Actividad I - Funciones

Exactas Programa

Invierno 2019

Esta serie de actividades están orientadas para practicar los contenidos vistos en la clase de introducción, específicamente la definición y uso de funciones.

> No copies y pegues el ítem anterior, escribí desde cero cada ejercicio. La idea es que aprendas a escribir en `Python`.
>
> Cada ejercicio que hagas, probalo en el Python Tutor (`http://pythontutor.com`).

Funciones

En programación, una *función* es nombre que se le da a un *pedazo* de programa. La idea es que este pedazo de programa resuelva *algo* y el definir una función permite utilizar ese *algo* varias veces sin necesidad de volver a escribir.

Hay dos puntos claves para una función:

- **Definición**: es el lugar del programa en donde se da el nombre a la función, qué parámetros recibe y qué hace. Se escriben las instrucciones en el lenguaje de programación que *implementan* lo que nos imaginamos que tiene que hacer. En `Python` las definiciones de funciones comienzan con la palabra clave `def`, con esto resulta fácil identificar dónde se define una función.
- **Uso**: también conocido como *invocación* o *llamado*. Corresponde con el lugar del programa en el que se *llama* a la función. Esto significa poner el nombre de la función que se quiere usar y los valores a los parámetros.

Veamos un ejemplo:

```
1 def devolver_2_por(a):
2     res = 2*a
3     return res
4
5 resultado = devolver_2_por(5)
6 print(resultado)
7
8 resultado = devolver_2_por(3)
9 print(resultado)
```

En las líneas 1 a 3 encontramos la definición de una función. A continuación de la palabra clave `def` viene el nombre que se le pone a la función. En este caso, a la función le pusimos `devolver_2_por` e intenta reflejar lo que hará la función (siempre es una buena práctica utilizar nombres descriptivos). En particular, las líneas 2 y 3 corresponden con la **implementación** de la función, que son las instrucciones que serán ejecutadas cuando se invoque a la función.

La línea 5 contiene un llamado a la función `devolver_2_por` y, a la vez, la asignación del resultado de la función a la variable llamada `resultado`. En este caso, a la función se la invoca con el valor 5 como parámetro, con lo que el valor que debería quedar asignado en `resultado` después de ejecutar esa línea es... pruebenlo y me cuentan.

La línea 8 tiene otro llamado a la función `devolver_2_por`. Hay dos cosas interesantes en esta línea, la primera es que el valor que devuelve esta función va a ser asignado a la misma variable que usamos antes (`resultado`) con lo que el valor anterior se perderá (se sobre-escribe). El segundo punto interesante es que ahora llamamos a la función con otro valor del parámetro (en este caso 3 en lugar de 5) con lo que el valor de retorno de la función (lo que devuelve) será diferente que en el caso anterior... tampoco les voy a decir cuánto, deberían poder probarlo o deducirlo.

Ejercicios para hacer

1. Completar y probar:

```
def devolver_numero_5():
    numero = <completar>
    return numero
```

2. Completar y probar:

```
def devolver_una_lista():
    lista = [5, 6, 7, 8, 9, 10]
    return <completar>
```

3. Completar y probar:

```
def devolver_misma_lista(una_lista):
    <completar>
```

4. Completar y probar:

```
def devolver_nombre(nombre):
    <completar>
```

5. Completar y probar:

```
def devolver_la_suma(numero1, numero2):
    suma = <completar>
    return suma
```

6. Completar y probar:

```
<Definir una funcion que sea la multiplicacion de dos numeros>
```