Buenas Practicas

Lección 4: Depuración y reglas de optimización de código.

eip ESCUELA INTERNACIONAL POSGRADOS

Indice

Introducción

En esta practica se pondrán a prueba los conocimientos adquiridos en esta lección respecto a la depuración y uso de herramientas para la misma y a la aplicación de reglas para la optimización de código.

Descripción del código

Librerías empleadas

Pdb, herramienta de depuración de código de python.

Actividad Primera:

Para la resolución de la primera actividad, se ha optado por implementar líneas de código “sueltas” a excepción de la función mayor, función que ha sido necesaria definir para el uso de las reglas de optimización de código. Se trata de una función cuyo cometido es devolver el numero de mayor valor de una lista de enteros.

```
ex = [[2, 4, 1], [1, 2, 3, 4, 5, 6, 7, 8], [100, 250, 43]]


def mayor(list):
    max = 0
    for elem in list:
        if elem > max:
            max = elem
    return max


pdb.set_trace()
maximos = list(map(mayor, ex))
print(maximos)
```

Uso de pdb

Al hacer uso de la librería pdb resulta muy cómodo examinar el código para hallar errores o simplemente entender como funciona un fragmento de código ya que permite ver de forma sencilla la evolución de las variables y el avance de los ciclos permitiendo hacer un seguimiento ágil y profundo del código. A termino personal, me ha resultado una herramienta muy útil y un gran sustituto para muchas situaciones del print().

```
> /home/elidas/Escritorio/Master/Buenas_Practicas/Leccion_4/Actividad_4.py(24)
<module>()
-> maximos = list(map(mayor, ex))
(Pdb) break 16
Breakpoint 1 at /home/elidas/Escritorio/Master/Buenas_Practicas/Leccion_4/Acti
vidad_4.py:16
(Pdb) continue
> /home/elidas/Escritorio/Master/Buenas_Practicas/Leccion_4/Actividad_4.py(16)
mayor()
-> max = 0
(Pdb) next
> /home/elidas/Escritorio/Master/Buenas_Practicas/Leccion_4/Actividad_4.py(17)
mayor()
-> for elem in list:
(Pdb) p list
[2, 4, 1]
(Pdb) next
> /home/elidas/Escritorio/Master/Buenas_Practicas/Leccion_4/Actividad_4.py(18)
mayor()
-> if elem > max:
(Pdb) p elem
2
(Pdb) next
> /home/elidas/Escritorio/Master/Buenas_Practicas/Leccion_4/Actividad_4.py(19)
mayor()
-> max = elem
(Pdb) p max
0
(Pdb) next
> /home/elidas/Escritorio/Master/Buenas_Practicas/Leccion_4/Actividad_4.py(17)
mayor()
-> for elem in list:
(Pdb) p max
2
```

Una vez definida la función, se ha empleado la función map para realizar la actividad primera, además se ha incluido la llamada a la función pbd para poder llevar a cabo el segundo apartado de esta primera actividad.

Segunda Actividad

Para la resolución de la segunda actividad, se han implementado tres funciones:

primo(n)

Dado un valor n devuelve “True” o “False” si este es primo o no.

verify_value()

Su cometido es verificar que no se introducen caracteres no admitidos en la lista que mas adelante será analizada, para ello hace uso de las estructuras Try Except

find_primos()

Haciendo uso de las dos funciones anteriores y de la función “filter” genera una lista de valores seleccionados por el usuario y devuelve los valores primos de la lista introducida.

```
def primo(n):
    for i in range(1, n):
        if n % i == 0 and i != 1:
            return False
    return True


def verify_value():
    while True:
        try:
            n = int(input('Introduzca un valor (-1 para finalizar):\n>>> '))
            if n < 0 and n != -1:
                raise SyntaxError
            return n
        except NameError:
            print('Valor no reconocido.')
        except SyntaxError:
            print('Valor no reconocido.')


def find_primos():
    val = list()
    n = 0
    while n != -1:
        n = verify_value()
        val.append(n)
    primos = list(filter(primo, val[:-1]))
    print(primos)


find_primos()
```