Introduction to Python

Foreword

`Python` is probably the most popular programming language by many measure these days, due to its simple learning experience and large ecosystem. In fact, I bet most of the people who are reading this note already know `python`. `python` has a long history and a huge community which you can probably do everything you want in `python`. However, being simple to learn also means it is easy to write bad code. Getting your calculation is one thing, building a nice package which people can use happily is another. There are many tricks and know-hows in `python` that you may not be aware of. And this is what we are going to focus in this lab.

In this lab, we are going to go through the following topics:

1. Some basic concepts and syntax for python, and try to write a simple insertion sort algorithm in python.
2. Basic of structuring and packaging a python project.
3. Building documentation with `mkdocs`.
4. Adding tests to the project.
5. Some best practices and development tips.
6. Some noteworthy libraries in python.

Key Concepts

Python is an interpreted language

Everything is an object

Basic Syntax

In this section, we are going to go through some basic syntax of python. We are only cover the minimum you need to know to write a simple insertion sort algorithm, especially most of you are already familiar with the `python` syntax.

Variables

To define a variable in python, you simply assign a value to a variable name. For example, to define a variable `a` with value `1`, you can do: `a = 1`. There are a couple of basic data types in `python`, like numbers, string, boolean. Three slightly more complicated datatypes are list, tuple, and dictionary.

List is a list of objects, which can be defined using the syntax `a = [0,1]`. A Tuple is an immutable list of objects, which can be defined using `a = (0,1)`. It is handy whenever you do not want things to change. A dictionary is basically a list but instead of accessing it by the index of the element, there are a list of key-value pairs, which you can access the values through their respective key. You can define a dictionary with the syntax `a = {"x": 1, "y": 2}` The dictionary in `python` is basically a hash table. They all can be accessed using `variable[index/key]`

One thing to remember is everything is an object in Python, meaning they (almost) all have some attributes and methods to themselves. If you have a background in `C` or some similiarly low level language, you may find be able to define something like `a = [0, "this", true]` blasphemous. There is certainly performance and stability implication to this feature, but I believe this flexibility is what makes `python` easy to get into.

Control flow

Introduction to Python

Functions

In order to define a function in python, you use the `def` keyword. For example, to define a function `add` which takes two arguments `a` and `b` and return the sum of `a` and `b`, you can do:

```
def add(a, b):
    return a + b
```

Now there is one tricky thing about `python`, which is the passed-by-object-reference. Some of you may have already ran into this in an unfortunate way, but for those who are not aware, here is a very sneaky failure mode one may spend hours trying figure out what's going on. Say I have defined a list `x = [0, 1]`, and I have a function that wants to modify

Running a python script

Exercise: Writing an insertion sort algorithm

Now given the information above, let's try to write an insertion algorithm.

Packaging code

Modules and import

Setuptools

Exercise: Setting up directory structure and build a binary

Building documentation

Mkdocs

Style it with Material with mkdocs

Exercise: Adding documentation to the code and build the documentation

Writing tests

Unit tests with pytest

How to write unit pytest?

Exercise: Adding tests and test it

Best practices/Development tips

Virtual environment

IPython and Jupyter

Typing

Linting and formatting

Debugging

Noteworthy libraries

Jax

I like to start with `jax` probably because I am a heavy `jax` user. Don't get me wrong, I use `PyTorch` too, but I use `jax` more because of its performance but also the workflow I have in `jax` is very close to the workflow I had before machine learning library became a thing. Basically, if you are familiar with `numpy` and `scipy`, you should find `jax` is basically `numpy` on steroids.

Flask

HoloViews

FastHTML

Too common/Too obscure

`numpy`, `scipy`, `pandas`, `matplotlib`, `seaborn`, `scikit-learn`, `pytorch`,

Checklist

By now, you should have completed all of the following items: