Contents

Preface

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. All the examples demonstrate coding features using the popular Python programming language and the book's screenshots illustrate the actual results produced by executing the listed code.

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' , 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, like this:

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. Browse to **www.ineasysteps.com** then navigate to Free Resources and choose the Downloads section

2. Find Coding for Beginners in easy steps 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

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 can only count from zero to one, and 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 of "batteries included".

Setting up

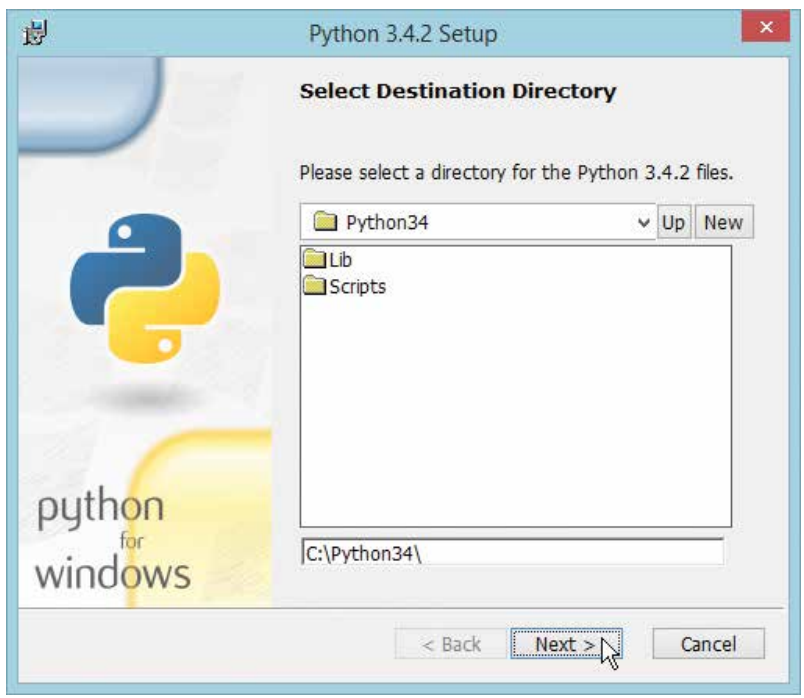
Before you can begin coding programs in the Python language you need to set up a programming environment on your computer by installing the Python interpreter and the standard library of tested code modules that comes along with it. This is available online as a free download from the Python Software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

Installers for Mac OS X 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 Downloads button to grab the latest version for your system – in this case it's "Python 3.4.2"

2. When the download completes run the installer and choose whether to install for all users or just yourself, then click the Next button to proceed

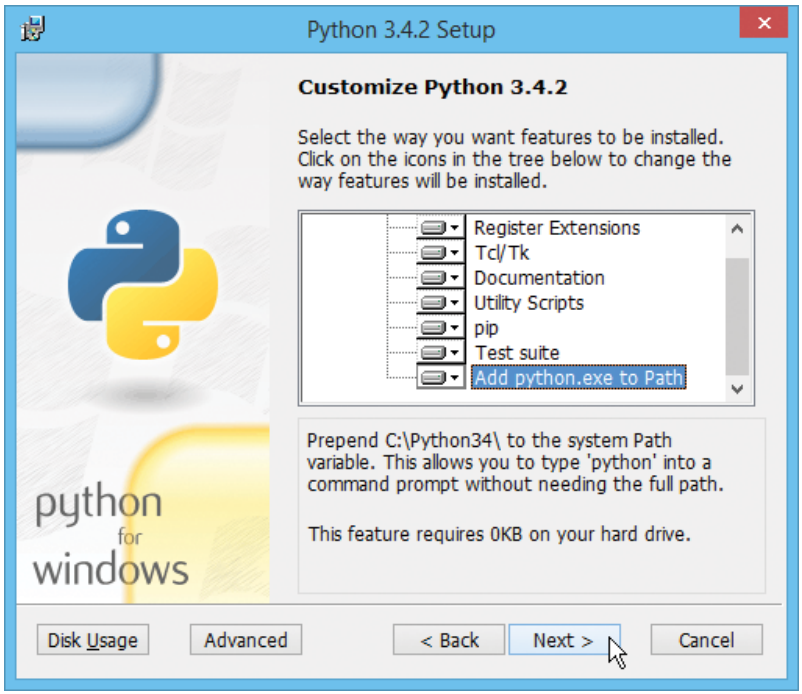
3. Now, accept the suggested default installation location, which will be a directory on your root **C:** drive named "Python" and version number – in this example it's a directory at **C:\Python34** for Python version 3.4.2



Hot tip

Do accept the suggested destination directory – such as **C:\Python34** that is suggested here.

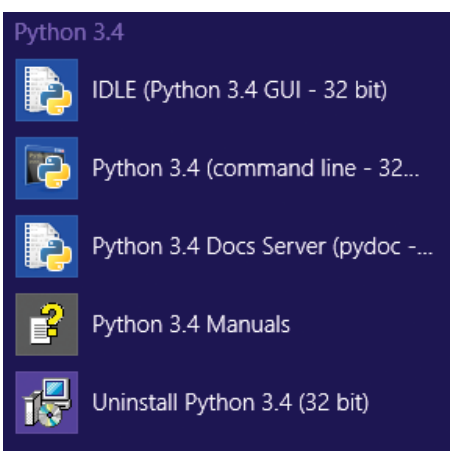
4 Click the Next button to proceed then be sure to select the feature to "Add python.exe to Path"



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

5 Click on Next to begin copying files onto your computer then click the Finish button to complete the installation

Upon completion the Python group is added to your Start/ Apps menu. Most important of this group is the IDLE item that launches the Python integrated development environment.



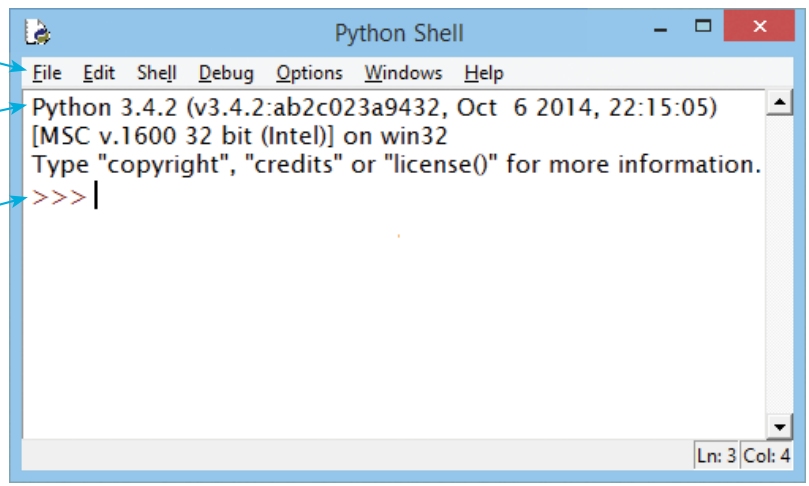
You will use the IDLE launcher often so right-click on its icon and choose "Pin to taskbar" to make it readily available from the Windows Desktop.

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.



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.

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 zero, the interpreter will print out an error message helpfully describing the nature of the error.

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

```
Python Shell
File  Edit  Shell  Debug  Options  Windows  Help
>>> print( 'Hello World!' )
Hello World!
>>>
```

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

```
Python Shell
File  Edit  Shell  Debug  Options  Windows  Help
>>> 8 + 4
12
>>>
```

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

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

```
Python Shell
File  Edit  Shell  Debug  Options  Windows  Help
>>> 8 / 0
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    8 / 0
ZeroDivisionError: division by zero
>>>
```

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

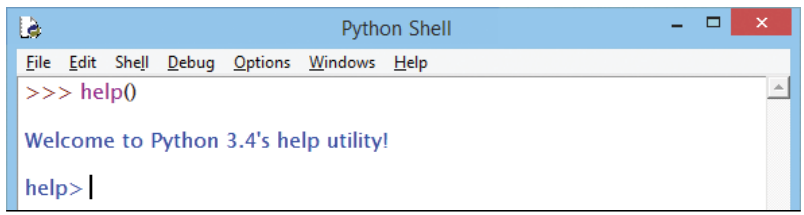
Getting help

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.

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

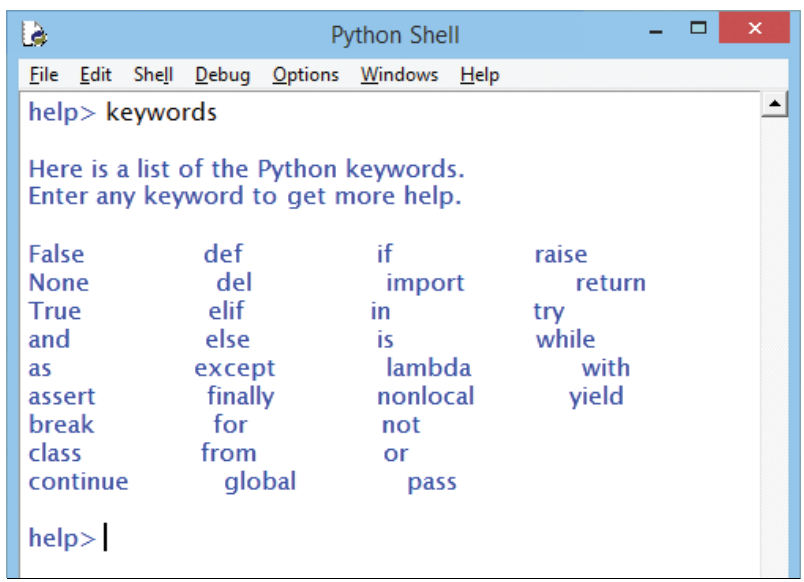
1. Open an IDLE Shell Window then precisely enter this statement at the interactive prompt
   help( )

2. Next, hit the Return key to enter Help mode



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 and return to an interactive Shell Window prompt

```
Python Shell
File  Edit  Shell  Debug  Options  Windows  Help
help> quit

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)".  Executing "help('string')"
has the same effect as typing a particular string at the help> 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

```
Python Shell
File  Edit  Shell  Debug  Options  Windows  Help
>>> help('keywords')

Here is a list of the Python keywords.
Enter any keyword to get more help.

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

>>>
```

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 the previous page, is a great place to try out snippets of code, but 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 re-run the code without re-typing 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:

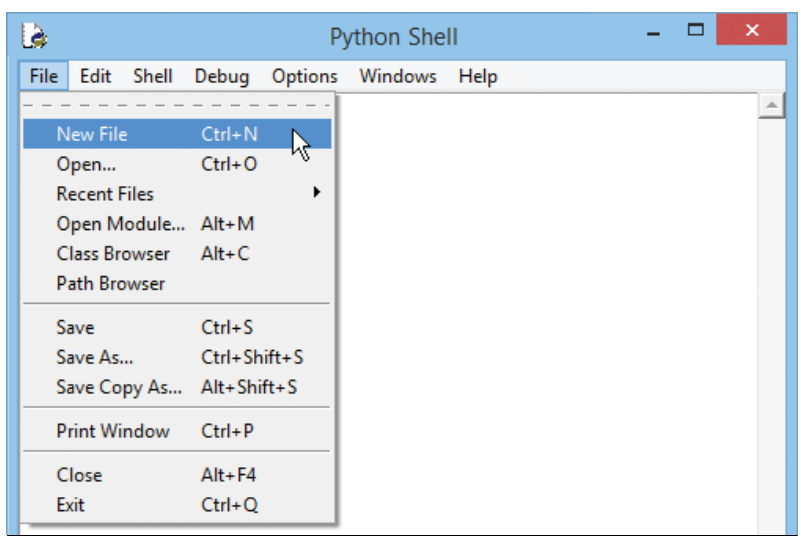
- Open an Edit Window from the Shell Window by selecting File, New File from the Shell Window menu items – or by pressing the Ctrl + N shortcut keys
- Type code into the Edit Window then save it by selecting File, Save from the Edit Window menu items – or by pressing the Ctrl + S shortcut keys
- Run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

Hot tip

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

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.

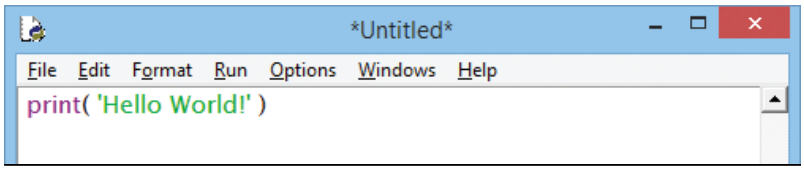
1 Open an IDLE Shell Window then select the File, New File menu item to open an IDLE Edit Window



Hot tip

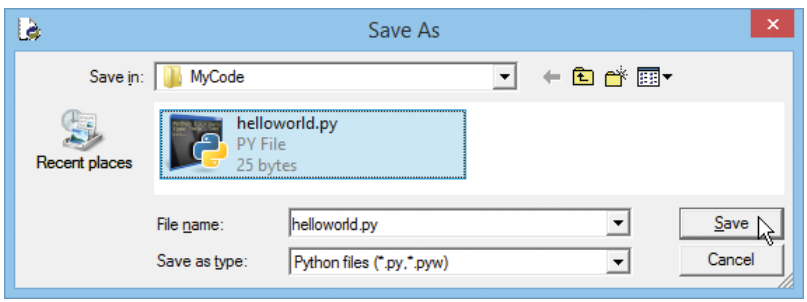
Notice the File, Open and File, Recent Files menu items that can be used to re-run program code previously saved.

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



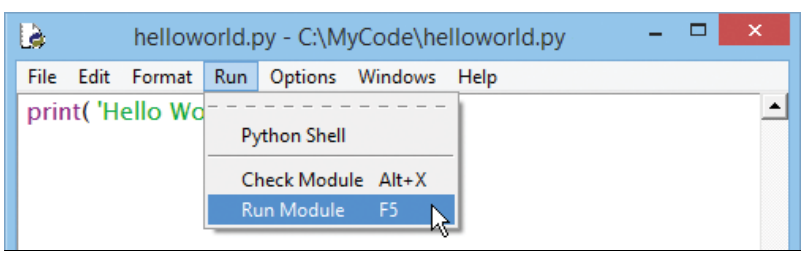
helloworld.py

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**

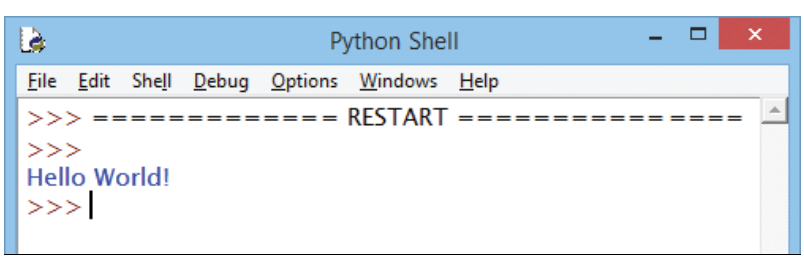


Your program code can be saved at any convenient location on your computer – here it is saved in a directory created at **C:\MyCode** that will be used for all examples in this book.

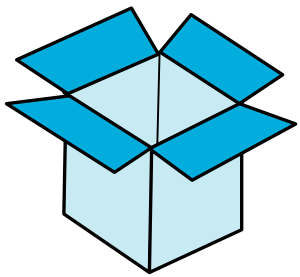
4 Finally, in the IDLE Edit Window, select the Run, Run Module menu items, to run your program code and see the output appear in the Shell Window



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 in 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 is assigned in a Python program declaration statement 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") in the statement that declares it in a program – otherwise 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 eight 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, text strings, or Boolean values of **True** or **False** as required.

Programming languages that require variable types to be specified are alternatively known as "strongly typed", whereas those that do not are alternatively known as "loosely typed".

1. Open an IDLE Edit Window then enter code to create a variable named "var" to store a whole number integer
```
var = 8
```

2. Next, add a statement to display the stored integer value
```
print( var )
```

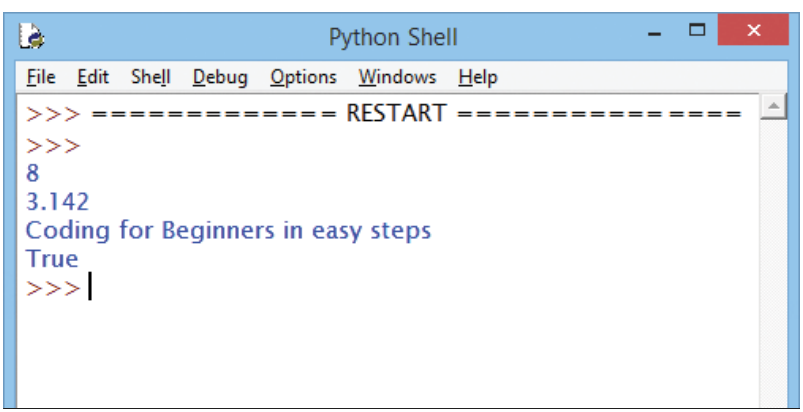
3. Assign a new floating-point number to the variable then add a statement to display the stored float value
```
var = 3.142
print( var )
```

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

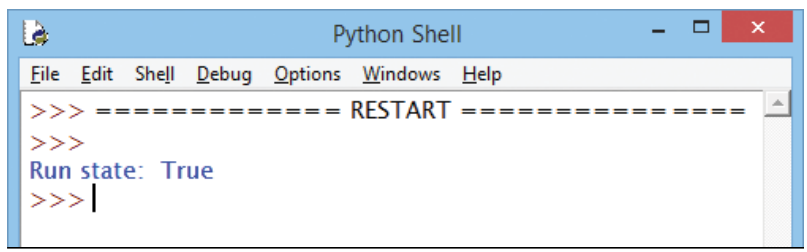
When you begin to code longer programs it is useful to add comments at the start of each piece of code describing the purpose of that piece. This makes the code more easily understood by others, and by yourself when revisiting the code at a later date. In the Python programming language everything on 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 status

```
# 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 output



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:

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

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

Naming rules

Keywords:		
False	**None**	**True**
and	**as**	**assert**
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 the 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 contain any keywords	**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 compilation, and includes a library of functions
- Python's development environment is called IDLE
- 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 there
- A variable data structure 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
- Comment lines can usefully be added to program code after beginning the line with a **#** hash character
- Variable names must not use any of the programming language keywords and must adhere to its naming rules