Curriculum: Learning Python for Beginners

Prerequisites.

- Install an Anaconda distribution/Python3 version and a code editor.
- *(optional)* download the course materials: https://github.com/AQuaintExpression/python_course_vf
- No prior knowledge needed.

Chapter 1: Introduction to Python

- Overview of Python and its uses
- Installing Python and a text editor
- Running Python code in the interpreter and in a file

Chapter 2: Basic Python Syntax and Data Types

- Variables and variable types
- Numbers and arithmetic operations
- Strings and string manipulation
- Lists and list manipulation
- Tuples and tuple manipulation
- Dictionaries and dictionary manipulation

Chapter 3: More on Data Types

- Advanced string manipulation
- Regular expressions

Chapter 4: Control Structures

- If-else statements
- Loops (for and while)
- Break and continue statements

Chapter 5: Functions

- Defining functions
- Parameters and arguments

- Return statements

Chapter 6: Working with Files

- Reading and writing text files
- Working with CSV files
- Working with JSON files
- Working with XML files
- Converting XML to JSON (xmltodict)
- Converting JSON to CSV
- Converting XML to CSV (advanced topic -> to be handled after Pandas or as a sum of the previous 2 subtopics)

Chapter 7: Introduction to Pandas

- What is Pandas and why is it useful?
- Pandas Series and DataFrame objects
- Loading data into a DataFrame
- Basic DataFrame manipulation

Prerquisites

Installing Python and a text editor.

Using an Anaconda distribution (what we will be using during this course)

Using a standalone distribution + VS Code.

Chapter 1: Introduction to Python

Overview of Python and its uses

Python is a high-level, interpreted programming language that emphasizes code readability and simplicity. It was first released in 1991 and has since become one of the most popular programming languages in the world. Python's popularity is due to its versatility and wide range of applications, which include web development, scientific computing, data analysis, machine learning, network automation, and artificial intelligence. It is known for its clear and concise syntax, which makes it easy to learn and use for both beginners and experienced programmers. Python also has a large and active community that contributes to its development and supports users through online resources and libraries.

Pros and Cons of Using Python

Pros

- **Easy to Learn:** Python has a simple and intuitive syntax that makes it easy to learn and use, especially for beginners.
- **Versatile:** Python has a wide range of applications, from web development to scientific computing to machine learning, making it a versatile language to learn.
- **Large and Active Community:** Python has a large and active community of users who contribute to its development and provide support through online resources and libraries.
- **Excellent Libraries and Frameworks:** Python has a rich set of libraries and frameworks that make it easy to develop complex applications quickly and efficiently.
- **Cross-Platform Compatibility:** Python code can be run on various operating systems, including Windows, macOS, and Linux, making it a flexible choice for developers.

Cons

- **Slower Execution Speeds:** Python is an interpreted language, which means that it can be slower than compiled languages like C++ or Java. This can be a disadvantage in performance-critical applications.
- **Global Interpreter Lock:** Python's Global Interpreter Lock (GIL) can limit the performance of multithreaded applications, which can be a concern for applications that require high concurrency.
- **Dynamic Typing:** Python is dynamically typed, which means that type checking is done at runtime rather than compile time. This can lead to more errors in code that are not caught until runtime.
- **Less Suitable for Mobile App Development:** Python is not well-suited for mobile app development due to its slower performance and limitations on mobile platforms.
- **Version Compatibility:** Python has multiple versions, and code written in one version may not be compatible with other versions, which can be a concern for developers. (less relevant since Python2 is no longer in scope. Most libraries and packages work well with all Python3 distributions)

Running Python code in the interpreter and in a file

Running Python code in the interpreter

The Python interpreter is a command-line interface that allows you to execute Python code directly in the terminal. This is useful for quick testing and debugging of code snippets. Here is an example of running Python code in the interpreter:

```
PS C:\github\python_course_vf> python
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC
v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more
information.
>>> print("Hello world!")
Hello world!
>>>
```

In the example above, we start the Python interpreter by running the `python` command in the terminal. Once inside the interpreter, we can execute Python code by typing it directly into the terminal. In this case, we print the string "Hello, world!" using the `print()` function.

Running Python code in a file

Python code can also be saved in a file with a `.py` extension and executed using the Python interpreter. This is useful for more complex programs and allows you to save and share your code. Here is an example of running Python code in a file:

`hello.py`

```
print("Hello, world!")
```

In this example, we create a file called `hello.py` and save it in our current working directory. The file contains a single line of Python code that prints the string "Hello, world!" using the `print()` function. To execute the code in the file, we run the following command in the terminal:

```
PS C:\github\python_course_vf\part_1> python .\hello.py
Hello, world!
```