



Argentina Programa 4.0

Universidad Nacional de San Luis

DESARROLLADOR PYTHON

Lenguaje de Programación Python: Introducción

Autor:

Dr. Mario Marcelo Berón

UNIDAD 1

LENGUAJE DE PROGRAMACIÓN PYTHON: INTRODUCCIÓN

1.1. Introducción

Python es un lenguaje de programación de propósito general. Su filosofía de diseño enfatiza la legibilidad de código haciendo uso de la indentación.

Python es un lenguaje *tipado dinámicamente*¹ que tiene un *recolector de basura*² que posibilita que el programador se centre únicamente en la tarea de programar dejando de lado tareas técnicas como por ejemplo la de liberar espacios de memoria.

Si bien Python surge como un *lenguaje imperativo*³ a medida que el mismo evolucionó incorporó soporte para los paradigmas de programación *orientado*

¹El tipo de una variable se establece en tiempo de ejecución.

²Proceso que tiene como principal objetivo liberar espacio de memoria que no se utiliza más.

³El paradigma imperativo o de procedimientos es, probablemente, uno de los paradigmas más conocidos en el mundo de la programación. Como su nombre lo indica, este es un método que permite desarrollar programas a través de procedimientos. Mediante una serie de instrucciones, se explica paso por paso cómo funciona el código para que el proceso sea lo más claro posible.



Figura 1.1: Guido van Rossum creador de Python

a objetos⁴ y funcional⁵.

Guido van Rossum (ver figura 1.1) creó el lenguaje Python en el año 1980 como sucesor del lenguaje de programación *ABC* y la primera versión fue presentada en el año 1991 como Python 0.9.0.

Python 2.0 apareció en el año 2000 en el se introdujeron nuevas características como *Comprensión de Listas*⁶, *Recolector de Basura*, *Contador de Referencias*⁷, *soporte Unicode*⁸.

Python 3.0 surge en 2008 como una revisión completa del lenguaje la cual no es compatible con las versiones anteriores. La última versión de Python 2 fue presentada en 2020.

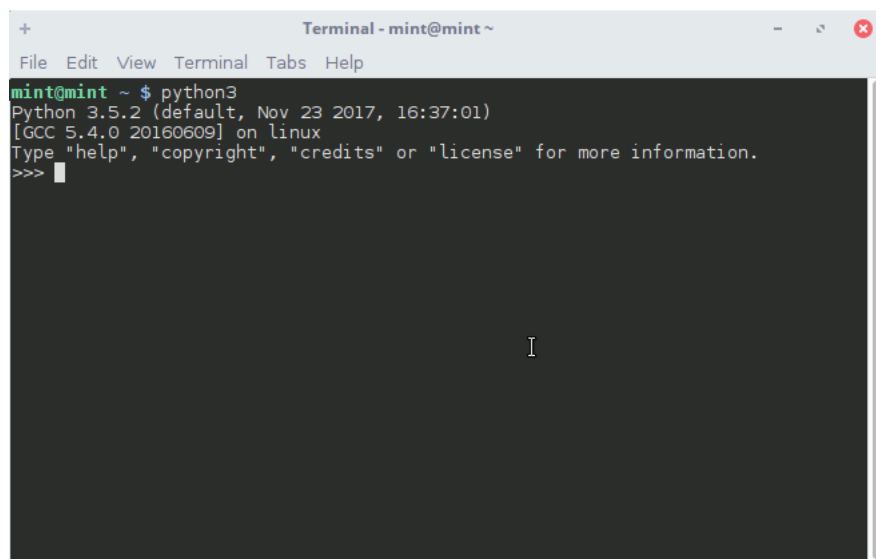
⁴Este tipo de paradigma de programación ofrece una guía que permite identificar cómo trabajar con él a través de objetos y planos de código. Este tipo de paradigma se constituye por piezas simples u objetos que al relacionarse entre sí forman diferentes componentes del sistema que estemos trabajando.

⁵Una de las características del paradigma funcional es que este, como su nombre lo indica, trabaja a través de determinadas funciones matemáticas. Este es un tipo de paradigma que se usa, principalmente, en el ámbito académico más que en el comercial. A diferencia del paradigma imperativo, aquí importa más el “qué” y no tanto el “cómo” se desarrolla un proyecto.

⁶Es una característica del soporte funcional de Python

⁷Una variable que cuenta la cantidad de apuntadores que referencian a una variable.

⁸Unicode es un estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión, y visualización de textos de numerosos idiomas y disciplinas técnicas, además de textos clásicos de lenguas muertas.



```
Terminal - mint@mint ~
File Edit View Terminal Tabs Help
mint@mint ~ $ python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
;>>>
```

Figura 1.2: El intérprete de Python

En este y en los próximos capítulos se presentan los conceptos básicos del lenguaje de programación Python. Estos conceptos permitirán al principiante iniciarse en el mundo de la programación y en la construcción de aplicaciones sencillas usando el lenguaje Python.

1.2. El Intérprete de Python

El intérprete de Python (ver figura 1.2) generalmente se instala como `/usr/local/bin/python3.xx` (xx hace referencia a una versión específica de Python) en aquellas máquinas que tienen un sistema operativo basado en Linux.

Poner `/usr/local/bin` en la ruta de búsqueda del intérprete de comandos de Linux hace posible iniciarlo escribiendo el comando: `python3.xx` en la terminal. Ya que la elección del directorio dónde se instalará el intérprete es una opción del proceso de instalación, puede estar en otros lugares como por ejemplo, `/usr/local/python` es una alternativa común.

En máquinas con Windows en las que haya instalado Python desde Microsoft Store, el comando `python3.11` estará disponible. Si tiene el lanzador



py.exe instalado, se puede usar el comando `py`.

Se puede salir del intérprete con estado de salida 0 (el estado de salida 0 indica que el programa finalizó sin errores) ingresando el carácter de fin de archivo (Control-D en Unix/Linux, Control-Z en Windows). Si eso no funciona, una alternativa posible es utilizar la sentencia: `quit()`.

Las características para edición de líneas del intérprete incluyen *edición interactiva*, *sustitución de historial* y *completado de código* en sistemas que soportan la biblioteca GNU Readline⁹.

Quizás la forma más rápida para comprobar si las características de edición se encuentran disponibles es presionar *Control-P* en la señal del intérprete de Python que aparezca. Si se escucha un sonido, se tiene edición de línea de comandos; si parece que no ocurre nada, o si se muestra `^P`, estas características no están disponibles; solo se podrá usar la tecla de retroceso (backspace) para borrar los caracteres de la línea actual.

El intérprete funciona de manera similar al intérprete de comandos de Linux cuando se llama con:

- Una entrada estándar (el teclado) conectada a un terminal, lee y ejecuta comandos de manera interactiva.
- Un argumento de nombre de archivo o con un archivo como entrada estándar, lee y ejecuta un script desde ese archivo.

Una segunda forma de iniciar el intérprete es `python -c comando [arg] ...`, que ejecuta las sentencias en comando, similar a la opción del intérprete de comandos `-c`.

Como las sentencias de Python a menudo contienen espacios u otros caracteres que son especiales para el intérprete de comandos, generalmente se recomienda escribir comando en su totalidad.

Algunos módulos de Python también son útiles como scripts. Estos pueden invocarse utilizando `python -m módulo [arg] ...`, que ejecuta el archivo

⁹GNU Readline es una biblioteca de software que proporciona funciones de edición e historial en línea para programas interactivos con una interfaz de línea de comandos, como Bash.

fuente para módulo como si se hubiera escrito el nombre completo en la línea de comandos.

Cuando se usa un script, a veces es útil poder ejecutar el script y luego ingresar al modo interactivo. Esto se puede hacer pasando la opción `-i` antes del nombre del script.

1.2.1. Pasaje de Argumentos

Cuando el intérprete reconoce los argumentos, el nombre del script y sus argumentos adicionales se almacenan en una lista de cadenas de texto asignada a la variable `argv` del módulo `sys`. Es importante notar que para hacer uso de esta característica se tiene que importar el módulo `sys` utilizando la sentencia `import sys`.

Esta lista tiene las siguientes características:

- Si se invoca sin scripts ni argumentos la lista tiene una cadena vacía.
- Cuando se pasa el nombre del script con `'-'` (lo que significa la entrada estándar), `sys.argv[0]` vale `'-'`.
- Cuando se usa `-c` comando, `sys.argv[0]` vale `'-c'`.
- Cuando se usa `-m` módulo, `sys.argv[0]` contiene el valor del nombre completo del módulo. Las opciones encontradas después de `-c` comando o `-m` módulo no son consumidas por el procesador de opciones de Python pero de todas formas se almacenan en `sys.argv` para ser manejadas por el comando o módulo.

A continuación se muestran algunos ejemplos de invocación.

Invocación

Asuma que se tiene el programa Ejercicio.py que imprime por pantalla "Hola Mundo"

Se invoca con el programa directamente

```
$python Ejercicio.py # En este caso la lista contiene ['Prueba.py']
```

Hola Mundo

Se invoca con la opción -i esto hace que se ejecute el programa y luego se ingrese al intérprete

```
$python -i Ejercicio.py #En este caso la lista contiene ['Prueba.py']
```

Hola Mundo

>>>

1.2.2. Modo Interactivo

Cuando se leen los comandos desde un terminal, se dice que el intérprete está en modo interactivo. En este modo, espera el siguiente comando con el *prompt*¹⁰ *primario*, generalmente *tres signos de mayor que* (>>>); para las líneas de continuación, aparece el *prompt secundario*, por defecto tres puntos (...).

El intérprete imprime un mensaje de bienvenida que indica su número de versión y un aviso de copyright antes de imprimir el primer prompt primario:

Modo Interactivo

```
$python3.11
```

Python 3.11 (default, April 4 2021, 09:25:04)

GCC 10.2.0 on linux

Type "help", copyright, credits or "license" for more information.

>>>

¹⁰Se llama prompt al carácter o conjunto de caracteres que se muestran en una línea de comandos para indicar que está a la espera de órdenes.

1.2.3. El Intérprete y su Entorno

De forma predeterminada, los archivos fuente de Python se tratan como codificados en UTF-8 ¹¹. En esa codificación, los caracteres de la mayoría de los idiomas del mundo se pueden usar simultáneamente en literales, identificadores y comentarios, aunque la biblioteca estándar solo usa caracteres ASCII para los identificadores, una convención que debería seguir cualquier código que sea portable. Para mostrar todos estos caracteres correctamente, el editor debe reconocer que el archivo es UTF-8, y debe usar una fuente que admita todos los caracteres del archivo.

Para declarar una codificación que no sea la predeterminada, se debe agregar una línea de comentario especial como la primera línea del archivo. La sintaxis es la siguiente:

Sintaxis de Codificación Interactivo

```
# -*- coding: encoding -*-
```

donde *encoding*¹² es uno de los *codecs*¹³ soportados por Python.

Por ejemplo, para declarar que se utilizará la codificación de *Windows-1252*, la primera línea del archivo de código fuente debe ser:

Ejemplo de Codificación Interactivo

```
# -*- coding: cp1252 -*-
```

Una excepción a la regla de primera línea es cuando el código fuente comienza con una línea UNIX *shebang*¹⁴. En ese caso, la declaración de co-

¹¹UTF-8 (8-bit Unicode Transformation Format) es un formato de codificación de caracteres Unicode e ISO 10646 que utiliza símbolos de longitud variable. UTF-8 fue creado por Robert C. Pike y Kenneth L. Thompson. Está definido como estándar por la RFC 3629 de la Internet Engineering Task Force. Actualmente es una de las tres posibilidades de codificación reconocidas por Unicode y lenguajes web, o cuatro en ISO 10646.

¹²Codificación

¹³Un códec es un programa o dispositivo hardware capaz de codificar o decodificar una señal o flujo de datos digitales.

¹⁴Shebang es, en la jerga de Unix, el nombre que recibe el par de caracteres `#!` que se encuentran al inicio de los programas ejecutables interpretados. En algunas ocasiones se le denomina también hash-bang o sharpbang.

dificación debe agregarse como la segunda línea del archivo. Por ejemplo:

Ejemplo de Codificación en Sistemas Tipo Unix Interactivo

```
#!/usr/bin/env python3 # -*- coding: cp1252 -*-
```

1.3. Python como una Calculadora

En esta sección se muestran algunos comandos simples de Python. Para llevar adelante esta tarea se debe iniciar el intérprete y esperar a que aparezca el prompt primario: `>>>`.

1.3.1. Números

El intérprete puede utilizarse como una simple calculadora; se puede introducir una expresión en él y este escribirá los valores.

La sintaxis es sencilla: los operadores `+`, `-`, `*` y `/` funcionan como en la mayoría de los lenguajes (por ejemplo, Pascal o C); los paréntesis `()` pueden ser usados para agrupar.

Por ejemplo:

Ejemplo de Operaciones Aritméticas con Números Interactivo

```
>>> 2 + 2 4
>>> 50 - 5*6 20
>>> (50 - 5*6) / 4 5.0
>>> 8 / 5 # La división siempre retorna un número en punto flotante.
1.6
```

Los números enteros (ej. 2, 4, 20) tienen tipo *int*, los que tienen una parte fraccionaria (por ejemplo 5.0, 1.6) tienen el tipo *float*. Más adelante se analizarán con mayor profundidad los tipos de datos proporcionados por Python.

La división (`/`) siempre retorna un número decimal de punto flotante. Para hacer el *floor division* y obtener un número entero como resultado puede

usarse el operador `//`; para calcular el *resto* se puede usar `%`:

Ejemplo de Operaciones Aritméticas con Números Interactivo

```
>>> 17 / 3 # La división clásica retorna un flotante 5.666666666666667
>>> 17 // 3 # El piso (floor) de la división descarta la parte fraccionaria.
5
>>> 17 % 3 # El operador % operator retorna el resto de la división 2
>>> 5 * 3 + 2 # La parte entera del cociente * divisor + resto (prueba
de que la división de 17 / 3 es correcta) 17
```

Con Python, es posible usar el operador `**` para calcular potencias:

Ejemplo de Potencias Interactivo

```
>>> 5 ** 2 # 5 al cuadrado
25
>>> 2 ** 7 # 2 a la 7
128
```

El signo igual (`=`) se usa para asignar un valor a una variable. Después, no se muestra ningún resultado antes del siguiente prompt interactivo:

Ejemplo de Asignación Interactivo

```
>>> ancho = 20
>>> alto = 5 * 9
>>> ancho * alto
900
```

Si una variable *no está definida* (no se le ha asignado un valor), al intentar usarla dará un error:

Uso de Variable no Creada Interactivo

```
>>> n # Se intenta acceder a una variable no definida.  
Traceback (most recent call last):  
File stdin, line 1, in module  
NameError: name 'n' is not defined
```

Hay soporte completo de *punto flotante*¹⁵; operadores con operando mezclados convertirán los enteros a punto flotante:

Soporte para Punto Flotante Interactivo

```
>>> 4 * 3.75 - 1 14.0
```

En el modo interactivo, la última expresión impresa se asigna a la variable `_`. Esto significa que cuando se está utilizando Python como calculadora, es más fácil seguir calculando, por ejemplo:

Soporte para Punto Flotante Interactivo

```
>>> tax = 12.5 / 100  
>>> precio = 100.50  
>>> precio * impuesto 12.5625  
>>> precio + _  
113.0625  
>>> round(_, 2)  
113.06
```

Esta variable debe ser tratada como de sólo lectura por el usuario. Si se le asigna un valor entonces se creará una variable local independiente con el mismo nombre enmascarando la variable con el comportamiento mágico.

¹⁵Hace referencia a los números reales.

Comentario Importante Modo Interactivo

También se pueden usar cadenas de caracteres en el modo interactivo. Es posible imprimir una cadena, concatenar una cadena, etc. Estas operaciones se verán con más detalle cuando se describa el tipo String.

Comentario Importante Modo Interactivo

Python proporciona muchos tipos de datos que facilitan la programación. Los tipos de datos principales serán explicados en las próximas unidades de este curso. Cubrir todos los tipos de datos no es posible dado que es muy amplio el abanico de tipos de datos que posee el lenguaje de programación.

1.4. Ejercicios

Ejercicio 1: Verifique si en su computadora tiene instalado Python. En caso afirmativo diga en qué carpeta está instalado el lenguaje. En otro caso, luego de instalar Python, indique en qué carpeta se instaló.

Ejercicio 2: Ejecute el intérprete de Python luego:

1. Describa en qué consiste.
2. Pruebe distintas formas de salir del intérprete.

Ejercicio 3: Ejecute la sentencia `print("Hola Mundo")` desde el intérprete de comandos del sistema operativo.

Ejercicio 4: Abra un editor de texto y copie el siguiente código:

```
import sys
print("Hola", sys.argv[0])
```



Grabe el archivo con el nombre Ejercicio1.py. Luego ejecútelo como un módulo de Python.

Ejercicio 5: Cuando se desea ver los argumentos que se reciben como parámetros en un programa Python se debe:

1. Importar el módulo sys (`import sys`)
2. Imprimir la variable argv (`print (sys.argv)`)

Teniendo en cuenta esta información se pide verificar si Python se invoca:

1. Sin scripts y sin módulos la lista contiene el string vacío (`''`)
2. Con `-` la lista contiene `-`
3. Con `-c` y un comando por ejemplo `print("Hola Mundo")` la lista contiene `-c`
4. Con `-m` módulo la lista contiene el nombre del módulo completo.

Ejercicio 6: Usando el intérprete de Python calcule:

1. El área de un triángulo cuya base es 10cm y altura 20cm.
2. El área de un cuadrado cuyos lados miden 40cm.
3. El Perímetro de un triángulo equilátero. Seleccione ud. la longitud del lado.
4. El perímetro de un rectángulo. Seleccione ud. la longitud de los lados.