¿Qué es Python?

Python es un lenguaje de programación creado por Guido Van Rossum, con una sintaxis muy limpia, ideado para enseñar a la gente a programar bien. Se trata de un lenguaje interpretado o de *script*.

Ventajas:

- **Legible**: sintaxis intuitiva y estricta.
- **Productivo**: ahorra mucho código.
- **Portable**: para todo sistema operativo.
- **Recargado**: viene con muchas librerías por defecto.

Editor recomendado: *Atom* o *Sublime Text*.

Instalación

Existen dos versiones de Python que tienen gran uso actualmente, *Python 2.x* y *Python 3.x*, para este curso puedes utilizar cualquiera de las dos, **pero te recomendamos usar una versión 2.x**

Para instalar Python solo debes seguir los pasos dependiendo del sistema operativo que tengas instalado.

Windows

Para instalarlo en Windows, debemos ir a https://www.python.org/downloads/release/python-2716/ el sitio reconocerá el sistema operativo y te dará las opciones para descargar.

Ejecuta el archivo que descargaste y sigue los pasos de instalación. Al finalizar vas a poder utilizar Python en tu computador y estás listo para continuar con el curso.

MacOS

La forma sencilla es tener instalado homebrew y usar el comando:

Para instalar la Versión 2.7

```
brew install python
```

Para instalar la Versión 3.x

```
brew install python3
```

Linux

Generalmente Linux ya lo trae instalado, para comprobarlo puedes ejecutar en la terminal el comando

Versión 2.7

```
python -v
```

Versión 3.x

```
python3 -v
```

Si el comando arroja un error quiere decir que no lo tienes instalado, en ese caso los pasos para instalarlo cambian un poco de acuerdo con la distribución de linux que estés usando. Generalmente el gestor de paquetes de la distribución de Linux tiene el paquete de Python

Si eres usuario de Ubuntu o Debian por ejemplo puedes usar este comando para instalar la versión 3.1:

```
$ sudo apt-get install python3.1
```

Si eres usuario de Red Hat o Centos por ejemplo puedes usar este comando para instalar python

```
$ sudo yum install python
```

Si usas otra distribución o no has podido instalar python en tu sistema Linux dejame un comentario y vemos tu caso específico.

Si eres usuario habitual de linux también puedes descargar los archivos para instalarlo manualmente.

Antes de empezar:

Para usar Python debemos tener un editor de texto abierto y una terminal o cmd (línea de comandos en Windows) como administrador.

No le tengas miedo a la consola, la consola es tu amiga.

Para ejecutar Python abres la terminal y escribes:

```
$ python
```

Te abrirá una consola de Python, lo notarás porque el *prompt* cambia y ahora te muestra tres simbolos de mayor que “ >>> “ y el puntero adelante indicando que puedes empezar a ingresar comandos de python.

```
>>>
```

En éste modo puedes usar todos los comandos de Python o escribir código directamente.

*Si deseas ejecutar código de un archivo sólo debes guardarlo con extension.py y luego ejecutar en la terminal:

```
$ python archivo.py
```

Ten en cuenta que para ejecutar el archivo con extensión “.py” debes estar ubicado en el directorio donde tienes guardado el archivo.

Cuando usamos Python debemos atender ciertas reglas de la comunidad para definir su estructura. Las encuentras en el libro PEP8.

Tipos de datos en Python

- **Enteros (int)**: en este grupo están todos los números, enteros y *long:*

ejemplo: 1, 2, 3, 2121, 2192, -123

- **Booleanos (bool):** Son los valores falso o verdadero, compatibles con todas las operaciones booleanas (and, not, or):

ejemplo: True, False

- **Cadenas (str):** Son una cadena de texto :

ejemplos: “Hola”, “¿Cómo estas?”

- **Listas:** Son un grupo o array de datos, puede contener cualquiera de los datos anteriores:

ejemplos: [1,2,3, ”hola” , [1,2,3]], [1,“Hola”,True]

- **Diccionarios:** Son un grupo de datos que se acceden a partir de una clave:

ejemplo: {“clave”:”valor”}, {“nombre”:”Fernando”}

- **Tuplas:** también son un grupo de datos igual que una lista con la diferencia que una tupla después de creada no se puede modificar.

ejemplos: (1,2,3, ”hola” , (1,2,3)), (1,“Hola”,True) (Pero jamás podremos cambiar los elementos dentro de esa Tupla)

En Python trabajas con **módulos** y **ficheros** que usas para importar las librerías.

Funciones

Las funciones las defines con **def** junto a un nombre y unos paréntesis que reciben los parámetros a usar. Terminas con dos puntos.

def nombre_de_la_función(**parametros**):

Después por indentación colocas los datos que se ejecutarán desde la función:

```
>>> def my_first_function():
...     return “Hello World!”
...
>>> my_first_function()
```

Hello World!

Variables

Las variables, a diferencia de los demás lenguajes de programación, no debes definirlas, ni tampoco su tipo de dato, ya que al momento de iterarlas se identificará su tipo. Recuerda que en Python todo es un objeto.

```
A = 3
B = A
```

Listas

Las listas las declaras con corchetes. Estas pueden tener una lista dentro o cualquier tipo de dato.

```
>>> L = [22, True, ”una lista”, [1, 2]]
>>> L[0]
22
```

Tuplas

Las tuplas se declaran con paréntesis, recuerda que no puedes editar los datos de una tupla después de que la has creado.

```
>>> T = (22, True, "una tupla", (1, 2))
>>> T[0]
22
```

Diccionarios

En los diccionarios tienes un grupo de datos con un formato: la primera cadena o número será la clave para acceder al segundo dato, el segundo dato será el dato al cual accederás con la llave. Recuerda que los diccionarios son listas de llave:valor.

```
>>> D = {"Kill Bill": "Tamarino", "Amelie": "Jean-Pierre Jeunet"}
>>> D["Kill Bill"]
"Tamarino"
```

Conversiones

De flotante a entero:

```
>>> int(4.3)
4
```

De entero a flotante:

```
>>> float(4)
4.0
```

De entero a string:

```
>>> str(4.3)
"4.3"
```

De tupla a lista:

```
>>> list((4, 5, 2))
[4, 5, 2]
```

Operadores Comunes

Longitud de una cadena, lista, tupla, etc.:

```
>>> len("key")
3
```

Tipo de dato:

```
>>> type(4)
< type int >
```

Aplicar una conversión a un conjunto como una lista:

```
>>> map(str, [1, 2, 3, 4])
['1', '2', '3', '4']
```

Redondear un flotante con x número de decimales:

```
>>> round(6.3243, 1)
6.3
```

Generar un rango en una lista (esto es mágico):

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

Sumar un conjunto:

```
>>> sum([1, 2, 4])
7
```

Organizar un conjunto:

```
>>> sorted([5, 2, 1])
[1, 2, 5]
```

Conocer los comandos que le puedes aplicar a x tipo de datos:

```
>>>Li = [5, 2, 1]
>>>dir(Li)
>>>['append', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse',
'sort']
```

‘append’, ‘count’, ‘extend’, ‘index’, ‘insert’, ‘pop’, ‘remove’, ‘reverse’, ‘sort’ son posibles comandos que puedes aplicar a una lista.

Información sobre una función o librería:

```
>>> help(sorted)
(Aparecerá la documentación de la función sorted)
```

Clases

Clases es uno de los conceptos con más definiciones en la programación, pero en resumen sólo son la representación de un objeto. Para definir la clase usas *class* y el nombre. En caso de tener parámetros los pones entre paréntesis.

Para crear un constructor haces una función dentro de la clase con el nombre **init** y de parámetros self (significa su clase misma), nombre_r y edad_r:

```
>>> class Estudiante(object):
...     def __init__(self,nombre_r,edad_r):
...             self.nombre = nombre_r
...             self.edad = edad_r
...
...     def hola(self):
...             return "Mi nombre es %s y tengo %i" % (self.nombre, self.edad)
...
>>> e = Estudiante(“Arturo”, 21)
>>> print e.hola()
Mi nombre es Arturo y tengo 21
```

Lo que hicimos en las dos últimas líneas fue:

1. En la variable e llamamos la clase Estudiante y le pasamos la cadena “Arturo” y el entero 21.

2. Imprimimos la función hola() dentro de la variable e (a la que anteriormente habíamos pasado la clase).

Y por eso se imprime la cadena “Mi nombre es Arturo y tengo 21”

Métodos especiales

cmp(self,otro)
Método llamado cuando utilizas los operadores de comparación para comprobar si tu objeto es menor, mayor o igual al objeto pasado como parámetro.

len(self)
Método llamado para comprobar la longitud del objeto. Lo usas, por ejemplo, cuando llamas la función len(obj) sobre nuestro código. Como es de suponer el método te debe devolver la longitud del objeto.

init(self,otro)
Es un constructor de nuestra clase, es decir, es un “método especial” que llamas automáticamente cuando creas un objeto.

Condicionales IF

Los condicionales tienen la siguiente estructura. Ten en cuenta que lo que contiene los paréntesis es la comparación que debe cumplir para que los elementos se cumplan.

```
if ( a > b ):
       elementos
elif ( a == b ):
       elementos
else:
       elementos
```

Bucle FOR

El bucle de for lo puedes usar de la siguiente forma: recorres una cadena o lista a la cual va a tomar el elemento en cuestión con la siguiente estructura:

```
for i in ____:
       elementos
```

Ejemplo:

```
for i in range(10):
       print i
```

En este caso recorrerá una lista de diez elementos, es decir el _print i _de ejecutar diez veces. Ahora i va a tomar cada valor de la lista, entonces este for imprimirá los números del 0 al 9 (recordar que

en un *range* vas hasta el número puesto -1).

Bucle WHILE

En este caso while tiene una condición que determina hasta cuándo se ejecutará. O sea que dejará de ejecutarse en el momento en que la condición deje de ser cierta. La estructura de un while es la siguiente:

```
while (condición):
       elementos
```

Ejemplo:

```
>>> x = 0
>>> while x < 10:
...    print x
...    x += 1
```

En este ejemplo preguntará si es menor que diez. Dado que es menor imprimirá x y luego sumará una unidad a x. Luego x es 1 y como sigue siendo menor a diez se seguirá ejecutando, y así sucesivamente hasta que x llegue a ser mayor o igual a 10.