Contents

How To Use This Book

The creation of this book has provided me, Mike McGrath, a welcome opportunity to produce an introduction to coding computer programs for readers with no previous coding experience. Although this is a book for beginners, it goes beyond the mere basics so some topics may be more easily understood after gaining some coding experience with the simpler listed programs. The examples demonstrate coding features using the popular Python programming language.

Conventions in this book

In order to clarify the code listed in the steps given in each example, I have adopted the same default colorization convention provided by Python's code editor. Keywords of the Python language itself are colored orange, built-in function names are purple, coder-specified function names are blue, text strings are green, comments are red, and all other code is black, like this:

```
# A function to display a greeting
def greet( reader ) :
    print( 'Welcome to Coding for Beginners in easy steps' , reader )
```

Additionally, in order to identify each source code file described in the steps, an icon and file name appears in the margin alongside the steps.

program.py

Grabbing the source code

For convenience I have placed source code files from the examples featured in this book into a single ZIP archive. You can obtain the complete archive by following these easy steps:

1. Visit **www.ineasysteps.com** and log in, then navigate to Free Resources and click the Browse Now button in the Source code downloads and other book resources section

2. Find Coding for Beginners in easy steps, 3rd edition in the list, then click on the hyperlink entitled All code examples to download the archive

3. Now, extract the archive contents to any convenient location on your computer

The screenshots in this book illustrate the actual results of executing the listed code steps. If you don't achieve the result illustrated in any example, simply compare your code to that in the original example files you have downloaded to discover where you went wrong.

1 Getting Started

Welcome to the exciting, fun world of computer coding! This chapter describes how to create your own programming environment and demonstrates how to code your very first program.

Programming Code

A computer is merely a machine that can process a set of simple instructions very quickly. The set of instructions it processes is known as a "program", and the instructions are known as "code".

People who write computer programs are known as "programmers" or "coders". Their programs have enabled computers to become useful in almost every area of modern life:

- **In the hand** – computers are found in cellphone devices for tasks such as communication via voice, text, and social media.
- **In the home** – computers are found in household devices such as TV sets, gaming consoles, and washing machines.
- **In the office** – computers are found in desktop devices for tasks such as word processing, payroll, and graphic design.
- **In the store** – computers are found in retail devices such as automatic teller machines (ATMs) and bar code scanners.
- **In the car** – computers are found in control devices for tasks such as engine management, anti-lock braking, and security.
- **In the sky** – computers are found in airplanes for piloting, and in air traffic control centers for safe navigation.

These are, in fact, just a few examples of how computers affect our lives today. Yet, computers are really dumb! They cannot think for themselves.

A computer is a collection of electronic components – collectively known as "hardware". To make the computer function it must be given a set of program instructions – known as "software".

It is important that each computer program provides clear step-by-step instructions that the computer can execute without errors. The coder must therefore break down the task required of the computer into simple unambiguous steps. For example, a program to move a mobile robot from indoors to outdoors must include instructions to have the robot locate a doorway and navigate around any obstacles. So the coder must always consider what possible unexpected difficulties a program may encounter.

Program instructions must be presented to the computer in a language it can understand. At the most basic level the computer can understand "machine code", which moves items around in its memory to perform tasks. This type of obscure low-level code is incredibly tedious as it requires many lines of instruction to perform even a simple task.

Fortunately, over the years, many "high-level" programming languages have been developed that allow the coder to compose instructions in more human-readable form. These modern high-level programs are automatically translated into the machine code that the computer can understand by a "compiler" or by an "interpreter". In order to become a coder you must typically learn at least one of these high-level programming languages:

- **C** – A powerful compiled language that is closely mapped to machine code and used to develop operating systems.
- **C++** – An enhanced compiled language developing on C to provide classes for Object Oriented Programming (OOP).
- **C#** – A modern compiled language designed by Microsoft for the .NET framework and Common Language Infrastructure.
- **Java** – A portable compiled language that is designed to run on any platform regardless of the hardware architecture.
- **Python** – A dynamic interpreted language that allows both functional and Object Oriented Programming (OOP).

Just as human languages have similarities – such as verbs and nouns – these programming languages have certain similarities as they each possess "data structures" in which to store information, and "control structures" that determine how the program proceeds.

The examples in this book use the Python language to demonstrate how to code computer programs, as it has a simple language syntax, requires no compilation, includes a large library of standard functions, and can be used to create both Console programs and windowed GUI (Graphical User Interface) apps.

Programs written in an interpreted language can be run immediately, but those written in compiled languages must first be compiled before they can be run.

Python is a total package that has a "batteries included" philosophy.

Setting Up

Before you can begin coding programs in the Python language you need to set up a programming environment on your computer by installing the Python interpreter and the standard library of tested code modules that comes along with it. This is available online as a free download from the Python Software Foundation.

Installers for macOS and other platforms are also freely available at python.org/downloads

1 Launch a web browser and navigate to **python.org/downloads** then click the **Download** button to grab the latest version for your system – in this example it's a Windows installer file named "python-3.13.0-amd64.exe"

2 When the download completes, find the executable (**.exe**) file in your Downloads folder, then run the file to launch the Python "Setup" dialog

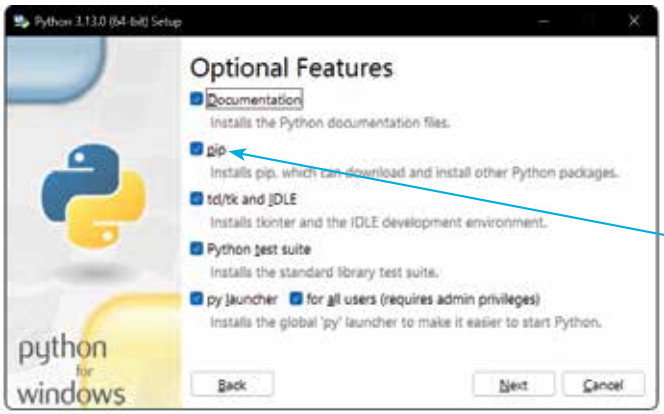
3 Next, be sure to check the Python Setup option box to select the feature to **Add python-exe to PATH**

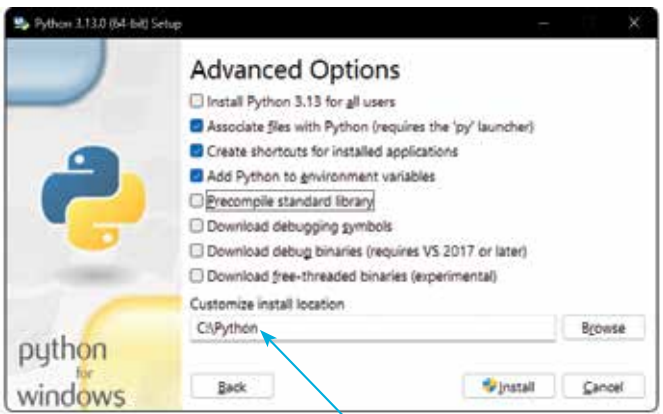
4 Now, click **Customize installation** to open an "Optional Features" dialog, and select all options

5 Click **Next** to open an "Advanced Options" dialog, then choose to **Associate files**, **Create shortcuts**, and **Add Python to environment variables**

Adding Python to the System Path makes it available from within any directory. After installation, you can enter the exact command **python -V** at a Command Prompt to see the interpreter respond with its version number.

Although the latest available Python version may be later than version 3.13, it does not affect Python commands and functions illustrated in this book using Python 3.13.





Be sure to install the Python package manager **pip**. It will be used later in this book to add a **PyInstaller** package with which you can create distributable apps.

6 Set the installation location to **C:\Python**, then click the **Install** button

7 If offered, click the option to **Disable path length limit** to avoid any path-related issues

8 Click the **Close** button to complete the Python Setup

Upon completion, the Python group is added to your Start/All Apps menu. Most important of this group is the **IDLE** item that launches the Python Integrated DeveLopment Environment.

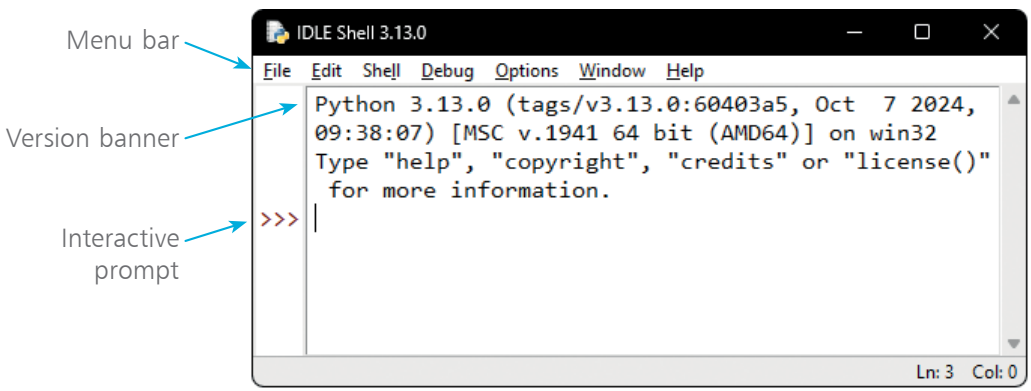
You will use the IDLE launcher often, so right-click on its icon and choose **More**, **Pin to taskbar** to make it readily available from the Windows Desktop.

Exploring IDLE

The installed Python software package includes the Integrated DeveLopment Environment (IDLE) in which you can easily code and run programs or snippets written in the Python language. IDLE provides two different windows for program development:

- Shell Window
- Edit Window

When you start up IDLE it opens a new window containing a menu bar, a banner describing the version, and a **>>>** prompt. This is the Shell Window in which you can interact directly with the Python interpreter by entering statements at the prompt.



If the interpreter understands your entry it will respond with an appropriate reply, otherwise it will report an error.

You can make the interpreter print out a string of text by entering a Python **print()** function statement that encloses your string within quote marks inside the parentheses at the interactive prompt.

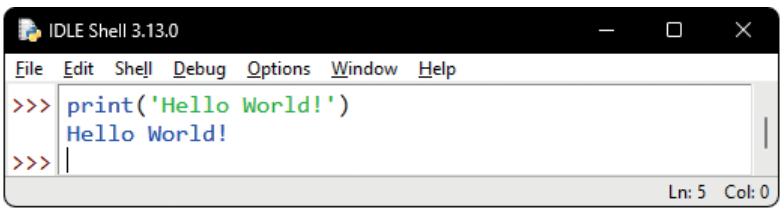
You can also make the interpreter print out the result of a simple arithmetic sum by entering a valid sum statement at the prompt.

If your statement is not valid, such as a sum that attempts to divide a number by 0, the interpreter will print out an error message that helpfully describes the nature of the error.

Most programming languages require text strings to be enclosed in quote marks to differentiate them from program code. By convention, Python coders use single quotes.

1. Open an IDLE Shell Window, then precisely enter this statement at the interactive prompt
   `print( 'Hello World!' )`

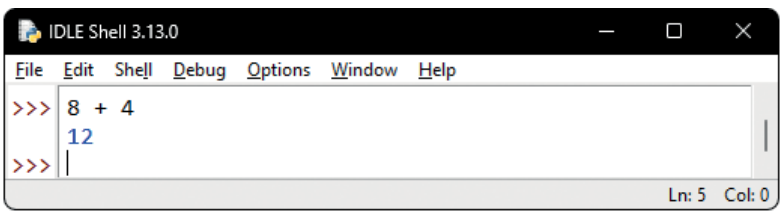
2. Next, hit the **Return** key to see the interpreter's response



Spaces in statements are ignored – so **8+4** can be entered without spaces.

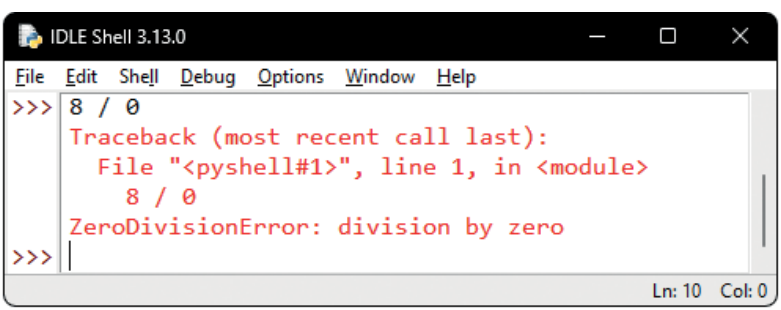
3. Now, enter this sum statement at the interactive prompt
   8 + 4

4. Hit **Return** to see the interpreter print the result total



5. Enter this invalid statement at the interactive prompt
   8 / 0

6. Hit **Return** to see the interpreter print an error message



The Shell Window is mostly used to test snippets of code.

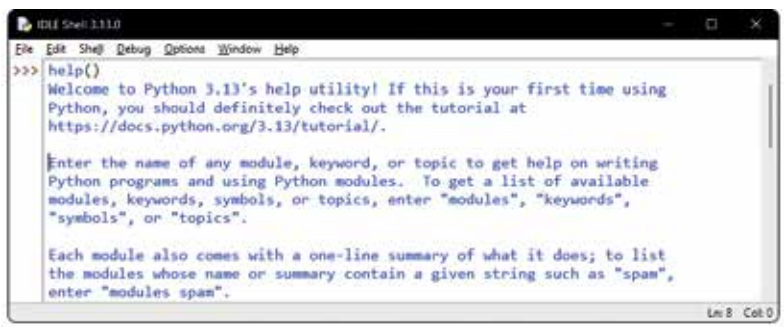
Getting Help

The IDLE Shell Window provides a great help utility where you can find help on any Python topic when coding Python programs. Help can be sought by entering a Python **help()** statement at the interactive **>>>** prompt. A welcome message appears, and the prompt changes to **help>** to denote you are now in "help mode".

1. Open an IDLE Shell Window, then precisely enter this statement at the interactive prompt
   help( )

2. Next, hit the **Return** key to enter help mode

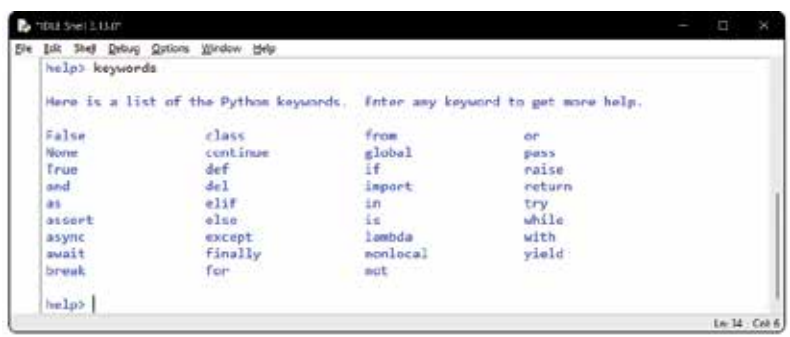
The help utility welcome message also contains handy hints – but are omitted here for brevity.



3. Now, enter this topic name at the help utility prompt
   keywords

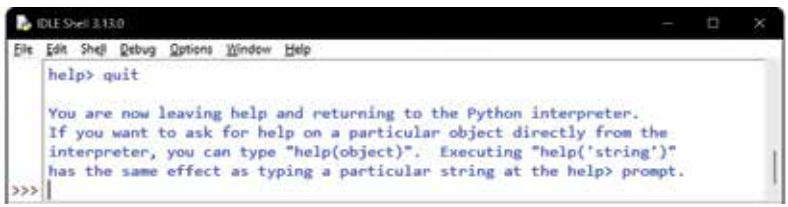
4. Hit **Return** to list all keywords of the Python language

Keywords are the vocabulary of a programming language. Note that Python keywords are case-sensitive – these are all in lowercase except **False**, **None**, and **True**.



5 Then, enter this command at the help utility prompt
quit

6 Hit **Return** to exit help mode and return to an interactive Shell Window prompt

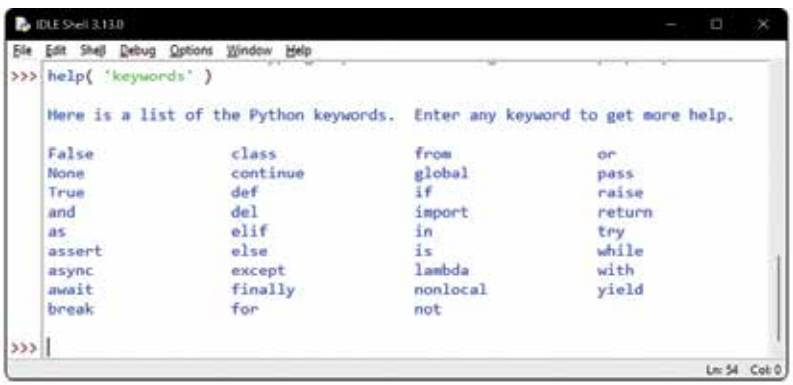


There are no parentheses required after the **quit** instruction – here, it is a help utility command, not a Python statement.

When you just want help on a single topic, you can simply enter the topic name within quote marks inside the parentheses of a **help()** statement at the interactive prompt:

7 Precisely enter this statement at the interactive prompt
`help( 'keywords' )`

8 Hit **Return** to list all keywords of the Python language and remain at an interactive Shell Window prompt



Keywords have special meaning in a programming language – they cannot be used to name items in your code.

Saving Programs

The IDLE Shell Window, described on pages 14-15, is a great place to try out snippets of code, but it cannot save your code. Happily, IDLE also provides an Edit Window where you can create longer pieces of programming code that can be stored in a (.py) file on your computer. This means you can easily rerun the code without retyping all the instructions at the Shell Window **>>>** prompt, and this lets you edit your code to try new ideas. The procedure to create, save, and run your code looks like this:

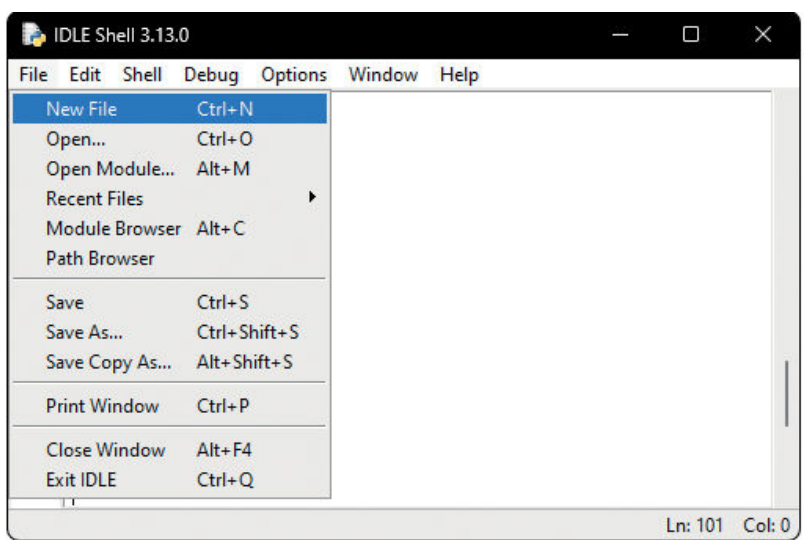
The procedure described here will be used to demonstrate the code examples given throughout this book.

- Open an Edit Window from the Shell Window by selecting **File**, **New File** from the Shell Window menu items – or by pressing the **Ctrl** + **N** shortcut keys.
- Type code into the Edit Window, then save it by selecting **File**, **Save** from the Edit Window menu items – or by pressing the **Ctrl** + **S** shortcut keys.
- Run saved code from the Edit Window by selecting **Run**, **Run Module** from the Edit Window menu items – or by pressing the **F5** shortcut key.

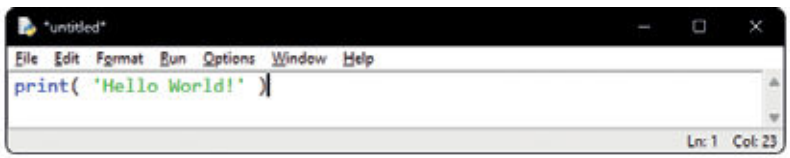
Output from your program code will appear in the Shell Window as the program runs, or a helpful error message will appear there if the interpreter discovers an error in your code.

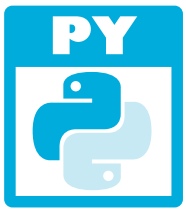
1 Open an IDLE Shell Window then select the **File**, **New File** menu item to open an IDLE Edit Window

Notice the **File**, **Open...** or **File**, **Open Module...** and **File**, **Recent Files** menu items that can be used to rerun program code previously saved.



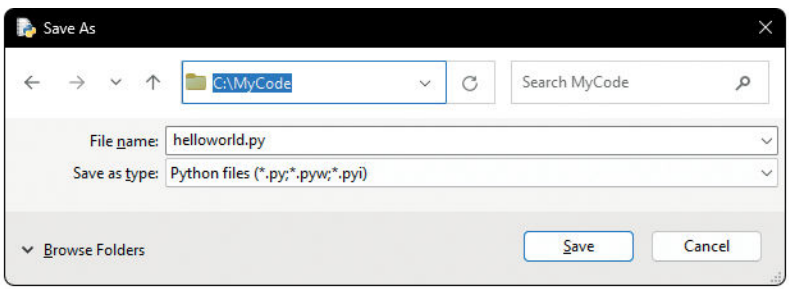
2 Now, in the IDLE Edit Window, precisely enter this code
`print( 'Hello World!' )`





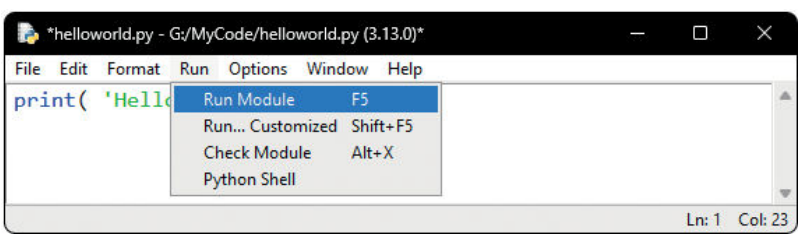
helloworld.py

3 Next, in the IDLE Edit Window, select the **File**, **Save** menu items, to open the "Save As" dialog, then save your program code as a file named **helloworld.py**

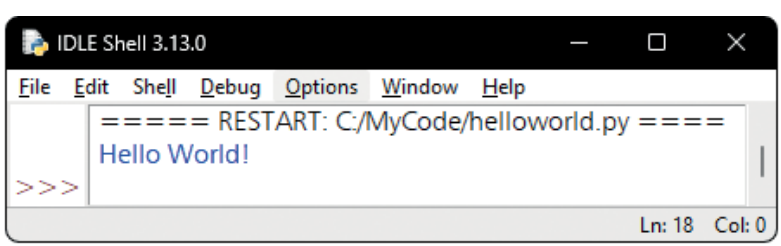


Your program code can be saved at any convenient location on your computer – here, it is saved in a directory created at **C:\MyCode** that will be used for all examples in this book.

4 Finally, in the IDLE Edit Window, select the **Run**, **Run Module** menu items, to run your program code and see the output appear in the Shell Window



Notice that the Shell Window restarts whenever it runs your program code afresh.



Storing Values

One essential feature of all computer programming languages is the ability to store data values by the program code. This ability is provided by a simple data structure called a "variable". A variable is a container in which an item of data can be stored, much like a real-life object can be stored in a box.

When creating a variable, you give it a name of your choice – subject to the naming conventions of the programming language – that acts like a label on a box. The data item stored within the variable can subsequently be retrieved using its given name – just as you can find a real-life object in a box by reading its label.

Data to be stored in a variable can be assigned in a Python program with the = assignment operator. For example, to store the numeric value eight in a variable named "a":

```
a = 8
```

The stored value can then be retrieved using the variable's name, so that the statement **print(a)** will output the stored value **8**. That variable can subsequently be assigned a different value, so its value can vary as the program proceeds – hence the term "variable".

In Python programming, a variable must be assigned an initial value ("initialized"), otherwise its value remains undefined and the interpreter will report a "not defined" error.

Multiple variables can be initialized with a common value in a single statement using a sequence of = assignments. For example, to initialize variables named "a", "b" and "c" each with a numeric value of 8, like this:

```
a = b = c = 8
```

Some programming languages, such as Java, demand you specify in its declaration what type of data a variable may contain. This reserves a specific amount of memory space and is known as "static typing". Python variables, on the other hand, have no such limitation and adjust the memory allocation to suit the various data values assigned to their variables ("dynamic typing"). This means they can store integer whole numbers, floating-point numbers (with a fractional part following the decimal point), text strings, or Boolean values of **True** or **False** as required.

With static typing, the variable type is checked at compile time, but with dynamic typing in Python, the variable type is checked at runtime.

1. Open an IDLE Edit Window, then enter code to create a variable named "var" to store a whole (integer) number

```
var = 8
```

2. Next, add a statement to display the stored integer value

```
print( var )
```

3. Assign a new floating-point (float) number to the variable, then add a statement to display the stored float value

```
var = 3.142
print( var )
```

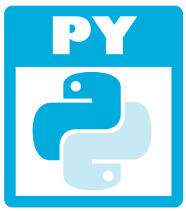
4. Now, assign a text string to the variable, then add a statement to display the stored string value

```
var = 'Coding for Beginners in easy steps'
print( var )
```

5. Finally, assign a logical truth value to the variable, then add a statement to display the stored Boolean value

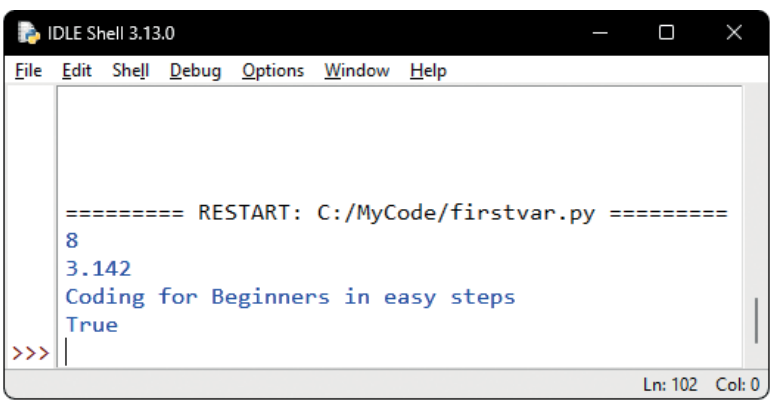
```
var = True
print( var )
```

6. Save the file (**File**, **Save**) then run the program (**Run**, **Run Module**) to see the stored values displayed in output



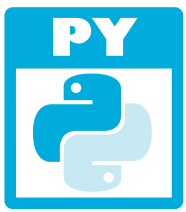
firstvar.py

Text string data must be enclosed within quote marks to denote the start and end of that particular string.



Adding Comments

When you begin to code longer programs it is useful to add comments at the start of each piece of code describing the purpose of that piece. This makes the code more easily understood by others, and by yourself when revisiting the code at a later date. In the Python programming language, everything on a single line after a **#** hash character is ignored by the interpreter. This means that a single-line comment can be inserted after a **#** character.

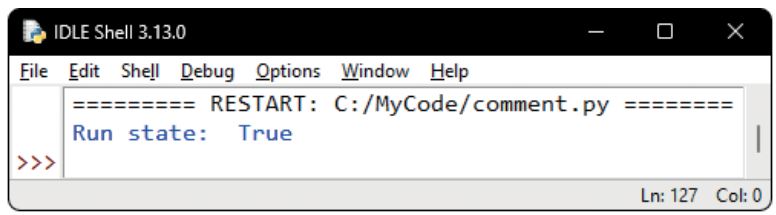


comment.py

1. Open an IDLE Edit Window, then enter commented code to initialize a variable and display its value

```
# Initialize program status
running = True
print( 'Run state: ' , running )
```

2. Save the file, then run the program to see the comment get ignored and the stored value displayed in the output



Code listed in the steps throughout this book also uses the default IDLE colors for consistency.

To readily identify aspects of your code, IDLE automatically colorizes your code, both in the Shell Window and the Edit Window, with the default colors listed in the table below:

Color	Description	Example
	Built-in function names	`print()`
	Strings in quote marks	`'Hello World!'`
	Symbols, numbers and names	`8 + 4`
	Shell Window output	`Hello World!`
	Coder-created function names in the function declaration	`my_function()`
	Keywords	`True`
	Edit Window comments and Shell Window errors	`# My comments` `ZeroDivisionError`

Naming Rules

Keywords		
False	None	True
and	as	assert
async	await	break
class	continue	def
del	elif	else
except	finally	for
from	global	if
import	in	is
lambda	nonlocal	not
or	pass	raise
return	try	while
with	yield	

It is good programming practice to choose meaningful names that reflect the nature of a variable's content.

Variable containers that you create in your code to store data within a program can be given any name of your choosing, providing you do not use any of the programming language keywords – such as the Python keywords in the table above – and the name adheres to the naming rules listed in the table below:

Naming Rule	Example
CANNOT be a keyword	**True**
CANNOT contain arithmetic operators	**a+b*c**
CANNOT contain symbols	**%$#@!**
CANNOT contain any spaces	**no spaces**
CANNOT start with a number	**2bad**
CAN contain numbers elsewhere	**good1**
CAN contain letters of mixed case	**UPdown**
CAN contain underscores	**is_ok**

Variable names are case-sensitive in Python – so variables named "VAR", "Var", and "var" would be treated as three separate variables.

Summary

- A computer program is a set of instructions written by a coder that enable computers to become useful.
- The electronic components of a computer are its hardware, whereas program instructions are its software.
- Computers understand low-level machine code.
- High-level programming languages in human-readable form get automatically translated into low-level machine code.
- Programming languages possess data structures to store information and control structures to determine progress.
- The Python programming language has simple syntax, requires no manual compilation, and includes a library of functions.
- Python's development environment is called IDLE (Integrated DeveLopment Environment).
- IDLE provides a Shell Window containing an interactive prompt for testing, and an Edit Window for coding programs.
- The IDLE help utility is accessed by entering a **help()** statement at a Shell Window prompt.
- After typing program code into an IDLE Edit Window it must first be saved as a file before the program can be run.
- Output from a program run from the Edit Window appears in the Shell Window, or a helpful error message appears.
- A variable is a named container that allows a single item of data to be stored for use by a program.
- Data stored in a variable can be retrieved using that variable's name, and may be replaced by assigning a new value.
- Variables in Python programming can store any type of data.
- Comments can usefully be added to program code after beginning the comment with a **#** hash character.
- Variable names must not use any of the programming language keywords, and must adhere to its naming rules.