**FUNCIONES EN PYTHON**

Las funciones se declaran con la palabra reservada **def**, seguido del nombre de la funcion y ()

Al igual que el resto de bloques de código en Python, tiene vital importancia la tabulación a la hora de definir el alcance del mismo.

def elevar(num):

    num \*\*= 2

    print(num)

elevar(6)

Este sería un ejemplo de una funcion que “eleva al cuadrado” el numero que se le pase por parámetro.

**DIFERENCIA ENTRE PARAMETRO Y ARGUMENTO**

La principal diferencia es de “ubicación”. Podríamos decir que los parámetros son las variables que se declaran en la función, y los argumentos, el valor que se le “pasa” al momento de ejecutarla.

En el ejemplo anterior, **num 🡪 es el parámetro**, mientras que **6🡪 es el argumento**

**PASO DE PARAMETROS**

Cuando se pasan parámetros a una funcion, los mismos se separan por comas dentro de los (), y al ser de un tipado dinámico, no es necesario declararles tipo alguno.

**FUNCIONAMIENTO DE UNA FUNCION**

Cuando se ejecuta una funcion, el flujo del programa se modifica para “trasladarse” hasta el lugar en que la funcion se encuentra declarada. Una vez que termine la ejecución (o no), el programa continuará desde la línea en que se invocó la function.

**RETURN**

La palabra return, funciona igual que en todos los lenguajes de programación, finaliza el flujo de la función, y “devuelve” en la línea que se invocó dicha función el valor de retorno, o lo que se haya encargado retornar. De esta forma, se puede guardar en una variable la invocación de una variable, para obtener su retorno en la misma.

**PASO DE PARAMETROS POR DEFAULT**

Los parámetros por default se utilizan en caso de que la funcion se invoque sin establecer argumentos. En caso de que se envíen argumentos, estos serán de mayor prioridad que los default.   
Para establecerlos, al lado de su declaración en los () se le asigna su valor:

def sumar(num1=1, num2=2):

    suma = num1 + num2

    print(suma)

**PARAMETROS VARIABLES**

Cuando nos referimos a parámetros variables, nos referimos a que se desconoce la cantidad de parámetros que van a enviarse.

Para especificar que se van a introducir parámetros variables, en la definición de la funcion, al parámetro se le antepone el signo \*

Conceptualmente, cuando se reciben parámetros con \*, se convierten a clase **tuple**

def listar(numero, \*nombres, \*\*bandas):

    for name in nombres:

        print(name)

    else:

        print(f'el numero introducido fue {numero}')

    for band in bandas:

        print(band)

listar(2, ('agustin', 'ignacio', 'silvana', 'alejandro'),

       ('foo', 'beatles', 'nirvana'))

En el ejemplo se rebuscó la situación, pidiendo un parámetro fijo, junto con 2 variables.

Cuando se pasan los argumentos, es más práctico separar entre paréntesis y comas correctamente.

**DICCIONARIOS COMO PARAMETROS**

Se pueden definir diccionarios como parámetros anteponiendo \*\*al parámetro en la def de la funcion

def listar(\*\*terminos):

    for k, v in terminos.items():

        print(f'Llave: {k}, Value: {v}')

Es importante reconocer que de este modo solo funcionaría si se enviasen los argumentos de la siguiente manera:

listar(a='agustin', b='ballena', c='cancion', d='diccionario')

Si se quisiese enviar una variable que contenga un diccionario, debería definirse de otro modo la variable.

def listar2(terminos):

    for k, v in terminos.items():

        print(k, v)

y enviar los argumentos así:

diccionario = {

    'a': 'agustin',

    'b': 'ballena',

    'c': 'cancion',

    'd': 'dinosaurio'

}

listar2((diccionario))

**FUNCIONES RECURSIVAS**

Funciones recursivas son funciones que se llaman así mismas hasta completar operación

Este es un ejemplo de un algoritmo con funciones recursivas

entrada = int(input('introduce un numero para calcular su factorial '))

def calcularFactorial(numero):

    if numero == 1:

        return 1

    else:

        return numero \* calcularFactorial(numero-1)

print(f'el factorial de {entrada} es {calcularFactorial(entrada)}')

Este algoritmo calcula el factorial de un número.

Procedimiento:

* Como se ve, se pide por parámetro el numero a calcular
* Si este numero es 1, su factorial será uno, por tanto, se detiene aquí. En caso de que no sea, el valor de retorno será la multiplicación de este numero introducido (distinto de 1) por el llamado a la funcion pero con el numero anterior.
* Esta acción hace que se mantenga “en espera” esa operación hasta que se obtenga el valor de (numero-1)
* Este llamado se repite hasta que numero sea == 1 y se comiencen a resolver las operaciones que quedaron “pospuestas”.
* De este modo, una vez llegado a uno comenzarán a resolverse las operaciones anteriores llevando el valor del producto a **calcularFactorial(numero-1)** de modo que cuando se llegue al numero introducido, se multiplicará este por el producto de todos sus anteriores.

**OTRA SOLUCION AL PROBLEMA SIN FUNCIONES RECURSIVAS**

numero = int(input('introduce un numero para calcular su factorial '))

fact = 1

for i in range(1, numero+1):

    mult = 1 \* i

    fact = fact \* mult

    print(fact)

Esta es otra solución, que en lugar de “correr hacia atrás”, avanza siguiendo hasta “toparse” con el numero introducido y arrastra el producto de los números anteriores.