Curso Desarrollo Web con Python y Django

Guía 3

Estructuras y Sentencias

Python no se usan llaves para separar bloques de código ni puntos y coma al final de cada línea

Para segmentar bloques de código siempre se hará con **4 espacios** o **1 tabulación** a este proceso le llamamos **IDENTACION**

Funciones

En Python, la definición de funciones se realiza mediante la instrucción def más un nombre de función descriptivo -para el cuál, aplican las mismas reglas que para el nombre de las variables- seguido de paréntesis de apertura y cierre. Como toda estructura de control en Python, la definición de la función finaliza con dos puntos (:) y el algoritmo que la compone, irá identado con 4 espacios:

Funciones sin parámetros

Ejemplo 1

```
1  def mi_primera_funcion():
2      return "Hola Mundo Cruel!"
3
4  mi_primera_funcion()
```

En el anterior código creamos una función y retornamos un mensaje luego en la línea 4 llamamos o invocamos a esa función pero no muestra nada en pantalla verdad?

```
1  def mi_primera_funcion():
2      return "Hola Mundo Cruel!"
3
4  frase = mi_primera_funcion()
5  print frase
```

Ahora si agregamos una variable llamada **frase** y le asignamos lo que retorna **mi_primera_funcion()** pues sencillamente **frase** tendrá almacenado la cadena **"Hola Mundo Cruel!"**

Funciones con Parámetros
Ejemplo 2

```
1  def my_segunda_funcion(nombre, apellido):
2      nombre_completo = nombre, apellido
3      print nombre_completo
4
5  nombre = "Mariana"
6  apellido = "Prado"
7  my_segunda_funcion(nombre, apellido)
```

Otro ejemplo más completo

```
1  def my_segunda_funcion(p1, p2):
2      nombre_completo = p1 + p2
3      return "Hola madre "+nombre_completo+" esta triunfando en la vida!"
4
5  nombre = "Diego "
6  apellido = "Prado"
7  frase = my_segunda_funcion(nombre, apellido)
8  print frase
```

Conversión de tipos

Para convertir un número real a entero es decir que elimina la parte decimal

```
>>> int(1.3)
1
```

Convertir un número 2 a un carácter “2”

```
>>> str(2)
'2'
```

Para convertir un entero a un número flotante o real

```
>>> float(1)
1.0
```

Convertir una lista a una tupla

```
>>> tuple([1,2,3])
(1, 2, 3)
```

Convertir una tupla a una lista

```
>>> list((1,2,3))
[1, 2, 3]
```

Para ver la longitud de una cadena o ver la cantidad de elementos de una lista

```
>>> len("Python Mola")
```

```
11
>>> len([1,2,3,4])
4
```

Para crear una lista desde 0 hasta 5

```
>>> range(5)
[0, 1, 2, 3, 4]
```

Para crear una lista desde 1 hasta 7

```
>>> range(1,7)
[1, 2, 3, 4, 5, 6]
```

Para crear una lista desde 1 ahasta 7 de 2 en 2

```
>>> range(1,7,2)
[1, 3, 5]
```

Si quieres averiguar cuál es el tipo de dato de una variable u objeto

```
>>> type(True)
<type 'bool'>
>>> type("Python Mola")
<type 'str'>
```

Quieres sumar una lista de números

```
>>> sum([0,1,2,3,4])
10
>>> sum(range(5))
10
```

Otros ejemplos

```
ord("s")          # codigo ascci
chr(98)           #codigo ascci a caracter
abs(-4)           #valor absoluto
float(num)        #convierte a flotante
int(num)          #convierte a entero
str(val)          #convierte a cadena
round(num)        #redondear numero
str(val)          #convierte a cadena el valor

cad = "estamos en la parte inicial"
cad.lower()       #texto en minusculas
cad.upper()       #textos en mayusculas
cad.title()       #textos con letras capital
cad.replace("inicial","final")

range(2,12,2)     #lista desde 2 hasta el 12 de 2 en 2
```

Sentencias

If-else

```
nombre = "Diego"
if nombre == "Diego":
    print("Bienvenido Diego")
elif nombre == "Fernando":
    print("Hola Fercho")
else:
    print("Quien eres?")
```

while

```
contador = 0
while contador < 5:
    print "Numero %i" % contador
    contador += 1
```

for

```
for i in range(10):
    print "Numero %i" % i
```

try-except

```
try:
    result = 3 / 0
except:
    print "Division por Cero"
```

Clases

Las clases son los modelos sobre los cuáles se construirán nuestros objetos. Podemos tomar como ejemplo de clases, el gráfico que hicimos en la página 8 de este documento. En Python, una clase se define con la instrucción class seguida de un nombre genérico para el objeto.

```
class Objeto:
    pass
class Antena:
    pass
class Pelo:
    pass
class Ojo:
    pass
```

PEP 8: clases
El nombre de las clases se define en singular, utilizando CamelCase.

Propiedades

Las propiedades, como hemos visto antes, son las características intrínsecas del objeto. Éstas, se representan a modo de variables, solo que técnicamente, pasan a denominarse “propiedades”:

```
class Antena():
    color = ""
    longitud = ""
class Pelo():
    color = ""
    textura = ""
class Ojo():
    forma = ""
    color = ""
    tamanio = ""
class Objeto():
    color = ""
    tamanio = ""
    aspecto = ""
    antenas = Antena() # propiedad compuesta por el objeto objeto Antena
    ojos = Ojo() # propiedad compuesta por el objeto objeto Ojo
    pelos = Pelo() # propiedad compuesta por el objeto objeto Pelo
```

PEP 8: propiedades
Las propiedades se definen de la misma forma que las variables(aplican las mismas reglas de estilo).

Métodos

Los métodos son "funciones" (como las que vimos en el capítulo anterior), solo que técnicamente se denominan métodos, y representan acciones propias que puede realizar el objeto (y no otro):

```
class Objeto():
    color = "verde"
    tamanio = "grande"
    aspecto = "feo"
    antenas = Antena()
    ojos = Ojo()
    pelos = Pelo()

    def flotar(self):
        pass
```

Notar que el primer parámetro de un método, siempre debe ser **self**.

```
1  class Student(object):
2      def __init__(self, name, age):
3          self.name = name
4          self.age = age
5
6      def hello(self):
7          return 'Mi nombre es %s' % self.name
8
9  s = Student("Diego", 30)
```

Accediendo a los métodos y propiedades de un objeto

Una vez creado un objeto, es decir, una vez hecha la instancia de clase, es posible acceder a sus métodos y propiedades. Para ello, Python utiliza una sintaxis muy simple: el nombre del objeto, seguido de punto y la propiedad o método al cuál se desea acceder:

```
objeto = MiClase()
print objeto.propiedad
objeto.otra_propiedad = "Nuevo valor"
variable = objeto.metodo()
print variable
print objeto.otro_metodo()
```

Ejercicio 1

Crear una función Sumar 2 variables **a** y **b** que permita sumar las variables e imprima en pantalla su resultado

Ejercicio 2

Cree las funciones multiplicar, restar, dividir, sumar para dos números **a** y **b** e imprima en pantalla el resultado correspondiente

Ejercicio 3

Repita el ejercicio 2 con la condición que las funciones de operaciones multiplicar, sumar, restar y dividir reciba parámetro para a y b, guarde el archivo como **calculadora.py**

Ejercicio 4

Abre tu archivo **calculadora.py** con nuestro editor de texto sublime text 2 y modifica el código donde tienes asignadas las variables **a** y **b,** deja que las variables puedas ingresarles los valores por teclado. Guarde el archivo como **calculadora.py** y ejecute el programa

Ejercicio 5

Nuestra calculadora funciona 1 sola ves que la ejecutamos ingresamos los valores para a y b y nos muestra los resultados de las operaciones.

Ahora haremos algo un poco más funcional para el usuario, crearemos una opción (menú) en el cual el usuario pueda seleccionar si suma, resta, multiplica o divide los números **a** y **b** que ha ingresado y luego muestre el resultado.

El programa debe permitir al usuario continuar o salir de la calculadora

Ejercicio 6

Cree en un nuevo documento llamado **clase_persona.py** en este documento python crea una clase **Persona** con los atributos como **nombre**, **identificación**, **edad** y **sexo**.

El programa debe permitir ingresar los datos correspondientes a una persona y debe permitir mostrarla en pantalla

Ejercicio 7

Complementa el archivo **clase_persona.py** de forma que permita agregar varias personas en una **lista** de **personas** agregue 3 personas utilizando la clase persona que ha creado anteriormente, guarda los cambios y ejecuta tu aplicación.