Data Analytics

Pasos para instalar Python en Windows 10 y agregarlo a las variables de entorno:

1. Descarga la última versión de Python para Windows desde la página oficial de descargas: https://www.python.org/downloads/windows/
2. Abre el archivo de instalación y sigue las instrucciones para instalar Python en tu equipo. Una vez que la instalación haya finalizado, abre el menú de inicio y busca "Editar las variables de entorno del sistema".
3. Haz clic en el botón "Variables de entorno".
4. En la sección "Variables del sistema", busca la variable "Path" y haz clic en "Editar".
5. Haz clic en "Nuevo" y agrega la ruta a la carpeta de instalación de Python. Por defecto, esta será "C:\Python39" si instalaste la versión 3.9. Si instalaste una versión diferente, asegúrate de agregar la ruta correcta
6. Haz clic en "Aceptar" para guardar los cambios. ¡Listo! Ahora deberías poder ejecutar Python desde la línea de comandos en cualquier directorio de tu equipo. Para verificar que todo esté funcionando correctamente, abre la línea de comandos y escribe "python --version". Deberías ver la versión de Python que instalaste.

Campias, Juan
Data Engineer

if, else e elif statement
Los condicionales if, else y elif permiten ejecutar diferentes bloques de código en función del cumplimiento o no de una o varias condiciones. Un ejemplo de uso sería el siguiente:

```
edad = 18

if edad >= 18:
    print("Eres mayor de edad")
else:
    print("Eres menor de edad")
```

En este caso, si la edad es mayor o igual a 18, se imprimirá "Eres mayor de edad", de lo contrario, se imprimirá "Eres menor de edad".
También podemos utilizar la sentencia elif para añadir más condiciones a la estructura:

```
edad = 18

if edad < 18:
    print("Eres menor de edad")
elif edad == 18:
    print("Tienes 18 años")
else:
    print("Eres mayor de edad")
```

En este caso, si la edad es menor de 18 se imprimirá "Eres menor de edad", si la edad es igual a 18 se imprimirá "Tienes 18 años" y si la edad es mayor de 18 se imprimirá "Eres mayor de edad".

for loops
Los bucles for permiten iterar sobre una colección de elementos y ejecutar un bloque de código para cada uno de ellos. Por ejemplo:

```
frutas = ["manzana", "naranja", "banana"]

for fruta in frutas:
    print(fruta)
```

En este caso, se imprimirá cada elemento de la lista frutas.

while loops
El bucle while se utiliza para ejecutar repetidamente un bloque de código mientras se cumpla una condición. Por ejemplo:

```
numero = 1

while numero <= 10:
```

```
print(numero)
numero += 1
```

En este caso, se imprimirá el valor de la variable **numero** mientras sea menor o igual a 10. La variable se incrementa en uno en cada iteración del bucle, gracias a la línea **numero += 1**.

listas y tuplas

Las listas y tuplas son dos tipos de colecciones de elementos en Python. La principal diferencia es que las listas son mutables, lo que significa que podemos modificar sus elementos, mientras que las tuplas son inmutables, lo que significa que no podemos modificar sus elementos una vez creadas.
Un ejemplo de creación de lista sería:

```
frutas = ["manzana", "naranja", "banana"]
```

Mientras que un ejemplo de creación de tupla sería:

```
dias_de_la_semana = ("lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo")
```

slice operators

Los slice operators permiten acceder a una parte específica de una lista o tupla. Por ejemplo:

```
frutas = ["manzana", "naranja", "banana", "kiwi", "pera"]
print(frutas[1:3])
```

En este caso, se imprimirá el segundo y tercer elemento de la lista (naranja y banana).

sets

Los sets son una colección de elementos no ordenados y sin elementos duplicados. Un ejemplo de creación de set sería:

```
frutas = {"manzana", "naranja", "banana"}
```

diccionarios

Los diccionarios son una estructura de datos que nos permite asociar un valor a una clave. Un ejemplo de creación de diccionario sería:

```
edades = {"Juan": 30, "María": 25, "Pedro": 40}
```

En este caso, la clave es el nombre de la persona y el valor es su edad.

comprehension lists

Las comprehension lists son una forma concisa de crear listas a partir de otras listas, aplicando una
aplicando una transformación o filtrado a sus elementos. Por ejemplo:

```
numeros = [1, 2, 3, 4, 5]

cuadrados = [numero**2 for numero in numeros]

print(cuadrados)
```

En este caso, se creará una lista llamada cuadrados que contendrá los valores de cada elemento de la lista numeros elevado al cuadrado. El resultado impreso será **[1, 4, 9, 16, 25]**.
También podemos aplicar un filtrado utilizando la estructura **if** dentro de la comprehension list. Por ejemplo:

```
numeros = [1, 2, 3, 4, 5]

pares = [numero for numero in numeros if numero % 2 == 0]

print(pares)
```

En este caso, se creará una lista llamada pares que contendrá los valores de cada elemento de la lista numeros que sean pares. El resultado impreso será **[2, 4]**.