Codo a Codo 4.0

FULL STACK PYTHON

html - css3 - bootstrap - javascript
vue.js - sql - python - django

Clase 28

Python – Parte 4

Temas: Funciones, Parametros vs Argumentos, Funcion Lambda, variables locales y globales

String

```
s = 'python'

type(s) → str

<str> + <str>
    → <str>

<str> * <int>
    → <str>

len(<str>)
s[i]

for c in <str>:

<str> in <str>
   → <bool>
```

Ordenado
Inmutable [i]

Listas

```
l = [8,'hola',False]

type(l) → list

<list> + <list>
    → <list>

<list> * <int>
    → <list>

len(<list>)
l[i]

for x in <list>:

<?> in <list>
   → <bool>
```

Ordenada
Mutable

Tuplas

```
t = (8,'hola',False)

type(t) → tuple

<tuple> + <tuple>
    → <tuple>

<tuple> * <int>
    → <tuple>

len(<tuple>)
t[i]

for x in <tuple>:

<?>in<tuple>→<bool>
```

Ordenada
Inmutable
Permite duplicados

Set

```
s={8,'hola',False}

type(s) → set

len(<set>)

for x in <set>:

<?> in <set>→<bool>
```

Desordenado
Inmutable
No permite duplicados
No Indexado

Dictionaries

```
d={1:'Juan',2:'Ana'}

type(d) → dict

len(<dict>)
d[i]

for i in diccionario.keys():

   print(diccionario[i])

)
```

Par de valores
Ordenado
Mutable
No acepta duplicados

FUNCIONES

La definición de la función comienza con:“def.”

Nombre de la función y sus argumentos

```
def obtener_cantidad(nombre_archivo):
    """String de documentación"""
    line1
    line2
    return contador
```

La identación importa…
La primer línea sin identación es considerada fuera de la función

La palabra clave ‘return’ indica que el valor será devuelvo a quien llamó a la función.

Parámetros vs Argumentos

En la definición de una función los valores que se reciben se denominan **parámetros**,

Pero durante la llamada de la funcion los valores que se envían se denominan argumentos

```
def superficieCuadrado(x):    # x es el parametro de la funcion
    return x*x


# Programa Principal
lado=float(input("Ingrese el valor del lado del cuadrado: "))
sup=superficieCuadrado(lado)   # lado es el argumento de la llamada de la funcion
print(f"La superficie del cuadrado de lado {lado} es igual a {sup}")
```

PARÁMETROS

Vamos Buenos Aires

Se encuentran al comienzo de la lista de parámetros y se relaciona con los argumentos por su posición.

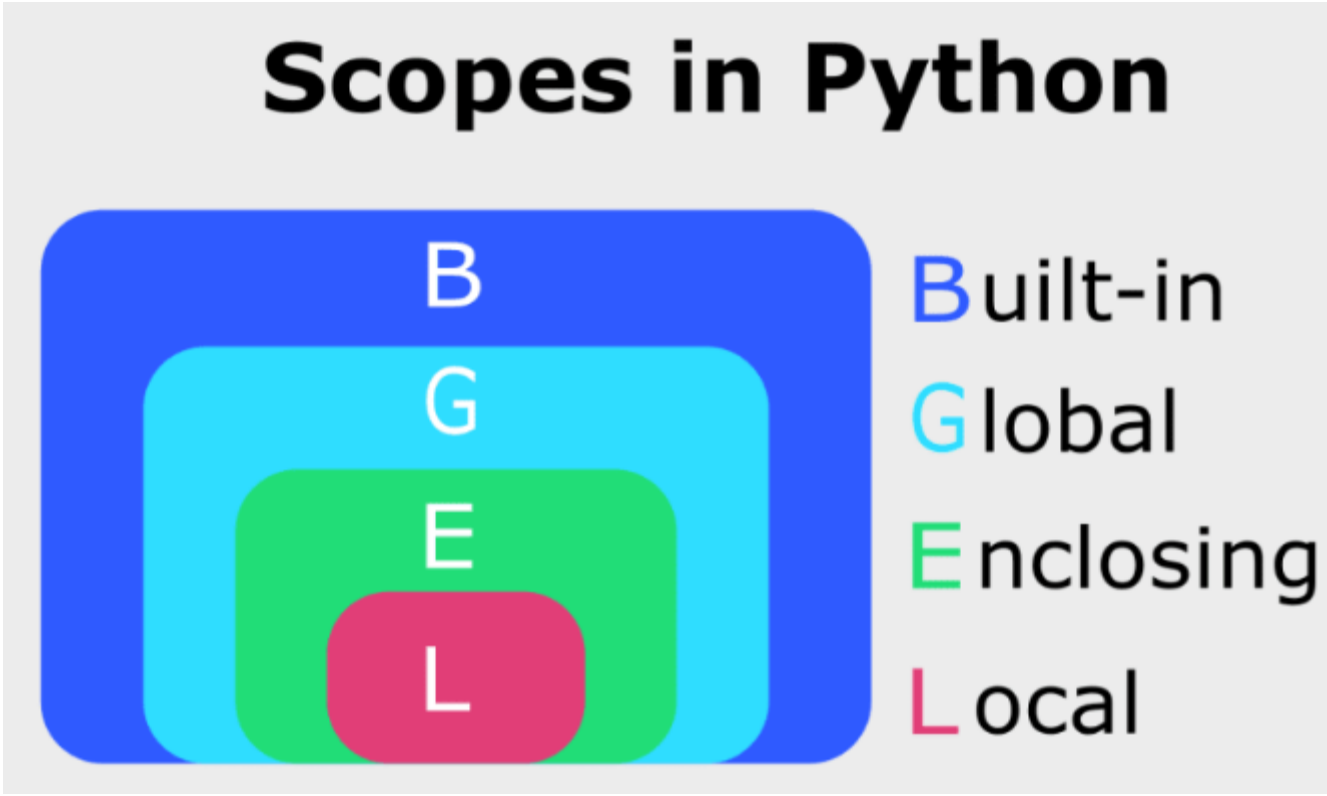
Posicionales

Se encuentran al final de la lista de parámetros y se relaciona con los argumentos por su posición o por nombre. Si no se especifica según nombre o posición toma el valor que tiene por defecto.
Tiene restricciones en la forma de uso para accederse por posición.

Por defecto

Sobrecarga de funciones?? NO

SCOPE DE LAS VARIABLES




```python
#Global Scope
x = 0
def funcion():
    #Enclosed scope
    x = 1
    def funcion_interna():
        #Local scope
        x= 2
        print(f"Local scope x={x}")

    funcion_interna()
    print(f"Enclosed scope x={x}")

funcion()
print(f"Global scope x={x}")
```

BUENAS PRÁCTICAS

- No utilizar variables globales desde dentro de una función
- No anidar definiciones de funciones
- Si se desea utilizar los valores de una variable del programa principal en una función, se debe pasar por parámetro

Funciones Lambda

- Permite definir funciones pequeñas y anónimas
  Tienen el mismo comportamiento que las funciones definidas con *def*
  Sintaxis
  Definición
  <variable> = lambda <lista de parám separados por ,> : <expresión>
  Invocación
  <variable> (<lista de argumentos separados por ,>
  Ejemplo:

```
>>> suma = lambda x , y : x + y
>>> suma ( 5 , 3 )
8
```

  Su equivalente usando una función *def* sería

```
>>> def suma ( x , y ):
        return x + y
>>> suma ( 5 , 3 )
8
```

- La funciones están restringidas a que devuelva un solo resultado
- La función lambda no se asocia con un nombre
- Antes de usar una función lambda verificar si esta es o no más clara que una función regular.