00_hello_python

December 10, 2023

1 Hello, Python

1.1 Introduction to Python

Data Sciences Institute, University of Toronto

Instructor: Kaylie Lau | TA: Tong Su

December 2023

2 Contents:

3 Welcome!

The course introduces the Python programming language starting from its basics, working up to loading, manipulating, and visualizing real datasets with popular data science libraries. You will learn how to design functions, repeat code using loops, store data in lists, test and debug your code, and manipulate data using various data analysis and visualization tools such as `numpy`, `pandas`, `matplotlib`, `seaborn`, and `plotly`.

3.1 Land Acknowledgement:

We wish to acknowledge this land on which the University of Toronto operates. For thousands of years it has been the traditional land of the Huron-Wendat, the Seneca, and the Mississaugas of the Credit River. Today, this meeting place is still the home to many Indigenous people from across Turtle Island and we are grateful to have the opportunity to work on this land.

4 Course Logistics

4.1 Course contacts

Instructor: Kaylie Lau (she/her)
Email: kaylie.lau@mail.utoronto.ca

TA: Tong Su (she/her)
Email: tong.su@mail.utoronto.ca

4.2 Pre-Course Work

Prior to the first class please: 1. Create a Google account that can use Google Colab: 1. Go to https://colab.research.google.com/. In the upper left corner, click File, then New Notebook 2. Enter `!python --version` in the code cell, then hit ctrl+enter to run the cell and confirm that your Python version is 3.6 or above. * If you are having issues with the set-up, the TA will be available to help with this **Monday 11 December from 5:30 PM - 6:00 PM and 8:30 PM - 9:00 PM**. 2. Complete the pre-course survey: https://forms.gle/tn8a5aFC4ZNUfvQs5

4.3 Design

The course runs synchronously over Zoom. It consists of eight classes over the span of two weeks. Classes are 6:00 PM - 8:30 PM EDT on weekdays, and 9:00 AM - 11:30 AM EDT on Saturday. Being mindful of online fatigue, there will be one or two breaks during each class where students are encouraged to stretch, grab a drink and snacks, or ask any additional questions.

Tutorial sessions with a TA will also be offered over Zoom. These will take place from 5:30 PM - 6:00 PM EDT and 8:30 PM - 9:00 PM on weekdays, and 8:30 AM - 9:00 AM EDT and 11:30 AM - 12:00 PM EDT on Saturday.

4.4 Schedule

Schedule is tentative and may be modified as needed. Learners will be notified of schedule changes. * **Day 1 (Monday 11 December, 6:00 PM - 8:30 PM):** Getting Started I (Introduction; Python fundamentals) * **Day 2 (Tuesday 12 December, 6:00 PM - 8:30 PM):** Getting Started II (Python fundamentals) * **Day 3 (Wednesday 13 December, 6:00 PM - 8:30 PM):** Dealing with Reality (Control flow using conditionals and loops; Lists, tuples, sets, and dictionaries) * **Day 4 (Thursday 14 December, 6:00 PM - 8:30 PM):** In/Out (Modules; Working with files; Object-oriented programming) * **Day 5 (Saturday 16 December, 9:00 AM - 11:30 AM):** Doing More with Data I (`numpy`) * **Day 6 (Monday 18 December, 6:00 PM - 8:30 PM):** Doing More with Data II (`pandas`) * **Day 7 (Tuesday 19 December, 6:00 PM - 8:30 PM):** Case Study Speaker * **Day 8 (Wednesday 20 December, 6:00 PM - 8:30 PM):** Visualizing Data (`matplotlib`; `seaborn`; `plotly`)

4.5 Prerequisites

Learners are expected to know how to operate a computer and are also expected to be familiar with the parts of a data table or spreadsheet, summary statistics, and basic data visualizations. No prior programming knowledge is required.

4.6 Expectations

The course is a live-coding class. Learners are expected to follow along with the coding in their own Python notebooks. Learners should be active participants while coding and are encouraged to ask questions throughout. Although slides will be available, they should be referenced before or after class, as class will be dedicated to coding with the instructor.

4.6.1 Technology requirements

- Learners must have an internet connection and a computer to participate in online activities
- Learners must have a Google account that can use Google Colab

4.7 Policies

- **Accessibility:** We want to provide an accessible learning environment for all. If there is something we can do to make this course more accessible to you, please let us know.
- **Course communications:** Communications take place over email. Please include "DSI-Python" or similar in the subject line, e.g. "DSI-Python: pandas question"
- **Camera:** Keeping your camera on is optional.
- **Microphone:** Please keep microphones muted unless you need to speak. Please indicate your name before speaking as some Zoom configurations make it hard to tell who is talking!
- **Assessment:** There will be homework which is **not** graded, but highly recommended, and there will be three assignments which **are** graded.

4.8 All course materials can be found on GitHub: https://github.com/UofT-DSI/03-python/tree/2023-course-materials

4.8.1 Assignments

Class attendance: To ensure everyone actively participates in class activities, attendance is mandatory and will be monitored. If you are unable to attend class, it is your responsibility to make up the work that was covered.

Format	Details	Submission Instructions
Assignment 1	Due on **Sunday December 17 at 11:59pm**	Submission Form for Assignment 1. Upload your code file (LASTNAME_FIRSTNAME_code.ipynb).
Assignment 2	Due on **Saturday December 22 at 11:59pm**	Submission Form for Assignment 2. Upload your (1) code file (LASTNAME_FIRSTNAME_code.ipynb), (2) original data file (LASTNAME_FIRSTNAME_orig.csv), and (3) processed DataFrame file (LASTNAME_FIRSTNAME_proc.csv)

4.9 Key Texts

- Gries, P., Campbell, J., Montojo, J., & Coron, T. (2017). *Practical programming: An introduction to computer science using python 3.6.*
- Adhikari, DeNero, and Wagner, (2021). *Computational and Inferential Thinking: The Foundations of Data Science.*
- Thomas, R. (2021). *Avoiding Data Disasters.* https://www.fast.ai/2021/11/04/data-disasters/
- Boykis, V. (2019). *Neural nets are just people all the way down.* https://vicki.substack.com/p/neural-nets-are-just-people-all-the

5 Google Colab

Jupyter Notebooks are web applications that lets us combine pieces of executable code, text, images, and visualizations in a single document, or notebook. *Google Colab* is a Jupyter Notebook environment hosted on the cloud – that is, on Google's servers. Google Colab provides a standardized notebook environment, with common data science libraries already installed.

Sections of a notebook are called cells. Cells can be run independently of each other, and in any order.

5.1 Check your Python version

Try adding and running your first code cells.

1. Add a cell by clicking the **+ Code**.
2. Enter the following: `!python --version`.
3. Press `ctrl` + `enter` to run the cell.

You should see Python 3.6 or higher.

```
[ ]: !python --version
```

```
Python 3.10.12
```

5.2 Adding text

Notebooks use markdown, a markup language, for formatting text. IBM publishes a Jupyter-specific markdown cheat sheet.

To add a text cell: 1. Click **+ Text** 2. Add some text. When you are done editing, press `ctrl` + `enter` to run the cell and format your text.

5.3 Writing Code

Google Colab provides an interactive environment for writing and executing code in a Jupyter Notebook format. Follow these steps to write and run your code:

1. **Writing Code Cells:**
   - Use code cells to write your Python code. Click on the '+ Code' button to create a new code cell.
   - Type or paste your Python code within the code cell.
2. **Running Code Cells:**
   - To execute a code cell, click the `Play` button on the left of the cell or use `Shift + Enter` as a shortcut.
   - Output, if any, will be displayed below the cell after execution.
3. **Keyboard Shortcuts:**
   - Utilize keyboard shortcuts for faster execution:
     - `Shift + Enter`: Run the current cell and move to the next cell.
     - `Ctrl + Enter`: Run the current cell and stay on the same cell.
     - `Alt + Enter`: Run the current cell and insert a new cell below.
4. **Code Assistance and Auto-completion:**

- Google Colab offers auto-completion suggestions to assist while writing code. Press `Tab` for suggestions.

5. **Saving and Renaming:**
   - Save your work by clicking on `File` > `Save` or `File` > `Save a copy in Drive`.
   - Rename the notebook by clicking on the notebook name at the top and entering a new name.
6. **Managing Cells:**
   - Use the '+ Code`or`+ Text' buttons to add new code or text cells respectively.
   - Drag and drop cells to rearrange the order.
   - Click on the three vertical dots (`...`) in the top right of a cell for more options like deleting or moving the cell.

[ ]:

5.4 Instructions for Accessing and Uploading Data to Google Colab

Follow these steps to access and upload the `data` folder from your GitHub repository to Google Colab:

1. **Access GitHub Repository:**
   - Go to your GitHub repository where the `data` folder is located: https://github.com/UofT-DSI/03-python/tree/2023-course-materials
   - Download the repository as a ZIP file and then extract the ZIP file to access the `data` folder on your local machine.
2. **Upload `data` folder to Google Drive:**
   - Upload the `data` folder into this directory: `/content/drive/MyDrive/Colab Notebooks/data`
3. **Mount Google Drive to access `data` folder in Google Colab:**
   - Open Google Colab in a browser and create/open a notebook.
   - Mount Google Drive using the following code snippet:

```
from google.colab import drive
drive.mount('/content/drive')
```

6 Course Tour

6.1 Getting Started

What are the core features of Python?

6.1.1 Topics

- Data types and variables
- Expressions and statements
- Functions
- Dealing with errors
- Comments and readability

6.1.2 What is Python?

Python is a general-purpose programming language first released in 1991. It has since become a popular language for data science, thanks to an enthusiastic community and a large ecosystem of code libraries and tools that make it easier to perform common tasks throughout the data science life cycle.

```
# create a variable
degrees_celsius = 25

# convert to Fahrenheit
(9 / 5) * degrees_celsius + 32

# make it a function
def c_to_f(degrees_c):
    return (9 / 5) * degrees_c + 32
```

6.2 Dealing with Reality

What tools does Python give us for working with collections of values and for writing programs that change their behaviour based on data?

6.2.1 Topics

- Logic
- Conditionals
- Iteration
- Sequences and collections of values

```
numbers = [2, 52, 18, 27, 5, 1937]

for number in numbers:
    if number % 2 == 0:
        print('Even')
    else:
        print('Odd')
```

6.3 In/Out

How can we bring in outside code and data to accomplish tasks? How can we get data out of our programs?

6.3.1 Topics

- Modules
- Working with files
- Object-oriented programming

```
import os

for f in os.listdir('folder'):
```

```
    if f.endswith('.csv'):
        # process file
```

6.4 Doing More with Data

How can we work with real data for analysis?

6.4.1 Topics

- `numpy` and `pandas`
- Loading tabular data
- Exploring data
- Wrangling data
- Reshaping and combining datasets

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
import pandas as pd

neighbourhoods = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/data/
 ↪neighbourhood-profiles-2016-140-model.csv')
neighbourhoods[['Characteristic', 'City of Toronto']].head()
```

[ ]:

	Characteristic	City of Toronto
0	Neighbourhood Number	NaN
1	TSNS2020 Designation	NaN
2	Population, 2016	2,731,571
3	Population, 2011	2,615,060
4	Population Change 2011-2016	4.50%

6.5 Visualizing Data

How do we create visualizations in Python?

6.5.1 Topics

- `matplotlib`, `seaborn` and `plotly`
- Bar charts, histograms, scatterplots, line charts
- Faceting and layering

[ ]:
```
neighbourhoods = (neighbourhoods
                  .query('Characteristic == "Population, 2016"')
                  .drop(columns=['_id',
                                 'Category',
                                 'Topic',
                                 'Data Source',
```

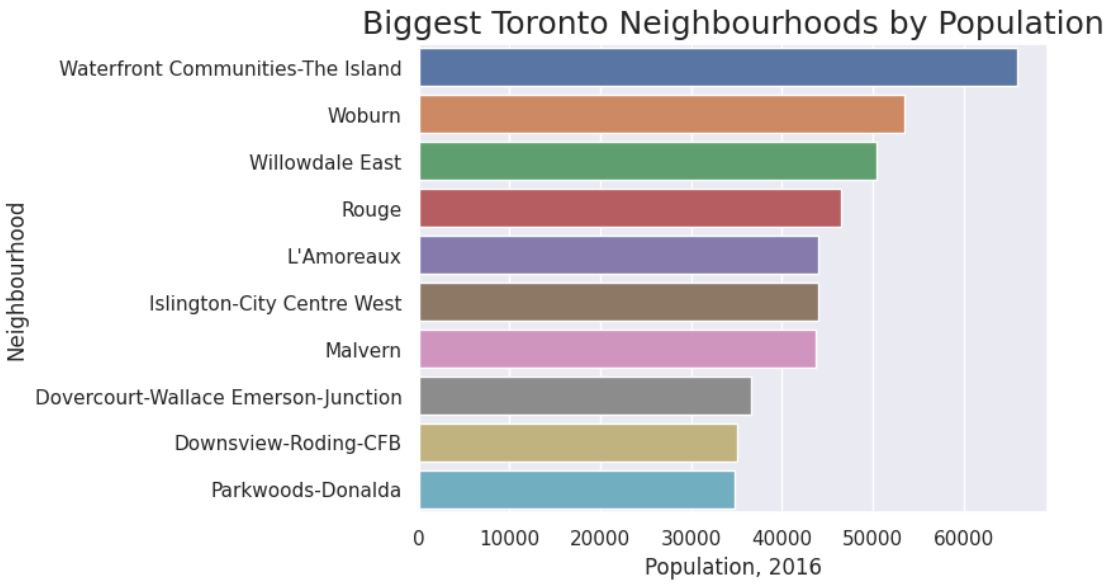
```
                                    'City of Toronto'])
                  .melt(id_vars=['Characteristic'],
                        var_name='Neighbourhood',
                        value_name='Population, 2016'))
neighbourhoods['Population, 2016'] = neighbourhoods['Population, 2016'].str.
 ↪replace(',', '')
neighbourhoods['Population, 2016'] = pd.to_numeric(neighbourhoods['Population,␣
 ↪2016'])
neighbourhoods = neighbourhoods.sort_values(by='Population, 2016',␣
 ↪ascending=False)
```

[ ]:
```
import seaborn as sns

sns.set_theme()

(sns.barplot(data=neighbourhoods.head(10),
             y='Neighbourhood',
             x='Population, 2016')
    .set_title('Biggest Toronto Neighbourhoods by Population',
               fontdict={'fontsize': 18}))
```

[ ]:
```
Text(0.5, 1.0, 'Biggest Toronto Neighbourhoods by Population')
```



6.6 Do It Again

How can we make our work easier to reproduce and/or maintain?

6.6.1 Topics

- Environments
- Testing and debugging
- Moving beyond notebooks

[ ]:
```
import doctest

def c_to_f(degrees_c):
    '''
    Convert degrees Celsius to Fahrenheit
    >>> c_to_f(0)
    32.0
    >>> c_to_f(10)
    50.0
    '''
    return (9 / 5) * degrees_c + 32

doctest.testmod()
```

```
PYDEV DEBUGGER WARNING:
sys.settrace() should not be used when the debugger is being used.
This may cause the debugger to stop working correctly.
If this is needed, please check:
http://pydev.blogspot.com/2007/06/why-cant-pydev-debugger-work-with.html
to see how to restore the debug tracing back correctly.
Call Location:
  File "/usr/lib/python3.10/doctest.py", line 1501, in run
    sys.settrace(save_trace)
```

[ ]:
```
TestResults(failed=0, attempted=2)
```

7 Let's get started!