Intro to Python & Jupyter Notebook

Coding 2: 8 Feb 2023

With Guest Lecturer Prof. Rebecca Fiebrink

Link for today's slides & resources:

https://tinyurl.com/coding2-8feb

Today

- What is Python? Why Python?
- Non-syntax basics
  - Running Python, Jupyter notebook
  - Modules, libraries
  - Conda, environments, versions
  - Learning the language and getting help
- Code along with Mick's syntax intro videos
- Homework review
- Activity
- Installation troubleshooting throughout

What is Python?

- A programming language
- Really useful for
  - Data analysis
  - Data visualization
  - Machine learning
  - Natural language processing (e.g., document analysis, chatbots)
  - Non-real-time work with audio, music, image, etc.
  - Server-side web programming
  - Interfacing with web APIs (e.g., interacting with Instagram, Spotify, YouTube, Twitter, etc.)
  - Teaching programming

Why should you learn Python?

- Simple syntax, easy to learn, easy to find documentation/tutorials/help
- Not as fast as C++, but many libraries are C++ underneath (and many run fast on GPU)
- Many great modules & libraries available
  - E.g., for working with text/language, web scraping, machine learning, making beautiful plots, analysing audio and image, running statistical tests in research, …
- Lots of existing github projects for you to run and adapt
- Do a lot with minimal effort & code, share your work with the world
- Much more employable if you know Python + JS

Stable Diffusion Public Release

It is our pleasure to announce the public release of stable diffusion following our release for researchers [https://stability.ai/blog/stable-diffusion-announcement]

Libraries

Python library

We provide a Python library, which you can install as follows:

```
$ pip install openai
```

Once installed, you can use the bindings and your secret key to run the following:

```
import os
import openai

# Load your API key from an environment variable or secret management service
openai.api_key = os.getenv("OPENAI_API_KEY")

response = openai.Completion.create(model="text-davinci-003", prompt="Say this i
```

The bindings also will install a command-line utility you can use as follows:

```
$ openai api completions.create -m text-davinci-003 -p "Say this is a test" -
```

Beautiful Soup Documentation

Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with your favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree. It commonly saves programmers hours or days of work.

These instructions illustrate all major features of Beautiful Soup 4, with examples. I show you what the library is good for, how it works, how to use it, how to make it do what you want, and what to do when it violates your expectations.

This document covers Beautiful Soup version 4.8.1. The examples in this documentation should work the same way in Python 2.7 and Python 3.2.

You might be looking for the documentation for Beautiful Soup 3. If so, you should know that Beautiful Soup 3 is no longer being developed and that support for it will be dropped on or after December 31, 2020. If you want to learn about the differences between Beautiful Soup 3 and Beautiful Soup 4, see Porting code to BS4.

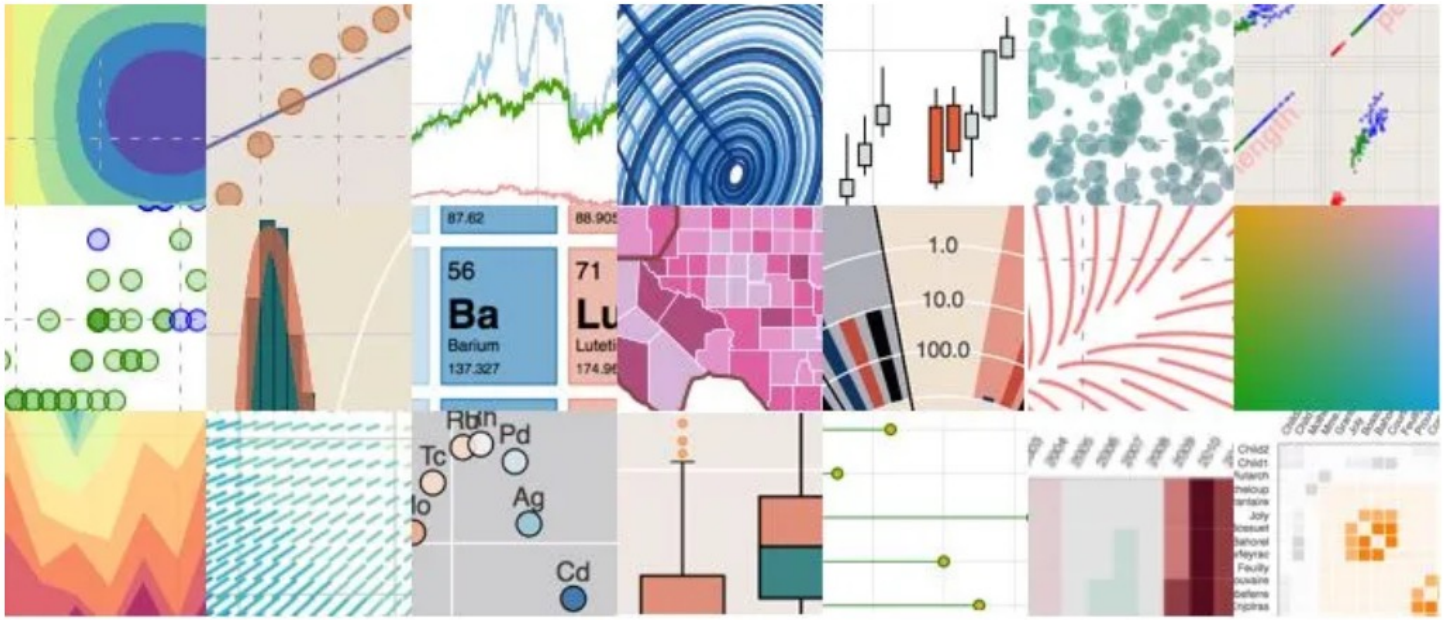
This documentation has been translated into other languages by Beautiful Soup users:

Jaouad Eddadsi

Jan 6, 2022 · 4 min read · Listen

Make Data Visualization Easy with Seaborn and Pandas

How to use pandas and seaborn libraries to generate different graphs



What should you *not* do in Python?

- Real-time audio, music, graphics, video, etc.
  - *But it's fine for loose interaction, e.g. responding to someone's action on a website*

What do you need to understand to learn Python?

- How to run Python
- Modules, packages, libraries
- Conda, environments, versions
- Learning the language and getting help

Running python

- Command line, Jupyter
  - Demos:
    - Running a .py file
    - Launching a notebook
    - Running code in a notebook
    - Magically printing the result of the last line in a cell
    - Adding markdown text to a notebook

Modules, packages, libraries

- All are ways of organizing and reusing Python code

Stuff.py

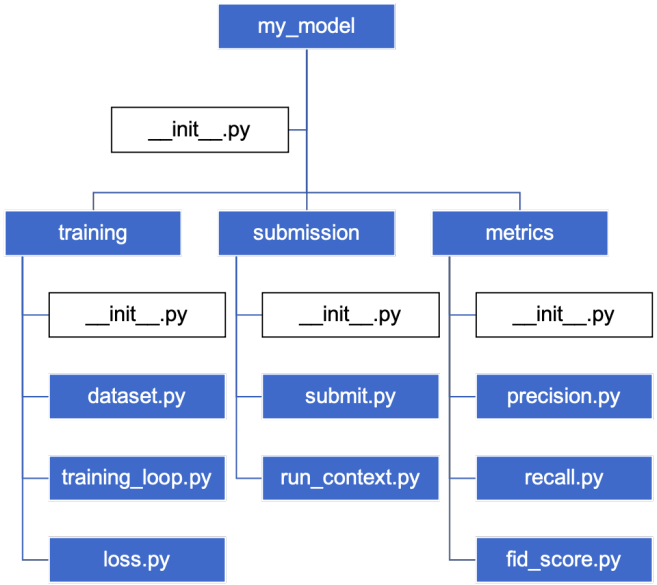
Module: a single file, ending in .py

You can easily write these yourself (similar to writing multiple .cpp or .js files)

See https://learnpython.com/blog/python-modules-packages-libraries-frameworks/

Modules, packages, libraries

- All are ways of organizing and reusing Python code



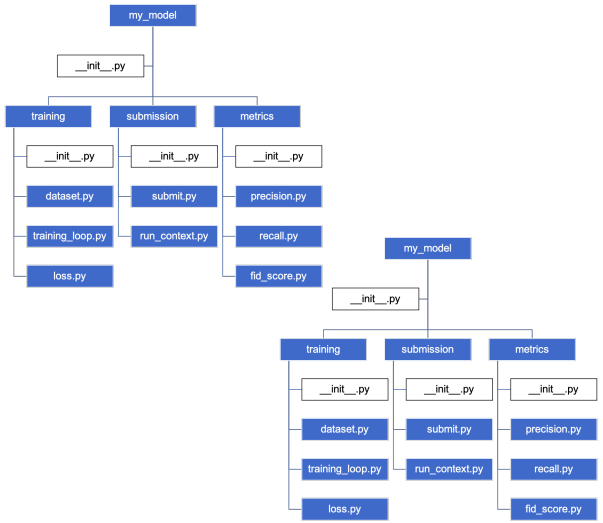
Package: A collection of modules, organized in a hierarchical structure

Convenient way of organizing a set of modules with a common purpose (e.g., visualization)

See https://learnpython.com/blog/python-modules-packages-libraries-frameworks/

Modules, packages, libraries

- All are ways of organizing and reusing Python code



Library: A collection of module(s) and/or packages that is meant to be reused (often by others)

You'll use lots of existing libraries for data analysis, visualization, etc. (e.g., numpy, pandas, keras, Requests, …)

See https://learnpython.com/blog/python-modules-packages-libraries-frameworks/

Installing new packages/libraries

- Pros:
  - It's easy!
  - Python package managers (e.g., PyPI, the "Python package index") keep track of popular third-party packages
    - You don't need to manually track down a library or keep track of dependencies (e.g., when one library requires a specific version of another library)
  - You can install new libraries (and their dependencies) by running a command-line command, e.g.

    `python3 -m pip install SomeProject` or `conda install SomeProject`
    - Then just use "import" in your Python code ("import SomeProject") to use that library in that code
    - No need to keep track of where you save libraries on your computer
- Cons:
  - If you use Python for different projects, they may need different packages, and these may conflict!
    - What if one project requires SomeProject v1.0 and another project requires SomeProject v1.1?

https://packaging.python.org/en/latest/tutorials/installing-packages/#installing-from-pypi

Solution: Environments

- We can install packages (and python versions) into specific, separate *environments* to avoid conflicts
  - Create a new environment for each project (or MSc course)
  - Activate this environment before installing new packages, and before running notebooks/code that relies on those packages

```
Python 3.9.0
numpy v. 1.23.2
keras v. 2.10.0
nltk v. 3.7
```

environment "`myProject1`"

```
python 3.10.6
numpy v. 1.23.2
keras v. 2.11.0
```

environment "`coding2`"

conda

- In Coding 2 and Coding 3, we'll be using *conda* to manage environments and packages
  - Conda can get packages from the Anaconda repository (very useful for data analysis, visualization, machine learning)
    - Can allow you to use packages that aren't written in Python
    - Better dependency management
- Follow our instructions to install conda via either miniconda or miniforge: https://git.arts.ac.uk/rfiebrink/ExploringMachineIntelligence_Spring2023/blob/main/1-Installation.md

More info: https://www.anaconda.com/blog/understanding-conda-and-pip , https://stackoverflow.com/questions/60532678/what-is-the-difference-between-miniconda-and-miniforge

Making a new environment

```
conda create --name coding2 python=3.9
```

See https://git.arts.ac.uk/rfiebrink/ExploringMachineIntelligence_Spring2023/blob/main/1-Installation.md

Launching a notebook with a specific environment

In terminal / commandline:

```
conda activate coding2
```
← Activate the environment

```
jupyter-notebook
```
← To begin writing code in a notebook that uses this environment

Note that this will by default take you to your current directory in the terminal! If you want to launch notebook somewhere else, change directories ("`cd directoryName`") sometime before calling `jupyter-notebook`

Adding new packages to an environment

In terminal / commandline:

conda activate coding2 ← Activate the environment

then

`conda install packageName` ← Try this first (use conda package manager)

or

`pip install packageName` ← If conda install doesn't work, use pip

Anything else?
Questions?

Reminder: Attendance

Rest of today:

1. Next 40 minutes: Watch Mick's videos and code along in Jupyter:
   - "Printing, Variables, Types, Arrays, Lists, Dicts"
   - "Conditionals, Loops, Functions"

   Pause for group check-in, tips
2. Homework review
3. Hands-on, self-directed:
   - For beginners: Work through Python tutorial at https://www.w3schools.com/python/ (start with "Python Syntax" section on left-hand menu)
   - Everyone: Try to get through the first 5 levels of the Python Challenge http://www.pythonchallenge.com/
4. Homework: Make your own challenge

If Jupyter Notebook is still not working for you, ask for help ASAP

These slides and all links are online at https://tinyurl.com/coding2-8feb

Getting help in python

- Type "?" before a function/variable name to read its documentation

Try

```
?print
x = 5
?x
```

Getting help in python

- In Jupyter, you can type the name of a variable or package then "." then **tab** to see autocomplete options (e.g., functions you can call)

```
In [35]: myList = ["a", "b", "c"]

In [ ]: myList.
        append
        clear
        copy
        count
        extend
        index
        insert
        pop
        remove
        reverse
```

```
In [46]: import numpy as np

In [ ]: np.
        abs
        absolute
        add
        add_docstring
        add_newdoc
        add_newdoc_ufunc
        all
        allclose
        ALLOW_THREADS
        alltrue
```

Getting help in python

- https://www.w3schools.com/python/default.asp is a great resource and tutorial collection
- Lots of other resources: web tutorials, youtube tutorials, books (including ebooks)
  - Make sure they're Python 3

Python vs. C++ vs. JavaScript

Python	C++	JS
Interpreted (errors not caught until code run); runs in a Python interpreter	Compiled (fast, compiler can catch some errors before code is run)	Interpreted; runs in a Web browser
Garbage collection (automatic memory management)	Manual memory management	Garbage collection
Slower	Faster	Slower
Far from the hardware	Closer to the hardware	Far from the hardware
Can be object-oriented (classes, objects, functions) or functional	Object-oriented (classes, objects, functions)	Can be object-oriented or functional
Dynamically typed (no need to declare variable type; type can change during runtime)	Statically typed (variables are declared with a type; type cannot be changed)	Dynamically typed
Back-end web programming	Not usually used for web programming, but can be (back-end)	Front-end web programming, sometimes back-end

Questions? Anything else?

Rest of today:

1. ~~Next 40 minutes: Watch Mick's videos and code along in Jupyter:~~
   - ~~"Printing, Variables, Types, Arrays, Lists, Dicts"~~

~~"Conditionals, Loops, Functions"~~

~~*Pause for group check-in, tips*~~

2. Homework review
3. Hands-on, self-directed:
   - For beginners: Work through Python tutorial at https://www.w3schools.com/python/ (start with "Python Syntax" section on left-hand menu)
   - Everyone: Try to get through the first 5 levels of the Python Challenge http://www.pythonchallenge.com/
4. Homework: Make your own challenge

If Jupyter Notebook is still not working for you, ask for help ASAP