**ABPro2**

**¿Qué es un dato booleano?**

Es un tipo de operador. Los booleanos en Python se implementan como una subclase de enteros. Solo hay dos valores booleanos, **Py\_False** y **Py\_True**. Como tal, las funciones normales de creación y eliminación no se aplican a los valores booleanos.

**¿Qué utilidad puede tener para el desarrollo de un programa?**

Son operadores fundamentales en programación junto con los operadores lógicos de comparación. Estos operadores se utilizan para comparar dos variables o expresiones y obtener un valor verdadero o falso.

**- Investigar qué significa que python sea un lenguaje de tipado dinámico.**

Quiere decir que en Python no tenemos que declarar el tipo de variable al asignar un valor a una variable. Otros lenguajes como C, C++, Java, etc., requieren una estricta declaración de variables antes de asignarles valores.

Python no tiene ningún problema incluso si no declaramos el tipo de variable. Indica el tipo de variable durante la ejecución del programa.

Ejemplo:

## asignar un valor a una variable

X = [1, 2, 3]

## x es una lista aquí

print(tipo (x))

## reasignando un valor a la 'x'

x = True

## x es un booleano aquí

print(tipo (x))

**Output**

<clase 'lista'>

<clase 'bool'>

**- Investigar y documentar sobre la creación de Módulos en Python.**

Los módulos en Python son simplemente archivos Python con la extensión .py, que implementa un conjunto de funciones. Los módulos son importados desde otros módulos usando el comando import.

Para importar un módulo, usamos el comando import. La primera vez que se carga un módulo dentro de un script Python en ejecución, éste se inicializa ejecutando el código del módulo una vez. Si otro módulo en tu código importa el mismo módulo de nuevo, éste no se cargará una segunda vez - de modo que las variables locales dentro del módulo actúan como un "singleton" - solo son inicializadas una vez.

Si queremos importar el módulo urllib, que nos permite crear datos de lectura desde URLs, simplemente import (importamos) el módulo:

# importamos la librería

import urllib

# la usamos

urllib.urlopen(...)

Hay dos funciones muy importantes que son útiles cuando se exploran módulos en Python: las funciones dir y help.

Podemos buscar qué funciones se implementan en cada módulo usando la función dir:

Cuando encontramos la función en el módulo que queremos usar, poder leer más sobre ella usando la función help, dentro del intérprete de Python:

help(urllib.urlopen)

**- Investigar y documentar sobre la creación de Paquetes en Python.**

**- Investigar e implementar el uso del archivo \_\_init\_\_.py:**

Los paquetes son espacios de nombre que contienen múltiples paquetes y módulos múltiples. Son simplemente directorios, pero dándole un giro.

Cada paquete en Python es un directorio que debe contener un archivo especial llamado \_\_init\_\_ py. Este archivo puede estar vacío, e indica que el contenido del directorio es un paquete Python, así que puede ser importado de la misma manera que se importa un módulo.

Si creamos un directorio llamado foo, que marca el nombre del paquete, podemos crear un módulo dentro del paquete llamado bar. Además no debemos olvidarnos de agregar el fichero \_\_init\_\_.py dentro del directorio foo.

Para usar el módulo bar, podemos importarlo de dos maneras:

import foo.bar

o:

from foo import bar

Con el primer método, debemos usar el prefijo foo cuando accedemos al módulo bar. En el segundo método no lo necesitamos, porque importamos el módulo al espacio de nombres de nuestro módulo.

El archivo \_\_init\_\_.py puede además decidir que módulos exporta el paquete como API, mientras mantiene los otros módulos internamente, sobreescribiendo la variable \_\_all\_\_, tal que así:

\_\_init\_\_.py

\_\_all\_\_ = ["bar"]