Seguridad Informática

Programación con Python

Virgilio Castro Rendón
Fernando Marcos Parra Arroyo

Programación funcional

- Paradigma de programación que (como bien dice su nombre), radica en el uso de funciones.

- Las **funciones de orden superior** son la parte central de este paradigma.

- Las funciones de orden superior reciben otras funciones como argumentos, o regresan otras funciones como resultados.

Programación funcional

- Pasar una función como argumento:

```
def aplica_dos_veces(func,arg):
        return func(func(arg))

def suma_cinco(x):
        return x + 5

print aplica_dos_veces(suma_cinco,10)  ->  20
```

Programación funcional

- ¿Qué imprime el siguiente código?:

```
def test(func, arg):
        return func(func(arg))

def mult(x):
        return x * x

print test(mult, 2) -> ___
```

Funciones puras e impuras

- La programación funcional busca utilizar funciones “puras”.
- Una función pura no tiene efectos secundarios y devuelven un valor que dependen **únicamente** de sus argumentos.
- Se habla de pureza pues son como las funciones matemáticas, cos(x) siempre devolverá el mismo resultado para el mismo valor de x.

Funciones puras e impuras

- La programación funcional busca utilizar funciones “puras”.
- Una función pura no tiene efectos secundarios y devuelven un valor que dependen **únicamente** de sus argumentos.
- Se habla de pureza pues son como las funciones matemáticas, cos(x) siempre devolverá el mismo resultado para el mismo valor de x.

Funciones puras e impuras

```
lista_1 = [‘a1’,’a2’,’a3’]

def func_pura(x,y):
    temp = x + 2*y
    return temp / (2*x+y)

def func_impura(arg):
    lista_1.append(arg)
```

Funciones puras e impuras

¿Son puras estas funciones?

```
def func(x):
    y = x ** 2
    z = x + y
    z *= 2
    return x
```

```
def func(x, lista):
    lista_i = lista[0]
    lista_f = lista[-1]
    return lista_i + lista_f
```

Funciones puras e impuras

- Utilizar funciones puras, como todo, tiene sus ventajas y desventajas.
- “Más fáciles de analizar y probar”
- Más fáciles de ejecutar en paralelo
- La principal desventaja de utilizar funciones puras es que complican en gran medida operaciones de I/O, pues inherentemente tienen efectos secundarios.
- En algunos casos pueden ser más difíciles de escribir.

Funciones Lambda

- Crear una función (def) asigna una variable automáticamente (el nombre de la función).
- Objetos como las cadenas y números pueden ser creados sin ser asignados a una variable.
- Una función puede ser “anónima” (sin ser asignada a una variable), si se usa la notación lambda.
- Las funciones lambda no son tan potentes como las funciones nombradas.
- Los problemas se resuelven utilizando recursividad

Funciones Lambda

- Función con nombre

```
def func1(x):
        return x**2 + 5*x + 4
print func1(10)
```

- Función anónima

```
print (lambda x: x**2 + 5*x + 4)(10)
```

Funciones Lambda

- Función lambda que eleva al cuadrado el argumento

(lambda x: x**2) (10)

- Función lambda que multiplica dos números

(lambda x,y: x*y) (4,5)

- Función lambda que valida un palíndromo

(lambda x: x == x[::-1]) (‘anitalavalatina’)

Funciones Lambda

- Es posible asignar una función lambda a una variable. (Aunque esto claramente le quita lo anónimo)

```
pal = lambda x: x == x[::-1]
pal('anitalavalatina')
```

Funciones Lambda

- Escribir función lambda que multiplique tres números
- Escribir función lambda que valide si una lista está vacía
- Escribir función lambda que valide si una lista tiene al menos ‘n’ elementos
- Escribir función lambda que calcule la raíz n de un número
- Escribir función lambda que obtenga la intersección de dos conjuntos

Map, Filter y Reduce

- Son funciones de orden superior, pues reciben otra función como argumento.
- Como segundo argumento reciben un conjunto de elementos (lista, tupla, conjunto)
- Map aplica la función a todos los elementos del conjunto
- Filter genera una nueva lista con todos los elementos que cumplan una condición
- Reduce aplica una operación a cada par de elementos, guardando el resultado anterior como primer operador de la siguiente operación.
- Es común que la función que se pasa como argumento, sea una función anónima (lambda)

Map

- Aplica la función a todos los elementos del conjunto

```
op_interna = [‘quintero’,’Fernando’,’yEudiEL’]
map(lambda nombre: nombre.upper(), op_interna)

[‘QUINTERO’,’FERNANDO’,’YEUDIEL’]
```

Filter

- Genera una nueva lista con todos los elementos que cumplan una condición

jefes_area = [‘quintero’,’angie’,anduin’,’juan’,’celica’]

filter(lambda nombre: ‘i’ in nombre, jefes_area)

[‘quintero’,’angie’,anduin’,’celica’]

Reduce

- Aplica una operación a cada par de elementos, guardando el resultado del anterior como el primer operador de la siguiente operación.

```
num = range(10)
reduce(lambda x,y: x+y,num)
```

45

Ejercicio de clase 5

- Escribir una expresión funcional en UNA SOLA línea que contenga lo siguiente:
  - Función lambda que sume las tres listas
  - Función “filter” que regrese todas las cadenas que tiene un solo nombre
  - Función “map” que convierta a mayúsculas los nombres del resultado anterior
  - Función “reduce” que obtenga una cadena con los nombres resultantes, separando los nombres con coma.

  Descargar: e5.py

Biblioteca estándar de Python

- Es un conjunto de módulos que vienen por defecto en cualquier instalación de Python.
- Contienen funciones y clases útiles para una gran cantidad de aplicaciones.
- Algunos módulos de la biblioteca estándar cambian entre Python 2.x y Python 3.x
- Para utilizar los módulos, basta con importarlos al inicio de un script
  - from **modulo** import **funcion1,funcion2**
  - from **modulo** import *
  - import **modulo**
  - import **modulo** as **alias**

Algunos módulos de la biblioteca estándar

Módulo	Descripción
re	Contiene operaciones comunes para expresiones regulares.
string	Contiene operaciones comunes para cadenas, así como algunas cadenas ya definidas (mayúsculas, minúsculas, etc.).
socket	Contiene la clase Socket que implementa métodos para establecer conexiones entre clientes y servidores.
base64	Contiene funciones que permiten codificar y decodificar archivos y cadenas a base64.
datetime	Contiene diversos tipos para fechas y horas (hora actual, fecha actual) y métodos para operar con ellos.
sys	Parámetros y funciones específicas del sistema. Sirve para obtener argumentos de la línea de comandos.
subprocess	Administración de subprocesos. Permite ejecutar comandos propios del sistema operativo.

https://docs.python.org/2/library/index.html

Recopilando todo lo visto

Descargar: modulos.py, passwd.xml

- Importar módulos
- Listas
- POO
  - Clases (métodos __init__, __str__)
  - Objetos
- Definición de funciones
- Abrir archivos
  - Lectura
  - Escritura
- Función __main__
- Programación funcional
  - Lambda
  - Map
- Uso de la biblioteca estándar

Tarea 4 – parte 1

- Reportar:
  - En salida estándar y en un archivo
  - Hora de ejecución
  - MD5 de archivo xml
  - SHA1 de archivo xml
  - Cantidad de hosts prendidos
  - Cantidad de hosts apagados
  - Cantidad de hosts con puerto 22 abierto
  - Cantidad de hosts con puerto 53 abierto
  - Cantidad de hosts con puerto 80 abierto
  - Cantidad de hosts con puerto 443 abierto
  - Cantidad de hosts que tienen nombre de dominio
  - Servidores HTTP usados
    - Cuántos usan Apache
    - Cuántos honeypots (Dionaea)
    - Cuántos usan Nginx
    - Cuántos usan otros servicios

Tarea 4 – parte 2

- Archivo csv con direcciones IP
  - Hosts apagados
  - Hosts prendidos
  - Hosts con puerto 22 abierto
  - Hosts que son honeypots
  - Nombre de dominio, en caso de tener

  Descargar: nmap.xml