Lección 4 de Buenas Prácticas de Programación en Python

Adolfo Omar Calderón

In [1]:
```
import pdb
```

Ejercicio 1

In [2]:
```
def mayor(lst):
    pdb.set_trace()
    mayor = 1
    for i in lst:
        if i > mayor:
            mayor = i
    return mayor
```

Haciendo uso de la comprensión de listas cree dos con valores diferentes

In [ ]:
```
lista1 = [i for i in range(1,9)]
lista2 = [i for i in range(100, 500, 50)]
```

In [ ]:
```
print(mayor(lista1))
```

En el IDE utilice PDB en la función mayor que me devuelve el número de mayor valor dentro de una lista, con este puse el break en la línea de la varible local mayor cuando se define y cuando se actualiza. Fue interesante ver como la variable local se iba actualizando a medida que el código iba procediendo, me parece una herramienta muy útil para ver con profundidad el funcionamiento de un programa. Así se ve desde la terminal.

In [ ]:
```
(base) Omars-MacBook-Pro:leccion4 omarcalderon$ python actividad.py
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(6)mayor()
-> mayor = 1
(Pdb) 6
6
(Pdb) 9
9
(Pdb) b
(Pdb) b
(Pdb) c
8
(base) Omars-MacBook-Pro:leccion4 omarcalderon$ python actividad.py
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(6)mayor()
-> mayor = 1
(Pdb) break 6
Breakpoint 1 at /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py:6
(Pdb) break 9
Breakpoint 2 at /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py:9
(Pdb) continue
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(9)mayor()
-> mayor = i
(Pdb) list
  4     def mayor(lst):
  5         pdb.set_trace()
  6 B       mayor = 1
  7         for i in lst:
  8             if i > mayor:
  9 B->              mayor = i
 10         return mayor
 11
 12     lista1 = [i for i in range(1,9)]
 13     lista2 = [i for i in range(100, 500, 50)]
 14
(Pdb) next
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(7)mayor()
-> for i in lst:
(Pdb) next
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(8)mayor()
-> if i > mayor:
(Pdb) next
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(9)mayor()
-> mayor = i
(Pdb) next
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(7)mayor()
-> for i in lst:
(Pdb) p mayor
3
(Pdb) next
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(8)mayor()
-> if i > mayor:
(Pdb) next
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(9)mayor()
-> mayor = i
(Pdb) next
> /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py(7)mayor()
-> for i in lst:
(Pdb) p mayor
4
(Pdb) clear all breaks
*** Non-numeric breakpoint number all
*** Non-numeric breakpoint number breaks
(Pdb) clear 1
Deleted breakpoint 1 at /Users/omarcalderon/Desktop/MasterPython/buenas_practicas/leccion4/actividad.py:6
(Pdb) clear 2
```

Ejercicio 2

Aquí defini una función que toma como argumento un número y dice si es primo o no

In [3]:
```
def primos(n):
    primo = True
    for i in range(2, n):
        if(n % i == 0):
            primo = False
    return primo
```

In [4]:
```
lista3 = list(range(1, 200))
```

La definición de esta variable la hice utilizando el método filter que usará la lista3

In [ ]:
```
primos = list(filter(primos, lista3))
```

In [ ]:
```
print(primos)
```