



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Marco Antonio Martinez Quintana

Asignatura: Estructura de Datos y Algoritmos I

Grupo: 17

No de Práctica(s): 9

Integrante(s): Issac Alexander Guerrero Prado

*No. de Equipo de
cómputo empleado:*

No. de Lista o Brigada: 17

Semestre: 2

Fecha de entrega:

Observaciones:

CALIFICACIÓN: _____

Objetivo:

Aplicar las bases del lenguaje de programación Python

Variables y tipos:

- ♣ Los nombres de las variables son alfanuméricos y empiezan con una letra en minúscula.
- ♣ No se especifica el tipo de valor que una variable contiene, está implícito al momento de asignar un valor.
- ♣ No se necesita poner r; al final de cada instrucción.

```
practica9.py - /Users/astianax/Documents/Practicas python/practica 9/practica9.py
x = y = z = 10
print(x,y,z)
```

```
Python 3.8.2 Shell
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hola mundo")
Hola mundo
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py ==
>>> print("Hola mundo")
Hola mundo
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py ==
>>> print("10")
10
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py ==
>>> print("10 10 10")
10 10 10
>>> |
```

```
practica9.py - /Users/astianax/Documents/Practicas python/practica 9/practica9.py
x = y = z = 10
cadena = "Hola mundo"
print(x,y,z)
print (type(x))
print (type(cadena))
```

```
Python 3.8.2 Shell
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py ==
>>> print(x,y,z)
10 10 10
>>> print(type(x))
<class 'int'>
>>> print(type(cadena))
<class 'str'>
>>> |
```

Este código tiene una variable entera y una variable string, y como se ve en el código no se le asigna el tipo, como se explica esta implícito el tipo al asignarle un valor a la variable

Cadenas:

Las cadenas pueden ser definidas usando comilla simple (') o comilla doble ("). Una característica especial de las cadenas es que son inmutables, esto quiere decir que no se pueden cambiar los caracteres que contiene.

```
practica9.py - /Users/astianax/Docu >>>
cadena1 = "Hola "
cadena2 = 'Mundo'
print(cadena1)
print(cadena2)
cadena3 = cadena1 + cadena2
print(cadena3)
num_cadena = cadena3 + ' ' + str(3)
print(num_cadena)

== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
Hola
Mundo
Hola Mundo
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
Hola
Mundo
Hola Mundo
Hola Mundo 3
>>> |
```

Ln: 28 Col: 4

```
practica9.py - /Users/astianax/Documents/Practicas python Python 3.8.2 Shell
cadena1 = "Hola "
cadena2 = 'Mundo'
print(cadena1)
print(cadena2)
cadena3 = cadena1 + cadena2
print(cadena3)
num_cadena = "{1}{2} {0}".format(cadena1,cadena2, 3)
print(num_cadena)

Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
Hola
Mundo
Hola Mundo
Hola Mundo 3
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
Hola
Mundo
Hola Mundo
Mundo3 Hola
>>> |
```

Ln: 16 Col: 4

En el primer código se declaran dos variables de string, una que contiene un “Hola” y otra que contiene un “Mundo” y lo que hace la operación + es juntar las dos variables formando “Hola mundo”. También luego a la cadena hola mundo se le agrega un espacio vacío y un 3 al final

En el segundo código se acomodan las cadenas mediante la función format y con los números dentro de las llaves se les da el orden en el que va a quedar la posición dentro del resultado.

Operadores

Aritméticos: +, -, *, /

Booleanos: and, not, or

Comparación: >, =, <=, ==



```
practica9.py - /Users/astianax/Python 3.8.2 Shell
print(1 + 5)
print(6 * 3)
print(10 - 4)
print(100 / 50)
print(10 % 2)
print(((20 * 3) + (10+1)) / 10)
print(2 ** 2)

Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
6
18
6
2.0
0
7.1
4
>>>
```

Ln: 13 Col: 4

Este código imprime operaciones con todos los operadores aritméticos

Listas:

- ♣ Son valores que están separados por comas dentro de corchetes
- ♣ Está compuesta por cualquier cantidad y/o tipo de datos, ya sean cadenas, caracteres, números e inclusive otras listas.
- ♣ Se puede acceder a las listas por medio de índices, estos índices comienzan desde 0 hasta el número de elementos menos 1.
- ♣ Las listas son mutables.

```
practica9.py - /Users/astianax/Documents/Practicas Python 3.8.2 Shell
lista_diasDelMes=[31,28,31,30,31,30,31,31,30,31,30,31]
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
[31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
31
31
31
>>> |
```

Ln: 10 Col: 4

```
practica9.py - /Users/astianax/Documents/Practicas python/practica 9/practica9.py (3.8.2) Python 3.8.2 Shell
lista_numeros = [['cero', 0],['uno', 1, 'UNO'],['dos',2],['tres', 3], ['cuatro', 4], ['X', 5]]
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
[['cero', 0], ['uno', 1, 'UNO'], ['dos', 2], ['tres', 3], ['cuatro', 4], ['X', 5]]
[['cero', 0]
 ['uno', 1, 'UNO']
 dos
 2
 uno
 1
 UNO
 ['cinco', 5]
>>> |
```

Ln: 16 Col: 0

Ln: 15 Col: 4

El primer código tiene una lista que contiene los días que hay en cada mes e imprime respectivamente los objetos que están en la posición 0, 6 y 11 además de la lista en sí.

El segundo código es una lista que se conforma de otras 6 listas, y el programa imprime primero toda la lista, luego la primera lista contenida en la lista, luego el primer elemento de la segunda lista y el segundo elemento de la segunda lista, luego los 3 elementos que hay en la primera lista, y por ultimo al primer elemento de la lista 6 se le asigna el valor de “cinco” y luego se imprime la lista 6.

Tuplas:

Son parecidas a las listas, valores separados por una coma.

- ♣ Comparadas con las listas, las tuplas no son mutables.
- ♣ Se pueden aplicar las mismas operaciones que en las listas y su ventaja es que consumen menos memoria para almacenarse.
- ♣ Se crean, ya sea utilizando paréntesis o simplemente separando los valores por comas.

```
practica9.py - /Users/astianax/Documents/Practicas python/practica 9/practica9.py
tupla_diasDelMes=(31,28,31,30,31,30,31,31,30,31,30,31)

print (tupla_diasDelMes)
print (tupla_diasDelMes[0])
print (tupla_diasDelMes[3])
print (tupla_diasDelMes[1])

tupla_numeros=(( 'cero', 0), ('uno', 1, 'UNO'), ('dos', 2))

print(tupla_numeros)

print(tupla_numeros[0])
print(tupla_numeros[1])

print(tupla_numeros[2][0])
print(tupla_numeros[2][1])

print(tupla_numeros[1][0])
print(tupla_numeros[1][1])
print(tupla_numeros[1][2])

Python 3.8.2 Shell
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
31
30
28
(( 'cero', 0), ('uno', 1, 'UNO'), ('dos', 2), ('tres', 3), ('cuatro', 4), ('x', 5))
)
('cero', 0)
('uno', 1, 'UNO')
dos
2
uno
1
UNO
>>>
```

Ln: 18 Col: 4

```
practica9.py - /Users/astianax/Documents/Practicas python/practica 9/practica9.py (3.8.2)
tupla_diasDelMes=(31,28,31,30,31,30,31,31,30,31,30,31)

print (tupla_diasDelMes)
print (tupla_diasDelMes[0])
print (tupla_diasDelMes[3])
print (tupla_diasDelMes[1])

tupla_numeros=(( 'cero', 0), ('uno', 1, 'UNO'), ('dos', 2), ('tres', 3), ('cuatro', 4), ('x', 5))

print(tupla_numeros)

print(tupla_numeros[0])
print(tupla_numeros[1])

print(tupla_numeros[2][0])
print(tupla_numeros[2][1])

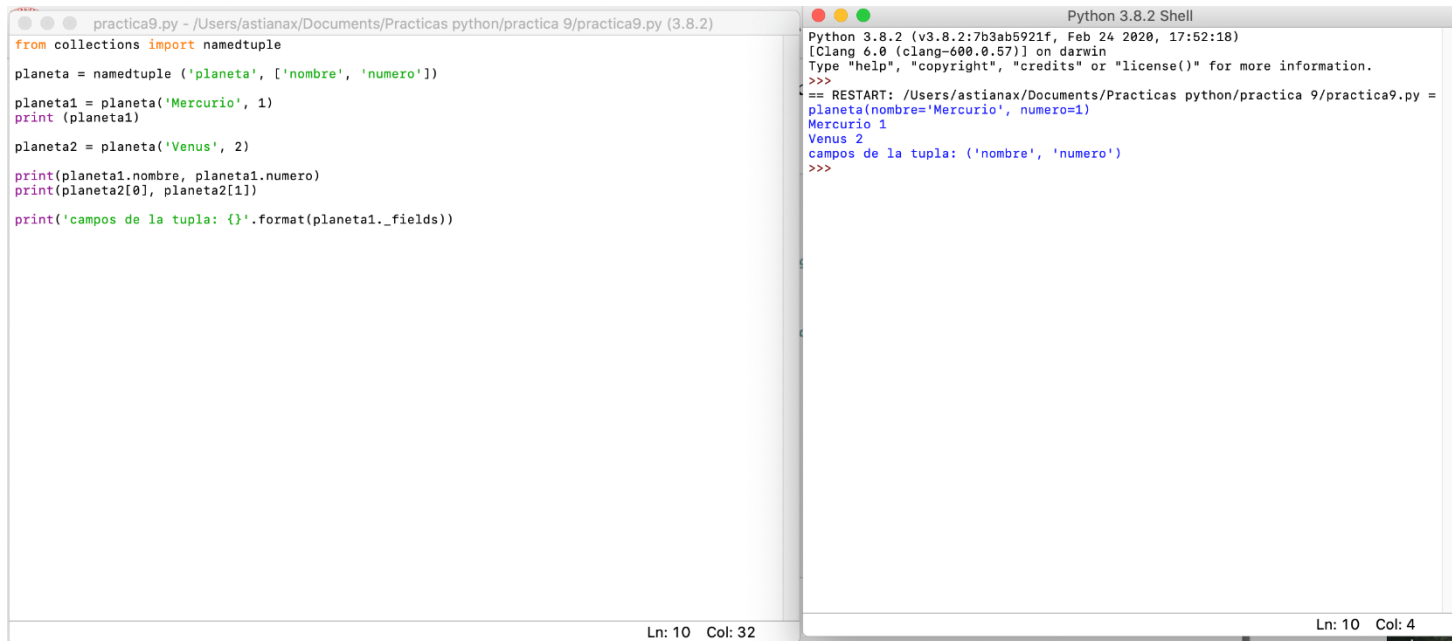
print(tupla_numeros[1][0])
print(tupla_numeros[1][1])
print(tupla_numeros[1][2])

tupla_diasDelMes[0] = 50

Python 3.8.2 Shell
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
31
30
28
(( 'cero', 0), ('uno', 1, 'UNO'), ('dos', 2), ('tres', 3), ('cuatro', 4), ('x', 5))
)
('cero', 0)
('uno', 1, 'UNO')
dos
2
uno
1
UNO
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
(31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
31
30
28
(( 'cero', 0), ('uno', 1, 'UNO'), ('dos', 2), ('tres', 3), ('cuatro', 4), ('x', 5))
)
('cero', 0)
('uno', 1, 'UNO')
dos
2
uno
1
UNO
>>>
Traceback (most recent call last):
  File "/Users/astianax/Documents/Practicas python/practica 9/practica9.py", line 22, in <module>
    tupla_diasDelMes[0] = 50
TypeError: 'tuple' object does not support item assignment
>>>
```

Ln: 22 Col: 24

Ln: 36 Col: 4



```
practica9.py - /Users/astianax/Documents/Practicas python/practica 9/practica9.py (3.8.2)
from collections import namedtuple

planeta = namedtuple ('planeta', ['nombre', 'numero'])

planeta1 = planeta('Mercurio', 1)
print (planeta1)

planeta2 = planeta('Venus', 2)

print(planeta1.nombre, planeta1.numero)
print(planeta2[0], planeta2[1])

print('campos de la tupla: {}'.format(planeta1._fields))

Python 3.8.2 Shell
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
planeta(nombre='Mercurio', numero=1)
Mercurio 1
Venus 2
campos de la tupla: ('nombre', 'numero')
>>>
```

El primer código consta de dos tuplas que son idénticas a las listas anteriores, una solo tiene una lista con elementos y accede a ellos mediante índices y el otro es una tupla que contiene más tuplas.

El segundo código solo cambia en que se intenta modificar el valor de la tupla y al momento de ejecutarlo marca error ya que la tupla es inmutable

El tercer código por medio de una biblioteca se hace una tupla con nombre. Que se comporta de forma similar a un struct en c

Diccionarios:

- ♣ Un diccionario se crea usando { } y consta de dos partes: llave y valor.
- ♣ Las llaves son inmutables, deben de tener un solo tipo de dato, una cadena o número. Una vez que es creado, no se puede cambiar su tipo.
- ♣ Mientras que el valor puede ser de cualquier tipo y se puede cambiar con el tiempo.
- ♣ Los elementos en un diccionario no están ordenados.

```
practica9.py - /Users/astianax/Documents/Practicas python/p
elementos = {'hidrogeno': 1, 'helio': 2, 'carbon': 6}

print (elementos)

print (elementos['hidrogeno'])

elementos['litio'] = 3
elementos['nitrogeno'] = 8

print (elementos)

print (elementos['helio'])

print (elementos.items())
print (elementos.keys())

Python 3.8.2 Shell
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
{'hidrogeno': 1, 'helio': 2, 'carbon': 6}
1
{'hidrogeno': 1, 'helio': 2, 'carbon': 6, 'litio': 3, 'nitrogeno': 8}
2
dict_items([('hidrogeno', 1), ('helio', 2), ('carbon', 6), ('litio', 3), ('nitro
geno', 8)])
dict_keys(['hidrogeno', 'helio', 'carbon', 'litio', 'nitrogeno'])
>>>
```

Ln: 12 Col: 4

Este código tiene una lista con 3 elementos, imprime primero los elementos, luego imprime el 1 contenido en el hidrogeno, después se agrega el litio que vale 3 y el nitrógeno que vale 8, se imprime el diccionario nuevamente con los elementos agregados y luego se imprime el numero del helio, por ultimo se imprime con el .items, todo el diccionario y con el .keys se imprimen solamente los elementos.

Funciones:

- ♣ Una función o procedimiento sirve para empaquetar código que sirve para ser reutilizado.
- ♣ Se puede usar ese mismo código con diferentes entradas y obtener resultados o comportamiento de acuerdo con esos datos.

```
practica9.py - /Users/astianax/Documents/Practicas python/p
def imprime_nombre(nombre):
    print("hola "+ nombre)

imprime_nombre("JJ")

def cuadrado(x):
    return x**2
x = 5

print("El cuadrado de {} es {}".format(x, cuadrado(x)))

def varios(x):
    return x**2, x**3, x**4
val1, val2, val3 = varios(2)
print("{} {} {}".format(val1, val2, val3))

val4, _, val5 = varios(2)
print("{} {}".format(val4, val5))

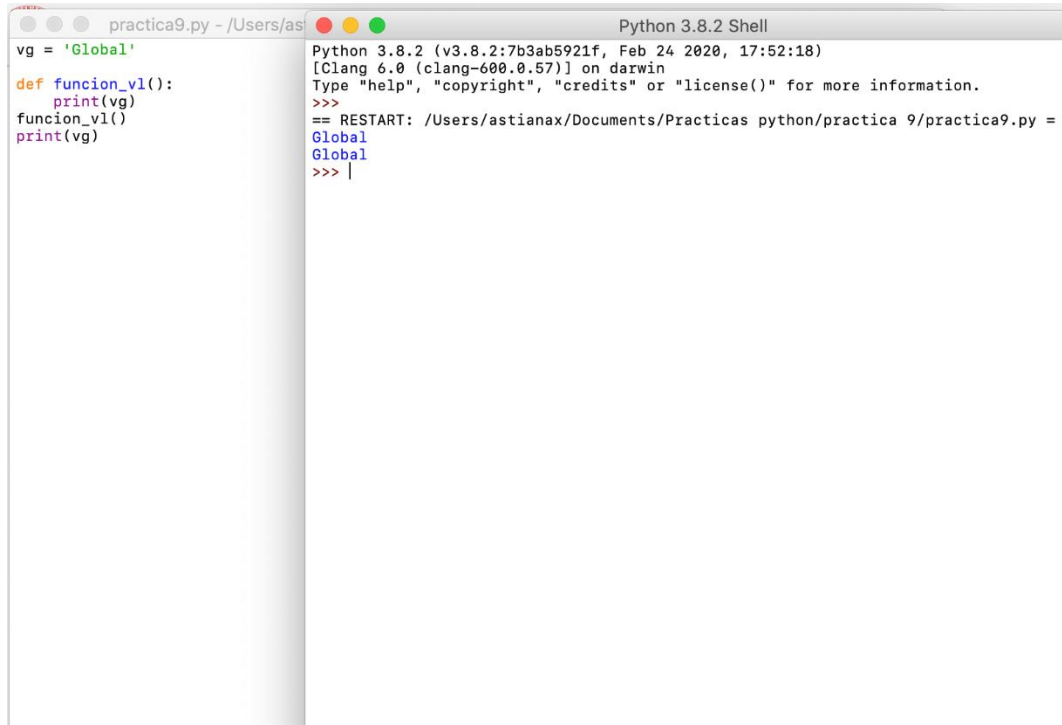
Python 3.8.2 Shell
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
hola JJ
El cuadrado de 5 es 25
4 8 16
4 16
>>> |
```

Ln: 10 Col: 4

Este código consta de 3 funciones, la primera imprime un “hola” y un nombre que se insertara en la función. La segunda función eleva al cuadrado un número y la última función eleva un numero al cuadrado, luego al cubo y luego a la cuarta potencia. Con ayuda del “_” se puede ignorar un valor, en este caso el número elevado al cubo.

Variables globales y locales:

Las variables globales son aquellas cuyo valor es reconocido en todas las partes del código y se pueden usar en cualquier situación y las variables locales son aquellas que se declaran en una función, y estas solo pueden ser utilizadas y solo son reconocidas dentro de la función en la que están contenidas.



```
practica9.py - /Users/as Python 3.8.2 Shell
vg = 'Global'
def funcion_v1():
    print(vg)
funcion_v1()
print(vg)

Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
Global
Global
>>> |
```

```
practica9.py - /Users/astianax/Documents/Practicas python/practica 9/practica9.py
vg = "Global"
def funcion_v2():
    vg = "Local"
    print(vg)
funcion_v2()
print(vg)
```

```
Python 3.8.2 Shell
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
Local
Global
>>> |
```

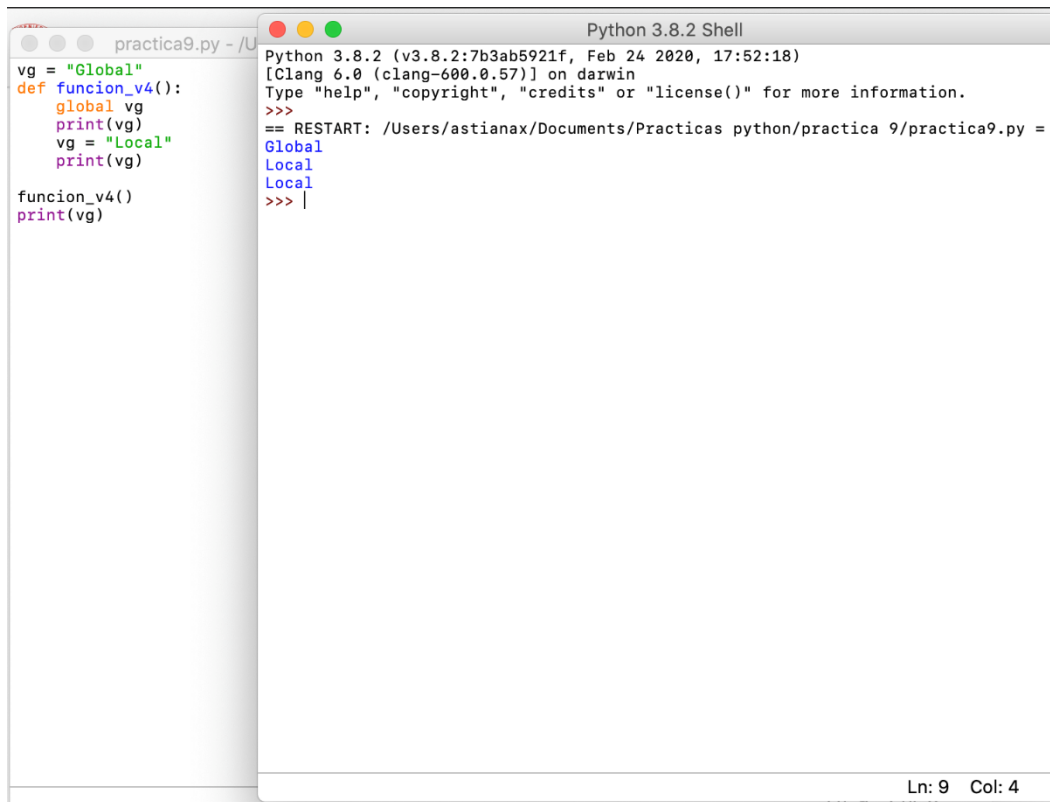
Ln: 8 Col: 4

```
practica9.py - /Users/astianax/Documents/Practicas python/practica 9/practica9.py
vg = "Global"
def funcion_v3():
    print(vg)
    vg = "Local"
    print(vg)
funcion_v3()
```

```
Python 3.8.2 Shell
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
Traceback (most recent call last):
  File "/Users/astianax/Documents/Practicas python/practica 9/practica9.py", line 6, in <module>
    funcion_v3()
  File "/Users/astianax/Documents/Practicas python/practica 9/practica9.py", line 3, in funcion_v3
    print(vg)
UnboundLocalError: local variable 'vg' referenced before assignment
>>>
```

Ln: 12 Col: 4

Ln: 1 Col: 12



```
practica9.py - /U
vg = "Global"
def funcion_v4():
    global vg
    print(vg)
    vg = "Local"
    print(vg)

funcion_v4()
print(vg)
```

```
Python 3.8.2 (v3.8.2:7b3ab5921f, Feb 24 2020, 17:52:18)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: /Users/astianax/Documents/Practicas python/practica 9/practica9.py =
Global
Local
Local
Local
>>> |
```

Ln: 9 Col: 4

En el primer código la variable `vg` es global entonces se puede usar con un `print` normal y también es válido imprimir la variable dentro de una función y el resultado es el mismo.

El segundo código tiene dos variables `vg` una global y una local, cada una con un valor diferente, como la que está en la función se mantiene de forma local, el cambio de valor solo se da dentro de la función, por lo que al imprimir la variable por fuera se imprime el valor global.

El tercer código intenta imprimir una variable global al igual que en el primer código, sin embargo, esta queda en el espacio local y no se le ha asignado ningún valor, entonces genera error.

El último código con la función global se especifica que se quiere usar la función global dentro de la función y con esto se resuelve el error que daba en el ejemplo anterior, hacer esto cambia el valor de la variable global en una función.

Conclusión

Python a diferencia de C en muchos aspectos tiene una forma de codificación más sencilla, pero en cuanto formato es bastante similar en varios aspectos, esto se resume en que si se maneja un lenguaje de programación se te hará más fácil poder ir aprendiendo más lenguajes, ya que aunque todos tengan sus diferencias, también van a tener varias cosas en común.

Bibliografía

<http://lcp02.fi-b.unam.mx/>

