Python for Digital Humanities

Theena Kumaragurunathan

Fri Nov 18 12:16:00 AM +0530 2022

Contents

1 Lesson 1 INTRODUCTION TO PYTHON FOR DIGITAL HUMANITIES 2
- 1.1 Resources . . . 2
- 1.2 Requirements . . . 2
- 1.3 Acknowledgements . . . 3
- 1.4 Copyright . . . 3
- 1.5 What is Python . . . 3
- 1.6 Why Python . . . 4
- 1.7 Installing Python . . . 6
- 1.8 Pythonic . . . 6

2 PART 1 WORKING WITH DATA IN PYTHON SCRIPTS 6
- 2.1 Lesson 2: Story Data in PYthon Script . . . 6
- 2.2 Lesson 3: Interacting with Strings in a Python Script . . . 6
- 2.3 Lesson 4: Interacting with Numbers in Python (Integers and Floats) . . . 6

3 PART 2 DATA STRUCTURES IN PYTHON 6
- 3.1 Lesson 5: Creating and Working with Types in Python . . . 6
- 3.2 Lesson 6: Creating and Working with Lists in Python . . . 6
- 3.3 Lesson 7: Creating and Working with Dictionaries in Python 6

4 PART 3 INTERACTING WITH DATA STRUCTURES IN PYTHON 6
- 4.1 Lesson 8: Python Conditionals . . . 6
- 4.2 Lesson 9: Python Loops . . . 6
- 4.3 Lesson 10: Python Functions . . . 6
- 4.4 Lesson 11: Python Classes . . . 6

5 PART 4 WORKING WITH TEXT DATA **6**
5.1 Lesson 12: Python and Text Files 6
5.2 Lesson 13: Python and Modules and Libraries 6
5.3 Lesson 14: Working with Regex Library (1) 6
5.4 Lesson 15: Working With Regex Library (2) 6

6 PART 5 PYTHON AND WORKING WITH DATA STORED IN EXCEL **6**
6.1 Lesson 16: Reading Data from Excel Using the XLRD Library 6
6.2 Lesson 17: Searching Data Imported from Excel 6
6.3 Lesson 18: Writing Data to Excel Files using the XLSWXWRITER Library 6

7 PART 6 PYTHON AND WORKING WITH DATA FROM THE WEB **6**
7.1 Lesson 19: Finding HTML Code from a website 6
7.2 Lesson 20: Using the Python Library Requests to Interact with a Website 6
7.3 Lesson 21: Using the Python Library Beautiful Soup to Rip Data from a Website 6

8 PART 7 PYTHON AND WAYS TO STORE DATA **6**
8.1 Lesson 22: Storing Data in Text Files 6
8.2 Lesson 23: Storing Data in XML Files 6

1 Lesson 1 INTRODUCTION TO PYTHON FOR DIGITAL HUMANITIES

1.1 Resources

This notebook is based on lessons found here: Python For Humanities

1.2 Requirements

1. Python 3.8 or >
2. A text editor of your choice, but preferably PyCharm

1.3 Acknowledgements

This notebook owes a debt of gratitude to Python tutors all over the web, who have spent the last few decades evangelizing its usage outside the traditional domains of computer programming, arguing cogently that here was a language for everybody who uses computers to do any manner of computational work, whether that is writing a novel, a thesis, analyzing excel sheets, or reading PDFs while researching. In all these use cases, and more, Python finds itself to be a useful utility.

This notebook follows the coursework made available by Dr William Mattingly] on PythonHumanities.Com

1.4 Copyright

This is an **open notebook** based on the course created and taught by Dr William Mattingly. As such, the **material belongs to Dr Mattingly**; these are merely my notes.

I am making this freely available in the hope that it helps a Humanities student who is looking to get a head start in their Python programming journey.

I am unsure which licensing bracket this repo falls into. I will only restate for absolute clarity that the structure and substance of this course material is the work of Dr William Mattingly.

While Dr Mattingly's course is geared towards a specific audience, my notes will have additional editorial notes meant for me. I hope this extra editorializing is useful to anyone who comes across this repo, but if it's not, you are free to adapt these notes to your own ways of thinking and recalling concepts. If you wish to fork this repo, please ensure that you make clear that the course material itself belongs to Dr William Mattingly.

I welcome any pull requests! The notebook was written in a hurry, to meet a specific work-related need on my end - all errors - whether in text or code, are mine. Please feel free to send a pull request if corrections are needed.

1.5 What is Python

Python is a general purpose **interpreted** programming lanuage.

It is general purpose because its usage is wide, finding utility across a diverse range of disciplines. Finance professionals use it to analyze Excel data, data scientists use it find insights in vast troves of statistical data, programmers themselves use it to create simple applications to solve specific

every-day problems they come across in their day-to-day work. Those of us outside the world of code and financial data, will find Python just as useful: automating the downloading of public domain images, summarizing an academic paper, finding keywords in a research report using textual analysis. The writer/programmer/educator Al Sweigart's Automate the Boring Stuff With Python highlights the many ways the menial tasks that we perform on computers can be relegated to Python. In this way, I liken it to a Swiss Army Knife of programming languages.

Programming languages are of two main types:

- Compiled
- Interpretted

The distinction between is useful to know.

Since we are speaking of *languages*, the distinction between the two can be likened to reading a set of instructions - let's assume a recipe for a curry; the instructions are written in Sinhala. There are two ways for you to follow the instructions: *either look for a translation of the recipe* or *have a friend who reads the language to read the instructions line by line.*

In this analogy, the translated version of the recipe is a *Compiled Language*, while the friend offering you line-by-line translations, is the *Interpretted Language*

The advantages of one over the other is beyond the scope of this course.

1.6 Why Python

Why Python specifically then? Why not the C programming language?

If we go back to the language analogy, we know from real life experience that some languages are easier to pick up than others. Python excels because its grammar and lexicon are easier to grasp for the non-computer programmer. Python's syntax, once internalized, has a logic that isn't unlike learning a human language - in fact, Python's code is especially known for being readable in a way, say, code written in *C* or *C++* or *Java* aren't.

As a result, Python users are diverse, encompassing people from the technical to the non-technical, from the sciences to the arts, and everyone in between.

There is of course nothing wrong with learning C or any other programming language if you are feeling motivated, but to employ C to do many of the tasks that non-programmers typically perform on computers, will be the equivalent of using *Excaliber* to slice a loaf of bread; it will get the job done, but you might damage your kitchen in the process.

1.7 Installing Python

1.8 Pythonic

2 PART 1 WORKING WITH DATA IN PYTHON SCRIPTS

2.1 Lesson 2: Story Data in PYthon Script

2.2 Lesson 3: Interacting with Strings in a Python Script

2.3 Lesson 4: Interacting with Numbers in Python (Integers and Floats)

3 PART 2 DATA STRUCTURES IN PYTHON

3.1 Lesson 5: Creating and Working with Types in Python

3.2 Lesson 6: Creating and Working with Lists in Python

3.3 Lesson 7: Creating and Working with Dictionaries in Python

4 PART 3 INTERACTING WITH DATA STRUCTURES IN PYTHON

4.1 Lesson 8: Python Conditionals

4.2 Lesson 9: Python Loops

4.3 Lesson 10: Python Functions

4.4 Lesson 11: Python Classes

5 PART 4 WORKING WITH TEXT DATA

5.1 Lesson 12: Python and Text Files

5.2 Lesson 13: Python and Modules and Libraries

5.3 Lesson 14: Working with Regex Library (1)

5.4 Lesson 15: Working With Regex Library (2)

6 PART 5 PYTHON AND WORKING WITH DATA STORED IN EXCEL

6.1 Lesson 16: Reading Data from Excel Using the XLRD Library

6.2 Lesson 17: Searching Data Imported from Excel

6.3 Lesson 18: Writing Data to Excel Files using the XLSWXWRITER Library