A Primer (Refresher) on Python

Javier Martinez Hernandez

Western University, Economics Department

fmarti23@uwo.ca

Introduction

- Python is a programming language created by Guido van Rossum, and released in 1991.
- Regularly used for:
  - software development,
  - athematics,
  - system scripting.
- Useful because
  - Python works on multiple platforms (Windows, Mac, Linux, etc).
  - Python has a simple syntax similar to the English language.
  - Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

Introduction

- We will use the recent major version of Python: Python 3 (but Python 2 is still widely used; main problem is that there is not backward compatibility).
- In this tutorial, Python will be taught in an Integrated Development Environment (IDE): Visual Studio Code.
- You can use Jupyter or text editors to write code in Python.
- Python was designed for readability: mathematics and English.
- Python relies on new lines to complete a command.
- Main drawback: Python relies heavily on indentation. So if you do not indent or miss one, Python will not execute the command.

Installation and Best Practices

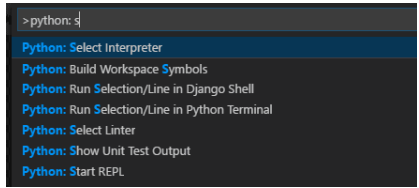
- Apple computers already have a version of Python installed.
- However, I recommend downloading the Anaconda distribution of Python. `https://www.anaconda.com/download`
- It contains Conda, a critical package and environment management system.
- Relevant because you don't want to mess with your native OS Python distribution (Apple computers) and you want to have environments for your projects to safeguard (sandbox) their operability.

Visual Studio

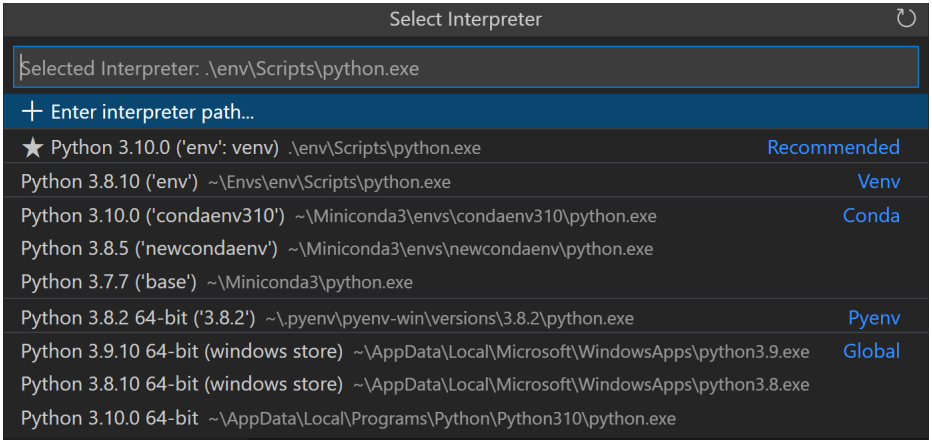
- Code editor with support for debugging, task running, and version control.
- Allows you to write in multiple languages with extensions found in the VS Code Marketplace.
- Integration with Github and Github Copilot.
- Download from `https://code.visualstudio.com/`
- You must install a Python interpreter yourself separately from the VS Code extension.

Visual Studio

- Download the Python extension for VS Code from: `https://marketplace.visualstudio.com/items?itemName=ms-python.python`
- Since we care about sandboxing our environment, we need to specify an interpreter for VS Code in case it does not automatically locate it.
- This allows us to select the conda environment we want to use for our project.



Visual Studio



Git, Github Desktop and Github Copilot

- Version control is the practise of tracking and managing changes to code.
- The coding language that lets you keep track of changes is called Git.
- Version control is useful because you may need to access previous versions of your files, need backups of code and results, and need to share code for collaboration purposes.
- Lets you avoid having multiple “v2”, “v2_final” files or creating new folders.

Git, Github Desktop and Github Copilot

- Git act locally (on your computer) to track changes in your local files.
- You can link your local repository to an online repository (like Github), so you can easily share your code.
- Online repository should only be back up.
- We will come back to Git, Github and Github copilot in the last class (time permitting).

NumPy

- Short for Numerical Python.
- It provides the data structures, algorithms, and library glue needed for most scientific applications involving numerical data in Python.
- A fast and efficient multidimensional array object ndarray.
- Functions for performing element-wise computations with arrays or mathematical operations between arrays.
- Tools for reading and writing array-based datasets to disk.

NumPy

- Linear algebra operations, Fourier transform, and random number generation.
- A mature C API to enable Python extensions and native C or C++ code to access NumPy's data structures and computational facilities.
- Primary use in data analysis is as a container for data to be passed between algorithms and libraries.
- For numerical data, NumPy arrays are more efficient for storing and manipulating data than the other built-in Python data structures.
- Libraries written in a lower-level language, such as C or Fortran, can operate on the data stored in a NumPy array without copying data into some other memory representation.

pandas

- pandas provides high-level data structures and functions designed to make working with structured or tabular data fast, easy, and expressive.
- The primary objects in pandas commonly used are:
  - the DataFrame, a tabular, column-oriented data structure with both row and column labels,
  - the Series, a one-dimensional labeled array object.
- pandas blends the high-performance, array-computing ideas of NumPy with the flexible data manipulation capabilities of spreadsheets and relational databases (such as SQL).

pandas

- pandas provides sophisticated indexing functionality to make it easy to reshape, slice and dice, perform aggregations, and select subsets of data.
- pandas is useful for data manipulation, preparation, and cleaning.

SciPy

- SciPy is a collection of packages addressing a number of different standard problem domains in scientific computing.

- scipy.integrate: Numerical integration routines and differential equation solvers

- scipy.linalg: Linear algebra routines and matrix decompositions extending beyond those provided in numpy.linalg

- scipy.optimize: Function optimizers (minimizers) and root finding algorithms

SciPy

- scipy.signal: Signal processing tools
- scipy.sparse: Sparse matrices and sparse linear system solvers
- scipy.special: Wrapper around SPECFUN, a Fortran library implementing many common mathematical functions, such as the gamma function.
- scipy.stats: Standard continuous and discrete probability distributions (density functions, samplers, continuous distribution functions), various statistical tests, and more descriptive statistics.

scikit-learn

- Premier general purpose machine learning toolkit for Python programmers.
- Classification: SVM, nearest neighbors, random forest, logistic regression, etc.
- Regression: Lasso, ridge regression, etc.
- Clustering: k-means, spectral clustering, etc.
- Dimensionality reduction: PCA, feature selection, matrix factorization, etc.
- Model selection: Grid search, cross-validation, metrics
- Preprocessing: Feature extraction, normalization

statsmodels

- statsmodels contains algorithms for classical (primarily frequentist) statistics and econometrics.
- This includes such submodules as:
  - Regression models: Linear regression, generalized linear models, robust linear models, linear mixed effects models, etc.
  - Analysis of variance (ANOVA)
  - Time series analysis: AR, ARMA, ARIMA, VAR, and other models
  - Nonparametric methods: Kernel density estimation, kernel regression
  - Visualization of statistical model results
- statsmodels is more focused on statistical inference, providing uncertainty estimates and p-values for parameters. scikit-learn, by contrast, is more prediction-focused.

Import Conventions

- naming conventions for commonly used modules:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import statsmodels as sm
```

- This means that when you see np.arange, this is a reference to the arange function in NumPy.
- This is done because it's considered bad practice in Python software development to import everything (from numpy import *) from a large package like NumPy.

Jargon

- *Munge/munging/wrangling*: Describes the overall process of manipulating unstructured and/or messy data into a structured or clean form.
- *Pseudocode*: A description of an algorithm or process that takes a code-like form while likely not being actual valid source code.
- *Syntactic sugar*: Programming syntax that does not add new features, but makes something more convenient or easier to type.

Indentation

- Python uses whitespace (tabs or spaces) to structure code instead of using braces as in many other languages like R, C++, Java, and Perl.

```
for x in array:
    if x<pivot:
        less.append(x)
    else:
        greater.append(x)
```

- A colon denotes the start of an indented code block after which all of the code must be indented by the same amount until the end of the block.

- I strongly recommend using four spaces as your default indentation and replacing tabs with four spaces. By and large, four spaces is the standard adopted by the vast majority of Python programmers.

Scalar Types

- "Single value" objects (numerical data, strings, boolean values, and dates and time) are sometimes called scalar types.

Type	Description
`None`	The Python "null" value (only one instance of the `None` object exists)
`str`	String type; holds Unicode (UTF-8 encoded) strings
`bytes`	Raw ASCII bytes (or Unicode encoded as bytes)
`float`	Double-precision (64-bit) floating-point number (note there is no separate `double` type)
`bool`	A `True` or `False` value
`int`	Arbitrary precision signed integer

Control Flow: if, elif, and else

- The if statement checks a condition that, if True, evaluates the code in the block that follows.

- An if statement can be optionally followed by one or more elif blocks and a catch-all else block if all of the conditions are False:

```
if x<0:
    print('Its negative')
elif x==0:
    print('Equal to zero')
elif 0<x<5:
    print('Positive but smaller than 5')
else:
    print('Positive and larger than or equal to 5')
```

Control Flow: for loops

- for loops are for iterating over a collection (like a list or tuple) or an iterater. The standard syntax for a for loop is:

```
for value in collection:
    # do something with value
```

Functions

- As a rule of thumb, if you anticipate needing to repeat the same or very similar code more than once, it may be worth writing a reusable function.
- Functions can also help make your code more readable by giving a name to a group of Python statements.
- Functions are declared with the def keyword and returned from with the return keyword:

```
def my_function(x, y, z=1.5):
    if z > 1:
        return z * (x + y)
    else:
        return z / (x + y)
```

Functions

- There is no issue with having multiple return statements.
- If Python reaches the end of a function without encountering a return statement, None is returned automatically.
- Each function can have positional arguments and keyword arguments.
- Keyword arguments are most commonly used to specify default values or optional arguments.
- In the preceding function, x and y are positional arguments while z is a keyword argument.

Functions

- This means that the function can be called in any of these ways:

```
my_function(5, 6, z=0.7)
my_function(3.14, 7, 3.5)
my_function(10, 20)
```

- The main restriction on function arguments is that the keyword arguments must follow the positional arguments (if any).

- You can specify keyword arguments in any order; this frees you from having to remember which order the function arguments were specified in and only what their names are.

Functions

- Functions can access variables in two different scopes: global and local.
- An alternative and more descriptive name describing a variable scope in Python is a namespace.
- Any variables that are assigned within a function by default are assigned to the local namespace.
- The local namespace is created when the function is called and immediately populated by the function's arguments.
- After the function is finished, the local namespace is destroyed (with some exceptions that are outside the purview of this primer).