Funciones en Python

Bibliografía

“Aprender a Pensar como un Programador en Python”:

- Capítulo 3
- Capítulo 5 hasta 5.4 inclusive

Temario

- Estructura de un programa
- Presentación de funciones
- Importación de funciones
- Definición de funciones
- Ejemplos
- Ventajas
- Preguntas
- Conclusiones

Estructura de un programa

Sentencia: orden ejecutable que compone un programa.

Flujo de ejecución: forma en que se ejecutan las sentencias.

Estructuras de control:

Son el medio por el cual los programadores pueden determinar el flujo de ejecución en un programa

Concepto de abstracción:
¿Saben qué es?
¿Qué tiene que ver con los programas?

Estructura de un programa

Por lo general, nuestros programas están formados por “subprogramas” que pueden implementar distintas funcionalidades de mi programa.

En Python usamos **funciones!**

Funciones

Definición: Es una estructura que me permite abstraer funcionalidad de mi programa.
Está compuesta de una secuencia de sentencias o instrucciones que permiten resolver una determinada operación.

Ejemplo:

type(x)

Argumentos o parámetros

Nombre de la función

Funciones

Nosotros ya hicimos uso de muchas funciones escritas por otros.

¿Cuáles?

```
x=”a”
print (ord(x))
```

```
x=3
print (str(x))
```

```
x=input('Ingresa tu edad')
```

```
X=”10”
print (int(x))
```

Todas tienen “Nombre” y “Argumentos”

Funciones

Invocación de una función

```
x=”a”
print (ord(x))
```

Cuando queremos que se ejecute su código, debemos **invocar** a la función. En ese momento se especifican el o los **argumentos** necesarios

Funciones

Ejemplo 1:

```
>>> cadena = ‘Estamos explicando cadenas’
>>> print(‘La longitud de esta cadena es:’, len(cadena))
La longitud de esta cadena es:  26
>>>
```

Invocación de la función **len**

Ejemplo 2:

```
>>> x = 2
>>> print('La raiz cuadrada es: ', sqrt(x))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqrt' is not defined
>>>
```

¿Y acá qué pasó?

No reconoce a la función **sqrt**

¿Cómo importamos funciones?

Podemos utilizar la clausula import.

Alternativa 1

```
import math
x=2
print ('La raíz cuadrada es: ', math.sqrt(x))
```

Alternativa 2

```
from math import sqrt
x=2
print ('La raíz cuadrada es: ', sqrt(x))
```

¿Cómo definimos una función?

```
def nombreFuncion(parametros):
    sentencias
    return <expresion>
```

El cuerpo de la función **debe** estar indentado!

Ejemplo sencillo:

```
def cuadrado(x):
    return x **x
```

x es un **parámetro** a la función.

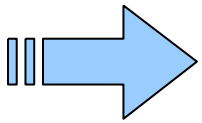
¿Cómo usamos o invocamos a la función?

```
print(cuadrado(3))
a = 2 + cuadrado(9)
```

Algunas consideraciones importantes

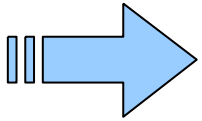
```
def mi_funcion(par1, par2, par3):
    print par1
    print par2
    print par3
```

```
mi_funcion(1,2,3)
```



Atención al orden de los parámetros!!!

```
a=1
b=23
c=34
mi_funcion(a,b,c)
```

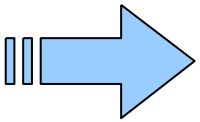


Parámetros reales vs. Parámetros formales

Algunas consideraciones importantes

```
def imprimo_mensaje():
    print "Hola a todos!!!!"

def retorno_mensaje():
    return "Hola a todos!!!!"
```



¿Qué diferencias hay?

Hagamos un ejemplo

Queremos trabajar con la lista de nuestros equipos de futbol. Queremos mostrar el equipo campeón, el equipo que más goles tuvo y el equipo que desciende de categoría (el que tiene menos puntos).

Podríamos pensar en tener:

.- una función que cargue los equipos

.- una función que calcule el campeon

.- una función que calcule el equipo goleador

.- una función que muestre el equipo que desciende

Hagamos un ejemplo

Queremos trabajar con la lista de nuestros equipos de futbol. Queremos mostrar el equipo campeón, el equipo que más goles tuvo y el equipo que desciende de categoría (el que tiene menos puntos).

```
def armo_lista():
    equipos=[["river", 10,1], ["boca", 3,2], ["racing", 50,50], ["estudiantes", 34, 2]]
    return equipos

def campeon(equipos):
    #resuelvo quien es el campeon
    #codigo
    #codigo
    #codigo
    #luego retorno el resultado
    return eq_campeon

def goleador(equipos):
    #resuelvo quien es el goleador
    #codigo
    #codigo
    #codigo
    #devuelvo el goleador
    return eq_goleador


def desciende(equipos):
    #resuelvo que equipo desciende (¿Racing?)
    #codigo
    #codigo
    #codigo
    #devuelvo el goleador
    return eq_desciende
```

```
mis_equipos=armo_lista()

print "El equipo campeón es " + campeon(mis_equipos)
print "El equipo goleador es " +goleador(mis_equipos)
print "El equipo que desciende es "+ desciende(mis_equipos)
```

¿Qué ventajas tiene esto?

- Nos enfocamos a resolver cada función y no todo el programa de un solo paso (Concepto de Abstracción)
- Podemos re-implementar una función sin modificar todo el programa
  - Por ejemplo, podríamos ingresar los equipos desde el teclado y el resto del programa sigue funcionando igual
- Podemos evitar escribir muchas veces el mismo código.
  - Simplemente, invocamos a la función.

¿Se les ocurre otras ventajas?

¿Calculamos?

```
def calculo_suma(x, y):
    return x+y

def calculo_producto(x, y):
    return x*y

def calculo_division(x, y):
    return x/y

def calculo_resta(x, y):
    return x-y


print "Ingrese el primer operando"
x=input()

print "Ingrese el segundo operando"
y=input()

print "La SUMA de los varlores que ingresaste es ", calculo_suma(x,y)

print "La RESTA de los varlores que ingresaste es ", calculo_resta(x,y)

print "El PRODUCTO de los varlores que ingresaste es ", calculo_producto(x,y)

print "La DIVISION de los varlores que ingresaste es ", calculo_division(x,y)
```

Mis Funciones

Acá las invocamos

¿Preguntas?

Conclusiones