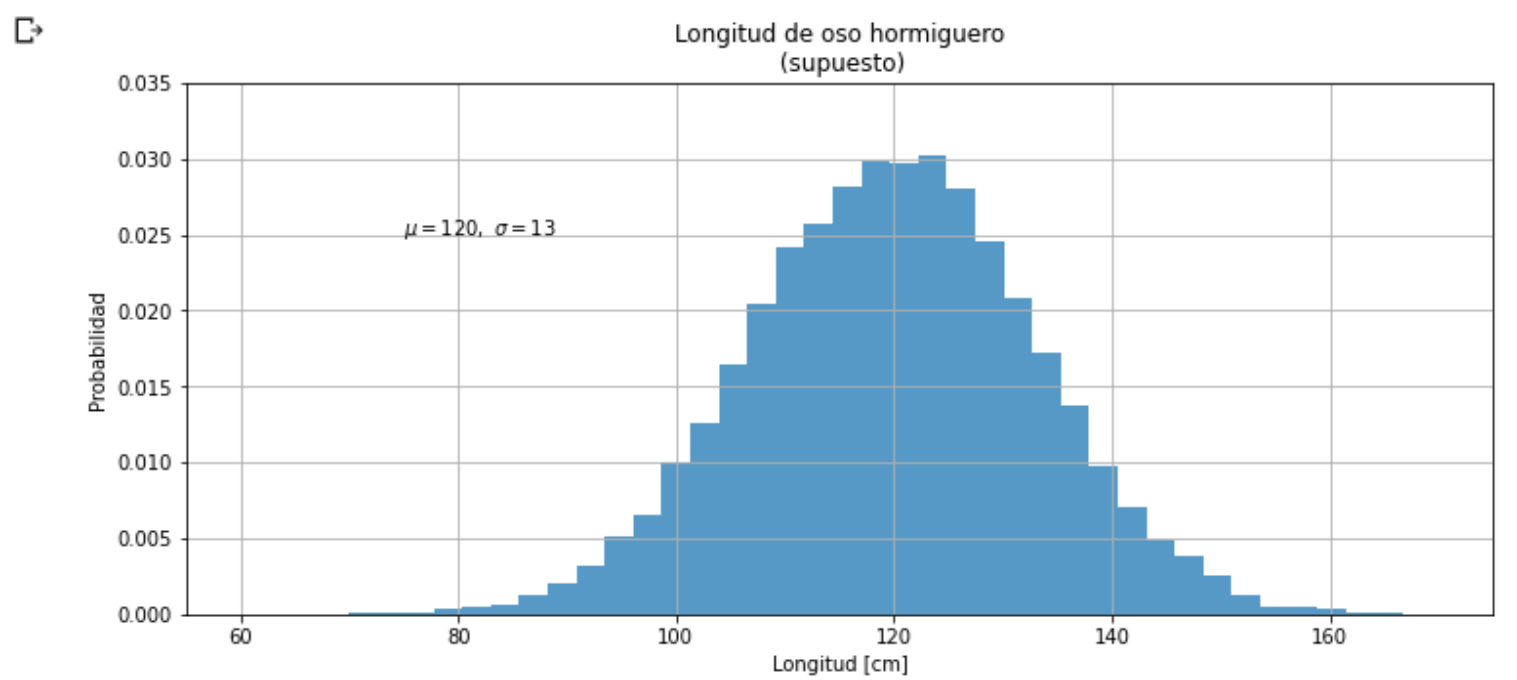
Universidad Tecmilenio

Otro gráfico comúnmente usado por la estadística es el **histograma**, que permite visualizar la forma de los valores proporcionados o bien su distribución, es un gráfico popular estadísticamente hablando, ya que ayudan a ver tanto la concentración de datos, como la extensión y su forma, además de con ello verificar su normalidad.

Tabla 3. Uso de la librería Matplotlib creando un histograma.

```
#Primer paso es importar las librerías necesarias.
import matplotlib as mpl
import matplotlib.pyplot as plt
#Segundo paso generamos los datos a graficar.
mu, sigma = 120, 13
x = mu + sigma * np.random.randn(15000)
fig, ax = plt.subplots(figsize=(12, 5))
#Tercer paso generamos el histograma.
n, bins, patches = ax.hist(x, 40, density=1, facecolor="C0", alpha=0.75)
#Cuarto paso definimos etiquetas y medidas a mostrar en gráfico.
ax.set_xlabel("Longitud [cm]")
ax.set_ylabel("Probabilidad")
ax.set_title("Longitud de oso hormiguero\n (supuesto)")
ax.text(75, 0.025, r"$\mu=120,\ \sigma=13$")
ax.axis([55, 175, 0, 0.035])
ax.grid(True)
```

Figura 3. Histograma realizado con Matplotlib



Bins se refiere al número de contenedores (barras) que tendrá el histograma y usar un **sigma** para la generación de datos que se está determinando a que siga esa forma de comportamiento de los datos.

Explicación

Explicación

Universidad Tecmilenio

Pandas es una librería en Python que además de graficar también puede dar apoyo en la limpieza o mejora de los datos, siendo esta parte a veces la que lleva más tiempo en los proyectos:

- **Remover *Outliers***
- **Trabajar datos Nulos**
- **Eliminar o transformar datos erróneos e irrelevantes**

Tabla 4. Funciones para mejora de datos con librería Pandas

```
#Paso cero, se crea archivo y agrega datos.
#Incluye datos nulos.
d = open("data.csv", "w")
d.write("Id,nombre,edad,salario,asistencia\n")
d.write("1,Miguel Hidalgo,55,1500,8\n")
d.write("2,Josefina Ortiz,25,,9\n")
d.write("3,Luis Perez,,1500,8\n")
d.write("4,Maria Lopez,25,1000,\n")
d.write("5,Rosa Hernandez,500,1000,6\n")
d.close()
#Paso uno, importa la librería y se cargan los datos en variable.
import pandas as pd
data=pd.read_csv("data.csv")
#Se imprime en pantalla lo que contiene la variable.
data
#Explora la función isnull para revisar datos nulos.
data.isnull()
#Explora la función notnull para revisar datos existentes.
data.notnull()
#Usa la función dropna para borrar renglones con nulos.
data2 = data.dropna()
data2
#Usa la función drop para borrar un renglón.
renglon=data2[data2["edad"]==500].index
data3=data2.drop(renglon)
data3
#Usa la función replace para cambiar valores.
data3["edad"]=data3["edad"].replace(55, 50)
data3
#Permite ver lo que contiene la variable.
data3.describe
```

Universidad Tecmilenio

Luego de observar los datos en gráficas, con estos comandos podemos visualizar los datos que vienen nulos: **describe, isnull, isnotnull.**

Para posteriormente tomar las acciones pertinentes como:

- Eliminar todos los renglones con datos nulos como con la función **dropna**.
- Borrar con la función **drop** aquellos renglones que cumplan con las condiciones que se proporcionen.
- Otra opción es reemplazar ciertos valores con la función **replace**.

Se pueden utilizar dentro de ciclos para que se ejecute en toda la información o en los renglones que se requieran. De esta manera dejamos los datos más limpios y objetivos para el análisis.

Explicación

La grafica más simple que podemos hacer es la de líneas, como en el siguiente ejemplo:

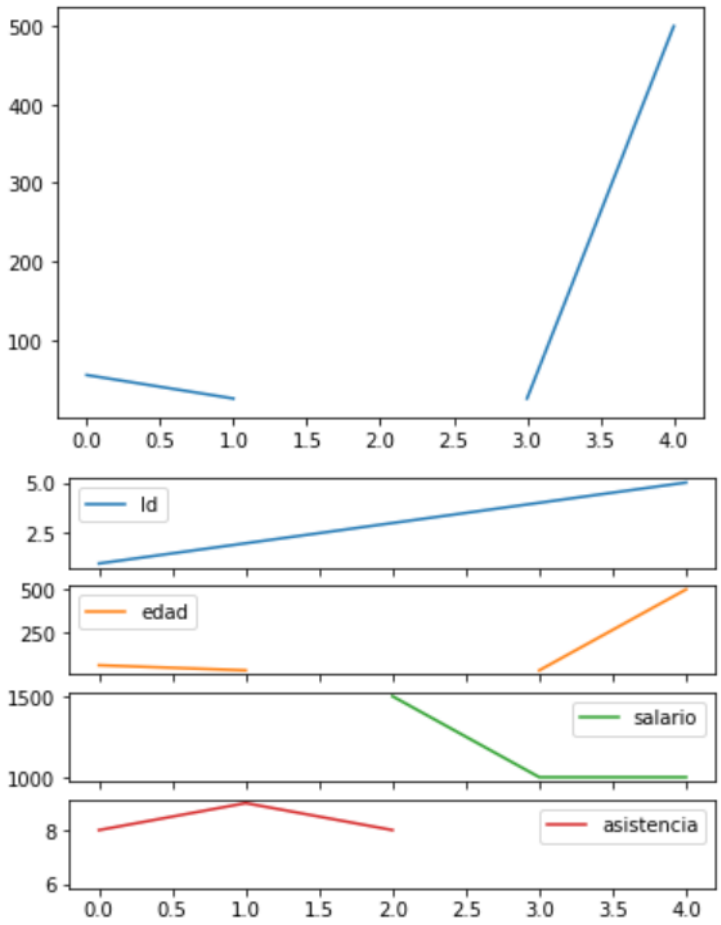
Tabla 5. Gráfica de líneas con librería Pandas.

```
#Paso uno, importa la librería y se cargan los datos en variable.
import pandas as pd
data=pd.read_csv("data.csv")
#Se realiza un gráfico de una de las columnas.
data["edad"].plot()
#Se realiza un gráfico de acuerdo con el número de columnas.
data.plot(subplots=4)
```

Explicación

Figura 4. Gráficas de líneas con librería Pandas.

```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f91d798bd10>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f91d79c39d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f91d797bd10>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f91d7928d90>],
      dtype=object)
```



Explicación

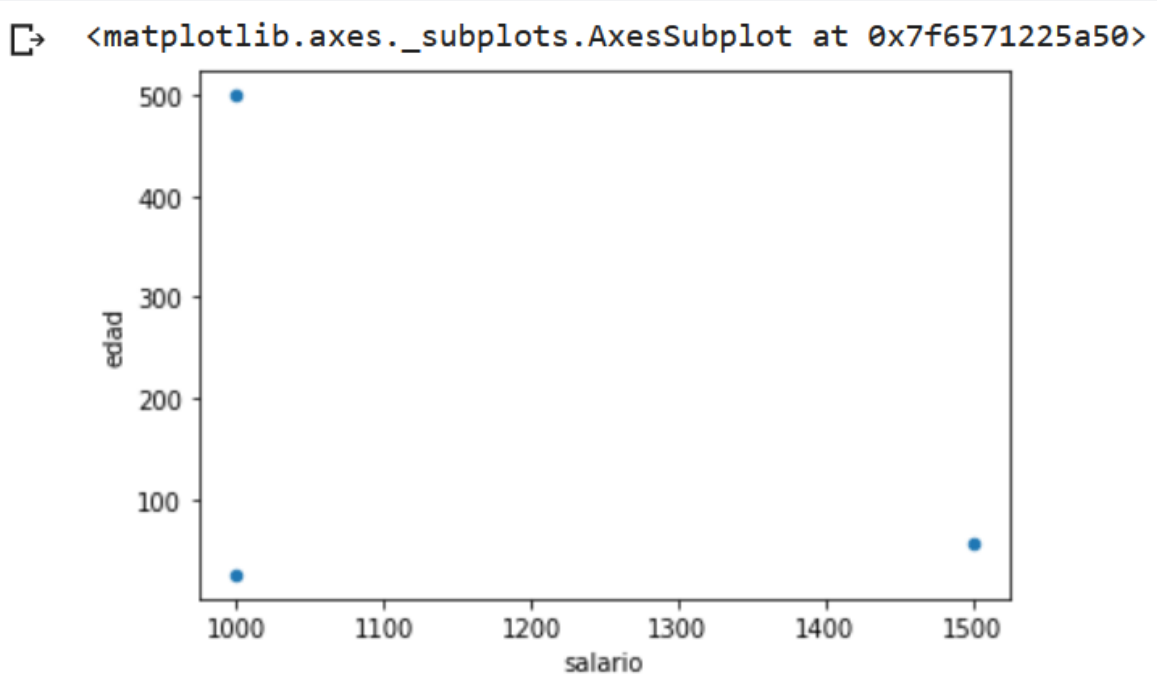
Universidad Tecmilenio

Otra de las graficas es la de **dispersión**:

Tabla 6. Gráfica de dispersión con librería Pandas

```
data.plot.scatter(x="salario", y="edad")
```

Figura 5. Graficas de dispersión con librería Pandas



Explicación

Universidad Tecmilenio

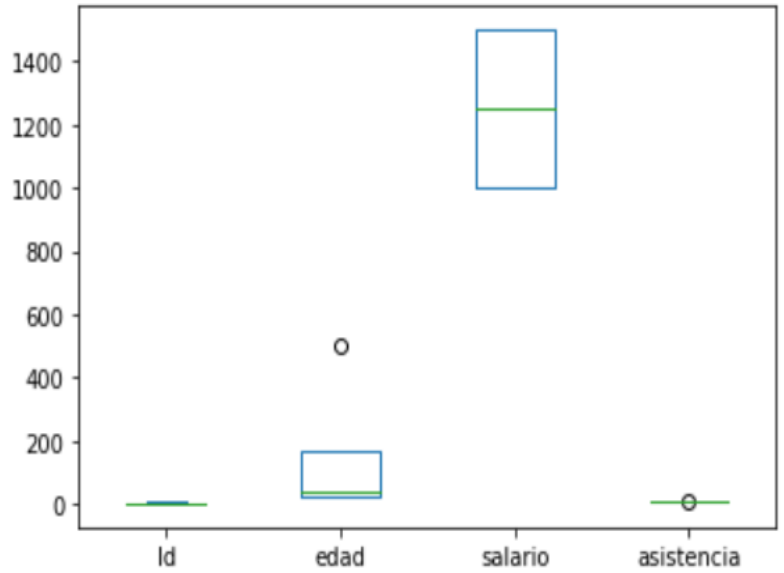
Otro ejemplo es con cajas de **gráficos**:

Tabla 7. Grafica de cajas con librería Pandas.

```
data.plot.box()
```

Figura 6. Gráficas de cajas con librería Pandas.

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/_
  X = np.atleast_1d(X.T if isinstance(X, np.ndarray) else
<matplotlib.axes._subplots.AxesSubplot at 0x7f6570f1eb10>
```



Explicación

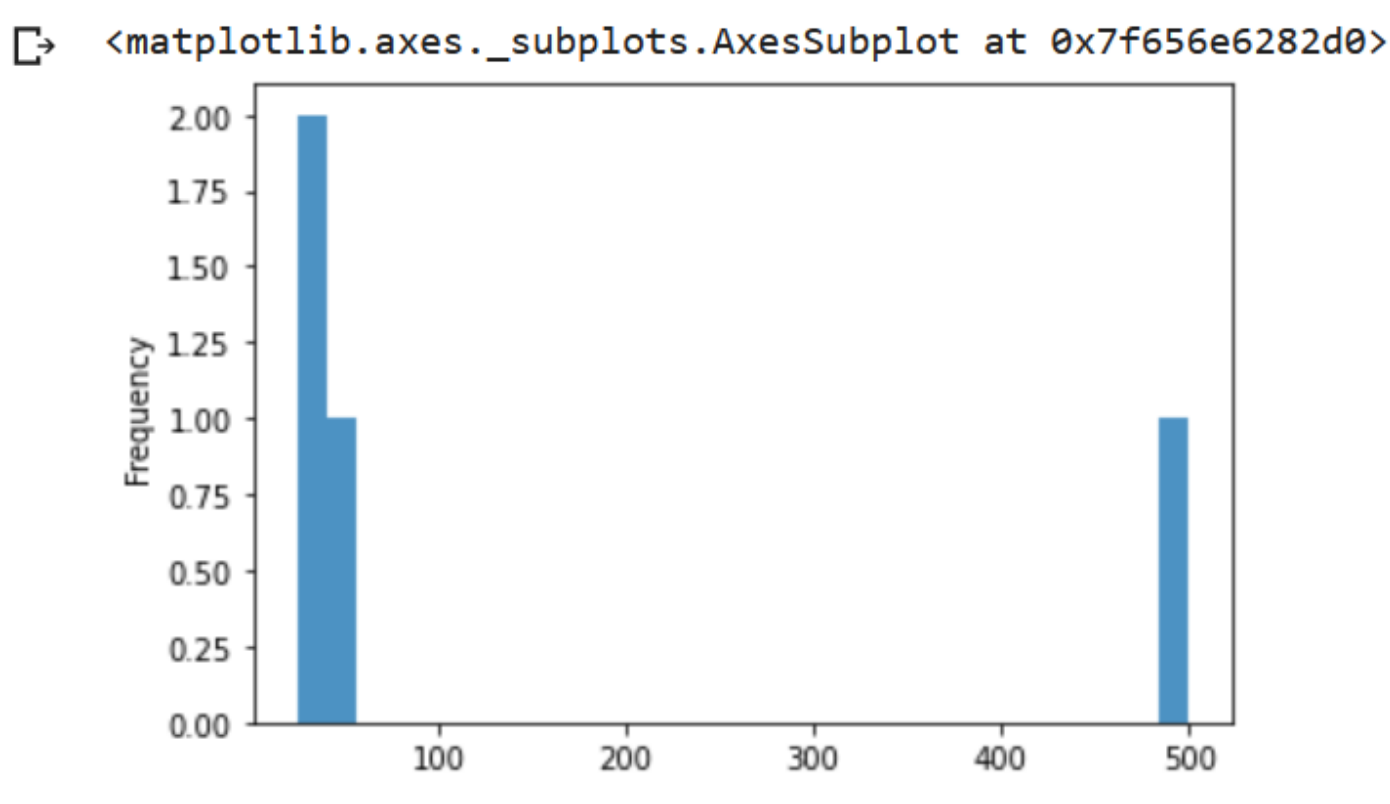
Universidad Tecmilenio

Está el histograma:

Tabla 8. Gráfica de cajas con librería Pandas

```
data["edad"].plot.hist(bins=30, alpha=0.8)
```

Figura 7. Histograma con librería Pandas



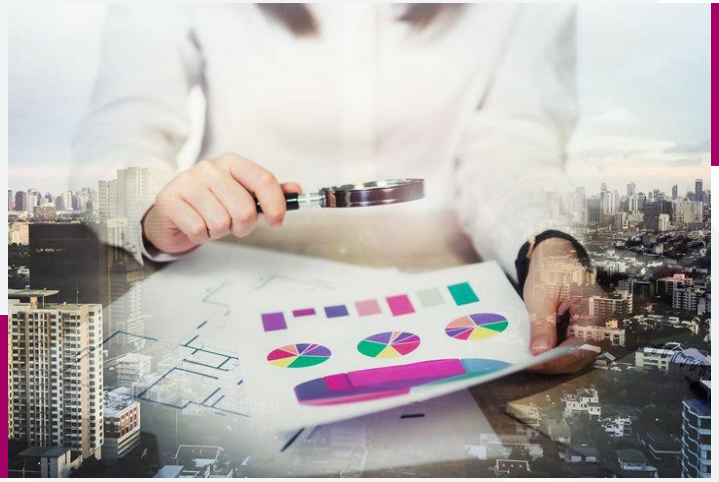
Explicación

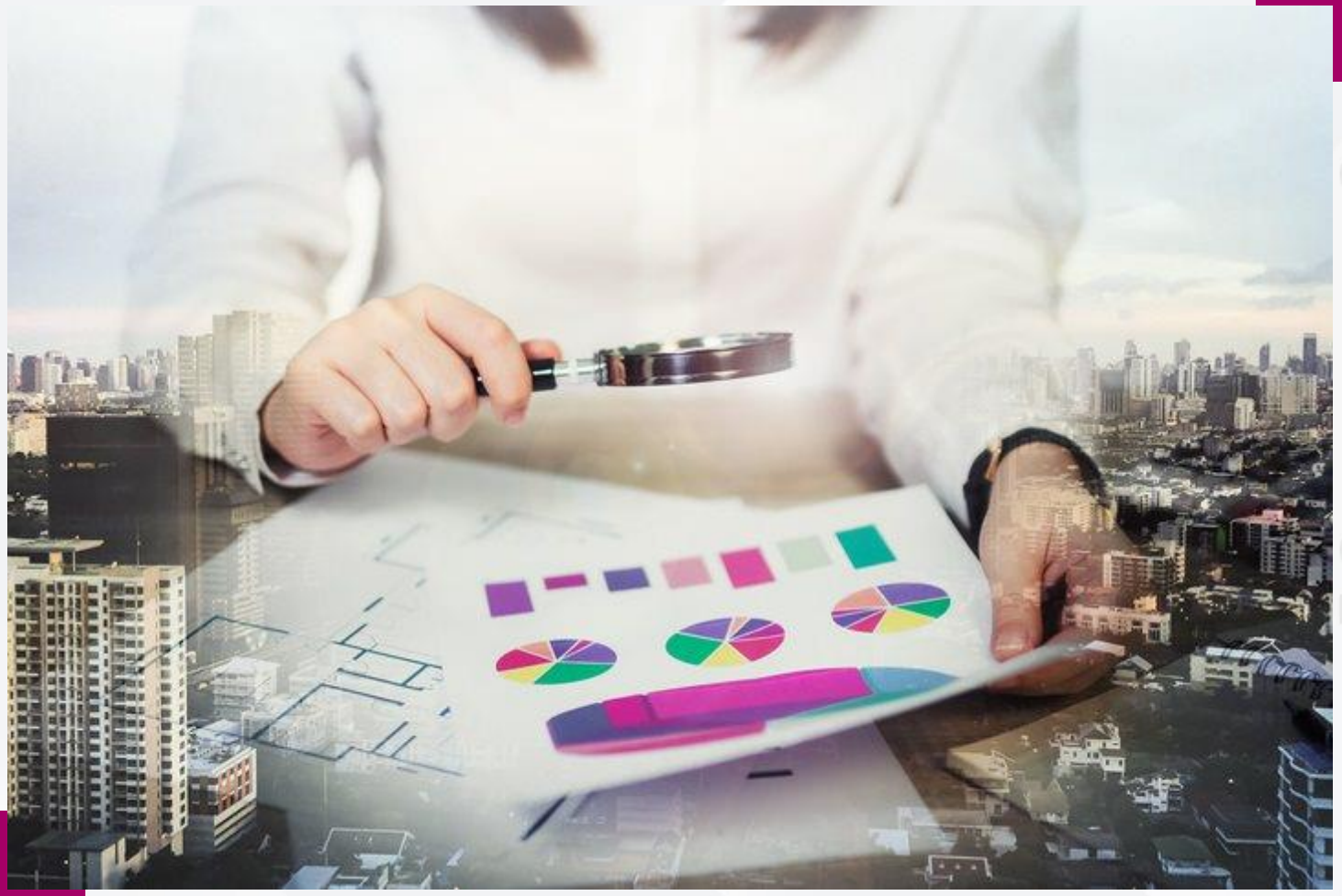
Universidad Tecmilenio

Todos los tipos de gráficos realizados con Pandas que se mencionan en **Pandas** (s.f) y de los cuales se han revisado los principales, da idea del impacto de esta librería tanto en exploración, análisis y limpieza de los datos, como para quedarse sin conocerla a fondo.

En este notebook de Google colab puedes ver todo el código desarrollado en el tema:

https://colab.research.google.com/drive/1ya6k3pFIZzsZBPk8eTq-N_PibYGj025I?usp=sharing





Explicación

Universidad Tecmilenio

Diagrama de flujo

Objetivo: Especificar los pasos para graficar con la librería pandas.

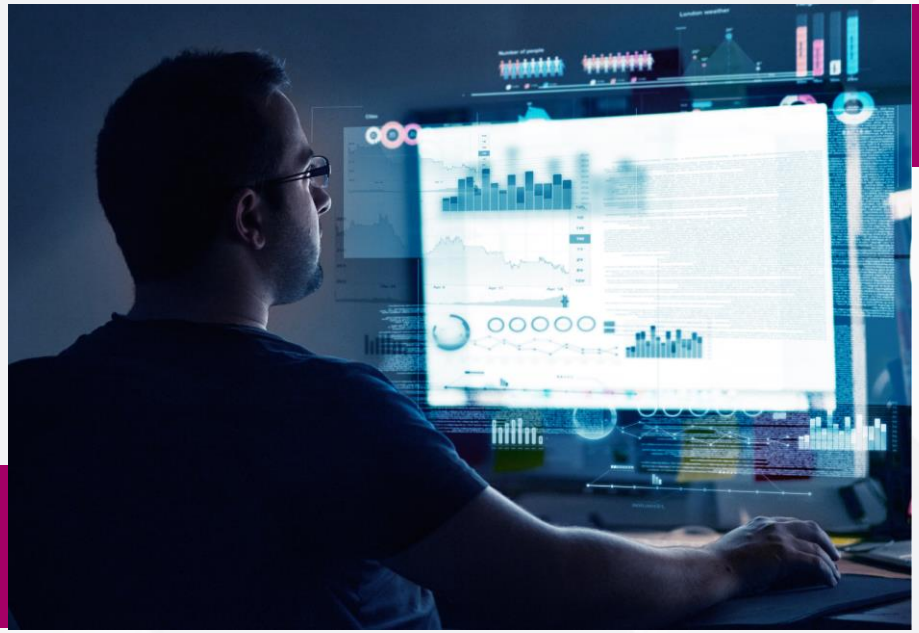
Instrucciones

- Para realizar esta actividad es importante que consultes el contenido del tema.
- Con base en lo aprendido elabora diagrama de flujo que incluya los conceptos y los pasos a seguir para realizar un gráfico en Python.
- Incluye un ejemplo.
- Deberá ser información sintetizada y precisa.
- En una sola hoja.

Actividad

Con estas dos librerías ya tienes una idea de cómo un gráfico puede ayudar a aumentar la calidad de los datos y al mismo tiempo comenzar a explorarlos y conocerlos para así plantear propuestas de análisis e interpretaciones de estos.

Cuando se trata de datos abiertos o externos muchas veces no se tiene esa oportunidad de mejorar la calidad de la información y queda del lado del analista esa decisión y función.



cierre

Referencias

Mulla, R. (2022). *ALL Python Data Visualization Libraries in 2022*. Recuperado de https://www.kaggle.com/code/robikscube/all-python-data-visualization-libraries-in-2022/notebook

Referencias

La obra presentada es propiedad de ENSEÑANZA E INVESTIGACIÓN SUPERIOR A.C. (UNIVERSIDAD TECMILENIO), protegida por la Ley Federal de Derecho de Autor; la alteración o deformación de una obra, así como su reproducción, exhibición o ejecución pública sin el consentimiento de su autor y titular de los derechos correspondientes es constitutivo de un delito tipificado en la Ley Federal de Derechos de Autor, así como en las Leyes Internacionales de Derecho de Autor.

El uso de imágenes, fragmentos de videos, fragmentos de eventos culturales, programas y demás material que sea objeto de protección de los derechos de autor, es exclusivamente para fines educativos e informativos, y cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por UNIVERSIDAD TECMILENIO.

Queda prohibido copiar, reproducir, distribuir, publicar, transmitir, difundir, o en cualquier modo explotar cualquier parte de esta obra sin la autorización previa por escrito de UNIVERSIDAD TECMILENIO. Sin embargo, usted podrá bajar material a su computadora personal para uso exclusivamente personal o educacional y no comercial limitado a una copia por página. No se podrá remover o alterar de la copia ninguna leyenda de Derechos de Autor o la que manifieste la autoría del material.