Undergraduate Topics in Computer Science

'Undergraduate Topics in Computer Science' (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems, many of which include fully worked solutions.

The UTiCS concept relies on high-quality, concise books in softback format, and generally a maximum of 275-300 pages. For undergraduate textbooks that are likely to be longer, more expository, Springer continues to offer the highly regarded Texts in Computer Science series, to which we refer potential authors.

More information about this series at http://www.springer.com/series/7592

John Hunt

A Beginners Guide to Python 3 Programming

Springer

John Hunt
Midmarsh Technology Ltd
Chippenham, Wiltshire, UK

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-030-20289-7 ISBN 978-3-030-20290-3 (eBook)
https://doi.org/10.1007/978-3-030-20290-3

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

This book was written for, and is dedicated to, my daughter Phoebe and son Adam; I could not be prouder of either of you.

Preface

There is currently huge interest in the Python programming language. This is driven by several factors; its use in schools with the Raspberry Pi platform, its ability to be used for DevOps scripts, its use in data science and machine learning and of course the language itself.

There are many books on Python, however, most assume previous programming experience or are focussed on particular aspects of Python use such as data science or machine learning or have a scientific flavor.

The aim of this book is to introduce Python to those with little or very little programming knowledge, and then to take them through to become an experienced Python developer.

As such the earlier parts of the book introduce fundamental concepts such as what a *variable* is and how a *for loop* works. In contrast, the later chapters introduce advanced concepts such as functional programming, object orientation, and exception handling.

In between a wide range of topics are introduced and discussed from a Python point of view including functions, recursion, operators, Python properties, modules and packages, protocols and monkey patching, etc.

After the core elements of Python are established, each new subject area is introduced by way of an introductory chapter presenting the topic in general, providing background on that subject, why it is of importance, etc. These introductions cover Structured Analysis, functional programming, and object orientation.

Some of the key aspects of this book are:

1. It assumes very little knowledge or experience of Python or programming.
2. It provides a basic introduction to Python as well as advanced topics such as generators and coroutines.
3. This book provides extensive coverage of object orientation and the features in Python 3 supporting classes, inheritance, and protocols.
4. Pythons' support for functional programming is also presented.

5. Following on from introducing the basic ideas behind functional programming, the book presents how advanced functional concepts such as closures, currying, and higher-order functions work in Python.
6. The book includes exercises at the end of most chapters with online solutions.
7. There are several case studies spread through the book that broaden understanding of preceding topics.
8. All code examples (and exercise solutions) are provided online in a GitHub repository.

Chapter Organization

Each chapter has a brief introduction, the main body of the chapter, followed by a list of (typically) online references that can be used for further reading.

Following this, there is typically an *Exercises* section that lists one or more exercises that build on the skills you will have learned in that chapter.

Sample solutions to the exercises are available in a GitHub online repository that supports this book.

What You Need

You can of course just read this book; however, following the examples in this book will ensure that you get as much as possible out of the content.

For this, you will need a computer.

Python is a cross-platform programming language and as such you can use Python on a Windows PC, a Linux box or an Apple Mac, etc. So you are not tied to a particular type of operating system; you can use whatever you have available.

However, you will need to install some software on that computer. At a minimum, you will need Python.

This book focusses on Python 3, so you will need that. Some guidance on this is provided in Chap. 2 on setting up your environment.

You will also need some form of editor in which to write your programs. There are numerous generic programming editors available for different operating systems with Vim on Linux, Notepad++ on Windows and Sublime Text on Windows, and Macs being popular choices.

However, using an integrated development environment (IDE) editor such as PyCharm will make writing and running your programs much easier.

Using an IDE

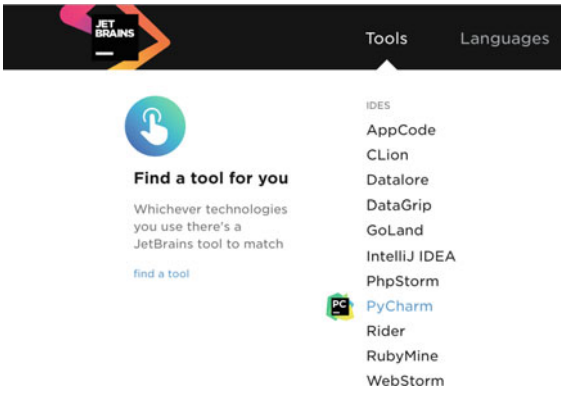
The IDE I prefer for Python is PyCharm, although it is not the only IDE for Python by any means, but it is a very widely used one.

Other IDEs available for Python include:

- Rodeo which is a lightweight, open source, IDE see https://rodeo.yhat.com.
- Jupyter Notebook which is a web-based IDE and is particularly good for data scientists https://jupyter.org.
- Visual Studio Code. This is a very good free editor from Microsoft that has really useful features https://code.visualstudio.com.
- Sublime Text is more of a text editor that color codes Python; however, for a simple project it may be all you need https://www.sublimetext.com.

Downloading the PyCharm IDE

PyCharm is provided by JetBrains who make tools for a variety of different languages. The PyCharm IDE can be downloaded from their site—see https://www.jetbrains.com/. Look for the menu heading 'Tools' and select that. You will see a long list of tools, which should include PyCharm.



Select this option. The resulting page has a lot of information on it; however, you only need to select the 'DOWNLOAD NOW'. Make sure that you select the operating system you use (there are options for Windows, Mac OS, and Linux).

There are then two download options available: Professional and Community. The Professional version is the charged for option, while the Community version is

free. For most of the work I do in Python, the Community version is more than adequate and it is therefore the version you can download and install (note with the Professional version you do get a free trial but will need to either pay for the full version at the end of the trial or reinstall the Community version at that point).

Assuming you selected the Community edition the installer will now download, and you will be prompted to run it. Note you can ignore the request to subscribe if you want.

You can now run the installer and follow the instructions provided.

Setting Up the IDE

You need to first start the PyCharm IDE. Once started, the first dialog shown to you asks if you want to import any settings you may have had for another version of PyCharm. At this point, select 'Do not import settings'.

Step through the next set of dialogs selecting the look and feel (I like the light version of the IDE), whether you want to share data with JetBrains, etc. Once you have completed this, click the 'Start PyCharm' option.

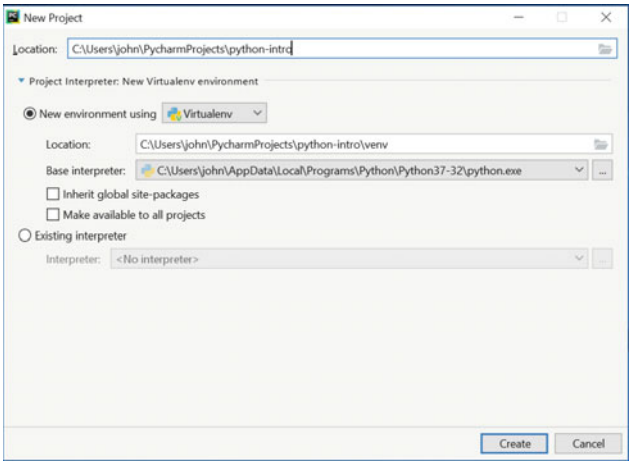
You should now be presented with the landing screen for PyCharm:

We will now create a project for you to work in. A project in PyCharm is where you write your programs and how you config what version of Python you are using and any libraries that you might need (such as graphics libraries, etc.).

Click on the 'Create New Project' option in the landing dialog.

You will now be asked where you want to create this new project. Again you can use the default location, but you will need to give it a name, I will call my project `python-intro`.

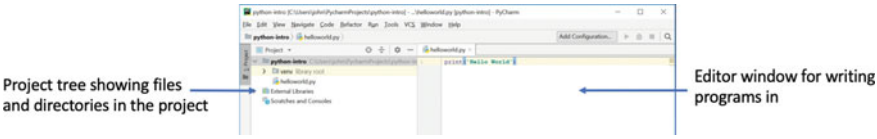
It is also worth making sure that the Python interpreter you installed has been picked up by the IDE. You can do this by opening the 'Project Interpreter: New Virtualenv environment' option and making sure that the base interpreter field is populated appropriately. This is shown below:



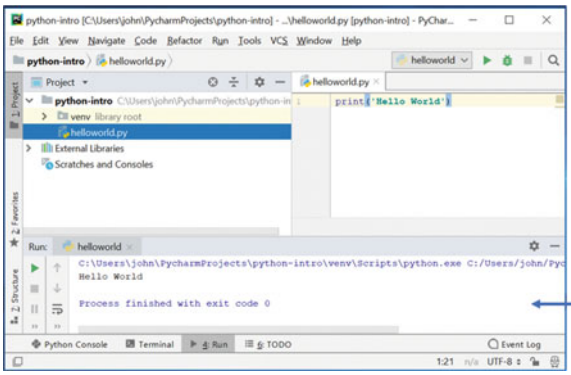
If all is OK, then select 'Create'; if the base interpreter is not specified or is incorrect, then click on the '…' button to the right of the field and browse to the appropriate location.

On opening the PyCharm project, you should see a Welcome message; click 'Close' and the project will be set up for you.

When you open the project, you will be shown at least two views. The left-hand view is the 'Project' view which shows you all the directories and files in your project. The right-hand area is where the editor is presented that allows you to type in your program. For example,



The third area that may be shown represents the output from your program. If this is the first time you have opened the project, then it may not yet be visible. However, if you run a program, it will be shown at the bottom of the IDE. For example:



Conventions

Throughout this book, you will find a number of conventions used for text styles. These text styles distinguish different kinds of information.

Code words, variable sand Python values, used within the main body of the text, are shown using a `Courier` font. For example:

> This program creates a top-level window (the `wx.Frame`) and gives it a title. It also creates a label (a `wx.StaticText` object) to be displayed within the frame.

In the above paragraph, wx.Frame and wx.StaticText are classes available in a Python graphical user interface library.

A block of Python code is set out as shown here:

```
num = int(input('Enter another number: '))
if num > 0:
    print(num,'is positive')
    print(num,'squared is ', num * num)
print('Bye')
```

Note that keywords and strings are shown in bold font.

Any command line or user input is shown in italics:

```
> python hello.py
```

Or

```
Hello, world
Enter your name:John
Hello John
```

Example Code and Sample Solutions

The examples used in this book (along with sample solutions for the exercises at the end of most chapters) are available in a GitHub repository. GitHub provides both a server environment hosting Git and a web based interface to that environment.

Git is a version control system typically used to manage source code files (such as those used to create systems in programming languages such as Python but also Java, C#, C++, Scala, etc). Systems such as Git are very useful for collaborative development as they allow multiple people to work on an implementation and to merge their work together. They also provide a useful historical view of the code (which also allows developers to roll back changes if modifications prove to be unsuitable).

If you already have Git installed on your computer, then you can clone (obtain a copy of) the repository locally using:

```
git clone https://github.com/johnehunt/beginnerspython3.git
```

If you do not have Git, then you can obtain a zip file of the examples using

```
https://github.com/johnehunt/beginnerspython3/archive/master.zip
```

You can of course install Git yourself if you wish. To do this, see https://git-scm.com/downloads. Versions of the Git client for Mac OS, Windows, and Linux/Unix are available here.

However, many IDEs such as PyCharm come with Git support and so offer another approach to obtaining a Git repository.

For more information on Git, see http://git-scm.com/doc. This Git guide provides a very good primer and is highly recommended.

Bath, UK John Hunt

Contents