

# Guia 1 Python

Objetivo General: Mostrar el uso de python a nivel básico

Objetivos Específicos:

1. Hacer una breve explicación de qué es python.
2. Mostrar la sintaxis de python.
3. Realizar ejemplos de estructuras de control en python.
4. Introducir los conceptos y aplicar la programación orientada a objetos en python.

## Introducción a Python

Python es uno de esos lenguajes raros que pueden reclamar ser simple y poderoso, luego de ser usado; se encontrará increíble lo fácil que es concentrarse en la solución del problema, en lugar de la sintaxis y la estructura del lenguaje de programación en el que se trabaja.

## Características de Python

- Simple: Es un lenguaje simple y minimalista. Leer un programa escrito en python es como leer inglés (uno muy estricto). Esta naturaleza de «pseudo-código» que tiene python es una de sus grandes fortalezas. Puesto que permite concentrarse en la solución.
- Fácil de aprender: Debido a su sintaxis simple.
- Libre y Código Abierto: Se puede distribuir libremente copias del software, leer la fuente, hacer cambios y usar porciones del código en nuevos programas.
- Lenguaje de Alto Nivel: No hay que preocuparse por detalles de bajo nivel como gestión de memoria.
- Portable: Los programas escritos en Python pueden funcionar en muchas plataformas.
- Interpretado: ... es interpretado.
- Orientado a Objetos: Es uno de los múltiples paradigmas que Python soporta.
- Extensible: Se pueden crear nuevas características o funciones enteras o parciales en C/C++ y usarlas en los programas en Python.
- Embebible: Se puede embeber Python en programas en C/C++ para añadir capacidades de script.
- Bibliotecas Extensas: Muchas bibliotecas y funcionalidades.

# Instalación de python en windows

- Instalación normal, descargar Python 3 de la pagina oficial y ejecutar el instalador.
- Instalación portable: existen algunos proyectos que permiten hacer portables los binarios de python, tales como [PythonXY](#), [Anaconda](#) y [WinPython](#); WinPython es el que se utilizará en este curso, pero puede usarse cualquiera. Para instalarlo seguir los pasos:
  - Ir a la [pagina de WinPython](#) y descargar la ultima version según sea la arquitectura de la computadora a usar.
  - Iniciar el instalador y escoger la ruta donde se instalará la version portable (escoger memoria usb)

## Primeros pasos

Lo primero que se hará es correr el tradicional «Hola Mundo» en Python, esto para mostrar como escribir, guardar y ejecutar programas en python.

Existen 2 formas de usar python para ejecutar los programas; Una es usar el intérprete interactivo; La otra es usar archivos fuente.

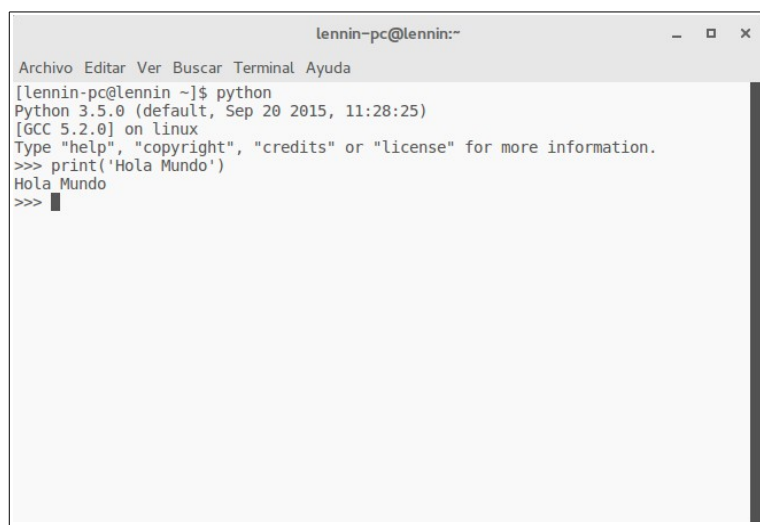
## Usar el intérprete interactivo.

Abre la terminal del sistema operativo de tu computadora y ejecuta el intérprete de Python usando el comando «python».

Una vez iniciado el interprete escribe:

```
print('Hola Mundo!')
```

Luego de apretar enter deberias ver algo semejante a:

A screenshot of a terminal window titled 'lennin-pc@lennin:~'. The window shows the command 'python' being executed, which starts the Python 3.5.0 interpreter. The prompt 'Python 3.5.0 (default, Sep 20 2015, 11:28:25)' is displayed, followed by '[GCC 5.2.0] on linux'. The user enters '>>> print('Hola Mundo')' and the output 'Hola Mundo' is printed. The prompt '>>>' is shown again, indicating the interpreter is ready for the next command. The terminal window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'.

Nota que Python muestra la salida de la linea inmediatamente, lo que se acaba de escribir es una sentencia de Python. Se usa print para imprimir el valor que se le envía como parametro. En este caso se envia el texto Hola Mundo para que sea mostrado en pantalla.

## Usar archivos con código.

Es el método mas utilizado para escribir programas, puesto que no podemos escribir el programa en el intérprete cada vez que deseemos ejecutar algo, para evitar esto; escribiremos las sentencias en archivos y las ejecutamos cualquier cantidad de veces. Para crear nuestros archivos de Python se suele usar un software editor donde se pueda escribir y guardar. Esto se discutirá más adelante en esta guía.

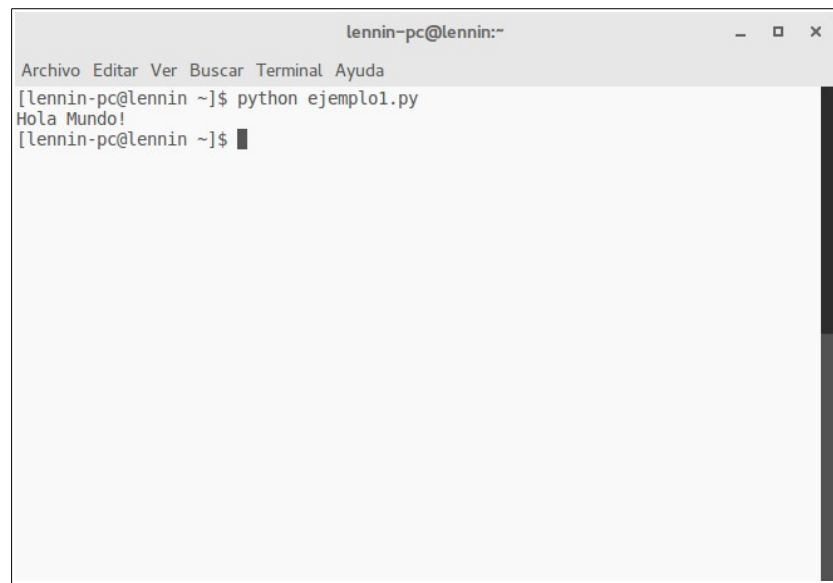
Abre el editor de archivos que mas te guste usar y/o tengas a la mano (ej, notepad, notepad++, leafpad, gedit, atom, brackets, sublime text, etc); y escribe la linea de código siguiente:

```
print('Hola mundo')
```

Guarda el archivo como ejemplo1.py y en la consola de comandos, sobre el directorio del archivo escribe:

```
python ejemplo1.py
```

El resultado debería ser similar a este:

A screenshot of a terminal window titled 'lennin-pc@lennin:~'. The window has a menu bar with 'Archivo', 'Editar', 'Ver', 'Buscar', 'Terminal', and 'Ayuda'. The terminal shows the command '[lennin-pc@lennin ~]\$ python ejemplo1.py' being executed, followed by the output 'Hola Mundo!' and the prompt '[lennin-pc@lennin ~]\$' with a cursor.

En este ejemplo se mira como se puede ejecutar el código en un archivo; esto es algo normal puesto que los programas escritos suelen ser mas complejos y la escritura de la serie de sentencias puede ser tediosa en el intérprete interactivo.

## Escojer un Entorno de Desarrollo Integrado (IDE)

Un IDE es un software que provee de facilidades a los desarrolladores para el desarrollo de software, un IDE normalmente consiste de un editor de código, utilidades de automatización de construcción, compilación, ejecución, testing y debugging; así como autocompletado de código. Para python existen una gran cantidad de entornos, en este curso se recomienda utilizar cualquiera de los siguientes:

- Eclipse con el complemento Pydev.
- Aptana Studio.
- PyCharm

Cualquiera que se escoja deberá ser configurado para utilizar la version de python que se instala con WinPython, la ventaja de usar cualquiera de los anteriores es que se puede utilizar como binarios portables puesto que dependen que esté instalado Java.

## Conceptos básicos

### Comentarios

Los comentarios son cualquier texto a la derecha del simbolo «#» y son utilizados como notas por los lectores del programa. Por ejemplo:

```
print('Hola Mundo') # comentario
```

### Constantes literales

Ejemplos de constantes literales son números como 7, 20, 0.9 o cadenas como 'Cadena de texto'. Se llama literal, por que se usa el valor literalmente, por ejemplo; el numero 2 siempre se representa a si mismo y nada mas, es un valor constante (inmutable) porque no se puede cambiar.

### Numeros

Los numeros son de dos tipos: enteros y flotantes.

### Cadenas de Texto

Son secuencia de caracteres, las cadenas son básicamente un grupo de palabras. Y pueden estar delimitadas por comillas simples, comillas dobles y comillas triples. (Son valores inmutables)

### Variables

Puesto que usar valores literales no se puede siempre, se necesita una forma de almacenar y modificar cualquier información. Para esto funcionan las variables. Para nombrarlas se necesita seguir las siguientes reglas:

- El primer carácter debe ser una letra del alfabeto o un guion bajo.
- El resto del identificador (nombre de variable) puede consistir de letras, numeros y guión bajo.
- Los nombres son sensitivos, es decir la variable hola no es la misma que Hola.
- No se permiten caracteres especiales como @, #, \$, %, ^, -, +, /, etc.

# Sintaxis

La sintaxis de Python es un conjunto de reglas que definen como un programa en Python debe ser escrito e interpretado (tanto para humanos y sistemas en tiempo de ejecución). Python fue diseñado para ser un lenguaje altamente legible.

## Palabras reservadas

and	else	in	raise
as	except	is	return
assert	exec	lambda	True
break	False	None	try
class	finally	nonlocal	while
continue	for	not	with
def	global	or	yield
del	if	pass	
elif	import	print	

Notas:

- Estas palabras no pueden ser usadas como identificadores o nombres de variables, funciones, clases o paquetes.
- exec y print desde python 3 no son palabras reservadas sino funciones.
- nonlocal es una palabra reservada desde python 3.

## Indentación

Python utiliza espaciado para delimitar bloques de programa, dividiendolos por nivel de indentación. En lugar de puntuaciones o palabras reservadas, utiliza la indentación para indicar la corrida de un bloque. En lenguajes como C/C++ o Java, los bloques de código son distinguidos por un set de llaves “{}”. En varias convenciones para estos lenguajes, los programadores indentan el código en bloques, para diferenciarlo de el código que le rodea.

C/C++	Python
<pre>void int potencia(int x, int p){     if (x==0){         return 0;     }else if (p==0){         return 1;     }else if (p&lt;0){         return (1/x)*potencia(x,p+1);     } else{         return x*potencia(x,p-1);     } }</pre>	<pre>def potencia(x, p):     if x==0:         return 0     elif p==0:         return 1     elif p&lt;0:         return (1/x)*potencia(x,p+1)     else:         return x*potencia(x,p-1)</pre>

## Sentencias de control de flujo

if	Ejecuta un bloque de código condicionalmente, junto a else y elif.
for	Recorre un objeto iterable, almacenando cada elemento en una variable local para su uso en el bloque.
while	Ejecuta un bloque de código siempre que se evalúe una condicion en True.
try	Permite que se emitan excepciones en el bloque de código asociado, para que sean capturadas y manejadas por la sentencia except, también se asegura la ejecución del bloque finally.
class	Ejecuta un bloque de código para ser usado en POO.
def	Define una función o método.
with	Encierra el bloque de código en un asistente de contexto.
pass	Sentencia que permite crear bloques de código vacíos.
yield	Usado para retornar generadores.

## Ejemplos (1)

1. Ejemplo if-elif-else.
2. Ejemplo for.
3. Ejemplo while.
4. Ejemplo try.
5. Ejemplo def.
6. Ejemplo class.
7. Ejemplo números primos.
8. Ejemplo crear una función que cuente la cantidad de caracteres 'a' en una cadena.
9. Ejemplo servidor Http.

## Ejercicios (1)

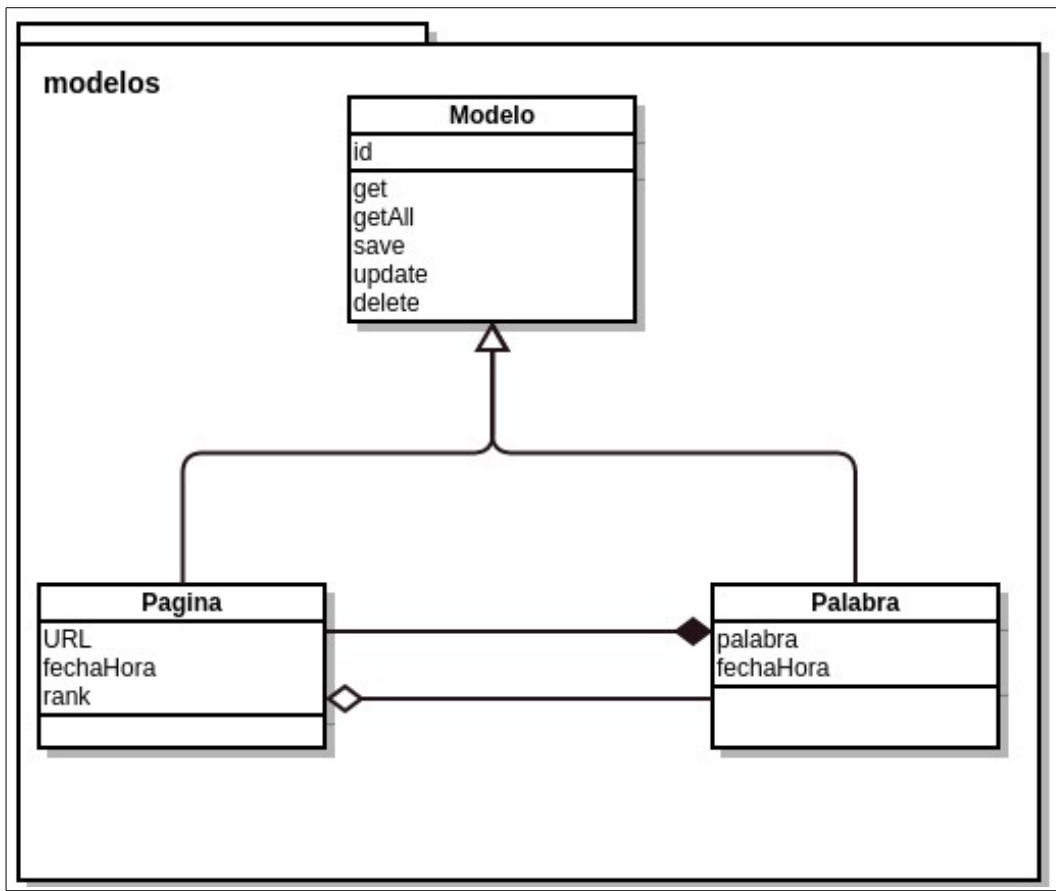
1. Completa el código en el archivo ejercicio1.py
2. Investigar sobre los diccionarios en python, como se usan y para que pueden servir.
3. Investigar sobre el uso de archivos en python, ¿Cómo se leen? y ¿Cómo se escriben?
4. Completa el código en el archivo ejercicio2.py

## Ejemplos (2)

1. Clases
2. Herencia
3. Paquetes

## Ejercicios (2)

1. Crear 3 Paquetes «modelos», «vistas» y «controladores».
2. En el paquete «modelos» crear las clases:



3. Investigar el uso de sqlite en python.
4. Codificar en la clase «Modelo» las funciones necesarias para persistir las clases que lo hereden, usando sqlite.