Que es None (es un null)

None es un tipo de dato especial en Python que representa la ausencia de un valor o la falta de definición de un objeto. Es similar al concepto de **null** en otros lenguajes de programación.

En Python, None es un singleton, lo que significa que solo hay una instancia de este objeto en todo el programa. Se utiliza comúnmente para indicar que una variable o una función no tiene un valor asignado o no devuelve nada.

Aquí tienes algunos ejemplos de cómo se usa None:

Inicialización de variables: A menudo se usa para inicializar variables cuando no se tiene un valor inicial específico.

x = None
x devuelve None para indicar que no hay un valor de retorno específico.

1) Ejemplo 1:

```
def funcion_sin_retorno ():
    print ("Esta función no devuelve nada")

# La función se ejecuta, pero no devuelve ningún valor
resultado = funcion_sin_retorno ()
print (resultado) # Esto imprimirá None
```

2) Ejemplo 2:

```
def funcion_con_argumento (arg=None):
    if arg is None:
        print ("No se proporcionó ningún argumento")
    else:
        print ("El argumento es:", arg)

funcion_con_argumento () # Imprimirá "No se proporcionó ningún argumento"
```

Que es break

break es una palabra clave en Python que se usa para salir de un bucle prematuramente, antes de completar su ciclo. Se utiliza principalmente en bucles **while y for**.

Cuando se encuentra la instrucción break dentro de un bucle, el control del programa sale inmediatamente del bucle y continúa con la siguiente línea de código después del bucle.

Ten en cuenta que break sirve solo para salir de los **ciclos y del switch** (según caso) pero como no existe en python el switch solo se una para los ciclos.

1) Ejercicio en ciclo while

```
while True:
    respuesta = input ("¿Desea continuar? (s/n): ")
    if respuesta.lower() == 'n':
        # Si la respuesta es 'n', salimos del bucle while
        break

# Aquí va el resto del código que se ejecutará mientras la respuesta no sea 'n'
Print ("El algoritmo sigue ejecutándose...")

# Continuamos con el resto del código fuera del bucle
Print ("El algoritmo ha finalizado.")
```

2) ejercicio ciclo for

```
for i in range (5):
    respuesta = input ("¿Desea continuar? (s/n): ")
    if respuesta.lower() == 'n':
        # Si la respuesta es 'n', salimos del bucle for
        break

    # Aquí va el resto del código que se ejecutará mientras la respuesta no sea 'n'
    print ("El algoritmo sigue ejecutándose...")

# Continuamos con el resto del código fuera del bucle
print ("El algoritmo ha finalizado.")
```

3) Ejemplo de búsqueda según un criterio llamado limite

```
numeros = [1, 5, 8, 10, 15, 20, 25, 30]
limite = 12
numero_encontrado = None

for num in numeros:
    if num > limite:
        numero_encontrado = num
        break # Salir del bucle una vez que se encuentra el primer número mayor que el límite

if numero_encontrado is not None:
    print ("El primer número mayor que", limite, "es:", numero_encontrado)
else:
    print ("No se encontró ningún número mayor que", limite)
```

Nota: No se usa para salir de un sí, solo de un ciclo.

Que es exit

En Python, exit es una función que forma parte del módulo sys y es un alias de sys.exit. Por lo tanto, en términos de funcionalidad, exit y sys.exit son lo mismo: ambos se utilizan para finalizar la ejecución de un script de Python de forma inmediata.
import sys

1) Ejemplo 1

```
import sys
nombre_usuario="pepe"
usuario_autenticado=True
# Verificar si el usuario está autenticado
if usuario_autenticado:
    print ("Bienvenido,", nombre_usuario)
else:
    # Si el usuario no está autenticado, salir del script
    sys.exit ("Error: El usuario no está autenticado")

# El resto del código aquí solo se ejecutará si el usuario está autenticado
Print (nombre_usuario,", Tu Panel del usuario esta activado...")
Print ("preparando las demás opciones del panel del usuario...")
```

2) Ejemplo 2

```
while True:
    opcion = input("Ingrese una opción (o escriba 'salir' para salir): ")
    if opcion.lower() == 'salir':
        Print ("Saliendo del script...")
        exit()  # dejar de ejecutar este script.
    else:
        Print ("Realizando la operación correspondiente a:", opcion)

#nunca ejecutara estas líneas por que se finalizara el script
Print ("Fuera del ciclo se continúa ejecutando líneas...")
#si no funciona import sys y usar sys.exit en vez de exit()
```

TALLER NONE, BREAK Y EXIT.

Ejercicio 1: Uso de None en Variables

Crea una variable llamada **valor** y asígnale **None**. Imprime el valor de la variable.

```
valor = None
print("El valor de la variable es:", valor)
```

Ejercicio 2: Función que Devuelve None

Escribe una función que no devuelva ningún valor explícito y llama a esa función. Imprime el resultado de la llamada.

```
def funcion_sin_retorno():
    print("Esta función no devuelve nada")

resultado = funcion_sin_retorno()
print("El resultado de la función es:", resultado)
```

Ejercicio 3: Uso de None en Funciones con Argumentos Opcionales

Escribe una función que reciba un argumento opcional. Si el argumento no es proporcionado, la función debe imprimir "No se proporcionó ningún argumento".

```
def funcion_con_argumento(arg=None):
    if arg is None:
        print("No se proporcionó ningún argumento")
    else:
        print("El argumento es:", arg)

funcion_con_argumento()
funcion_con_argumento("Hola")
```

Ejercicio 4: Uso de break en un Ciclo while

Escribe un ciclo **while** que continúe preguntando al usuario si desea continuar. Si el usuario ingresa 'n', usa **break** para salir del ciclo.

```
while True:
    respuesta = input("¿Desea continuar? (s/n): ")
    if respuesta.lower() == 'n':
        break
print("El algoritmo ha finalizado.")
```

Ingeniero Freddy Muñoz

Ejercicio 5: Uso de `break` en un Ciclo `for`

Escribe un ciclo **for** que se repita 5 veces y pregunte al usuario si desea continuar en cada iteración. Si el usuario ingresa 'n', usa **break** para salir del ciclo.

```
for i in range(5):
    respuesta = input("¿Desea continuar? (s/n): ")
    if respuesta.lower() == 'n':
        break
    print("El algoritmo sigue ejecutándose...")
print("El algoritmo ha finalizado.")
```

Ejercicio 6: Búsqueda con `break`

Escribe un ciclo **for** que busque el primer número mayor que un valor límite en una lista de números. Usa **break** para salir del ciclo una vez que el número es encontrado.

```
numeros = [1, 5, 8, 10, 15, 20, 25, 30]
limite = 12
numero_encontrado = None

for num in numeros:
    if num > limite:
        numero_encontrado = num
        break

if numero_encontrado is not None:
    print("El primer número mayor que", limite, "es:", numero_encontrado)
else:
    print("No se encontró ningún número mayor que", limite)
```

Ejercicio 7: Uso de `exit` en una Función

Escribe una función que verifique si un usuario está autenticado. Si no está autenticado, usa **exit** para finalizar el script.

```
import sys

def verificar_autenticacion(usuario_autenticado):
    if usuario_autenticado:
        print("Bienvenido")
    else:
        sys.exit("Error: El usuario no está autenticado")

verificar_autenticacion(False)
print("Esta línea no se ejecutará si el usuario no está autenticado")
```

Ejercicio 8: Uso de exit en un Ciclo while

Escribe un ciclo **while** que continúe ejecutándose hasta que el usuario ingrese 'salir'. Usa **exit** para finalizar el script cuando el usuario ingrese 'salir'.

```
while True:
    opcion = input("Ingrese una opción (o escriba 'salir' para salir): ")
    if opcion.lower() == 'salir':
        print("Saliendo del script...")
        exit()
    else:
        print("Realizando la operación correspondiente a:", opcion)
```

Ejercicio 9: Verificar None en una Lista

Crea una lista que contenga algunos elementos y uno de ellos sea **None**. Escribe un algoritmo que verifique si **None** está en la lista e imprima un mensaje apropiado.

```
lista = [1, 2, None, 4, 5]
if None in lista:
    print("La lista contiene None")
else:
    print("La lista no contiene None")
```

Ejercicio 10: Filtrar Elementos None de una Lista

Escribe un algoritmo que elimine todos los elementos **None** de una lista.

```
lista = [1, 2, None, 4, None, 5]
lista_filtrada = [x for x in lista if x is not None]
print("Lista sin elementos None:", lista_filtrada)
```

Ejercicio 11: Contar Elementos None en una Lista

Escribe un algoritmo que cuente cuántos elementos **None** hay en una lista.

```
lista = [1, 2, None, 4, None, 5]
contador = lista.count(None)
print("Número de elementos None en la lista:", contador)
```

Ejercicio 12: Uso de break para Encontrar un Elemento

Escribe un algoritmo que busque un elemento específico en una lista. Si el elemento es encontrado, imprime un mensaje y usa **break** para salir del ciclo.

```
lista = [1, 2, 3, 4, 5]
elemento_buscado = 3

for elemento in lista:
```

```
    if elemento == elemento_buscado:
        print("Elemento encontrado:", elemento_buscado)
        break
else:
    print("Elemento no encontrado")
```

Ejercicio 13: Uso de `None` en una Lista de Listas

Crea una lista de listas donde algunas sublistas contengan **None**. Escribe un algoritmo que imprima la primera sublista que contenga **None**.

```
lista_de_listas = [[1, 2, 3], [4, None, 6], [7, 8, 9]]

for sublista in lista_de_listas:
    if None in sublista:
        print("Sublista que contiene None:", sublista)
        break
```

Ejercicio 14: Ciclo con `break` y `continue`

Escribe un ciclo **for** que recorra una lista de números. Si el número es par, imprímelo y usa **continue** para pasar al siguiente número. Si el número es impar y mayor que 10, usa **break** para salir del ciclo.

```
numeros = [2, 4, 6, 11, 14, 17, 20]

for num in numeros:
    if num % 2 == 0:
        print("Número par:", num)
        continue
    if num > 10:
        print("Número impar mayor que 10:", num)
        break
```

Ejercicio 15: Salida Temprana de una Función con `None`

Escribe una función que reciba una lista y devuelva **None** si la lista está vacía. De lo contrario, devuelve el primer elemento de la lista.

```
def primer_elemento(lista):
    if not lista:
        return None
    return lista[0]

resultado = primer_elemento([])
print("El resultado es:", resultado)

resultado = primer_elemento([1, 2, 3])
print("El primer elemento es:", resultado)
```

Ejercicio 16: Uso de `break` en una Búsqueda de Texto

Escribe un algoritmo que busque una palabra específica en una lista de frases. Si la palabra es encontrada en una frase, imprime la frase y usa **break** para salir del ciclo.

```
frases = ["Hola mundo", "Python es genial", "Me gusta programar", "Adiós mundo"]
palabra_buscada = "Python"

for frase in frases:
    if palabra_buscada in frase:
        print("Frase encontrada:", frase)
        break
else:
    print("Palabra no encontrada en ninguna frase")
```

Ejercicio 17: Uso de `exit` en una Condición de Error

Escribe una función que reciba un número y verifique si es positivo. Si no lo es, usa **exit** para finalizar el script con un mensaje de error.

```
import sys

def verificar_positivo(numero):
    if numero <= 0:
        sys.exit("Error: El número no es positivo")
    print("El número es positivo:", numero)

verificar_positivo(-5)
print("Esta línea no se ejecutará si el número no es positivo")
```

Ejercicio 18: Bucle Infinito con `break`

Escribe un bucle infinito que pregunte al usuario por un número. Si el número es mayor que 100, usa **break** para salir del bucle.

```
while True:
    numero = int(input("Ingrese un número: "))
    if numero > 100:
        print("Número mayor que 100. Saliendo del bucle.")
        break
```

Ejercicio 19: Verificar Palabra Clave con `exit`

Escribe un algoritmo que solicite al usuario una contraseña. Si la contraseña es incorrecta, usa **exit** para finalizar el script.

```
import sys
```

```
contraseña_correcta = "12345"
contraseña_ingresada = input("Ingrese la contraseña: ")

if contraseña_ingresada != contraseña_correcta:
    sys.exit("Error: Contraseña incorrecta")

print("Contraseña correcta. Bienvenido.")
```

Ejercicio 20: `None` en Diccionarios

Crea un diccionario donde algunas claves tengan el valor **None**. Escribe un algoritmo que elimine las claves con valores **None**.

```
diccionario = {'a': 1, 'b': None, 'c': 2, 'd': None}
diccionario_limpio = {k: v for k, v in diccionario.items() if v is not None}
print("Diccionario sin valores None:", diccionario_limpio)
```