Programming

Adventure Games in Python

Geert-Jan Kruijff

Story Byte Studios

Contents

1. Introduction

1.1. Creating Adventures

This book is about creating adventures. Adventures that can take place anywhere where your creativity takes you. Adventures you can play by yourself, or with others.

Yes, these adventures are computer games. And in this book that means two things. For one, this book teaches you how to write *computer programs*. We start very simple, with a text adventure game – and gradually build this out to complex massively multi-player online role-playing games. (Note the 'gradually' here – it will take some effort to get there!)

And then there is the other aspect, namely *game design*. Creating a game really starts with designing a game. What is the story? What should the player try to achieve? What can the player do? What are the obstacles that a player needs to overcome? These are several key ingredients of what makes up your adventure – which you then implement as a computer game.

1.2. For Whom Is This Book?

This book is for anyone with an interest in learning how to program, and to create adventure games. You do not need to have any experience in either programming or game design – we will learn things as we go.

Naturally, you may be interested in one aspect more than in the other – and that's perfectly fine. For those who are more interested in the programming side: You will learn about programming in Python, ranging from basic programming concepts like variables and control structures, to complex concepts like internet protocols, client/server architectures, and graphical user interfaces. And all of that you will then be able to apply to building a game.

For those who are more interested in creating adventure games: That is great too! All the code in this book is available online (as we will see below), so even if you do not want to dive in each and every aspect you can still run the code, and explore creating ever-more fascinating adventure games. It's all there for you.

1.3. Getting Things Set Up

In this section we explain how to get three things set up for your first steps in programming: the Python programming language, a graphical user interface to write Python

programs in, and *versioning system* called `git`. You will learn how to use these in Part 1, Getting Started.

1.3.1. Mac

To download the resources for the Python programming language,

1. Go to `https://python.org`. This is the official website for all matters Python.
2. In the menu bar on the main website, click on the "Downloads" button.
3. A new webpage will appear, stating "Download the latest version for Mac OS.X"
4. On that page you will see a (yellow) button saying "Download Python" plus a version number – at the time of writing, the latest version is 3.7.0.
5. Click on that button and follow the instructions to install Python.

We will use the Atom "integrated development environment" or *IDE* to write and test our programs. To download the resources,

1. Go to `https://atom.io`. This is the official website for Atom. Don't forget to admire that animated logo!
2. At the right hand side of the page, there is a box that says "Atom", a version number, and "mac OS", and that has a "Download" button.
3. Click on the "Download" button to download Atom. A zip file will be downloaded.
4. Once downloading has finished, open your browser's download menu, and drag the Atom icon to the "Applications" folder to install Atom.
5. Click on the Atom icon your Applications folder, to start Atom
6. A welcome screen will pop up – Welcome to Atom!
7. Click on the "Install a package" button, and then on the "Open Installer" button
8. A left pane called "+ Install Packages" appears, with a text box in which you can type package names.
9. One after the after, install the packages `atom-ide-ui` (the complete IDE), `ide-python` (language support for Python), `atom-python-run` (run your Python program straight from Atom)
10. Over the course of this book we will install more packages in Atom, but for the moment this will do! If you want, play around with *Themes* to make Atom look like you want.

Finally, we will use a versioning system called `git`. What is a versioning system, you ask? Well, think of it as an archive, where you can safely store previous versions of your programs. If you are working on something and all over sudden all goes horribly wrong (like, your cat walked over your keyboard and managed to delete half of your code), you can always go back to your archive, and take the latest version stored there. Of course, this does mean you need to regularly put your latest code into the archive otherwise ... but we will build up that "software engineering" discipline as we go along. Moreover, Atom will be a great help here as we will see.

1. `git` may already be installed on your Mac.
2. Open a Terminal window, and type in `git --version`.
3. If you get a response like "git version 2.10.1 (Apple Git-78)" then all is fine.
4. Else you will be immediately prompted whether you want to install `git` (which, of course, you do).

For the truly adventurous, you can also set up an account on GitHub. GitHub is basically "git in the cloud." You can use it to safely version everything not just locally on your laptop, but also in the cloud – useful if you want to share your code with others, or access your programs on different computers.

1.3.2. Windows

Too bad. Don't use Windows.

1.4. Conventions Used in This Book

We use the following type-setting conventions in this book:

Italic
for email addresses, URLs, filenames, pathnames, and emphasizing new terms when we first introduce them

`Constant width`
for the contents of files and the output from commands, and to designate modules, methods, statements, and commands

`Constant width bold`
used in code sections to show commands or text that you type in, and occasionally, to highlight potions of code

1.5. Using Code Examples

This book is here to help you learn more about programming, and creating adventures. The code is there to help in that. In general, you may use the code in this book in your own programs, and documentation (yes, programmers do -or should!- write documentation). You do not need to contact the author for permission unless you are using a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing the code *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Using a significant amount of example code described here in a commercial product or in your product's documentation *does* require permission.

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *storybytestudios@gmail.com*.

1.6. Contacting the Author

You can contact the author at *storybytestudios@gmail.com*.

Part I.

Getting Started

Introduction to Part 1

Let's get started! In this part we are going to cover quite some ground ... We start right away with building a simple, text-based adventure. Hello adventure world! We then continue with learning how we can let the player make decisions (i.e. actually play the game!), and how we can keep track of what all is happening. Towards the end of this first part, we build things out to a fully-fledged game engine that can run "any" kind of text adventure.

Specifically, you will learn:

- How to use *Atom*, `git`, to edit, run, and version your programs
- How to let your program show something in a Terminal (`print`), and get some input from your user (`input`)
- How to implement basic control structures in Python, for decisions (`if..then..else`) and loops (`while...` loops, and `for` enumerations)
- How to define functions, classes, and modules in Python (and what these things are! and why they are useful!)
- How to store your adventures as files, using a structured file format (*JSON*)
- How to save games, and load them, and how to load game resources

2. Hello Adventure World

2.1. Introduction

A program is basically a set of instructions what to do. There are different programming languages we can write such instructions in – Python is one of them. Other languages are for example Java, C, C++, Swift – by now (October 2018) probably close to 300 different programming languages exist! But, we will just focus on Python for the moment.

Python is actually more than "just" a language. It is also a program that can interpret your program and then translate it into instructions for your computer to execute. Let us see that in action right away.

Open a terminal, type in `idle3 &` after the $-prompt, and press return:

```
$ idle3 &
```

Up pops a little window, like the one below.

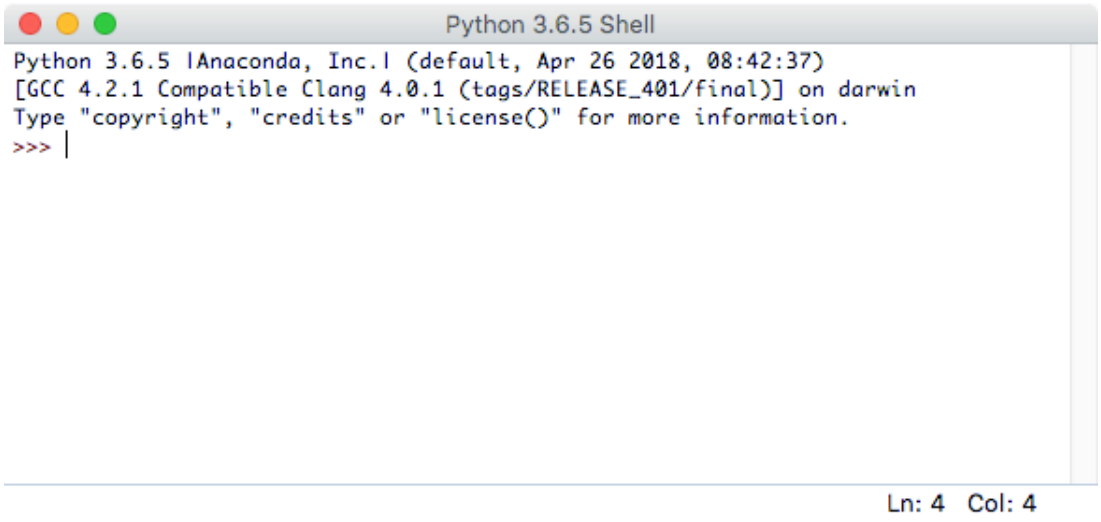

Figure 2.1.: The Python IDLE3 interactive editor

This is the `idle3` window. It is an interactive editor for Python. Everything you type in is immediately interpreted and executed – so you can see right away what it does! Let's see this at work. After the ">>>" prompt in the editor, type in the following:

```
>>> print("Hello adventure world!")
```