**Conceptos**

Estos son los conceptos y temas de programación de Python que debes aprender durante esta lección.

**¿Qué es una función?**

Una *función* es un grupo de instrucciones que juntas realizan una tarea. En términos generales, existen cuatro tipos de funciones en Python:

1. Funciones integradas
2. Funciones de la biblioteca estándar
3. Funciones de terceros
4. Funciones definidas por el usuario

En la lección anterior, aprendió a llamar a los dos primeros tipos de funciones. En la semana 3, aprenderá a instalar módulos de terceros y a llamar a funciones de terceros. En esta lección, aprenderá a escribir y llamar a funciones definidas por el usuario.

**¿Qué es una función definida por el usuario?**

Una *función definida por el usuario* es una función que no es una función incorporada, una función estándar o una función de terceros. Una función definida por el usuario es escrita por un programador como usted como parte de un programa. Para algunos estudiantes, el término "función definida por el usuario" es confuso porque el usuario de un programa no define la función. En cambio, el programador (usted) define las funciones definidas por el usuario. Quizás un término más correcto sea función definida por el programador. Escribir funciones definidas por el usuario tiene varias ventajas, entre ellas:

1. haciendo que su código sea más reutilizable
2. Haciendo que su código sea más fácil de entender y depurar
3. Haciendo que su código sea más fácil de cambiar y agregar capacidades

**Cómo escribir una función definida por el usuario**

Para escribir una función definida por el usuario en Python, simplemente escriba el código que coincida con esta plantilla:

|  |  |
| --- | --- |
|  | def function\_name(param1, param2, … |
|  | paramN): |
|  | """documentation string""" |
|  | statement1 |
|  | statement2 |
|  | ⋮ |
|  | statementN |
|  | return value |

La primera línea de una función se llama *encabezado* o *firma* e incluye lo siguiente:

1. la palabra clave **def**(que es una abreviatura de "definir")
2. el nombre de la función
3. la lista de parámetros (con los parámetros separados por comas)

Aquí está el encabezado de una función llamada **draw\_circle** que toma tres parámetros llamados **x**, **y**, y **radius**:

**def draw\_circle(x, y, radius):**

Podrías leer la línea de código anterior como: “Define una función llamada **draw\_circle**que toma tres parámetros llamados *x* , *y* y *radio* ”.

El *nombre de la función* debe comenzar con una letra o un guión bajo (\_). El resto del nombre debe estar formado por letras, dígitos (0 a 9) o un guión bajo. El nombre de una función no puede incluir espacios ni otros signos de puntuación. El nombre de una función debe ser significativo y debe describir brevemente lo que hace la función. Las funciones bien nombradas suelen comenzar con un verbo.

Las declaraciones dentro de una función se denominan *cuerpo* de la función. Al igual que otras declaraciones de bloque en Python, como **if**, **else**, **while**y **for**, todas las cuales terminan con dos puntos (:), debe sangrar las declaraciones dentro del cuerpo de una función. El cuerpo de una función debe comenzar con una *cadena de documentación* que es una cadena entre comillas triples que describe el propósito de la función, los parámetros y el valor de retorno. El cuerpo de una función puede contener tantas declaraciones como desee escribir dentro de él. Sin embargo, es una buena idea limitar las funciones a menos de 20 líneas de código.

Un cilindro circular recto con radio *r* y altura *h*

El ejemplo 1 contiene una función denominada **print\_cylinder\_volume()**sin parámetros que obtiene dos números del usuario: *radio* y *altura* , y utiliza esos números para calcular el volumen de un cilindro circular recto y luego imprime el volumen para que el usuario lo vea.

|  |  |
| --- | --- |
|  | *# Example 1* |
|  | import math |
|  | *# Define a function named print\_cylinder\_volume.* |
|  | def print\_cylinder\_volume(): |
|  | """Compute and print the volume of a cylinder. |
|  | Parameters: none |
|  | Return: nothing |
|  | """ |
|  | *# Get the radius and height from the user.* |
|  | radius = float(input("Enter the radius of a cylinder: ")) |
|  | height = float(input("Enter the height of a cylinder: ")) |
|  | *# Compute the volume of the cylinder.* |
|  | volume = math.pi \* radius\*\*2 \* height |
|  | *# Print the volume of the cylinder.* |
|  | print(f"Volume: {volume:.2f}") |

Como la **print\_cylinder\_volume**función del ejemplo 1 no acepta parámetros, debe llamarse sin ningún argumento de esta manera:

**print\_cylinder\_volume()**

**Cómo hacer que una función definida por el usuario sea reutilizable**

Debido a que la **print\_cylinder\_volume**función del ejemplo 1 obtiene información de un usuario e imprime sus resultados en una ventana de terminal, solo se puede usar en un programa que se ejecuta cuando un usuario está presente. No se puede usar en un programa que se ejecuta automáticamente y obtiene información de un archivo, la red o un sensor. En otras palabras, la **print\_cylinder\_volume**función del ejemplo 1 no se puede reutilizar en otros programas. Las *funciones más reutilizables* son las que toman parámetros, realizan cálculos y devuelven un resultado, pero *no realizan la entrada y salida del usuario* .

La lista de parámetros en el encabezado de una función contiene datos almacenados en variables que la función necesita para completar su tarea. Un *parámetro* es una variable cuyo valor proviene de fuera de la función. Una forma de obtener información para una función es pedirle información al usuario llamando a la función integrada de Python **input** . Otra forma de obtener información para una función es a través de los parámetros de la función. Obtener información a través de parámetros es mucho más flexible que pedirle información al usuario porque la información a través de parámetros puede provenir del usuario o de un archivo en un disco duro o de la red o de un sensor o incluso de otra función.

El ejemplo 2 contiene otra versión de la **print\_cylinder\_volume**función. Esta segunda versión no obtiene el radio ni la altura del usuario, sino que obtiene la información a través de sus dos parámetros llamados *radio* y *altura* .

|  |  |
| --- | --- |
|  | *# Example 2* |
|  | import math |
|  | *# Define a function named print\_cylinder\_volume.* |
|  | def print\_cylinder\_volume(radius, height): |
|  | """Compute and print the volume of a cylinder. |
|  | Parameters |
|  | radius: the radius of the cylinder |
|  | height: the height of the cylinder |
|  | Return: nothing |
|  | """ |
|  | *# Compute the volume of the cylinder.* |
|  | volume = math.pi \* radius\*\*2 \* height |
|  | *# Print the volume of the cylinder.* |
|  | print(volume) |

Dado que la segunda versión de la **print\_cylinder\_volume**función acepta dos parámetros, debe llamarse con dos argumentos de la siguiente manera:

**print\_cylinder\_volume(2.5, 4.1)**

Para *devolver* un resultado de una función, simplemente escriba la palabra clave **return**seguida del resultado que desea devolver a la función que realiza la llamada. El ejemplo 3 contiene una tercera versión de la función de volumen del cilindro. Observe que la versión del ejemplo 3 devuelve el volumen en lugar de imprimirlo, lo que hace que la función sea más reutilizable. Observe también que en el ejemplo 3 cambiamos el nombre de la función de **print\_cylinder\_volume**a **compute\_cylinder\_volume**porque esta versión no imprime el volumen sino que lo devuelve.

|  |  |
| --- | --- |
|  | *# Example 3* |
|  | import math |
|  | *# Define a function named computer\_cylinder\_volume.* |
|  | def compute\_cylinder\_volume(radius, height): |
|  | """Compute and return the volume of a cylinder. |
|  | Parameters |
|  | radius: the radius of the cylinder |
|  | height: the height of the cylinder |
|  | Return: the volume of the cylinder |
|  | """ |
|  | *# Compute the volume of the cylinder.* |
|  | volume = math.pi \* radius\*\*2 \* height |
|  | *# Return the volume of the cylinder so that the* |
|  | *# volume can be used somewhere else in the program.* |
|  | return volume |

Muchas funciones que has usado en el pasado, como **input**, **float**y **round**, devuelven un resultado. Cuando una función devuelve un resultado, normalmente escribimos código para almacenar ese resultado devuelto en una variable para usarlo más adelante en el programa, como se muestra a continuación:

**text = input("Please enter your name: ")**

Dado que la **compute\_cylinder\_volume**función del ejemplo 3 acepta dos parámetros y devuelve un resultado, podría llamarse así: