Curso Básico de Python

1. Introducción a Python

Contenido

Python es un lenguaje de programación de alto nivel, moderno, con una extensa colección de bibliotecas para aplicaciones científicas y técnicas. Su diseño y características lo hacen apto tanto para desarrollo como para enseñanza e investigación.

1 Conocer Python

Usaremos como texto de apoyo el Tutorial de Python, parte de la documentación oficial de Python.

Lecturas recomendadas:

1. Tutorial de Python, presentación, en esa misma página.
2. Características del lenguaje, sección 1 Abriendo el apetito.
3. Uso del intérprete de comandos, sección 2 Usando el intérprete de comandos.

2 Python en línea

La forma más fácil e inmediata de usar Python es en línea; hay varios sitios en Internet donde se puede escribir un comando de Python y ver su ejecución inmediatamente. Algunos de estos sitios:

- W3 Schools. Python Compiler (Editor).
- Online Python.
- Online GDB.

En una ventana escribes el código, con el botón Run lo ejecutas, y ves el resultado en la otra ventana. Puedes ver los ejemplos mostrados, y probar escribiendo algunos comandos:

```
print("Hola Mundo")

for i in range(0,10):
    print(i, end=" ")
```

Debes escribir los comandos exactamente como están mostrados, usando espacios para indentar. El primer comando escribe "Hola Mundo", las siguientes dos líneas escriben los número del 0 al 9.

Una vez instalado Python en nuestra máquina ya no necesitaremos el acceso a Internet; podremos escribir y correr programas en nuestra máquina.

3 Instalar Python

Referencia: Configuración y uso de Python.

En Linux. En las máquinas Linux, Python suele venir instalado. Para comprobarlo, abre una terminal de comandos y escribe `python3`; deberás ver algo parecido a esto:

```
$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Los signos `>>>` indican al programador dónde escribir los comandos.
Para salir del intérprete, digitar Ctrl-D en Linux, Ctrl-Z en MS Windows, escribir `exit()`, o escribir `quit()`.
Para verificar si está instalado el ambiente de programación IDLE, busca en el menú la palabra IDLE; debe abrir una ventana similar a la de la figura 1. Si así no fuera, deberás instalar el paquete denominado “idle”, con un instalador de paquetes como Synaptic.

Referencia: Uso de Python en plataformas Unix.

En MS Windows. El paquete Microsoft Store es fácil de instalar, permite el uso interactivo, y contiene IDLE. Una explicación detallada sobre este paquete se puede ver en 4.2 El paquete Microsoft Store; en la referencia se explica más en detalle el uso de Python en Windows.

Referencia: Uso de Python en Windows.

En macOS, el sistema operativo de Mac, Python viene instalado. Para obtener la versión más actualizada, ir al sitio de Python, Downloads, macOs: https://www.python.org/downloads/macos/.
En la referencia se puede ver una explicación detallada de instalación y uso de Python en Mac.

Referencia: Usando Python en un Mac.

4 Python en la máquina local

Una vez instalado Python en nuestra máquina, tenemos varias formas de acceder a él.

- En Linux, en una terminal de comandos, escribe `python3`; verás en la pantalla algo parecido a esto:

```
$ python3
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- En MS Windows 10, en el menú Inicio, inicia PowerShell; allí escribe `Python –version`. Así podrás confirmar que Python está instalado.
- Usando IDLE, tanto en MS Windows como en Linux. IDLE es un ambiente de desarrollo donde es posible usar Python en forma interactiva o escribir archivos de comandos (programas, o scripts) y ejecutarlos. En el menú, buscar IDLE; al invocar la aplicación debe aparecer una ventana similar a la de la figura 1; allí puedes escribir comandos y ver su ejecución.

El manejo de IDLE es bastante intuitivo. Para un manejo más especializado, aquí están los manuales de IDLE en inglés y en español.

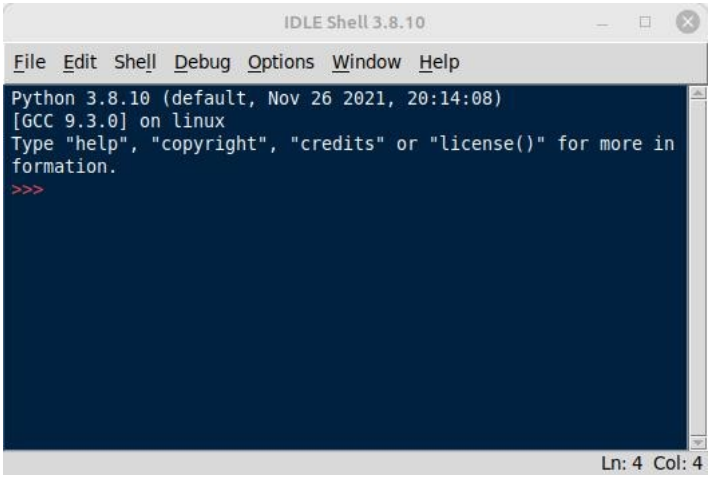

Figura 1: Ventana principal de IDLE, ambiente de desarrollo para Python.

Probemos ahora el primer comando en todos los lenguajes de programación: en una terminal, o en IDLE, escribe el saludo universal “Hola Mundo”, así:

```
>>> print("Hola Mundo")
Hola Mundo
```

Una prueba un poco más avanzada:

```
>>> for i in range(0,5):
...     print("Número", i)
...
Número 0
Número 1
Número 2
Número 3
Número 4
```

Asegúrate de escribir los comandos exactamente como se muestra, incluidos los espacios (puedes usar el tabulador para la segunda línea).

Referencias:

- Python docs. IDLE, manual en español, en inglés.

5 Formas de usar Python

Python se puede usar de diferentes formas:

- *interactiva,* en un sitio web intérprete de Python, en una terminal de comandos, o en IDLE. Digitamos los comandos en el intérprete, y cuando cada comando se completa aparece inmediatamente el resultado.
- *módulo ejecutable,* un archivo con comandos ejecutable como un programa. Desde una terminal podemos escribir por ejemplo:

  ```
  $ python3 hola.py
  Hola Mundo
  0 1 2 3 4 5 6 7 8 9
  $
  ```

  Vemos el resultado de la ejecución inmediatamente; no hemos tenido que invocar nosotros el intérprete de Python.

Un archivo con comandos de Python se llama *módulo*. Veremos enseguida el contenido de uno de estos archivos.

- *función dentro de un módulo.* En el intérprete de comandos, primero importamos el módulo, para poder invocar las funciones contenidas en él:

```
>>> import saludo
>>> saludo.personal()
¿Cómo te llamas? Antonio Molonio
Hola Antonio Molonio ; bienvenido a Python.
>>> saludo.al_mundo()
Hola Mundo
>>>
```

6 IDLE, guía rápida

Al iniciar IDLE nos muestra una ventana con un intérprete de comandos; es el núcleo o *shell* de IDLE. Son útiles las siguientes teclas rápidas, también disponibles en el menú, Shell:

`Alt-P` : muestra el comando anterior en la historia de comandos;
`Alt-S` : muestra el comando siguiente en la historia de comandos;
`Ctrl-C` : interrumpe la ejecución;
`Ctrl-D` : sale de IDLE.

Para escribir:

`Backspace` : borra hacia la izquierda;
`Del` : borra hacia la derecha;
`flechas, PgDn, PgUp` : mueven el cursor;
`Home, End` : principio y fin de línea;
`Ctrl-Home, Ctrl-End` : principio y fin de archivo.

Para escribir comandos en un archivo:

`File, New, o Ctrl-N` : abre un nuevo archivo en el editor;
`File, Save, o Ctrl-S` : guarda los cambios en el archivo.

En la ventana del editor,

`Run, Run module o F5` : ejecutan el módulo.
`Run, Python Shell` : vuelve al intérprete de comandos.

7 Módulo Python

El programa `hola.py` solo contiene un par de comandos, con una explicación de cada uno:

```
print("Hola Mundo")      # imprime el texto entre paréntesis

for i in range(0,10):    # la variable i recorre de 0 a 10-1
    print(i, end=" ")    # imprime el número sin cambiar de línea
```

Los textos después del numeral `#` son comentarios, no se ejecutan.

Examinamos ahora el módulo `saludo.py`, un programa ya más completo. Si abres el archivo en IDLE podrás ver resaltada la sintaxis. Luego de leer el código, puedes ver a continuación una explicación más detallada de cada parte.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
#
# saludo.py : primer módulo en Python, saludo.
#
# comentario de una línea; lo que viene después de # no se ejecuta.

'''El siguiente código define algunas funciones de ejemplo.
```

```
Este es un comentario de varias líneas; se llama "docstring".
'''

def personal():
    '''Pregunta nombre y saluda.

    Este docstring dice lo que hace la función.
    '''
    nombre = input("¿Cómo te llamas? ")       # asignación a una variable
    print("Hola", nombre, "; bienvenido a Python.")  # uso de la variable
    return

def al_mundo():
    '''Saluda a todo el mundo.
    '''
    print("Hola Mundo")
    return

def contar(n):
    '''Imprime números de 0 a n-1.

    @param n: un entero, hasta dónde contar.
    '''
    for i in range(0,n):     # la variable i recorre de 0 a n-1
        print(i, end=" ")    # imprime el número sin cambiar de línea
    return

if __name__ == "__main__":
    al_mundo()          # ejecuta esta función
    nro_hasta = 10      # asigna 10 a la variable nro_hasta
    contar(nro_hasta)   # invoca la función con nro_hasta como parámetro
```

Análisis del código.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

Indicaciones al intérprete. En la primera línea, ubicación del intérprete (Linux). En la segunda, la codificación de caracteres a usar. Estas dos líneas deberán ir siempre en todos los scripts de Python.

```
# saludo.py : primer módulo en Python, saludo.
```

Nombre del programa; no es obligatorio, pero sí un buen hábito escribir el nombre del programa y una descripción breve de su propósito, en la misma línea.
El texto después del `#` es un comentario, no se ejecuta.

```
'''El siguiente código define algunas funciones de ejemplo.

Este es un comentario de varias líneas; se llama "docstring".
'''
```

El texto entre tres comillas, sean dobles `"""` o simples `'''`, es un comentario multilínea, no se ejecuta. Este primer comentario multilínea define el propósito de módulo. En el primer renglón un resumen en una línea, seguido de un renglón en blanco, y luego una explicación más detallada. Este formato permitirá mostrar las explicaciones con el comando `help` así como la generación automática de documentación.

```
def personal():
    '''Pregunta nombre y saluda.

    Este docstring dice lo que hace la función
    '''
    nombre = input("¿Cómo te llamas? ")       # asignación a una variable
    print("Hola", nombre, "; bienvenido a Python.")  # uso de la variable
    return
```

Define una función denominada `personal`; dentro de los paréntesis irán, cuando los haya, uno o más parámetros, valores que la función deberá procesar. El comentario, en el formato de línea resumen, línea en blanco y explicación detallada, describe el propósito de la función. Vienen luego los comandos encargados de realizar las tareas propias de la función. El comando final, `return`, indica el final de la función, retomándose la ejecución del programa en el punto donde se había llamado a la función. La sentencia `return` permite también devolver valores que podrán guardarse en una variable.

```
def contar(n):
    '''Imprime números de 0 a n-1.

    @param n: un entero, hasta dónde contar.
    '''
    for i in range(0,n):     # la variable i recorre de 0 a n-1
        print(i, end=" ")    # imprime el número sin cambiar de línea
    return
```

La función anterior recibe un parámetro `n`; al invocarla, se deberá indicar ese valor, ya sea con un número o con una variable que contenga un número.

```
if __name__ == "__main__":
    al_mundo()          # ejecuta esta función
    nro_hasta = 10      # asigna 10 a la variable nro_hasta
    contar(nro_hasta)   # invoca la función con nro_hasta como parámetro
```

Esta parte final permite ejecutar el módulo como un programa, mediante un comando tal como

```
$ python3 saludo.py
```

La primera sentencia habilita ejecutar el módulo como programa; las siguientes invocan dos de las funciones definidas en el módulo. Este módulo se puede importar desde otro programa o desde un intérprete de comandos para usar las funciones definidas el él.

Puedes ver más detalles sobre el intérprete de comandos y su entorno en la sección 2 Usando el intérprete de comandos del Tutorial de Python.

8 Rutas de búsqueda

La primera condición para poder cargar y usar un módulo es encontrarlo. Los intérpretes de Python buscan módulos en ciertos directorios predeterminados, uno de ellos el directorio de trabajo. Par aagregar rutas de búsqueda, en el intérprete de comandos o en IDLE,

```
>>> import sys
>>> sys.path = sys.path + ["/tmp"]    # agrega /tmp a las rutas de búsqueda
```

Este cambio no es permanente. Para agregar un directorio a las rutas de búsqueda en forma permanente, es conveniente fijar en la variable de entorno `PYTHONPATH` las rutas a todos los directorios donde se encuentran los módulos. En Linux esto puede hacerse agregando en el archivo `.bashrc` del usuario esta sentencia:

```
PYTHONPATH=$PYTHONPATH:<directorio donde están los módulos>
```

Para activar el cambio, es preciso dar en una terminal el comando Linux `source ~/.bashrc`. De ahora en más los módulos se buscarán en todos los directorios contenidos en la variable `PYTHONPATH`.
Para evitar tener que alterar la variable `PYTHONPATH`, puede invocarse IDLE desde la línea de comandos, en el directorio donde se encuentran los módulos a importar:

```
$ cd <directorio del módulo a importar>
$ idle-python3.8      # o similar según la instalación
```

Para ver el nombre exacto de IDLE:

```
$ apropos idle
idle-python3.8 (1)   - An Integrated DeveLopment Environment for Python
idle (2)             - make process 0 idle
```

En MS Windows, se deberá fijar la variable de entorno a través del menú.

9 Usar un módulo Python

En una terminal, luego de invocar Python, o en IDLE:

```
>>> import saludo
>>> saludo.al_mundo()
Hola Mundo
>>> dir(saludo)
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__',
'__name__', '__package__', '__spec__', 'al_mundo', 'contar', 'personal']
>>> help(saludo)
```

Con `dir(saludo)` vemos todas las funciones definidas en el módulo. Con `help(saludo)` vemos los comentarios presentados como una página de ayuda, gracias al formato empleado al escribirlos.

Para importar todas las funciones al espacio de nombres actual:

```
>>> from saludo import *
>>> contar(10)
0 1 2 3 4 5 6 7 8 9
>>> al_mundo()
Hola Mundo
>>>
```

10 Ejercicios

Usar Python.

1. Python en línea. Prueba alguno de los sitios de Python en línea. Escribe `print("Hola Mundo")` o algún otro comando simple, toca el botón Run, y mira el resultado.
2. Luego de instalar Python, verifica la instalación: a) en una terminal, b) entrando en IDLE. Escribe algún comando simple para ver la ejecución.

Módulo Python.

1. Abre una terminal de comandos (Linux) y ve al directorio donde hayas bajado el archivo `saludo.py`. Con el comando `apropos idle` verifica la invocación de IDLE Python, por ejemplo, `idle-python3.8`. Inicia IDLE con ese comando. Importa el módulo `saludo`: `import saludo`.
   Escribe el comandos `dir(saludo)`. ¿Reconoces algún elemento en la salida?
   Escribe ahora `help(saludo)`. ¿Qué ves?
2. Inicia IDLE desde el menú. Intenta importar el módulo `saludo`. ¿Por qué no puedes? Ingresa estos comandos:

   ```
   >>> import sys
   >>> sys.path = sys.path + ["<ruta hacia donde está el archivo saludo.py>"]
   ```

   Importa ahora el módulo `saludo`. ¿Qué hemos hecho?

3. Importa el módulo `saludo`. Prueba invocar la función `contar(10)`. ¿Funciona? Compara la salida de estos dos comandos: `dir()` y `dir(saludo)`. ¿Cómo debemos invocar la función `contar`? Respuesta: `saludo.contar(10)`. Reinicia ahora el shell de IDLE: en el menú, Shell, Restart Shell. Escribe el comando `dir()`; analiza su salida. Escribe ahora el comando

`from saludo import *`. Vuelve a escribir `dir()` y compara con la salida anterior. ¿Qué ha pasado? ¿Puedes invocar ahora `contar(10)` directamente? ¿Por qué?