False : El tipo booleano sólo puede tener dos valores: True (verdadero) y **False** (falso).
ejemplo:

"Edwin" -> True

"" -> False

[3, 4] -> True

[] -> False

{"0": "Cero"} -> True

{} -> False

30 -> True

0.0 -> False

None -> False

Class: En python, la palabra clave class crea una clase. Un objeto se crea utilizando el constructor de la clase. Este objeto se llamará entonces la instancia de la clase.

ejemplo:

```
class Alumno:

    pass
```

```
class Alumno:

    def saludar(self):

        """Imprime un saludo en pantalla."""

        print("¡Hola, mundo!")
```

```
        def __init__(self):
            self.nombre = "Pablo"


        def saludar(self):
            """Imprime un saludo en pantalla."""
            print(f"¡Hola, {self.nombre}!")
```

Finally:puede ubicarse un bloque finally donde se escriben las sentencias de finalización, que son típicamente acciones de limpieza.
ejemplo:

```
try:
    archivo = open("miarchivo.txt")
    # procesar el archivo
except IOError:
    print "Error de entrada/salida."
    # realizar procesamiento adicional
```

```
except:
    # procesar la excepción
finally:
    # si el archivo no está cerrado hay que cerrarlo
    if not(archivo.closed):
        archivo.close()
```

is: El operador **is** se utiliza para comparar la identidad de dos objetos.
ejemplo:

```
a = 3
b = 3
c = 4
print a is b # muestra True
print a is not b # muestra False

print a is not c # muestra True
```

`Return:` La instrucción return indica el final de la función pero también el valor que devuelve la función.

ejemplo:

```
def escribe_media():
    media = (a + b) / 2
     print(f"La media de {a} y {b}
es: {media}")
    return

a = 3
b = 5
escribe_media()
print("Programa terminado")
```

None: El objeto None de Python, denota falta de valor. Este objeto no tiene métodos.
ejemplo:
if posicion is None: **print("No se encontró el elemento.")**

Continue: La instrucción continue da la opción de omitir la parte de un bucle en la que se activa una condición externa, pero continuar para completar el resto del bucle.
ejemplo:

```
number = 0

for number in range(10):
    if number == 5:
        continue    # continue here

    print('Number is ' + str(number))

print('Out of loop'
```

`For:`Un bucle for establece la variable iteradora en cada valor de una lista, arreglo o cadena proporcionada y repite el código en el cuerpo del bucle for para cada valor de la variable iteradora.
ejemplo:

```
nums = [4, 78, 9, 84]
for n in nums:
    print(n)

4
78
9
84
```

`Lambda:`Las expresiones lambda en Python son una forma corta de declarar funciones pequeñas y anónimas (no es necesario proporcionar un nombre para las funciones lambda). Las funciones Lambda se comportan como funciones normales declaradas con la palabra clave def
ejemplo:

```
mi_lista = [18, -3, 5, 0, -1, 12]
lista_nueva = list(filter(lambda x: x > 0, mi_lista))
print(lista_nueva) # [18, 5, 12]
```

Try: El bloque try es el bloque con las sentencias que quieres ejecutar. Sin embargo, podrían llegar a haber errores de ejecución y el bloque se dejará de ejecutarse.
ejemplo:

```
try:
    res = dividir(num, div)
    print(res)
except ZeroDivisionError:
    print("Trataste de dividir entre cero :( ")
```

True: Los booleanos son un tipo de dato primitivo comúnmente usado en lenguajes de programación. Por definición, un booleano tiene 2 posibles valores: true (verdadero) o false (falso).
ejemplo:

```
def esPar(n):
resultado = False
if n:
    n = True
return n


resultado = esPar(6)
print(resultado)
```

def:def es una **palabra reservada que indica a Python que una nueva función está siendo definida**. Luego viene una función con un nombre válido de tu elección. Los nombres válidos empiezan con una letra o un guion bajo, pero pueden incluir números.
ejemplo:

```
def mi_funcion(nombre, apellido):
    nombre_completo = nombre, apellido
    print nombre_completo

# Retornará el error: NameError: name 'nombre' is not defined

print nombre
```