TEMA 1

Introducción

Completado100 XP

- 1 minuto

Python es uno de los lenguajes de programación más populares y con un crecimiento más rápido del mundo. Se usa para todo tipo de tareas, como las de programación web y análisis de datos, y se ha convertido en *el lenguaje* que hay que conocer para el aprendizaje automático. Esa popularidad significa que los desarrolladores de Python están muy solicitados y los trabajos de programación de Python pueden ser rentables.

Las razones mencionadas anteriormente representan el por qué podría ser una buena idea aprender a programar en Python. En este módulo se proporcionará una introducción al uso de Python para compilar una aplicación, que puede ser un punto de partida para convertirse en programador de Python.

Objetivos de aprendizaje

En este módulo, aprenderá a:

- Explorar las opciones disponibles para ejecutar aplicaciones de Python.
- Usar el intérprete de Python para ejecutar instrucciones y scripts.
- Obtener información sobre cómo declarar variables.
- Compilar una aplicación sencilla de Python que toma una entrada y genera una salida.

TEMA 2

¿Qué es Python?

Completado100 XP

- 3 minutos

Python es uno de los lenguajes de programación más populares del mundo. Creado a principios de la década de los 90, Python posee una amplia gama de usos, desde automatizar tareas repetitivas y escribir aplicaciones web hasta compilar modelos de Machine Learning e implementar redes neuronales. A los investigadores, matemáticos y, en particular, a los científicos de datos les gusta Python gracias a su sintaxis completa y fácil de entender y a la amplia gama de paquetes de código abierto disponibles. Los **paquetes** son bibliotecas de código compartido que están disponibles de forma gratuita para todos los usuarios.

Python cuenta con una sintaxis sencilla y fácil de aprender que enfatiza la legibilidad. Las aplicaciones escritas en Python se pueden ejecutar en casi cualquier equipo, como los que ejecutan Windows, macOS y las distribuciones populares de Linux. Asimismo, el ecosistema contiene un amplio conjunto de herramientas de desarrollo para escribir, depurar y publicar aplicaciones de Python.

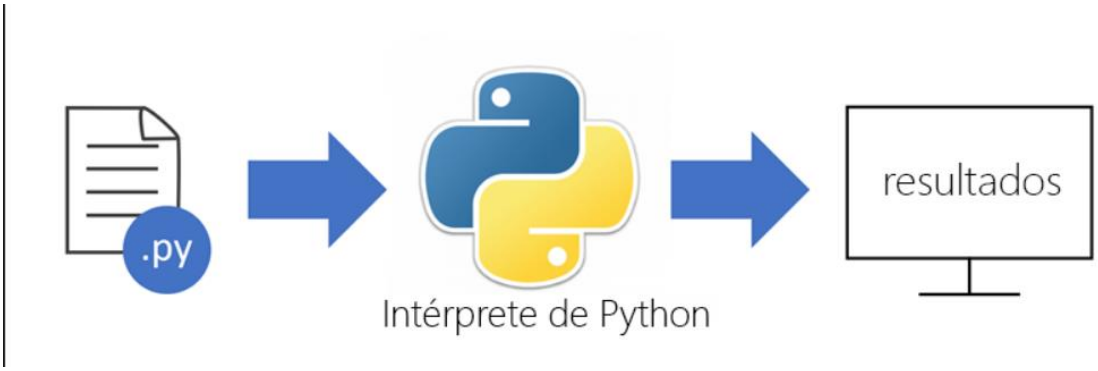
Por último, Python se halla respaldada por una comunidad de usuarios activos dispuesta a ayudar a que los programadores nuevos aprendan el **estilo de Python**, por el cual no solo se obtiene la sintaxis correcta, sino que el lenguaje se usa para el fin para el que estaba previsto.

Ejecución de código de Python

Python es un lenguaje *interpretado*, lo que reduce el ciclo editar-probar-depurar porque no se requiere ningún paso de compilación. A fin de ejecutar aplicaciones de Python, necesita un entorno o un intérprete de runtime para ejecutar el código.

La mayoría de los entornos de runtime admiten dos formas de ejecutar código de Python:

- **Modo interactivo:** en este modo, cada comando que se escribe se interpreta y ejecuta inmediatamente, y se ven los resultados cada vez que se presiona Entrar. El modo interactivo es el modo predeterminado si no se pasa un nombre de archivo al intérprete.
- **Modo de script:** en este modo, se coloca un conjunto de instrucciones de Python en un archivo de texto con una extensión **.py**. Después, se ejecuta el intérprete python y se apunta al archivo. El programa se ejecuta línea por línea y se muestra la salida. No hay ningún paso de compilación, como se muestra en el diagrama siguiente:



Implementaciones de Python

Python se autoriza bajo la licencia de código abierto OSI y, en función de sus necesidades, existen varias implementaciones disponibles. Aquí se muestran algunas de las opciones disponibles:

- **CPython, la implementación de referencia**: La más popular es la implementación de referencia (CPython), disponible en el sitio web de Python. CPython se usa habitualmente para desarrollo web, desarrollo de aplicaciones y creación de scripts. Hay paquetes de instalación para Windows y macOS. Los usuarios de Linux pueden instalar Python mediante los administradores de paquetes integrados tales como **apt**, **yum** y **Zypper**. También hay un área de juegos en línea en la que se pueden probar las instrucciones de Python directamente en el sitio web. Por último, está disponible el código fuente completo, lo que permite compilar su propia versión del intérprete.
- **Anaconda**: Anaconda es una distribución especializada de Python adaptada para tareas de programación científicas, como la ciencia de datos y el aprendizaje automático. Consulte más detalles sobre Anaconda aquí.
- **IronPython**: ironPython es una implementación de código abierto de Python compilada en el runtime de .NET. Más información sobre IronPython.
- **Jupyter Notebook**: Jupyter Notebook es un entorno de programación interactivo basado en web que admite diversos lenguajes de programación, incluido Python. Jupyter Notebooks son de uso generalizado en la investigación y la enseñanza de modelos matemáticos, aprendizaje automático, análisis estadístico y de enseñanza y aprendizaje sobre cómo codificar. Instale cuadernos de Jupyter Notebook.

En este módulo vamos a usar Azure Cloud Shell para desarrollar Python, pero el resumen tiene vínculos para descargar e instalar Python en el equipo local una vez que se haya completado este módulo.

Tema 3

Usar el REPL

Completado100 XP

A veces querrá probar un fragmento de código sin tener que primero crear un archivo para él. En esas ocasiones, es buena idea usar un programa integrado denominado REPL que le permite escribir instrucciones más cortas y evaluar esas instrucciones.

REPL de Python

Python es compatible con una experiencia de consola interactiva que permite escribir en comandos y ver los resultados inmediatamente. Esto se conoce a veces como "read–eval–print loop" o **REPL**.

Para usar REPL, escriba python en la consola. Recibirá un mensaje similar a la salida siguiente que, después, espera a que escriba comandos:

OutputCopiar

Python 3.9.14 (main, Oct 29 2022, 22:18:10)

[GCC 11.2.0] on linux

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

>>>

¿Qué puede hacer?

Con REPL puede hacer la mayoría de las cosas que podría hacer en un archivo de código. Por ejemplo:

- **Ejecutar y evaluar instrucciones**. Puede hacer que REPL evalúe una instrucción como:

OutputCopiar

>>> 1+1

2

>>>

- **Declarar variables y funciones**. También puede crear variables y funciones y REPL recordará que existen, en caso de que intente usarlas más adelante:

OutputCopiar

>>> PI = 3.14

>>> PI

3.14

- **Usar la ayuda integrada**. Obtener ayuda sobre un tema puede ser útil, ya que no tiene que salir del editor de su elección para navegar en la Web, pero puede continuar creando código.

Uso de la ayuda

El REPL tiene una función de ayuda integrada que se puede usar para buscar funciones y palabras clave. La sintaxis generalizada para esta función es la siguiente:

PythonCopiar

Donde [object] es una función específica o palabra clave de la que quiere obtener ayuda.

Consola de ayuda interactiva

Si no pasa un argumento a la función de ayuda, Python iniciará una ayuda interactiva.

Especifique la consola de ayuda interactiva escribiendo help(). Este comando mostrará una lista con algunas instrucciones básicas sobre cómo usar el sistema de ayuda.

Desde aquí puede escribir en el elemento que le interesa. Al escribir string, por ejemplo, obtendrá información sobre el tema string, que tiene un aspecto similar al siguiente:

Tema 4

Variables y tipos de datos básicos en Python

Completado

100 XP

4 minutos

Las variables son uno de los bloques de creación fundamentales de los programas escritos en Python. Las variables retienen los datos en la memoria. Tienen nombres y se puede hacer referencia a ellas por esos nombres. Las variables también tienen tipos, que especifican el tipo de datos que pueden almacenar (como una cadena y un entero) y se pueden usar en expresiones que usan operadores (como + y -) para manipular los valores.

Variables

En Python, se declara una variable y se le asigna un valor mediante el operador de asignación =. La variable está en el lado izquierdo del operador y el valor asignado (que puede ser una expresión como 2 + 2 e incluso puede incluir otras variables) se encuentra en el lado derecho. Por ejemplo:

Python

Copiar

```
x = 1       # assign variable x the value 1
y = x + 5   # assign variable y the value of x plus 5
z = y       # assign variable z the value of y
```

Estos ejemplos asignan números a variables, pero los números son solo uno de los varios tipos de datos que se admiten en Python. Observe que no hay ningún tipo declarado para las variables. Python es un lenguaje escrito dinámicamente, lo que significa que el tipo de variable viene determinado por los datos que se le asignan. En los ejemplos anteriores, las variables x y z son tipos enteros, capaces de almacenar números enteros positivos y negativos.

Los nombres de variables distinguen entre mayúsculas y minúsculas y pueden usar cualquier letra, número y el carácter de subrayado (_). Sin embargo, no pueden empezar con un número.

Trabajo con números

La mayoría de los programas manipulan números. Los equipos tratan números enteros y números decimales de forma distinta. Tenga en cuenta el código siguiente:

Python

Copiar

```
x = 1     # integer
```

x = 1.0 # decimal (floating point)

Python crea enteros a partir de un tipo de datos integrado llamado int y de decimales (números de punto flotante) como instancias de float. La función integrada type() de Python devuelve el tipo de datos de una variable. El código siguiente da como resultado tipos de datos:

Python

Copiar

```
x = 1

print(type(x)) # outputs: <class 'int'>


x = 1.0

print(type(x)) # outputs: <class 'float'>
```

La incorporación de .0 al final de 1 supone una gran diferencia en cómo trata el lenguaje de programación a un valor. El tipo de datos afecta a cómo se almacena el valor en la memoria, cómo controla los datos el procesador (CPU) al evaluar expresiones, cómo unos datos se relacionan con otros y qué tipos de operaciones puede realizar.

Trabajo con valores booleanos

Otro tipo de datos habitual es el tipo booleano, que contiene el valor True o False:

Python

Copiar

```
x = True

print(type(x)) # outputs: <class 'bool'>
```

Internamente, bool se trata como un tipo especial de entero. Técnicamente, True tiene un valor de 1 y False tiene un valor de 0. Normalmente, los valores booleanos no se usan para realizar operaciones matemáticas, sino que se usan para tomar decisiones y realizar una bifurcación. No obstante, es interesante comprender la relación entre los tipos. Muchos tipos no son más que versiones especializadas de tipos más generales. Los enteros son un subconjunto de los números de punto flotante y los valores booleanos son un subconjunto de los enteros.

Trabajo con cadenas

Las cadenas son, junto con los números, uno de los tipos de datos más usados. Una cadena es una colección de cero o más caracteres. Las cadenas se declaran normalmente mediante comillas simples, pero también puede utilizar comillas dobles:

Python

Copiar

```
x = 'This is a string'

print(x) # outputs: This is a string

print(type(x)) # outputs: <class 'str'>

y = "This is also a string"
```

Puede agregar cadena a otras cadenas (una operación que se conoce como "concatenación") con el mismo operador + que suma dos números:

Python

Copiar

```
x = 'Hello' + ' ' + 'World!'

print(x) # outputs: Hello World!
```

Aprenderá más acerca de las cadenas en otra lección, incluido cómo analizarlas y manipularlas de varias maneras. También aprenderá sobre otros tipos de datos importantes, como listas, que almacenan colecciones de datos y normalmente se usan para retener colecciones de cadenas.

Impresión en la consola

En Python, la función print, que es una de más de 60 funciones integradas en el lenguaje, genera texto en la pantalla.

La siguiente instrucción muestra "Hola mundo!" en la pantalla:

Python

Copiar

```
print('Hello World!')
```

El argumento pasado a print es una cadena, que es uno de los tipos de datos fundamentales en Python y que se usa para almacenar y administrar el texto. De forma predeterminada, print genera un carácter de nueva línea al final de la línea, de modo que una llamada posterior a print se inicie en la línea siguiente.

TEMA 5

Prueba de la primera instrucción de Python

Python es compatible con una experiencia de consola interactiva que permite escribir en comandos y ver los resultados inmediatamente.

En la consola de Cloud Shell, escriba python para iniciar el intérprete de Python en el modo interactivo.

python

Debería ver una salida similar a:

Output

Copiar

Python 3.9.14 (main, Oct 29 2022, 22:18:10)

[GCC 11.2.0] on linux

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

>>>

>>> es el intérprete que está esperando a que escriba un comando de Python.

Escriba la instrucción siguiente en el intérprete.

Python

Copiar

print('Hello World!')

La instrucción debe devolver directamente el texto y, después, crear otro aviso esperando el comando siguiente.

Output

Copiar

Hello World!

>>>

Declaración y generación de variables

Con REPL en ejecución, vamos a crear variables a continuación.

Escriba el código siguiente para declarar una variable:

Python

Copiar

PI = 3.14

Escriba el nombre de la variable PI para repetir su valor:

Python

Copiar

PI

La salida debe mostrar su valor:

Output

Copiar

3.14

TEMA 6

Lectura de la entrada de teclado

Completado

100 XP

3 minutos

Muchos programas son interactivos. La compatibilidad con interactividad significa que tiene un programa que se ejecuta de forma diferente en función de la entrada. Normalmente, quien introduce datos en un programa es un usuario, pero puede ser otro programa. Hay muchas maneras de enviar datos a un programa. Dos maneras comunes son a través de una interfaz gráfica o una consola.

Entrada de usuario

Para leer la entrada hecha desde el teclado, Python proporciona la función input(). input() lee lo que el usuario escribe en el teclado y lo devuelve como una cadena. Aquí tiene un ejemplo en el que se combinan input() y print() para capturar el nombre de una persona y mostrarlo en la pantalla:

Python

Copiar

```
name = input('Enter your name:')

print(name)
```

La cadena se ha pasado como un argumento a la función input, que es el aviso que el usuario verá. En este ejemplo, le pide al usuario que escriba su nombre ("Escriba su nombre"). Una vez que el usuario escriba un nombre y pulse Entrar, la función input volverá. El valor devuelto de la función es el texto que ha escrito el usuario y este texto se asigna a la variable llamada name. Después, la variable name se usa como una entrada o un argumento para la función print, que dará como resultado el nombre que el usuario ha introducido.

También se puede llamar a la función input sin un parámetro:

Python

Copiar

```
print('What is your name?')

name = input()

print(name)
```

El comportamiento de este programa será casi igual que el primero de ellos. La diferencia es que print agrega, de forma predeterminada, una nueva línea a la salida.

Lectura de números como entrada

La función input siempre devuelve el valor escrito como una cadena (texto). Esta elección tiene sentido porque el usuario puede escribir cualquier valor que quiera. Incluso si la entrada es un número válido, se devuelve como un tipo de cadena desde la función input. Por ejemplo:

Python

Copiar

```
x = input('Enter a number: ')

print(type(x))
```

Ejecutar este código y escribir el valor de "5" mostraría <class 'str'>, aunque el propio valor es numérico. Para convertir el valor en una variable de entero verdadera, se puede usar la función int():

Python

Copiar

```
x = int(input('Enter a number: '))

print(type(x))
```

Para el valor "5", este código dará como resultado <class 'int'>. Se puede usar la función float igual que cuando se espera un componente fraccional.

Importante

¿Qué ocurre si la entrada no es numérica y se pasa a la función int()? Como puede imaginar, esto sería un error y produciría un error de runtime. El programa finalizará en esta declaración: puede probarlo por su cuenta en la consola interactiva de Python. Hablaremos sobre diversas soluciones para controlar estos tipos de errores en módulos futuros.

Conversión de números en cadenas

También puede ir en la otra dirección. El método str() tomará un valor entero o flotante y lo convertirá en una cadena. Es necesario llamar al método str() si quiere que funcione el ejemplo de código siguiente. La conversión garantiza que el entero, en su forma de cadena, se concatena con la cadena de la izquierda.

Python

Copiar

```
x = 5

print('The number is ' + str(x))
```