Course Checkpoint 5

¿Qué es un condicional?

Los condicionales en Python son estructuras de control que te permiten tomar decisiones en tu código. Estas decisiones se basan en si ciertas condiciones son verdaderas o falsas. Los condicionales te permiten ejecutar diferentes bloques de código dependiendo del resultado de estas condiciones.

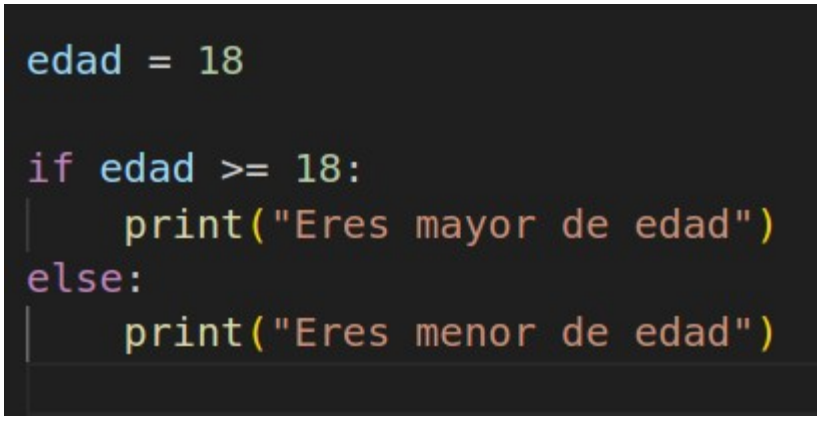
en Python, los condicionales más comunes son if, elif (que significa "else if") y else.

if: Se utiliza para ejecutar un bloque de código si una condición es verdadera.

elif Se utiliza para verificar múltiples condiciones si la primera condición no es verdadera. Puedes tener cero o más bloques **elif** en un condicional.

else Se utiliza para ejecutar un bloque de código si ninguna de las condiciones anteriores es verdadera.

Aquí tienes un ejemplo simple de cómo se utilizan los condicionales en Python:



```
edad = 18

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

En este ejemplo, el condicional **if** verifica si la variable **edad** es mayor o igual a 18. Si es así, imprime "Eres mayor de edad". Si no lo es, el bloque **else** se ejecuta, imprimiendo "Eres menor de edad".

Es como un toma de decisiones de la vida real, imagina que estas en un supermercado y quieres comprar unas cuantas cervezas. Pero hay reglas :

1. si tienes suficientes dinero, puedes comprar cervezas.
2. si no tienes sufiente dinero, no puedes comprar cervezas

entonces, ¿lo podriamos hacer en python ? Vamos a ello :

```
money = 12

price_sixpack = 12
price_onbeer = 3
price_two_sixpack = 18

if money >= price_two_sixpack:
    print("Puedes comprar dos six-packs.")
elif money >= price_sixpack:
    print("Puedes comprar un six-packs.")
elif money >= price_onbeer:
    print("Puedes comprar una cerveza.")
else:
    print("no tienes suficente dinero")
```

aqui, “money” es la cantidad de dinero que posees y “price_sixpack , price_onebeer y price_two_sixpack” es la cantidad de cervezas que puedes comprar segun el dinero que poseas, el sistema nos imprimira la mayor cantidad de cervezas segun el dinero que tengas.

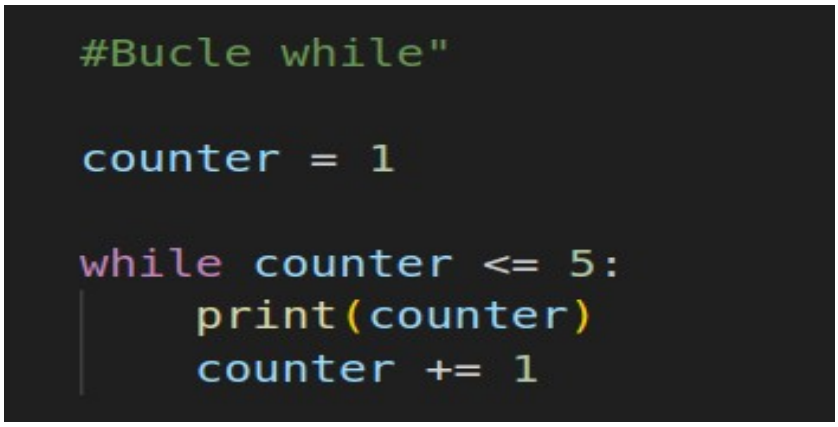
¿Cuáles son los diferentes tipos de bucles en Python?

los bucles son estructuras de control que nos permiten repetir un bloque de código varias veces, tenemos dos tipos de bucles :

bluce ‘for’ : Este bucle se utiliza cuando se conoce de antemano la cantidad de veces que se quiere repetir un bloque de código. Se utiliza especialmente para iterar sobre una secuencia (como una lista, tupla, diccionario, etc.) y ejecutar un bloque de código una vez para cada elemento de esa secuencia por ejemplo :

con este bucle ‘for’ podemos ver que con solo dos lineas de codigo podemos imprimir los numeros del 1 al 6 y no es necesario escrbir 5 linas de codigo mas para imprimir el numero individualmente

Bucle ‘while’ : Este bucle se utiliza cuando no se sabe de antemano cuántas veces se repetirá un bloque de código, pero se repetirá mientras se cumpla una condición especificada. Se ejecuta siempre que la condición sea verdadera.



el bucle while se ejecuta mientras el valor de contador sea menor o igual a 5. Dentro del bucle, imprimimos el valor actual de contador y luego lo incrementamos en 1. El bucle se detiene cuando contador es mayor que 5.

¿Por qué son útiles?

Automatización de tareas repetitivas: Los bucles permiten ejecutar un bloque de código varias veces sin tener que escribir las mismas líneas una y otra vez. Esto es especialmente útil cuando necesitas realizar una tarea repetitiva, como procesar elementos de una lista o realizar cálculos iterativos..

Manipulación de grandes conjuntos de datos: Cuando trabajas con grandes cantidades de datos, como listas, diccionarios o archivos, los bucles te permiten procesar cada elemento de manera individual. Esto facilita la manipulación y el

análisis de grandes conjuntos de datos, ya que puedes aplicar las mismas operaciones a cada elemento de manera eficiente.

Flexibilidad y control del flujo de ejecución: Los bucles te permiten controlar el flujo de ejecución de tu programa. Puedes repetir un bloque de código un número específico de veces (bucle for) o continuar repitiéndolo mientras se cumpla una condición particular (bucle while). Esto te da flexibilidad para adaptar el comportamiento de tu programa a diferentes situaciones y requisitos.

Reducción de la redundancia: Al utilizar bucles, puedes reducir la redundancia en tu código al evitar la repetición de instrucciones similares. En lugar de escribir el mismo código una y otra vez para cada elemento de una lista, puedes encapsularlo dentro de un bucle y reutilizarlo fácilmente.

¿Qué es una lista por comprensión en Python?

Una lista por comprensión en Python es una forma compacta y elegante de crear listas. Te permite construir una lista a partir de otra lista, o de algún otro tipo de iterador, aplicando una expresión a cada elemento de la secuencia original y, opcionalmente, filtrando los elementos según una condición.

La sintaxis básica de una lista por comprensión es la siguiente:

```
cuadrados = [x**2 for x in range(1, 6)]
```

`x**2` : es la expresion que se aplica a cada elemento
`x` : es una variable que representa cada elemento del iterable
`range(1, 6)` : es cualquier objeto iterable, como una lista, tupla, o rango.

¿Qué es un argumento en Python?

una función es como una receta de cocina. Cuando haces una receta, necesitas ingredientes específicos, ¿verdad? De manera similar, cuando usas una función en Python, necesitas pasarle ciertos valores para que pueda hacer su trabajo. Estos valores se llaman "argumentos". Por ejemplo :

```
def suma(a, b):
    return a + b
```

Aquí, a y b son los argumentos de la función suma. Cuando llamas a esta función, debes pasarle dos valores para a y b, de lo contrario, no sabrá qué números sumar.

```
result = sum(5 , 7)
```

Aqui le dimos los valores a=5 y b=7 , 5 y 7 son nuestros argumentos que utilizaremos para que nuestra funciona nos de un resultado al darle print :

En conclusion los argumentos son los ingredientes necesarios de una funciona (receta).

¿Qué es una función Lambda en Python?

Una función lambda en Python es una función anónima y pequeña que se define sin un nombre utilizando la palabra clave lambda. A diferencia de las funciones normales definidas con la palabra clave def, las funciones lambda se pueden usar para crear funciones de una sola expresión de forma más concisa.

La sintaxis básica de una función lambda es la siguiente:

```
lambda argumentos: expresion
```

Donde:

argumentos es una lista de parámetros separados por comas, similar a los parámetros de una función normal.

expresion es la expresión que se evalúa y devuelve como resultado de la función.

Por ejemplo, considera una función lambda que suma dos números:

```
suma = lambda a, b: a + b
```

En este caso, lambda a, b: define una función lambda que toma dos argumentos a y b, y luego devuelve la suma de estos dos números.

Las funciones lambda son útiles cuando necesitas definir una función simple y rápida sin tener que escribir una función completa con la declaración def. Se utilizan comúnmente en combinación con funciones como map(), filter() y reduce(), así como en expresiones de listas y diccionarios, donde se requieren funciones breves y concisas.

En resumen, una función lambda en Python es una forma de crear funciones anónimas y pequeñas en una línea sin necesidad de darles un nombre. Son convenientes para situaciones donde necesitas una función rápida y simple sin tener que definirla por separado.

¿Qué es un paquete pip?

Un paquete pip es una colección de código Python que se distribuye para facilitar su instalación y uso en proyectos de Python. Pip es el sistema de gestión de paquetes de Python, y se utiliza para instalar y gestionar paquetes de software escritos en Python que no están incluidos en la biblioteca estándar de Python.

Cuando trabajas en un proyecto de Python y necesitas utilizar funcionalidades adicionales que no están disponibles en Python por defecto, puedes buscar y descargar paquetes específicos para instalarlos en tu entorno de desarrollo. Estos paquetes pueden proporcionar una amplia gama de funcionalidades, desde herramientas de desarrollo hasta bibliotecas especializadas para tareas específicas, como el procesamiento de datos, el desarrollo web, la inteligencia artificial, entre otros.

Para instalar un paquete pip, normalmente utilizas el comando pip install nombre_del_paquete. Por ejemplo, si quieres instalar el paquete requests, que se utiliza para hacer solicitudes HTTP en Python, simplemente ejecutas pip install requests en tu terminal y pip descargará e instalará automáticamente el paquete requests y todas sus dependencias.