Programming Principles in Python (CSCI 503)

Introduction

Dr. David Koop
Why Python?
Productivity
Libraries, Libraries, Libraries
What about speed?
Administrivia

- Course Web Site
- TA: Palak Jalota (Blackboard Collaborate)
- Syllabus
  - Plagiarism
  - Accommodations
- Assignments
- Tests: 2 (Feb. 17, Mar. 29) and Final (Apr. 26)
Academic Honesty

• Do not cheat!
• You will receive a zero for any assignment/exam/etc. where cheating has occurred.
• Misconduct is reported through the university's system
• You may discuss problems and approaches with other students
• You may not copy or transcribe code from another source
Online Synchronous Course

- Lectures will be 2:00-3:15pm MW via Zoom (login via Blackboard)
  - Better for learning if you are engaged
  - Ask questions
    - Please mute your microphone if you are not asking a question or discussing
    - Please advise me of any issues
- Slides will be posted to the course website
- Recordings will be made available via Blackboard
Office Hours & Email

• Ms. Jalota's office hours will be held via Blackboard Collaborate
  - TuTh: 11:00am-2pm

• Prof. Koop's office hours will be held via Zoom via Blackboard
  - MW: 10:30-11:30am, or by appointment (Prof. Koop’s Office Hours)

• You do not need an appointment to join during scheduled office hours

• If you need an appointment outside of those times, please email me with details about what you wish to discuss and we can set up a time

• Many questions can be answered via email. Please consider writing an email before scheduling a meeting.
Course Material

• Textbook:
  - Recommended: Python for Programmers
  - Good overview + data science examples

• Many other resources are available:
  - http://www.pythontutor.com
  - https://www.python-course.eu
  - https://software-carpentry.org/lessons/
Course Material

- Jupyter Notebook: Web-based interface for interactively writing & executing Python code
- JupyterLab: An updated web-based interface that includes the notebook and other cool features
- JupyterHub: Access everything through a server
Python

• Started in December 1989 by Guido van Rossum
• “Python has surpassed Java as the top language used to introduce U.S. students to programming…” (ComputerWorld, 2014)
• Python is also a top language for data science
• High-level, interpreted language
• Supports multiple paradigms (OOP, procedural, functional)
• Help programmers write readable code, use less code to do more
• Lots of libraries for python
• Designed to be extensible, easy to wrap code from other languages like C/C++
• Open-source with a large, passionate community
Python adoption is increasing

[D. Robinson, StackOverflow blog, 2017]
Python adoption is increasing

Growth of major programming languages
Based on Stack Overflow question views in World Bank high-income countries

[D. Robinson, StackOverflow blog, 2017]
Comparison to smaller, growing technologies

[D. Robinson, StackOverflow blog, 2017]
StackOverflow Languages

% of developers who are developing with the language or technology and have expressed interest in continuing to develop with it

<table>
<thead>
<tr>
<th>Language</th>
<th>Loved</th>
<th>Dreaded</th>
<th>Wanted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rust</td>
<td>86.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TypeScript</td>
<td>67.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Python</td>
<td>66.7%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kotlin</td>
<td>62.9%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Go</td>
<td>62.3%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Julia</td>
<td>62.2%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dart</td>
<td>62.1%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C#</td>
<td>59.7%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Swift</td>
<td>59.5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JavaScript</td>
<td>58.3%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Stack Overflow Developer Survey, 2020]
StackOverflow Languages

<table>
<thead>
<tr>
<th>Language</th>
<th>% of developers who are not developing with the language or technology but have expressed interest in developing with it</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>30.0%</td>
</tr>
<tr>
<td>JavaScript</td>
<td>18.5%</td>
</tr>
<tr>
<td>Go</td>
<td>17.9%</td>
</tr>
<tr>
<td>TypeScript</td>
<td>17.0%</td>
</tr>
<tr>
<td>Rust</td>
<td>14.6%</td>
</tr>
<tr>
<td>Kotlin</td>
<td>12.6%</td>
</tr>
<tr>
<td>Java</td>
<td>8.8%</td>
</tr>
<tr>
<td>C++</td>
<td>8.6%</td>
</tr>
<tr>
<td>SQL</td>
<td>8.2%</td>
</tr>
<tr>
<td>C#</td>
<td>7.3%</td>
</tr>
</tbody>
</table>

[Stack Overflow Developer Survey, 2020]
Modes of Computation

• Python is interpreted: you can run one line at a line without compiling
• Interpreter in the Shell
  - Execute line by line
  - Hard to structure loops
  - Usually execute whole files (called scripts) and edit those files
• Notebook
  - Richer results (e.g. images, tables)
  - Can more easily edit past code
  - Re-execute any cell, whenever
Python Differences

• Dynamic Typing
  - A variable does not have a fixed type
  - Example: `a = 1; a = "abc"

• Indentation
  - Braces define blocks in Java, good style is to indent but not required
  - Indentation is critical in Python
    
    ```python
    z = 20
    if x > 0:
        if y > 0:
            z = 100
        else:
            z = 10
    ```

JupyterLab and Jupyter Notebooks

In this Notebook we explore the Lorenz system of differential equations:

\[
\begin{align*}
\dot{x} &= \sigma (y - x) \\
\dot{y} &= px - y - xz \\
\dot{z} &= -\rho x + xy
\end{align*}
\]

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

```python
from Lorenz import solve_lorenz
t, x, y = solve_lorenz(N=10)
```

![Image of JupyterLab interface with Lorenz system code and output graph]

JupyterLab Documentation

D. Koop, CSCI 503, Spring 2021
Jupyter Notebooks

- Display rich representations and text
- Uses Web technology
- Cell-based
- Built-in editor
- GitHub displays notebooks
Jupyter Notebooks

- An interactive programming environment
- Runs in your web browser
- Displays results (even interactive maps) inline
- Originally designed for Python
- Supports other languages, too
- You decide how to divide code into executable cells
- Shift+Enter to execute a cell
Notebooks in JupyterLab

• Directory view on left
• Create new notebooks using "+" button, "New" from the File menu, or Launcher window
  - Notebook originally has name "Untitled"
  - Click on "Untitled" to change the name (do this!)
• Save a notebook using the command under the File menu
• Shutting down the notebook — use Close and Shutdown Kernel
  - Web browser is interface to display code and results
  - Kernel actually runs the code: usually see messages in a console/terminal window
Notebooks in JupyterLab

• Open a notebook by going back to the file browser and clicking on it like you would in a desktop view
• Past results are displayed—does not mean they are loaded in memory
• Use "Run All" or "Run All Above" to re-execute past work
  - If you shut down the kernel, all of the data and variables you defined need to be redefined (so you need to re-run all)
  - **Watch Out—Order Matters**: If you went back and re-executed cells in a different order than they are shown, doing "Run All" may not produce the same results!
• Edit mode (green) versus Command mode (blue == **Be Careful**) 
• Learn keyboard shortcuts
Notebooks in JupyterLab

• Can write code or plain text (can be styled Markdown)
  - Choose the type of cell using the dropdown menu

• Cells break up your code, but all data is global
  - Defining a variable \( a \) in one cell means that variable is accessible in any other cell
    - This includes cells above the cell \( a \) was defined in!

• Remember Shift+Enter to execute
• Enter just adds a new line
• Use \(?<function_name>\) for help
• Use Tab for auto-complete or suggestions
JupyterLab

- More than just notebooks:
  - Text editor
  - Console
  - Custom components (Many extensions)

- Arrange multiple documents and views

- [JupyterLab Documentation](#)
Using Python & JupyterLab on Course Server

- https://tiger.cs.niu.edu/jupyter/
- Login with your Z-ID
- You will receive an email with your password
- Advanced:
  - Can add your own conda environments in your user directory
Using Python & JupyterLab Locally

- www.anaconda.com/download/
- Anaconda has JupyterLab
- Use Python 3.8
- Anaconda Navigator
  - GUI application for managing Python environment
  - Can install packages
  - Can start JupyterLab
- Can also use the shell to do this:
  - $ jupyter lab
  - $ conda install <pkg_name>
Chicago Food Inspections

• Data: Information about food facility inspections in Chicago
• Data Source: https://data.cityofchicago.org/Health-Human-Services/Food-Inspections/4ijn-s7e5/data
• Fields: Name, Facility Type, Risk, Violations, Location, etc.
Chicago Food Inspections Exploration

- Based on David Beazley's PyData Chicago talk
- YouTube video: https://www.youtube.com/watch?v=j6VSAsKAj98
- Our in-class exploration:
  - Don't focus on the syntax
  - Focus on:
    - What is information is available
    - Questions are interesting about this dataset
    - How to decide on good follow-up questions
    - What the computations mean