Python

Module 6

Reading, Writing, and Organising Files with Python

Bits and Bots
study group

Inhoud

This module

For this module, make sure you:

- Recap what you have learned in module 5. Check your own notes, or read the summaries provided at the end of the chapters of Al Sweigart's books.
- Read and work through chapters 9 and 10 of *Automate the Boring Stuff with Python*.

Further learning:

- Practice with File Handling on W3Schools.

Recap of Module 5

Like lists, a **`dictionary`** is a mutable collection of many values. But, *unlike* lists, indexes for dictionaries can use many different data types - not just integers. The indexes for dictionaries are called **`keys`**. Dictionary items are ordered, changeable, and do not allow duplicates.

The dictionary is typed with braces `{}` in a code. Dictionaries are, unlike lists, unordered and that's why they can't be sliced like lists. This is because items do not have a defined order, so you cannot refer to an item by using an index. While a newer version of Python does provide you with this option, the books by Sweigart are not yet updated to include this. However, as of Python version 3.7, dictionaries are ordered. This means that the items do have a defined order, and that order will not change. Be aware of this and which version of Python you are using.

Dictionaries are useful because you can map one item (the key) to another (the value), as opposed to lists, which simply contain a series of values in order. Values inside a dictionary are accessed using square brackets just as with lists.

Reading and Writing Files

The first modules of this course have been focused on learning the Python basics, while making games. The last two modules will centre a bit more on how to use Python in your day to day work, by learning how to read, write, organise, delete, rename, and move files. We recommend creating a folder with some files that you can use to practice on. You can simply copy some files that are on your computer into your new folder, or create some word, pdf and png files that you place in the new folder.

Create a folder on your computer and add some files to it for practice purposes.

This module starts with the basics of how to navigate to files and folders on your computer by understanding what a directory and a file path is. You should be able to recognise some of this already, as GitHub uses paths as well. Have a look at the Bits and Bots repository and check out different files in the folder. Do you notice the file path at the top of your screen?

After you have had a look at the basics, you will learn the difference between an absolute and a relative path, and how to create a new folder and write to files. And because games are such a fun element to learn Python, you can use your new skills to create a random quiz generator.

Read and work through Chapter 9 of *Automate the Boring Stuff with Python* until 'Project: Generating Random Quiz Files'.

Extra challenge: Have a look at the quiz project at the end of the chapter and alter the quiz to make it more digital preservation themed.

Organising Files

Now that you've learned how to read and write files with Python, it is time to start organising them. With a Python program, you can copy, rename, move, delete or compress multiple files in mere seconds, which would have taken a lot of time by hand.

Read and work through Chapter 10 of *Automate the Boring Stuff with Python* until 'Project: Renaming Files with American-Style Dates to European-Style Dates'.

Extra challenge: If you work in digital preservation, you should know a thing or two about back-ups. But did you know you can create a Python program that does the job for you? Follow the project 'Backing Up a Folder into a ZIP File' at the end of the chapter.

Introduction to APIs

This month’s expert session will focus on building and using APIs with Python. You'll learn what an API is, how to request data from an API in Python, and how to create your own basic API. You'll also explore how to integrate APIs into your website using JavaScript. Exercises for both Python and front-end development will be shared and explained during the session, and later made available on our channels.

APIs—short for *Application Programming Interfaces*—are sets of rules that allow different software applications to interact. They act as messengers between systems, enabling one program to request data or services from another. For example, when a website displays weather updates, it likely uses an API to fetch real-time data from a weather service.

Here are some key reasons APIs are useful:

- **Automation**: APIs allow programs to perform tasks automatically, such as retrieving data, sending emails, or processing payments, without manual intervention.
- **Integration**: They enable different systems and platforms to work together. For instance, a website can use APIs to connect with social media platforms, payment gateways, or databases.
- **Scalability**: APIs help developers build modular systems that can grow and evolve. You can update or replace parts of a system without affecting the whole, making maintenance easier.
- **Efficiency**: Instead of building every feature from scratch, developers can use APIs to access existing services—saving time and resources.
- **Security**: APIs can control access to data and services, ensuring that only authorized users or systems can interact with sensitive information.

Understanding how APIs work and how to use them effectively is a vital skill for anyone building modern applications or websites.