# Unidad 5

## Repaso

En la unidad pasada estuvimos trabajando sobre estructuras de control tanto condicionales como de repetición**.** En esta unidad abordaremos la forma de desarrollar nuestras propias funciones con el objetivo de reutilizar código en forma sistemática.

## Introducción

Una función, es la forma de agrupar expresiones y sentencias (algoritmos) que realicen determinadas acciones, pero que éstas, solo se ejecuten cuando son llamadas. Es decir, que, al colocar un algoritmo dentro de una función, al ejecutar el programa, el algoritmo no será invocado si no se ha hecho una referencia a la función que lo contiene.

En Python, la definición de funciones se realiza mediante la palabra clave **def** más un nombre de función descriptivo, para el cuál, aplican las mismas reglas que para el nombre de las variables seguido de paréntesis de apertura y cierre. Como toda estructura de control en Python, la definición de la función finaliza con dos puntos (:) y el algoritmo que la compone llevará indentación.

```
def mi_funcion():
    # aquí el algoritmo
```

Una función no será ejecutada hasta tanto no sea invocada. Para invocar una función, simplemente se la llama por su nombre:

```
def mi_funcion():
    print("Hola Mundo")

mi_funcion()
```

Cuando una función realice un retorno de datos, éstos, pueden ser asignados a una variable:

```
def funcion():
    return("Hola Mundo")
```

```
frase = funcion()
print(frase)
```

## Parámetros de una función

Un parámetro es un valor que la función espera recibir cuando sea llamada (invocada), a fin de ejecutar acciones en base al mismo. Una función puede esperar uno o más parámetros (que irán separados por una coma) o ninguno.

```
def mi_funcion(nombre, apellido):
    # algoritmo
```

Los parámetros que una función espera serán utilizados por ésta, dentro de su algoritmo, a modo de variables de ámbito local. Es decir, que los parámetros serán variables locales, a las cuáles solo la función podrá acceder:

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

Si quisiéramos acceder a esas variables locales, fuera de la función, obtendríamos un error.

## Parámetros por omisión

En Python, también es posible, asignar valores por defecto a los parámetros de las funciones. Esto significa, que la función podrá ser llamada con menos argumentos de los que espera:

```
def saludar(nombre, mensaje='Hola'):
        print(mensaje, nombre)

saludar('Pepe Grillo')
```

En este caso, se imprime: Hola Pepe Grillo ya que solo pasamos el primer parámetro y el segundo parámetro tomo el valor por omisión.

## Conclusiones

Una función, puede tener cualquier tipo de algoritmo y cualquier cantidad de ellos y, utilizar cualquiera de las características vistas hasta ahora. No obstante, ello, una buena práctica, indica que la finalidad de una función, debe ser realizar una única acción, reutilizable y por lo tanto, tan genérica como sea posible.

Nos encontramos cerrando esta quinta unidad, que tuvo por objetivo familiarizarnos con la idea de funciones y crear nuestras propias herramientas. En la próxima unidad trabajaremos sobre la manipulación de información y su almacenamiento mediante archivos.