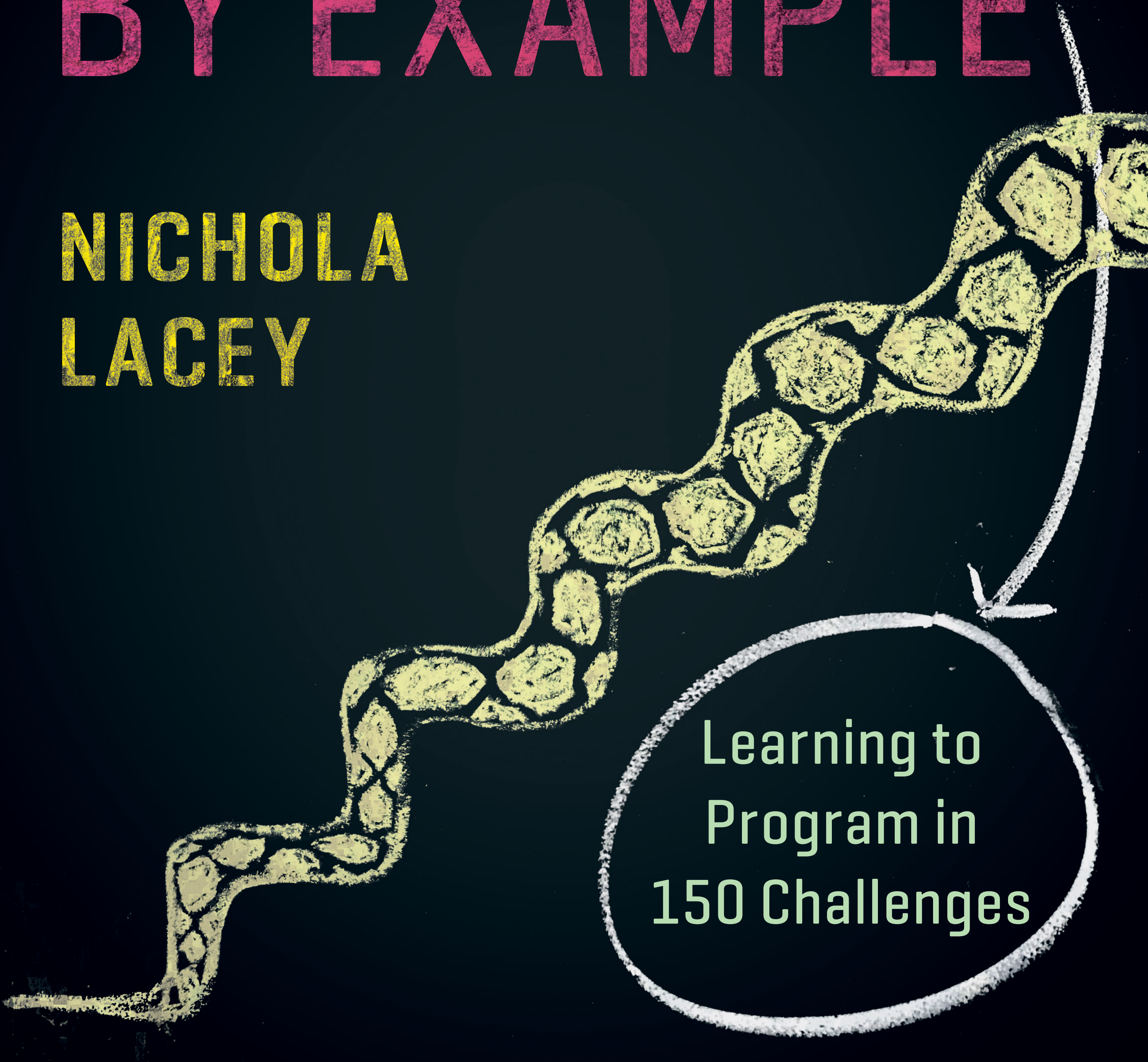
PYTHON BY EXAMPLE

NICHOLA LACEY



Learning to Program in 150 Challenges

CAMBRIDGE

more information - www.cambridge.org/9781108716833

Python by Example

Learning to Program in 150 Challenges

Python is today's fastest growing programming language. This engaging and refreshingly different guide breaks down the skills into clear step-by-step chunks and explains the theory using brief easy-to-understand language. Rather than bamboozling readers with pages of mind-numbing technical jargon, this book includes 150 practical challenges, putting the power in the reader's hands. Through creating programs to solve these challenges the reader will quickly progress from mastering the basics to confidently using subroutines, a graphical user interface, and linking to external text, csv and SQL files. This book is perfect for anyone who wants to learn how to program with Python. In particular, students starting out in computer science and teachers who want to improve their confidence in Python will find here a set of ready-made challenges for classroom use.

NICHOLA LACEY is Director of Nichola Wilkin Ltd. She is a trusted source for teaching resources, having sold thousands of resources to schools around the world. As one of the most popular authors on TES, Nichola enjoys an extremely high review rating with hundreds of thousands of downloads. She was a programmer before moving into corporate training and then retraining as a teacher, and she gained a unique skill set of programming and practical classroom experience after being promoted to head of computer science in a private boys' school.

PYTHON BY EXAMPLE

Learning to Program in 150 Challenges

NICHOLA LACEY
Nichola Wilkin Ltd

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE
UNIVERSITY PRESS

University Printing House, Cambridge CB2 8BS, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre, New Delhi – 110025, India

79 Anson Road, #06-04/06, Singapore 079906

www.cambridge.org
Information on this title: www.cambridge.org/9781108716833
DOI: 10.1017/9781108591942

First published 2019

Printed in Singapore by Markono Print Media Pte Ltd

A catalogue record for this publication is available from the British Library.

ISBN 978-1-108-71683-3 Paperback

Contents

Image Credits

Animal Drawings:
Pages 1, 9, 11, 31, 35, 37, 41, 45, 92, 125, 127, 138, 150, 165: HelenField/Shutterstock.com
Pages 16, 20, 94 and141(bottom): Victoria Novak/Shutterstock.com
Pages 27 and 157: Dimonika/Shutterstock.com
Pages 46, 58, 73, 93, 95, 103, 128, 135, 147, 169: mart/Shutterstock.com
Pages 59, 68, 151, 154: lynea/Shutterstock.com
Page 80: MoreVector/Shutterstock.com
All other animal drawings: Olga_Angelloz/Shutterstock.com

Other decorative icons:
MG Drachal/Shutterstock.com
Mila Petkova/Shutterstock.com
Nikolaeva/Shutterstock.com
Tiwat K/Shutterstock.com

Introduction

If you have ever picked up a programming manual and felt your forehead go clammy and your eyes cross as you attempt to make sense of the long-winded explanations, this is the guide for you.

I have been in your position, attempting to learn how to program and having to rely on the traditional style of guides. I know from painful experience how quickly I glaze over and my brain solidifies; after only a few pages the tedium leaves me blindly reading words without any real notion of what they mean any more. Inevitably I give up and the whole process makes me feel like a limp failure, gasping for breath after I surface from drowning in technical jargon.

I hated having to read through pointless drivel and then be presented with a short program telling me exactly what to type in and then spend the next 20 pages reading about what I have just done and the 101 ways I could run it. I hated having no control over trying things out for myself and I hated the way these guides would only contain one or two challenges at the end of a chapter of theory.

I knew there had to be a better way, and thankfully there is. I wrote it and you are presently reading it, so aren't you lucky? This guide is refreshingly different and helps you learn how to program with Python by using practical examples rather than self-important explanations.

Many programmers learn through experimentation, looking at others' code and working out what method is best for a given situation. This book is a hands-on approach to learning programming. After minimal reading you are set a number of challenges to create the programs. You can explore and experiment with the programming language and look at the example solutions to learn how to think like a programmer. There are no chapters entitled "the architecture of a computer", "the theory of programming" or any other gobbledy-gook other authors like to waste time with. I don't want to baffle you with theory or blind you with overbearing explanations that suck out your enthusiasm for learning to program.

Hopefully, you want to get stuck into creating programs, solving problems and enjoying the sense of accomplishment that you get as you proudly look over your lines of code, knowing that you created something that works. That is great, your eagerness is to be applauded and I salute those who are reading this while already sitting at their computers, fingers poised and ready to get going. If that is the case, that you already have Python open on your screen and are itching to get going, then away you go and I'll see you in the first chapter called "The Basics" on page 11.

For everyone who is still with us and is feeling a little more timid, there are just a few more things to tell you about before you take the plunge.

How to Use This Book

This book builds from very simple programs to more complex ones. If you are new to programming or new to Python, start with "The Basics" and work through the chapters in order.

If you are familiar with Python programming and feel confident with the basics, the theory and logic surrounding programming, then you can just dip in and out of the book to get help on the specifics you need.

The book is split into two sections:

Part I

In Part I, each chapter takes you through some basic programming rules and challenges for you to complete and includes:

- a **simple explanation** giving you pointers, which is useful if you are new to programming in Python;
- **examples of code** with a short explanation, which you can use as a basis to solve the challenges;
- a **list of challenges** for you to work through that get harder as you move through them. Each challenge should only take between a couple of minutes and 20 minutes to solve; however, some of the more complex challenges near the end of Part I will take longer as you build up the techniques you will be using. Don't panic if you take longer than this, as long as you solve the problems without *too* much copying from the suggested solution, you are doing fine;
- code containing a **possible solution** for each challenge; there is often more than one answer available, but we include just a single program as a possible solution that you can refer to if you get stuck on a particular aspect of the code.

Part II

In Part II, you are given some larger challenges which utilize the programming skills you learnt in Part I and allow you to consolidate and reinforce the techniques you have been practising. In this section, you are not given the help and example code that is given in Part I and it will take longer to solve each challenge. After each challenge, you are given one possible answer that you may find useful if you are stuck. However, you may have found another solution that works just as well.

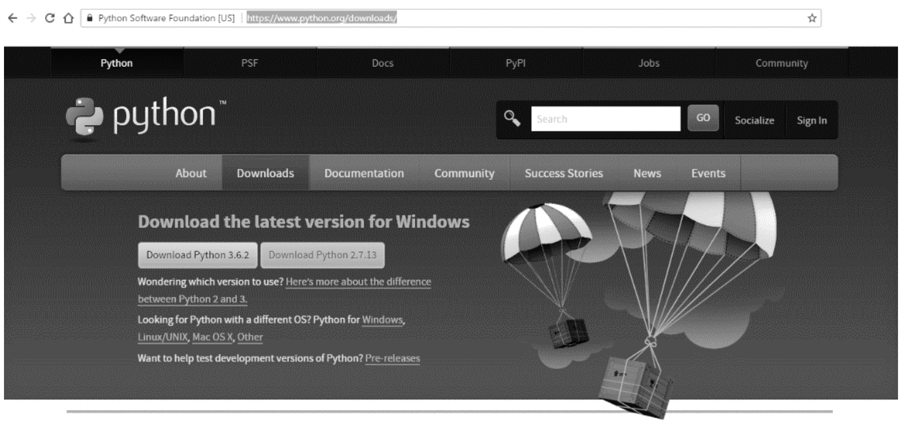
Who Is This Book For?

This book is suitable for anyone who wants to learn how to program with Python. It is an essential tool for teachers and students in Key Stage 3 or those studying computer science who need help and ready-made examples to practise programming techniques and build confidence. It can also be used to help with a computer science programming project resource bank, to help pupils needing additional support or just a quick reminder of the syntax when creating programs.

Downloading Python

You can download Python for free from the official Python website:

www.python.org/downloads/



Download the latest version for Windows

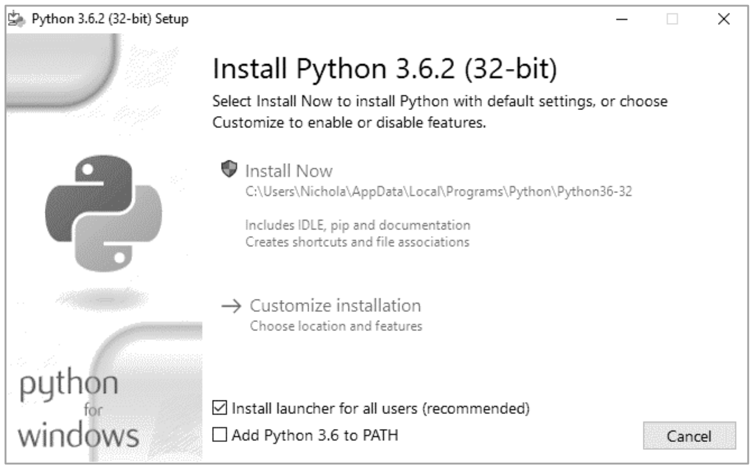
Wondering which version to use? Here's more about the difference between Python 2 and 3.

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases

Click on the latest version (in the example above, click on the Download Python 3.6.2 button) to start the installation.

The program will download an executable (.exe) file. When you run this program, you will see an install window like the one shown below.



Install Python 3.6.2 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Nichola\AppData\Local\Programs\Python\Python36-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

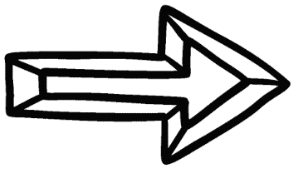
→ Customize installation
Choose location and features

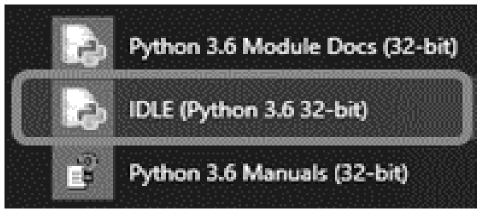
☑ Install launcher for all users (recommended)
☐ Add Python 3.6 to PATH

Click the Install Now option and the program will start installing Python onto your system.

Running Python

To start Python on a Windows system, click on the **Windows** icon or **Start** menu and select the **IDLE (Python version number)** option as highlighted below.





Some Tips

File Location

On a Windows system, the Python folder is usually found in the C:\ drive and will be named Python36 (or similar) and the files will automatically be saved in the same location, unless you save them specifically in another location.

Using Comments

Comments are a very useful tool for programmers. They serve two purposes:

- adding an explanation of how the program works;
- stopping parts of the program from working temporarily so you can run and test other parts of the program.

The first, and original, purpose of explaining how a program works is so other programmers can make sense of your programs in case your code needs to be altered and updated in the future and to remind you about why you wrote particular lines of code.

```
print("This is a simple program")
print() #Outputs a blank line to help with layout
name = input("Please input your name: ")#Asks for an input
print("Hello", name) #Joins "Hello" and their name together
```

In this example, comments have been added at the end of the last three lines. They are shown in red and start with the # symbol.

In reality, you would not add comments on lines which contain obvious code as it would clutter the screen; you would only add comments where necessary.

As Python knows to ignore anything after a # symbol, programmers soon started to use # at the start of lines of their code to block out sections they do not want to run so they can focus on and test others.

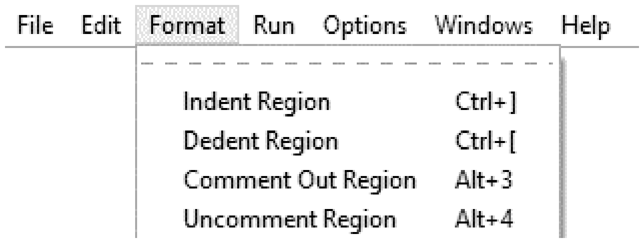
```
#print("This is a simple program")
print()
name = input("Please input your name: ")
print("Hello", name)
```

In this example, the # has been added to the first line of the program to temporarily stop it from running. To bring it back into the running order simply delete the # and the code will be reactivated.

In this guide, we have not included any comments to the programs so you have to read the code to make sense of it. That way you will really learn how to code! If you are creating programs as part of your coursework you should add comments to explain your programming to the examiner.

Formatting Python

In most versions of Python IDLE it is possible to quickly add comments and indent code using the menus. This way, if you need to block out entire areas using a comment you simply highlight the lines and then select the **Format** menu and select **Comment Out Region**. Similarly, if you need to indent a region (we will look at the reason for indenting code later) then you can also easily do this with the menu.



Okay, that is all the “housekeeping” out of the way. No more procrastinating; take a deep breath and away we go…

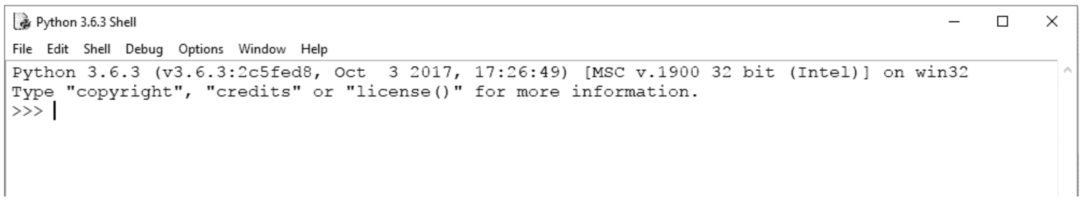
Part 1

Learning Python

The Basics

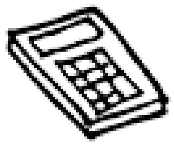
Explanation

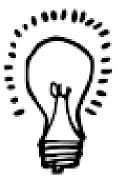
This is the **shell** window and is the first screen you see when you launch Python.



```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

It is possible to write Python code straight into the shell, but as soon as you hit [Return] at the end of a line, it will run that line of code. This may be suitable for using Python as a quick calculator; for instance, you can type in **3*5** at the prompt and Python will show the answer **15** on the next line; however, this style of inputting is not useful for more complex programs.

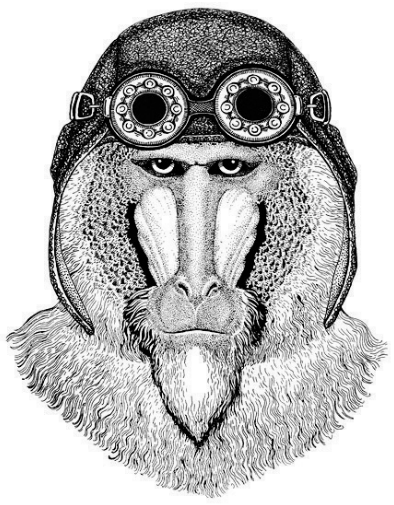




It is much better to start a new window, create all the code in the new window, save your code and run it.

To create a new window in which to write your code, select **File** and **New**. Once you enter your code in this new window you can save it and run it all in one go. This will then run the code in the shell window.

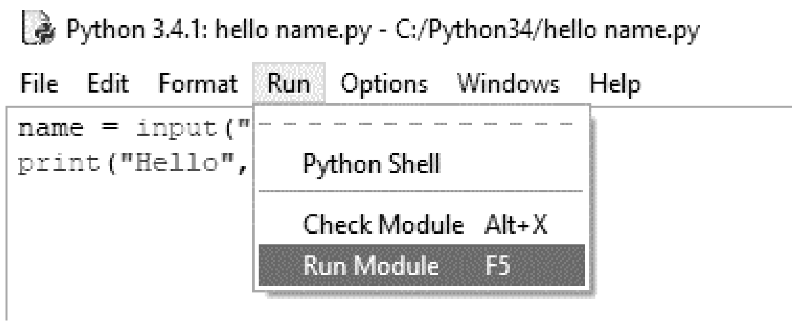
Alternatively, Python programs can be written using any text editor and must be saved with the file name extension .py in order to work. These programs can then be run from the command prompt by typing in the full directory root and file name.



Running Your Program

Every time you run the code your program will need to be saved afresh in case there have been any changes to it.

In this version of Python, you can run the program by selecting the **`Run`** menu and selecting **`Run Module`**. Alternatively, you can press the **`[F5]`** key. If this is the first time the program is saved, Python will prompt you to name and save the file before it will allow the program to run.


Python 3.4.1: hello name.py - C:/Python34/hello name.py

```
name = input("
print("Hello",
```

Important Things to Note When Writing Your Programs

Python is case sensitive so it is important that you use the correct case, otherwise your code <u>will not work</u>.

Text values need to appear in speech marks (") but numbers do not.

When naming **`variables`** (i.e. values that you want to store data in) you cannot use any dedicated words such as print, input, etc. otherwise your code will not work.

When saving your files **do not save them with any dedicated words** that Python already uses, such as print, input, etc. If you do this it will not run and you will need to rename the file before it works.

To edit a program you have saved and closed, right-click on the file and select **`Edit with IDLE`**. If you just double-click on the file it will only try to run it and you will not be able to edit it.