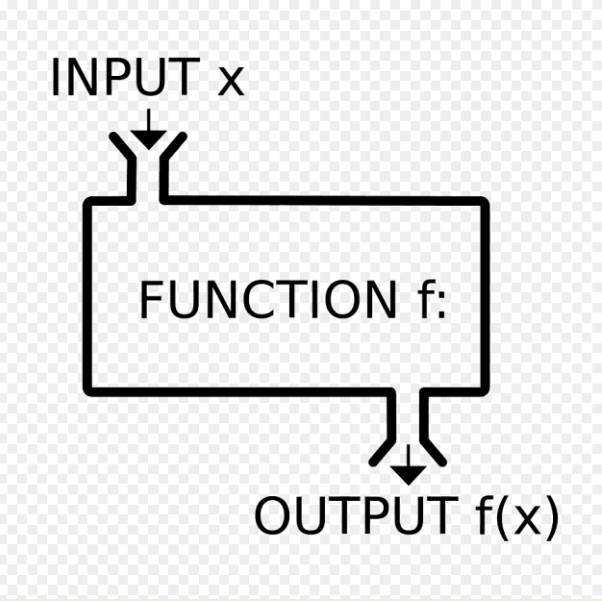
INTRODUCCIÓN A LA PROGRAMACIÓN

6.- MÓDULOS Y FUNCIONES

Rodrigo López A.

rilopez3@uc.cl

Funciones



Una función es una porción o bloque de código reutilizable que se encarga de realizar una determinada tarea.

Es fácil identificar una función por la presencia de paréntesis luego de su nombre.

Una función puede recibir o utilizar variables para su funcionamiento, a estas variables se les conoce como parámetros, y van en el interior de sus paréntesis.

Una función puede retornar variables o no retornar nada.

Ventajas de su uso:

- modularización: permite segmentar un programa complejo en una serie de partes o módulos más simples, facilitando así la programación y el depurado.
- reutilización: permite reutilizar una misma función en distintos programas.

Funciones

X = Función(parámetro1, parámetro2, …)

Para implementar funciones existen 2 grandes métodos:

- Incorporar grupos de funciones prediseñadas.
- Desarrollar nuestras propias funciones.

De momento nos enfocaremos en la primera opción, estos conjuntos de funciones prediseñadas, se suelen llamar “librerías”, “paquetes”, “módulos”, entre otros nombres.

Los módulos que uno utiliza están diseñados en forma de árbol o ramas, por lo cual uno puede decidir si agregar al proyecto todas las funciones del grupo o una de manera particular.

La forma en que se agrega la función influenciara la forma en que la utilizamos.

Importar módulos/librerías

```
#se agrega el modulo
import random
# llamado básico de random, nos genera un numero aleatorio entre [ 0 y 1 ]
print(random.random())


#otra forma de incorporar, en este caso solo se incorpora la función random, dentro de la librería random
from random import random
print(random())


#también es posible dar otro nombre a la función, esto es útil si tenemos funciones con nombres similares
from random import random as nombre_nuevo
print(nombre_nuevo())
```

- De momento comenzaremos con 2 librerías útiles, math y random
- Random: contiene funciones para generar números aleatorios
- Math: contiene un buen puñado de funciones para manejar números, hacer redondeos, sumatorios precisos, truncamientos… además de constantes.

```
import random   #se agrega el modulo


# Entero aleatorio >= 1 y <= 10
print(random.randint( 1, 10 ))

# Flotante aleatorio >= 0 y < 1.0
print(random.random())

# Flotante aleatorio >= 1 y <10.0
print(random.uniform( 1, 10 ))

# Entero aleatorio de 0 a 9, 10 excluido
print(random.randrange( 10 ))

# Entero aleatorio de 0 a 100
print(random.randrange( 0, 101 ))

# Entero aleatorio de 0 a 100 cada 2 números, múltiples de 2
print(random.randrange( 0, 101, 2 ))
```

```
import math   #se agrega el modulo


# Potencia de 2 a la 3, 2 elevado a 3
print(math.pow(2, 3))

# Raíz cuadrada de 9
print(math.sqrt( 9 ))

# Constante pi
print(math.pi )

# Constante e (Euler)
print(math.e )

#  Truncar un numero, quitar decimales
print(math.trunc( 10.15 ))

# Redondea un numero a la baja
print(math.floor( 10.15 ))

# Redondea un numero al alta
print(math.ceil( 10.15 ))
```

Importar funciones

Cuando trabajamos con Python, nos encontraremos con distintas categorías de funciones

- ***Modulos Built-in**: grupos de funciones que vienen incorporadas de manera base en Python (print, input, round…)*
- ***Módulos ya instalados**: son librerías que se encuentran instaladas en el entorno de trabajo, pero aun no se incorporan, por lo cual uno las importa cuando las necesita usar (math, random…)*
- ***Módulos no instalados**: muchas veces querremos usar un modulo nuevo o poco conocido, o queremos usar su ultima versión, en estos casos uno suele requerir instalar el modulo para luego incorporarlo (durante el curso no veremos esta categoría ☹)*
- ***Módulos propios**: en la medida que hacemos nuestras funciones útiles, las podemos incorporar en otros proyectos, y de esta forma acelerar el progreso.*

Módulos propios

De manera default, uno guarda los script de Python en archivos con extensión .py

El tipo de archivos con ejecuciones parciales, que hemos utilizado en colab o en un IDE, se logran mediante los cuadernos jupyter y se almacenan con extensión .ipynb.

Estos archivos, también se pueden guardar como archivos .py, en ese momento se unen todas las celdas de código en una sola ejecución secuencial.

Para incorporar funciones que hayamos realizado previamente tendremos que guardar esas funciones en un archivo con extensión .py, y luego incorporarlos en un nuevo proyecto.

Funciones propias

- Para avanzar hacia diseñar nuestras propias funciones tenemos que conocer los componentes de una función.

```
def nombre_funcion( parametro1, parametro2, ...):

    #algoritmo

    return dato_que_entrega_la_funcion #opcional
```

```
def nombre_funcion( parametro1, parametro2, ...):

    #algoritmo

    return dato_que_entrega_la_funcion #opcional
```

- Los componentes de una función consideran:
  - *Sentencia **def** que permite indicar que se procederá a definir una función*
  - *Se debe asignar un nombre único a la función (como buena practica, se recomienda usar el sistema snake_case para el nombre)*
  - *Uno puede definir si la función recibe datos para su ejecución, estos datos se conocen como **parámetros**, van dentro de un **paréntesis**, se puede crear la función **sin** necesitar parámetros o **con** tantos parámetros como unos desee.*
  - *Importante colocar los **dos puntos** **:** luego de los paréntesis*
  - *De manera opcional podemos definir si la función retornara un dato con la sentencia **return***

Ejemplo función hola

```
1  def hola_1(nombre):
2      print('Hola',nombre)
3
4  nombre = input()
5  hola_1(nombre)
```

```
Mundo
Hola Mundo
```

```
[2]
1  def hola_2():
2      nombre = input()
3      print('Hola',nombre)
4
5  hola_2()
```

```
Mundo
Hola Mundo
```

```
[3]
1  def hola_3():
2      nombre = input()
3      return 'Hola ' + nombre
4
5  print(hola_3())
```

```
Mundo
Hola Mundo
```

```
[4]
1  def hola_4(nombre):
2      return 'Hola ' + nombre
3
4  nombre = input()
5  print(hola_4(nombre))
```

```
Mundo
Hola Mundo
```

Scope, scope, scope

- Un concepto clave al trabajar en Python es la importancia del SCOPE (de momento no ha sido un gran problema)
- Cuando uno ejecuta una función, la función no se ejecuta de manera secuencial en el resto del código, se crea una instancia nueva… (es como si la ejecución de la función se hiciera en otro archivo, proyecto, los parámetros y el return es la forma de comunicación)

- SCOPE: es la forma en que Python identifica los nombre de objetos y las variables
- La disponibilidad de una función o variable depende de en que parte del código se creo
  - *Las variables que se definen dentro de una función, existen solo dentro de la función*
  - *Las variables que se definen fuera de una función, no existen en la función*
  - *AUNQUE TENGAN EL MISMO NOMBRE*

```
[1]  1  def numero_por_dos_1():
     2      print( x * 2 )
     3
     4  x = 5
     5  numero_por_dos_1()
     6
```

```
[2]  1  def numero_por_dos_2(x):
     2      print( x * 2 )
     3
     4  x = 5
     5  numero_por_dos_2(x)
```

¿Diferencias ?

¿Que podemos esperar en cada caso?

[1]
```python
def sucesor_1():
    x = input()
    print('El sucesor es', x + 1)

sucesor_1()
print('El numero era', x )

```

[2]
```python
def sucesor_2():
    x = input()
    return x

x = sucesor_1()
print('El sucesor es', x + 1)
print('El numero era', x )
```

¿Diferencias ?

¿Qué podemos esperar en cada caso?

- Archivo utiles.py:

```
def hola():
    print('hola mundo')
def minimo( x, y):
    if x < y:
        return x
    else:
        return y
```

```
#Nuevo proyecto:


import utiles


utiles.hola()
>> hola mundo


print(utiles. minimo(2, 5))
>> 2


utiles. minimo(2, 5)
```

Consideraciones

Para incorporar archivos es importante donde están alojados

Les recomiendo que si hacen pruebas en sus equipos locales, tengan los archivos que incorporaran en la misma carpeta
En caso contrario deberán usar sentencias para llegar al archivo, las cuales escapan a los contenidos vistos

ejemplo:

carpeta/carpeta/archivo.py
../../archivo.py

En colab, uno puede subir archivos, pero estos se almacenan solo mientras este la “sesión” activa, por lo cual uno requerirá volver a cargarlos posteriormente