Funciones y archivos con Python

@anaFranzoni

Notas tomadas del material de la Mtra. Teresa Sóla

Funciones

Una función es una secuencia de instrucciones que realizan alguna operación útil y se identifican con un nombre.

Las funciones pueden o no tomar argumentos y pueden o no producir un resultado.

Crear una función permite:

- da legibilidad al código
- facilita la depuración
- reduce el tamaño total del código, eliminando código repetitivo.
- permite reutilizar código.
  - cumple el principio DRY (don’t repeat yourself)

Las funciones bien diseñadas suelen ser útiles para muchos programas.

Funciones

Una función siempre inicia con la palabra reservada **def** *nombre***(***args***):**

Con la palabra reservada **return** se indica lo que devolverá la función.

Una función puede regresar más de un valor.

La función puede o no tener argumentos de entrada.

Los argumentos de entrada pueden ser llamados por nombre o tener valores x default o ser ignorados.

Funciones

- **Python** es un lenguaje que adopta el paradigma programación funcional permitiendo utilizar:
  - Funciones de orden superior
  - Funciones lambda o anónimas:
    - el contenido de este tipo de funciones debe ser una expresión y no un bloque de acciones.
    - No cuentan ni con el **def**, ni con el **return** que caracterizan a las funciones.

Python es un lenguaje que adopta el paradigma programación funcional permitiendo utilizar:

Iteraciones de orden superior

map, filter, reduce

los cuales

permiten reducir la cantidad de ciclos utilizados en los lenguajes imperativos.

función **lambda**, es una forma de crear funciones anónimas, (funciones sin nombre). Estas funciones son desechables, es decir, solo se necesitan donde se han creado. Utilizan principalmente en combinatorio con las funciones Map, Filter y Reduce. La sintaxis general de una función lambda es bastante simple:
lambda argument_list : expresison

El operador **Map(función, lista)**, toma una función y un iterable como argumentos, y devuelve un nuevo iterable con la función aplicada a cada argumento .

El operador **filter(función, lista)** filtra todos los elementos de una lista, necesita una función f (devuelve un valor booleano) como primer argumento, esta función se aplicará a cada elemento de la lista. Solo si f devuelve True, el elemento de la lista se incluirá en la lista de resultados.

La función **Reduce(función, lista)** reduce los valores de la *lista* a un solo valor aplicando una *función reductora*. El primer argumento es la *función reductora* que vamos aplicar y el segundo argumento es la lista.

Lectura de archivos de texto

```
id = open('archivo.txt','r')
contenido = id.read()
lista = contenido.split('\n')
id.close()
```

- El contenido del archivo de texto queda almacenado en una lista, cada renglón del archivo es un elemento de la lista (se utilizó el '\n' como separador de elementos.
- Una buena práctica de programación es no olvidar cerrar el archivo de lectura al terminar de utilizarlo.

Ejercicios

- 1)
- # crea una función que reciba una lista de CURP y devuelva el % de M y % H
- # IN lista de CURP
- # OUT diccionario
- 2)
- # crea una función que reciba una lista de CURP una lista con la cantidad de personas nacida x mes
- # IN lista de CURP
- # OUT lst de personas nacidas x mes [1] nacidas en enero, etc

Ejercicios

- 3)
- # QUE cuenta la cantidad de personas para cada estado existen en la lista de CURPs
- # IN   lista CURP
- # OUT diccionario llave = estado   valor = cantidad de personas
- 4) Inventa cuatro ejercicios semejantes en pareja.