

Sumario

UT 02: Entornos de desarrollo.....	2
1 Entorno de desarrollo en pseudocódigo (PSEInt).....	2
2 Herramientas y entornos para el desarrollo de programas.....	4
2.1 Ensambladores, Compiladores e intérpretes.....	4
2.2 El JDK.....	5
2.3 Editores de texto.....	5
2.4 Entornos integrados de desarrollo.....	6
3 El lenguaje Python.....	8
4 Entornos de desarrollo para Python.....	9
4.1 Instalando el intérprete de Python en Linux.....	9
4.2 Instalando el intérprete de Python en Windows.....	10
5 Instalando un IDE para Python (Spyder3).....	11



Documentación oficial de Python: <https://docs.python.org/3/>

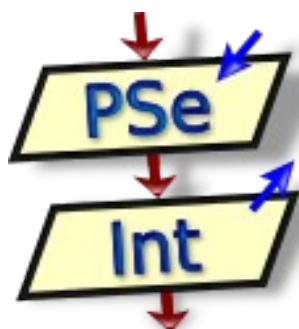
Referencia **w3schools**: https://www.w3schools.com/python/python_reference.asp



Realizado bajo licencia [Creative Commons Reconocimiento-NoComercial CC-BY-NC 4.0](https://creativecommons.org/licenses/by-nc/4.0/)

UT 02: Entornos de desarrollo

1 Entorno de desarrollo en pseudocódigo (PSEInt)

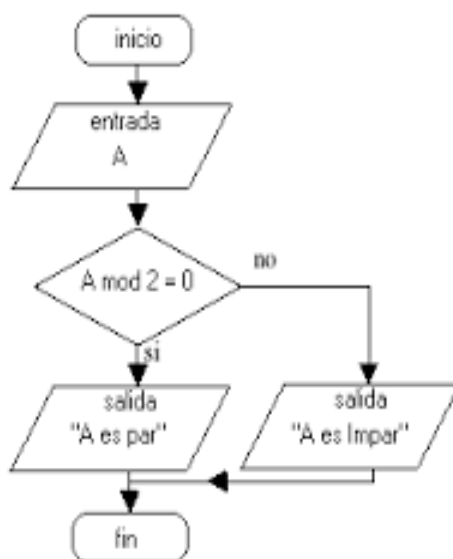


Como avanzábamos en la unidad anterior, en este curso utilizaremos la herramienta **PseInt**, basada en software libre. Esta aplicación permite diseñar programas con pseudocódigo. Mediante este programa podemos representar los diagramas de flujo que corresponden a un programa o fragmento de código.

El software puede descargarse de:


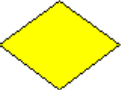








<http://pseint.sourceforge.net/>

Un diagrama de flujo es una representación gráfica que describe un proceso, sistema o algoritmo informático. Se usan ampliamente en numerosos campos para documentar, estudiar, planificar, mejorar y comunicar procesos que suelen ser complejos en diagramas claros y fáciles de comprender. Los diagramas de flujo emplean rectángulos, óvalos, diamantes y otras numerosas figuras para definir el tipo de paso, junto con flechas conectoras que establecen el flujo y la secuencia. Pueden variar desde diagramas simples y dibujados a mano hasta diagramas exhaustivos creados por computadora que describen múltiples pasos y rutas.



Una vez instalada la aplicación, hay que realizar la configuración correcta accediendo al menú "Opciones del Lenguaje / Perfiles" y seleccionando el perfil "FP Clara del Rey".

En el Aula Virtual del módulo se podrá encontrar un manual de uso de la herramienta PSEInt, así como una batería de ejercicios para realizar. Se recomienda practicar con el entorno tecleando los ejemplos proporcionados.

	Inicio/Final Se utiliza para indicar el inicio y el final de un diagrama; de Inicio sólo puede salir una línea de flujo y al final sólo debe llegar una línea		Decisión Indica la comparación de dos datos y dependiendo del resultado lógico (falso o verdadero) se toma la decisión de seguir un camino del diagrama u otro
	Entrada/Salida Entrada/Salida de datos por cualquier dispositivo (scanner, lector de código de barras, micrófono, parlantes, etc.)		Impresora/Documento. Indica la presentación de uno o varios resultados en forma impresa
	Entrada por teclado. Entrada de datos por teclado. Indica que el computador debe esperar a que el usuario teclee un dato que se guardará en una variable o constante		Pantalla Instrucción de presentación de mensajes o resultados en pantalla
	Acción/Proceso Indica una acción o instrucción general que debe realizarse (operaciones aritméticas, asignaciones, etc.)		Conector Interno Indica el enlace de dos partes de un diagrama dentro de la misma página
	Flujo/Flecas de Dirección Indica el seguimiento lógico del diagrama. También indica el sentido de ejecución de las operaciones		Conector Externo Indica el enlace de dos partes de un diagrama en páginas diferentes

Estos son algunos de los símbolos más usados en diagramas de flujo:

- Óvalo o Elipse: Inicio y Final (Abre y cierra el diagrama).
- Rectángulo: Actividad (Representa la ejecución de una o más actividades o procedimientos).
- Rombo: Decisión (Formula una pregunta o cuestión).
- Círculo: Conector (Representa el enlace de actividades con otra dentro de un procedimiento).
- Triángulo boca abajo: Archivo definitivo (Guarda un documento en forma permanente).
- Triángulo boca arriba: Archivo temporal (Proporciona un tiempo para el almacenamiento del documento).

2 Herramientas y entornos para el desarrollo de programas

2.1 Ensambladores, Compiladores e intérpretes

Cuando los procesadores fueron creciendo en potencia, el código máquina, basado en unos y ceros, se hizo imposible de manejar directamente por los humanos. Surgieron "traducciones" de las instrucciones en código máquina para hacerlas más legibles (por ejemplo, la instrucción "1101...." se podía traducir por algo como "ADD A1,A2", es decir, la suma de dos registros). Este lenguaje consistente en una traducción literal del código máquina es el "**ensamblador**", y constituye el nivel más bajo de abstracción, como vimos en apartados anteriores. Este lenguaje es completamente dependiente del hardware de la máquina en que se ejecuta, es decir, no es portable a otros ordenadores.

Los **compiladores** son programas que traducen el código de alto nivel a código binario. Un programa escrito en lenguaje de alto nivel se denomina "programa fuente" o "código fuente". Cuando se traduce a código binario se conoce como "programa objeto" o "código objeto". El compilador realiza la tarea de conversión del programa fuente en programa objeto. El programa objeto ya puede ejecutarse en la máquina en la que fue compilado u otra similar. Si se usa programación modular es necesario un proceso previo de enlace ("linkado") de los diferentes módulos.

El **intérprete** traduce el código de alto nivel a código binario en tiempo de ejecución, ejecutando instrucción a instrucción secuencialmente. Esto provoca una mayor lentitud en la ejecución del programa, si se compara con el tiempo de ejecución de un programa compilado. Sin embargo, permite que el programa sea portable. Un mismo programa fuente se puede interpretar en diferentes plataformas o sistemas operativos.

Un ejemplo de lenguaje interpretado es **Javascript**, que permite ejecutar código sobre un navegador web capaz de entender el código, independientemente de la plataforma sobre la que se está trabajando. Un ejemplo de lenguaje compilado es C/C++.

El lenguaje **Java** es un caso peculiar, ya que se diseñó para que fuera altamente portable y altamente eficiente. Para ello, se diseñó una fase intermedia o "semicompilación" en la que se genera un código binario que no corresponde a ninguna máquina real, sino a una máquina simulada, con su hardware y registros correspondientes. Cada sistema puede emular esta máquina virtual mediante la instalación de un software especial denominado **JVM (Java Virtual Machine)**. El código binario generado a partir de un programa en Java se denomina "bytecode" y es interpretado por la JVM de cada sistema.



Esta peculiaridad de Java permite que los **programas escritos y compilados en cualquier arquitectura puedan ejecutarse en otra**, simplemente con el requisito de disponer de la máquina virtual (JVM) en el ordenador de destino. Para desarrollar programas en Java disponemos del software de desarrollo proporcionado por el fabricante, **Java Development Kit (JDK)**.

2.2 El JDK

El Java Development Kit es un paquete de software que incluye todo lo necesario para el desarrollo de programas en Java, como por ejemplo:

- Biblioteca de clases estándar de Java
- Máquina virtual de Java (comando "java")
- Compilador de Java (comando "javac")
- Desensamblador de clases (comando "javap")
- Depurador de consola (comando "jdb")
- Generador automático de documentación (comando "javadoc")

El JDK puede descargarse gratuitamente desde la web de Oracle:

<http://www.oracle.com/technetwork/es/java/javase/downloads/index.html>

2.3 Editores de texto

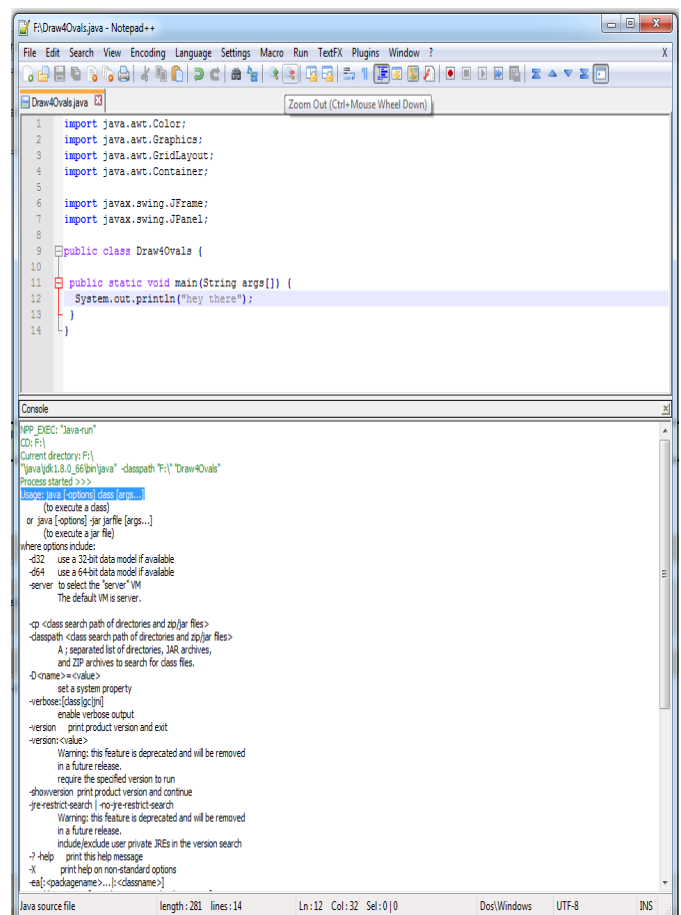
Para desarrollar un programa necesitamos una herramienta de edición de texto, que nos permita guardar los archivos en texto plano para posteriormente compilarlo y ejecutarlo. Muchos editores permiten seleccionar la codificación de caracteres e incorporan funciones de ayuda al desarrollo, como la posibilidad de reconocer el lenguaje en que estamos programando, coloreando ciertas palabras clave, identificadores, etc...

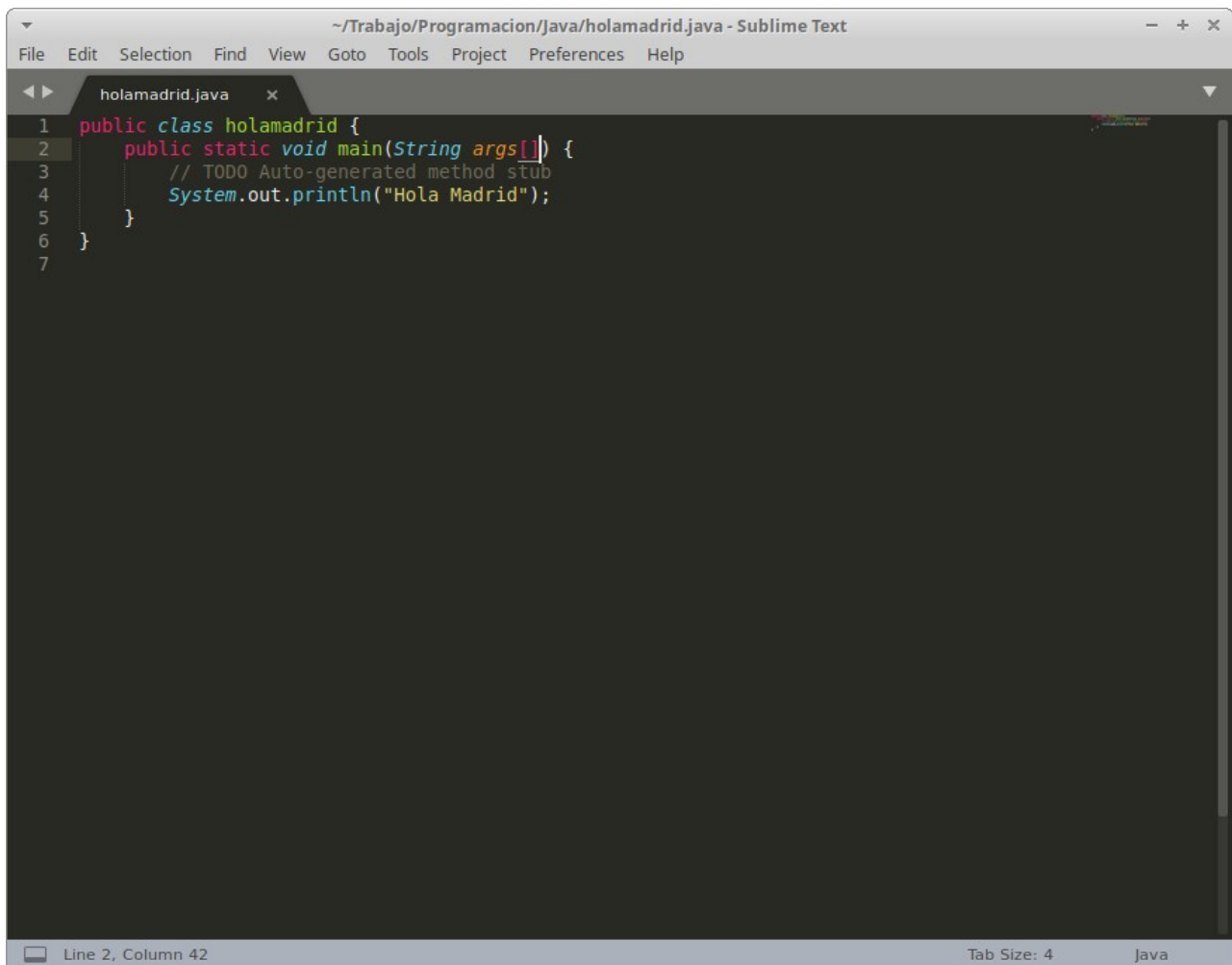
El editor de código libre y gratuito más usado en Windows es **Notepad++**

En Linux también se puede utilizar una versión de Notepad++, así como otros muchos editores, incluso comandos clásicos como emacs, vim, gedit, kwrite, etc...

En entornos Mac tenemos editores como Editra, Xemacs, etc...

También hay editores de pago que ofrecen ayudas contextuales y dan funcionalidades que los convierten en preferidos por muchos desarrolladores, como **Sublime Text**.





```
~/Trabajo/Programacion/Java/holamadrid.java - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help

holamadrid.java x
1 public class holamadrid {
2     public static void main(String args[]) {
3         // TODO Auto-generated method stub
4         System.out.println("Hola Madrid");
5     }
6 }
7

Line 2, Column 42 Tab Size: 4 Java
```

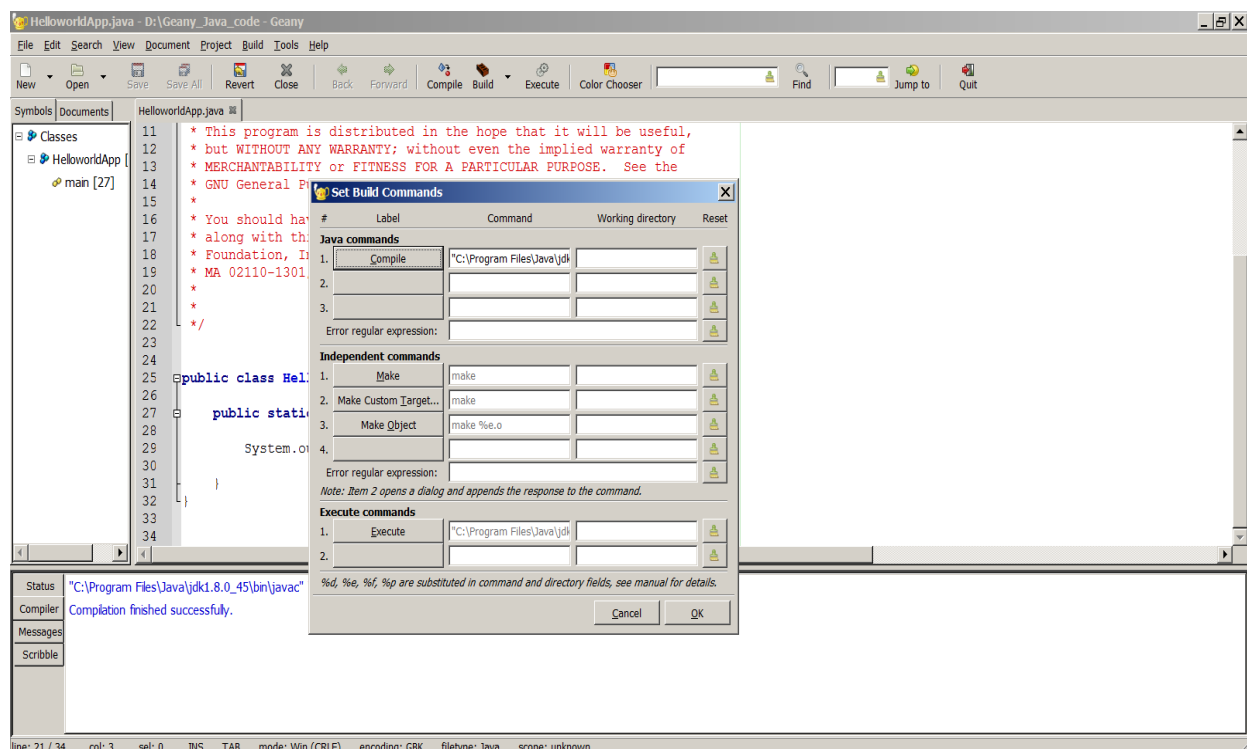
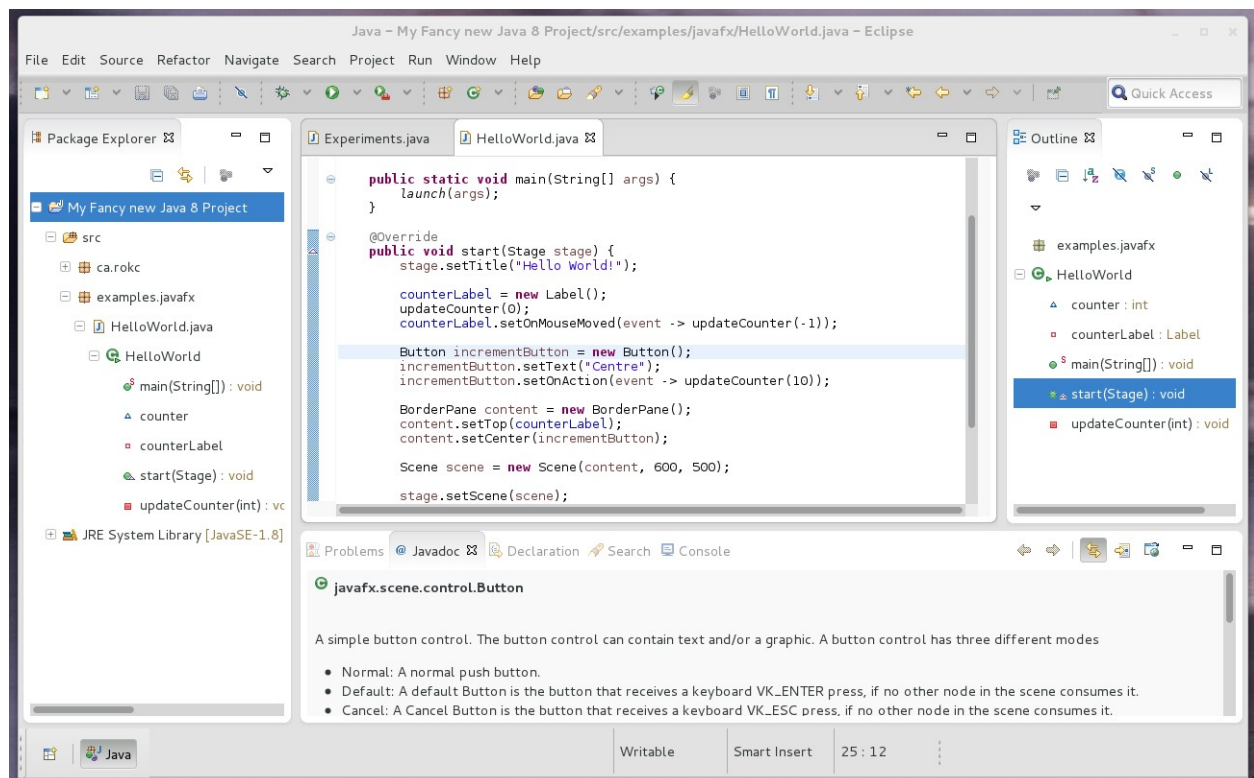
2.4 Entornos integrados de desarrollo

Los conocidos IDE (Integrated Development Environment), o Entornos de Desarrollo Integrado, proporcionan en una sola interfaz el acceso a todas las funciones anteriores: edición del código fuente, compilación, depuración y ejecución. Realizan una invocación transparente a las herramientas del JDK para ahorrar trabajo al programador.

En la primera parte del curso preferiremos utilizar las herramientas básicas y programar con ficheros de texto, para tener una mejor visión y control de los procesos involucrados.

Ejemplos de IDEs:

- IDEs ligeros y sencillos: Geany, Spyder3, BlueJ...
- IDEs pesados y complejos: NetBeans, Eclipse, IntelliJ...



3 El lenguaje Python

El lenguaje **Python** es interpretado, es decir, las instrucciones se ejecutan secuencialmente, sin necesidad de ser compiladas en un archivo ejecutable.

Para ejecutar programas en lenguaje Python, el sistema debe tener instalado el intérprete de Python. Existen versiones para Windows, Linux/Unix, MacOS y otros, disponibles en este enlace:

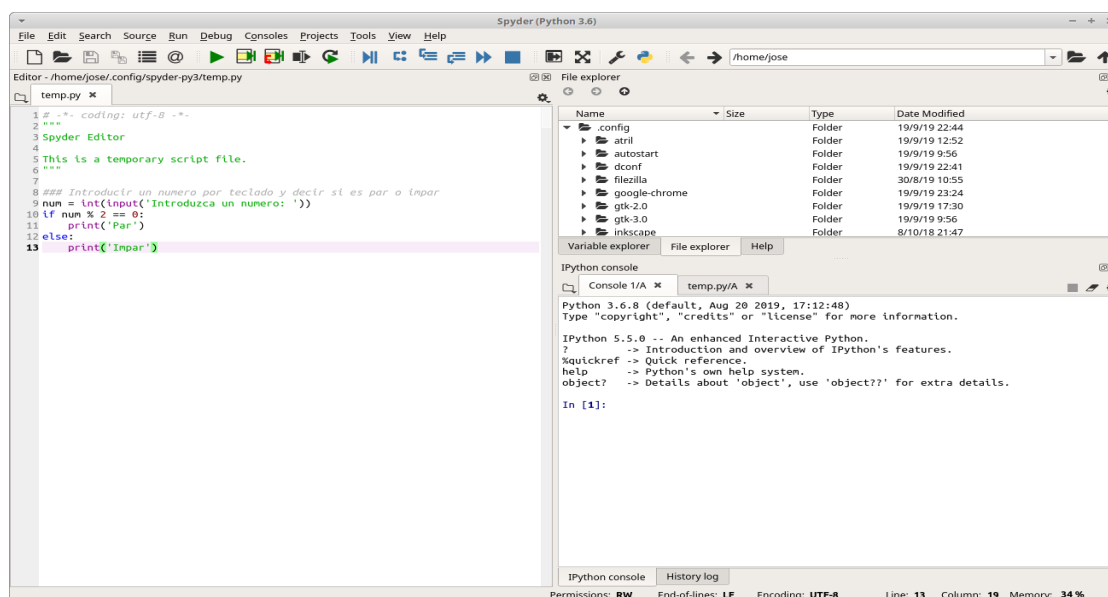
<https://www.python.org/downloads/>

Ejemplo de trabajo con el intérprete de comandos de Python en Linux:

```
jose@Audax:~$ Uso de herramientas de diseño. PSeInt
$ python
Python 2.7.15+ (default, Jul 9 2019, 16:51:35)
[GCC 7.4.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 3+2
5
>>> 5/2.
2.5
>>> num=5
>>> num
5
>>> if num%2==0:
...     print('Par')
... else:
...     print('Impar')
...
Impar
```



Existen entornos integrados para desarrollar programas en Python, como **Spyder3**:



4 Entornos de desarrollo para Python

4.1 Instalando el intérprete de Python en Linux

Antes de empezar a programar, hay que verificar que el sistema (puede ser Windows, Linux, etc...) tiene instalado el intérprete de Python.

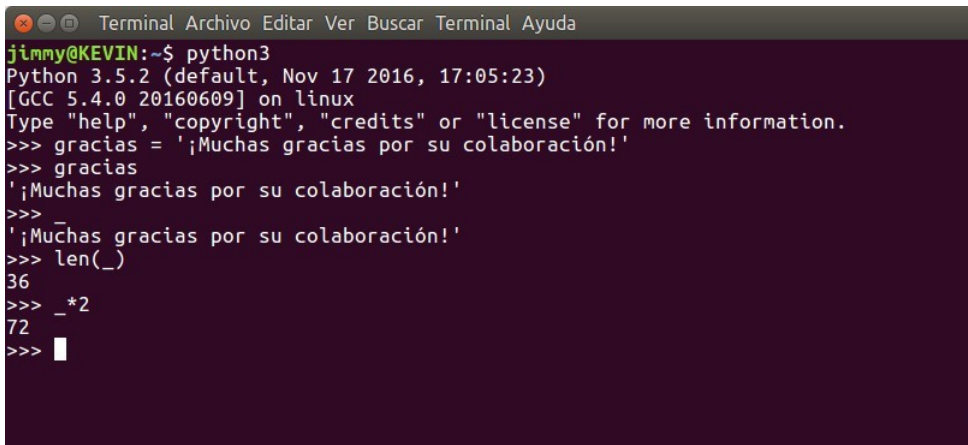
Este intérprete se puede arrancar directamente desde un terminal y permite ser usado como consola para realizar operaciones sencillas y ejecutar comandos, que son interpretados directamente. Hay que diferenciar entre la versión 2 y 3, que son incompatibles entre sí, aunque pueden convivir instaladas en el mismo sistema:

```
$ python -V
Python 2.7.15+
$ python
Python 2.7.15+ (default, Jul 9 2019, 16:51:35)
[GCC 7.4.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

```
$ python3 --version
Python 3.6.8
$ python3
Python 3.6.8 (default, Aug 20 2019, 17:12:48)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> print(sys.version)
3.6.8 (default, Aug 20 2019, 17:12:48)
[GCC 8.3.0]
```

En Linux se puede instalar con el gestor de paquetes (en Ubuntu, con "apt-get"):

```
$ sudo apt-get update
$ sudo apt-get install python3.6
```

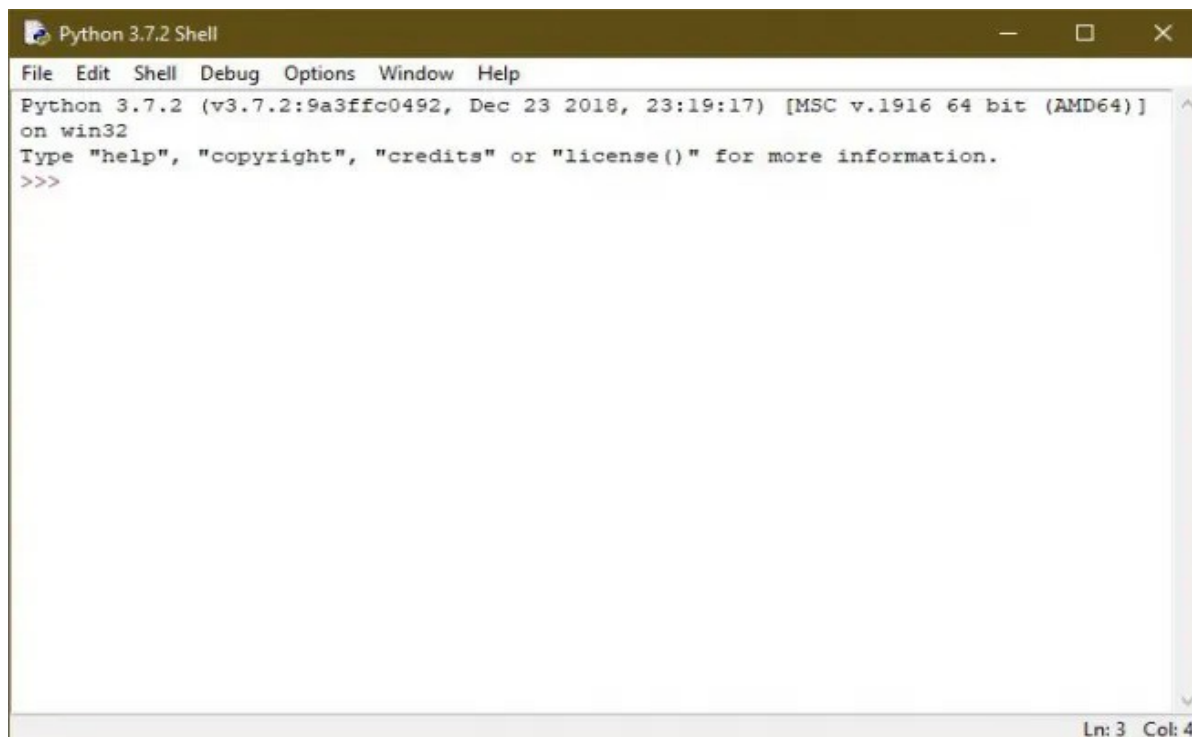


```
Terminal Archivo Editar Ver Buscar Terminal Ayuda
jimmy@KEVIN:~$ python3
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> gracias = '¡Muchas gracias por su colaboración!'
>>> gracias
'¡Muchas gracias por su colaboración!'
>>> _
'¡Muchas gracias por su colaboración!'
>>> len(_)
36
>>> _*2
72
>>> 
```

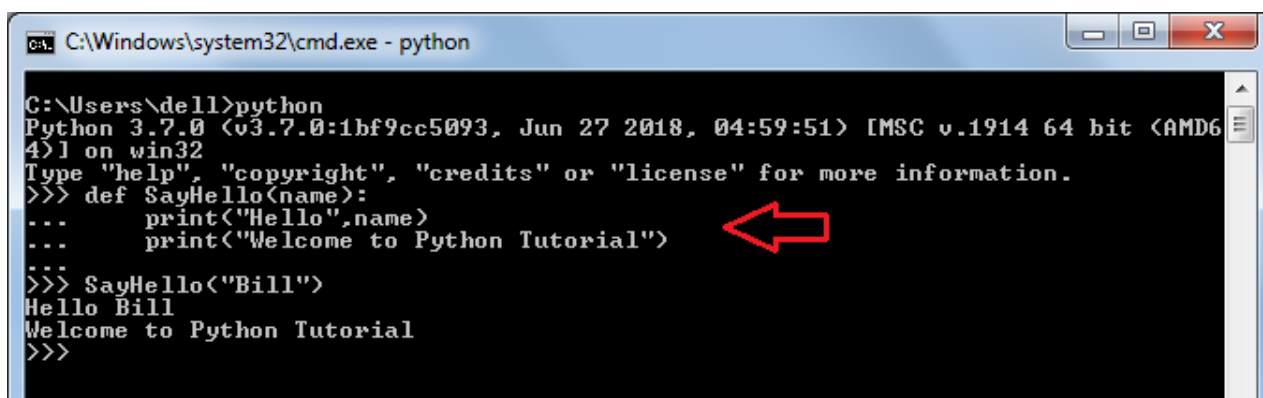
4.2 Instalando el intérprete de Python en Windows

En Windows 10 podemos descargar e instalar gratuitamente el intérprete Python 3.7 desde la Microsoft Store:

<https://www.microsoft.com/es-es/p/python-37/9nj46sx7x90p?activetab=pivot:overviewtab>

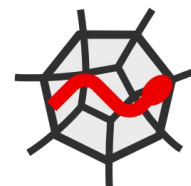


Una vez instalado, se integra en la línea de comandos (cmd), desde la que se puede invocar directamente el entorno Python.

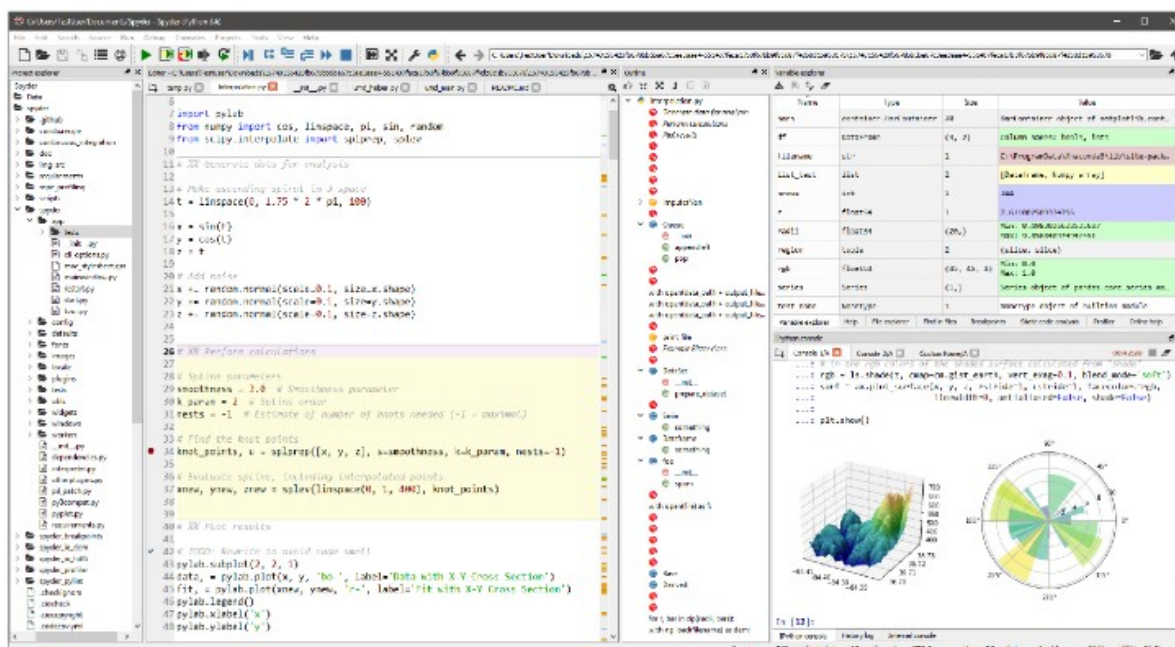


5 Instalando un IDE para Python (Spyder3)

Hay diferentes entornos integrados de desarrollo para trabajar con Python. Uno de los más completos es Spyder3, disponible para diferentes sistemas operativos. Este es el enlace de descarga del instalador:



<https://www.anaconda.com/download/>



En el caso de Linux, Spyder también se encuentra en el repositorio de programas, por lo que la instalación es muy sencilla:

```
$ sudo apt install spyder
```