El siguiente manual tiene la función de darte unas pinceladas de las operaciones/funciones que más vas a utilizar en Python mientras estes trabajando con nosotros. Esto es solo el inicio con el tiempo llegaras a perfeccionar tu perfil

El siguiente manual consta de los siguientes apartados:

1. ¿Qué es un condicional?
2. ¿Cuáles son los diferentes tipos de bucles en Python? ¿Por qué son útiles?
3. ¿Qué es una lista por comprensión en Python?
4. ¿Qué es un argumento en Python?
5. ¿Qué es una función Lambda en Python?
6. ¿Qué es un paquete pip?
7. *¿Qué es un condicional?*

Te preguntaras ….¿**que es un condicional**?, pues bien un condicional no es mas que un conjunto de instrucciones que se dan al programa , para que ejecute un código u otro si se cumplen las condiciones impuestas. Un ejemplo simple en la vida real…. Si vas a comprarme el pan te dejo ir al cine. Como puedes ver solo podrás ir al cine si me compras el pan. Te estarás preguntando, ¿en Python como seria?

Pues la respuesta es sencilla: con los operadores if, elif y else. Te defino a continuacion los conceptos

* ***if:*** ejecuta el código al que va asociado si la condición es verdadera. En el ejemplo, sería mostrar el mensaje “No puedes beber alcohol” si el/la usuario/a tiene menos de 18 años.
* ***else***: va después de un if. Aplica el código al que está conectado si la condición que el if marca es falsa. Es decir, si A no se cumple, else B.
* ***elif:*** si hay varias condiciones que comprobar antes de “decidir”, se utiliza if en la primera y elif en las siguientes. En la tabla, expresa que, si el/la usuario/a no es mayor de 18 años, sino que tiene justo los 18, puede beber alcohol legalmente.

Te adjunto una tabla que me parece que esta muy bien resumido todo esto de una forma sencilla, con ejemplos sencillos:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Recordar que la estructura del condicional es if + la condición: (importante poner los dos puntos) y después acordaros de poner la indentacion 2 o 4 espacios) .

Luego tenemos los condicionales anidados, es decir un condicional dentro de otro, que no hay que confundir con el elif.

Diferencias entre elif y condicionales anidados:

Elif solo valora la condición o requisito al que va asociado si los anteriores son falsos. Simplifica el código y lo hace más fácil de leer y comprender.

Condicionales anidados valoran cada requisito independientemente del resto y permiten mayor precisión en las casuísticas complicadas.

Te pongo unos ejemplos de condicionales anidados para que puedas comprender esto más fácilmente:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Errores mas comunes al emplear condicionales:

* **No escribir los dos puntos “:”** después de los condicionales.
* **Abusar de las anidaciones**, que hacen que el código sea cada vez más complejo y que el software esté menos optimizado.
* **No contemplar todos los requisitos** posibles.
* Realizar una**indentación incorrecta** o confundir los operadores.

1. *¿Cuáles son los diferentes tipos de bucles en Python? ¿Por qué son útiles?*

***Ahora pasaremos a definir que son los bucles el tipo de bucles que hay y para que son importantes***

Un bucle no es mas que otra herramienta para alterar el flujo normal de un programa. Nos permite repetir una parte del código tantas veces como queramos (por ejemplo, en las condicionales que antes te he contado).

Los diferentes bucles son **For…in y while**

**While:** ejecuta la parte de código una y otra vez hasta que la condición especificada sea falsa, es decir mientras sea verdadera se ejecutara. Por esto hay que tener cuidado al definir un bucle while, ya que podemos crear bucles infinitos si no lo definimos bien. A continuación, te pongo un ejemplo de bucle infinito:

a = 1

while a < 10:

print("¡Hola, mundo!")

como puedes ver a siempre va a ser menor de 10, ya que siempre es uno, el programa estará repitiéndose infinitamente.

Para que no sea infinito deberíamos definirlo por ejemplo de esta forma

a = 1

while a < 10:

print("¡Hola, mundo!")

a = a + 1

ahora solo se repetirá 9 veces ya que el valor de a se esta incrementando.

Como has visto la forma de aplicar while es la siguiente: while + la condición que quieras+: y en la siguiente línea lo que quieres ejecutar con la indentación de 2 o 4 espacios.

El bucle For…in es el mas usado ya que no genera el problema de bucles infinitos y es el mas sencillo para navegar por listas, diccionarios etc…

La forma de aplicarlo es : For + elemento+ in + colección/diccionario/lista : y en la siguiente línea lo que quieres ejecutar con la indentación de 2 o 4 espacios. Te adjunto un ejemplo:

lenguajes = ["Python", "C", "C++", "Java"]

for lenguaje in lenguajes:

print(lenguaje)

en este ejemplo el bucle lo que hace es imprimir cada elemento de la lista (primero imprimirá “Python”, luego “C” así hasta “Java”.

Los bucles como has podido ver tienen su importancia para poder navegar por colecciones (listas, tuplas y diccionarios) y también por strings (cadena de caracteres).

1. *¿Qué es una lista por comprensión en Python?*

Ya has entendido los condicionales y los bucles, ahora te voy a explicar el siguiente concepto que veras como te va a servir mucho en tu día a día. Listas de comprensión nos proporcionan una forma corta y concisa de crear listas. Se usan con corchetes [] y en su interior contienen una expresión seguida de un bucle for y cero o más sentencias for o if. La expresión puede ser cualquier cosa que se te ocurra, lo que significa que puedes usar cualquier tipo de objetos en la lista. El resultado es una nueva lista creada tras evaluar las expresiones que haya dentro. Ahora te pondré un ejemplo sencillo:

squared = []

**for** x **in** range(10):

squared.append(x\*\*2)

en este ejemplo estamos haciendo un bucle for…in (explicado anteriormente), va a ir recorriendo el rango (del 0 al 9) y elevándolo al cuadrado y añadiéndolo a la lista, es decir, squared al final del bucle tendrá los siguientes valores[0, 1, 4, 9, 16,…,81]

1. *¿Qué es un argumento en Python?*

Ahora te explicare que son los argumentos en Python

Los argumentos en funciones se refieren a los valores que se pasan a una función para que realice una tarea específica. Como desarrolladores, es común encontrarnos con situaciones en las que necesitamos que una función tome distintos valores para realizar una tarea en particular.

Ejemplo 1, queremos hacer una suma (def= definición de función, a y b (argumentos)

def suma(a, b):

return a + b

result = suma(2, 3)

print(result) # Imprime 5

Ejemplo 2, mismo que el anterior pero marcando valores de referencia en caso de que no se incluyan al llamar la función (en este caso es el argumento b, sino se introduce valor tomara el valor 2)

def multiplicacion(a, b=2):

return a \* b

result1 = multiplicacion(3) # Imprime 6

result2 = multiplicacion(3, 5) # Imprime 15

1. *¿Qué es una función Lambda en Python?*

Enhorabuena ya estas teniendo un conocimiento muy avanzado, pues como tienes un nivel bastante avanzado, voy a explicarte una función que te va a parecer muy interesante y práctica, es la función Lambda (no te asustes por el nombre), esta función es una forma muy sencilla de definir una función sencilla, y anónima (sin nombre) y sin mucha línea de código. Tienes que tener en cuenta que se comportan como las funciones normales (las que se definen con def), la forma de escribirla es la siguiente -> lambda a1, a2, … : salida (a1, a2, a…, son los argumentos de entrada de la función, luego seguido de : y la salida, es lo que queremos obtener (una operación, etc…)

Te pongo a continuación un ejemplo sencillo:

Función lambda

mi\_lista = [1, 2, 3, 4, 5, 6]

lista\_nueva = list(map(lambda x: x \* 2, mi\_lista))

print(lista\_nueva) # obtendremos lo siguiente :[2, 4, 6, 8, 10, 12]

Ahora forma normal de definirla sin una función lambda

def doble(x):

return x\*2

mi\_lista = [1, 2, 3, 4, 5, 6]

lista\_nueva = list(map(doble, mi\_lista))

print(lista\_nueva) #obtendremos lo siguiente: [2, 4, 6, 8, 10, 12]

1. *¿Qué es un paquete pip?*

Ya estamos al final de nuestra guía, te preguntaras si todo lo debes codificar tu, pues la respuesta es no, podemos usar cosas que hayan codificado otros, para ello tenemos por ejemplo los paquetes pip (pip es un acrónimo que se puede interpretar como Pip Instalador de Paquetes o Pip Instalador de Python), es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python. Muchos paquetes pueden ser encontrados en el Python Package Index (PyPI).

Como usamos estos paquetes:

Ponemos en la consola pip install nombre\_paquete

Hay muchas librerías como numpy, en el siguiente link puedes consultar como subir, descargar, buscar el paquete que necesites

<https://pypi.org/>