Unit 3

Overview

- “A data scientist knows more about statistics than a software engineer, and more about programming than a statistician.”
- In this unit we will build the core skills needed to communicate and work with software engineers.
- You not only have to know how to build the latest and greatest machine learning models, but build and approach it with software development best practices.
- To do this, we will go beyond Python notebooks.

Sprint 1

- Module 1 - Python Modules, Packages, and Environments
- Module 2 - OOP, Code Style, and Reviews
    - Basic principles of object-oriented programming.
- Module 3 - Containers and Reproducible Builds
    - Docker is a common standard and tool for containers.
- Module 4 - Software Testing, Documentation, and Licensing

Module 1

Python Modules, Packages, and Environments

- Learning Objectives
    - Understand Modules, Packages and Environments
    - Create a Python module and package
    - Install dependencies in a dedicated environment

Module

- The code we write for data science and machine learning grows over time.
- Keeping the code in one file makes it difficult to maintain and debug.
- We may want to split it into several files.
- We may also want to use a function that we have written without copying its definition into each program.
- The solution: python modules.

Module

- Modules help us to organize our code.
- A module is re-useable and importable code.
- A module is a file with the “.py” extension that contains python code.
- This code can also be imported into other modules or python programs.
- The name of the module is the name of the file.
    - Eg: Filename = foo.py, Module = foo

Create Python Modules

```
def welcome(module_name):
    print(“Welcome “ + module_name + “ to the World of Modules!”)
```

Use Python Module

- Usage: import module
    - Call: module.function
- Usage: from module import function
    - Call: function
- Usage: import module as alias
    - Call: alias.function
- Usage: from module import function as alias
    - Call: alias

Python Package

- Python package is a collection of modules.
- It is a folder containing modules
    - And maybe other folders that may contain other modules.
- An __init__.py file is required to make Python treat directories containing the file as packages.
    - It can just be an empty file.

Namespace

- A system that has a unique name for each and every object - variable, function.
- A mapping between variable names and objects.
- Local — All the names of variables declared within a function.
- Global — All the names of variables and functions that are included in the modules being used in the program and Local namespace.
- Builtin — All the built in names live here and this namespace is created when the interpreter starts up. Contains global which in turn contains local.
- When a function starts execution, a namespace is created, and is dropped when it finishes execution.

Virtual Environment

- A virtual environment is a Python environment such that the Python interpreter, libraries and scripts installed into it are isolated from those installed in a “system” Python - as part of your operating system or other virtual environments. (https://docs.python.org/3/library/venv.html)

- Pipenv (https://pipenv.pypa.io/en/latest/)

- venv (https://docs.python.org/3/tutorial/venv.html)