Ciclos con while.

Python cuenta con la palabra reservada while para ejecutar un bloque de código recursivamente mientras se cumpla una condición determinada.

Cuando la expresión lógica evaluada por while sea **false**, el flujo de ejecución continuará sin ejecutar el bloque dentro de while.

*<flujo principal>*

*while <expresión lógica>:*

*<bloque inscrito a while>*

*<flujo principal>*

Interrupciones de ejecución de un bloque.

En ciertas circunstancias es necesario interrumpir el flujo lógico de un programa. Python cuenta con los siguientes recursos para hacerlo.

* La palabra reservada continue.
* La palabra reservada break.
* La función exit().
* La palabra reservada continue.

La palabra reservada continue termina de forma prematura la ejecución de un bloque dentro de un ciclo.

*repeticion = 0*

*while repeticion < 10: # 10 repeticiones, del 0 al 9*

*repeticion += 1*

*numero = int(input("Ingrese un número: "))*

*if numero % 2 == 1: # Número impar*

*continue*

*print("Es un número par!")*

* La palabra reservada break.

La palabra reservada break termina prematuramente la ejecución del bloque de código en el que se encuentra y restablece el flujo de ejecución al bloque de código que lo precede.

*numero = 0*

*while True: # Esto crea un bucle infinito*

*numero += 1*

*print(numero)*

*if numero == 10:*

*break*

* La función exit().

La función exit() termina la ejecución de un programa y cierra el intérprete de Python.

Objetos iterables.

Una de las grandes fortalezas de Python es su capacidad de realizar iteraciones de forma dinámica a partir de diversos tipos de objetos con la capacidad de ser iterables.

Algunos de estos objetos son los de tipo:

str.

list.

tuple.

dict.

set.

frozenset.

bytes

La estructura for ... in.

Para iterar un objeto iterable se utiliza la siguiente sintaxis:

*for <contador> in <objeto iterable>:*

Iteraciones incrementales/decrementales.

La forma más común de realizar iteraciones en otros lenguajes de programación es por medio algo similar al uso de la función range().

La función range()

Para definir rangos numéricos se usa la función range().

* range(n, m, s) cumple: rango >= n and rango < m en incrementos de s.
* range(n, m) cumple: rango >= n and rango < m en incrementos de 1.
* range(m) cumple: rango >= 0 and rango < m en incrementos de 1.

*>>> frutas = {'Fresa':'roja', 'Limon':'verde', 'Papaya':'naranja', 'Manzana':'amarilla', 'Guayaba':'rosa'}*

*>>> for llave in frutas:*

*... print llave, 'es de color', frutas[llave]*

*...*

*Fresa es de color roja*

*Limon es de color verde*

*Manzana es de color amarilla*

*Papaya es de color naranja*

*Guayaba es de color rosa*

Condiciones

La condicion que vaya a evaluar la sentencia if, sera un valor booleano: por True ejecutará algún cierto código, por False ejecutará otro código.

Sentencias IF: La notación en Python es:

*if (condicion):*

*# Ejecuta algun codigo por Verdadero*

*else:*

*# ejecuta algun codigo por Falso*

Ejemplo:

print("\*\*\*Calcule su etapa\*\*\*")

edad=int(input("Ingrese su edad: "))

if edad<=5:

edad=int(input("Ingrese su edad: "))

elif edad>=5 and edad<=13:

print("\nUsted es un niño")

elif edad>=14 and edad<=17:

print("\nUsted es adolescente")

elif edad>=18 and edad<=35:

print("\nUsted es un adulto joven")

elif edad>=36 and edad<=64:

print("\nUsted es un adulto")

elif edad>=65:

print("\nUsted es un anciano")

Librerías

**Datos binarios**

Para manejar estos datos tenemos la librería “struct” que nos proporciona dos funciones “pack()” y “unpack()” para trabajar con formatos binarios de longitud variable. En el siguiente ejemplo os vamos a mostrar cómo leer la información de cabecera de un fichero ZIP sin utilizar la librería “zipfile”. “Pack” codifica la “H” y la “I” como números sin signo de dos bytes y cuatro bytes respectivamente. También comentaros que el símbolo “<” indica que son de tamaño estándar y en orden “Little-endian” byte.

**Log**

La librería de “logging” nos ofrece un sistema “log” completo y muy flexible. Además es muy simple, ya que los mensajes de “log” son enviados a un fichero o al “sys.stderr”.

**Herramientas para listas**

La librería “array” nos proporciona el objeto “array()” que se comporta como una lista, que almacena la información de una manera más compacta aunque esta tiene que ser toda homogénea. En el siguiente ejemplo mostramos una “array” de números binarios sin signo de 2 bytes (typecode “H”) en vez de los 16 bytes que normalmente se utilizan si estuviéramos utilizando el tipo “int”.

**Decimales en coma flotante**

Los decimales en coma flotante son muy utilizados ya que tienen diversas aplicaciones. Esta librería se puede utilizar para multiplicar decimales, cálculo de %, aplicar redondeos, etc. Además puede llegar a proporcionarnos gran exactitud.