Guía de ejercicios I - Funciones

Verano 2023

Esta serie de actividades están orientadas para practicar los contenidos vistos en la parte introductoria. Se enfoca, específicamente, en la definición y uso de funciones.

> Escribí **desde cero** cada ejercicio, no copies y pegues el ítem anterior. 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 el nombre que se le da a una *parte* del programa (un pedazo) que cumple una tarea *particular*. La idea es que este pedazo de programa resuelva *algo* y, el definir una función, permite realizar ese *algo* **varias veces** sin necesidad de escribirlo de nuevo. Además, al ponerle un nombre, podemos identificar qué hace.

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 (con qué valores va a trabajar) y qué hace. En el cuerpo de la función (también se la conoce como su *implementación*) se escriben las instrucciones en el lenguaje de programación que *implementan* lo que queremos que haga. En `Python`, las definiciones de funciones comienzan con la palabra clave **def**, con esto resulta fácil identificar dónde se define una función. La línea en la que se define una función termina con un dos puntos (**:**).
- **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_el_doble_de(un_numero):
2     multiplicacion = 2 * un_numero
3     return multiplicacion
4
5 resultado = devolver_el_doble_de(5)
6 print(resultado)
7
8 resultado = devolver_el_doble_de(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, le pusimos **devolver_el_doble_de** y refleja lo que hará la función (siempre es una buena práctica utilizar nombres descriptivos). Además, toma un único parámetro que, dentro de la función, será identificado como **un_numero**. Esta línea debe tener al final dos puntos (:) que indican que a partir de ahí comienza su definición.

Las líneas 2 y 3 corresponden con la **implementación** de la función, que son las instrucciones que serán ejecutadas cuando se invoque o llame a la función. Notar que estas dos líneas son las únicas dentro de la función y eso se indica escribiendo ese texto más a la derecha (dejamos cuatro espacios a modo de *tabulación*).

La línea 5 contiene un llamado a la función devolver_el_doble_de 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... prueben y me cuentan.

La línea 8 tiene otro llamado a la función devolver_el_doble_de. 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

Completar y probar las siguientes funciones:

1.
```
def devolver_la_suma(numero1, numero2):
    suma = <completar>
    return suma
```

2.
```
def fahrenheit_a_celsius(temp_far):
    temp_cel = <completar, para pasar hay que restar 32, luego
    multiplicar por 5 y finalmente dividir por 9>
    return temp_cel
```

3.
```
def perimetro_cuadrado(lado):
    perim = <completar>
    return perim
```

4.
```
def area_rectangulo(lado1, lado2):
    area = <completar>
    return <completar>
```

5.
```
def obtener_valor(precio):
    <completar para obtener el valor a pagar
    dado un 35 porciento de descuento>
```

6.
```
def obtener_valor(precio, descuento):
    <completar para obtener el valor a pagar
    dado el descuento a aplicar>
```

7.
```
def es_tripla_pitagorica(cateto1, cateto2, hipotenusa):
    cuadrados_catetos = cateto1 * cateto1 + <completar>
    cuadrado_hipotenusa = <completar>
    es_tripla = cuadrado_catetos == cuadrado_hipotenusa
return es_tripla
```