JOHNS HOPKINS
CAREY BUSINESS SCHOOL

Python for Data Analysis

BU.510.615.81

Tuesdays 6:00 – 9:00 PM

1/23/2024 - 3/12/2024

Spring I – 2024

Baltimore MD – 203A/203B

Instructor
Mohammad Ali Alamdar Yazdi

Contact Information
yazdi@jhu.edu

Required Texts & Learning Materials
There is no required textbook for this course. Instead, there are an elaborate set of handouts that are prepared to fit the scope of this course. These references will be provided to you on a weekly basis.

Recommended Texts
Students may find the following references quite useful:

1- Python online documentation (https://docs.python.org/3/)
   This is Python's online documentation. It provides a very thorough and organized reference for most of the Python questions; specially Python set-up and syntax.
2- Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter, Wes McKinney, O'Reilly Media, 3rd edition, 2022
3- Effective Pandas: Patterns for Data Manipulation, Matt Harrison, Independently Published, 2021
4- Intro to Python for Computer Science and Data Science, Paul J. Deitel, Harvey Deitel, Pearson, 1st edition, 2019
5- Introduction to Computation and Programming Using Python: With Application to Understanding Data, John Guttag, The MIT Press, 2nd edition, 2016
6- Introduction to programming in Python; Robert Sedgewick, Kevin Wayne, Robert Dondero, Addison-Wesley Professional; 1st edition, 2015.
   Another excellent reference for beginners. Contents of this book is available via https://introcs.cs.princeton.edu/Python/home, as well.

Course Description
This is an introductory course in using Python for analytical purposes. Python (www.Python.org) is a general-purpose cross-platform programming language that has a strong presence in the diverse areas of analytics. This course will provide a pragmatic and hands-on introduction to the fundamental aspects of Python programming language with a focus on data exploration, analysis, and driving insights from data. Additionally, towards the end of the semester, students will be exposed to using Python for introductory optimization and machine learning. Class time will be used for short overview lectures followed by analysis of worked-out examples and in-class coding exercises. As the course progresses, students will learn to work with libraries such as statistics, random, numpy, scipy, pandas, matplotlib, and seaborn. By the end of this course, students should be able to start writing useful Python programs on their own or to understand and modify Python code written by others.

This course does not assume any prior coding experience. If you have an extensive knowledge of Python, you might experience significant repetition.

Prerequisite(s)
Statistical Analysis (BU.510.601)

Learning Objectives
By the end of this course, students will be able to:

1. Readily use the Python programming language
2. Work comfortably with various Python data and control structures
3. Gain functional knowledge of Python data analysis libraries

4. Learn about using Python for optimization and machine learning

To view the complete list of the Carey Business School's general learning goals and objectives, visit the Carey website.

Attendance
Class attendance is expected and students are responsible for all material covered in class and via the posted videos.

Classroom Protocol
I will come to class prepared each day and I expect you to do the same. Please come to class on-time, follow the suggested social distancing protocols and all the provided health guidelines. Upon arrival please turn off your cell-phone. While you are coding on your computer, please refrain from visiting any websites.

Stackoverflow, Github, and other programming resources:
Almost all programmers use the internet (Stackoverflow, github, etc) to find solutions to programming problems. I'm not asking you to ignore the online resources at your disposal when programming for this course; however, I would like to bring the following points to your attention:

1- Do not use any external resources during in-class exercises.
2- The value you get from this course is up to you and I would like to see every student learn Python programming for data analysis to the best of their abilities. I want you to develop programming skills by practicing, making mistakes, and debugging your own mistakes. There is no other alternative for honing your programming skills. That is how everyone develops programming skills.
3- Force yourself to understand the programming exercises better and engage your analytical mind by attempting to solve all problems by yourself before looking up external resources. Understand every code you write; and be ready to explain your code and defend it.
4- Do not copy and paste code directly from other resources. This count as cheating.
5- If you are stuck and hints from me and your TA are not helpful and end up going to the external resources for help, please do not just copy the code; please make sure to study and understand the external solution.

Assignments

Assignment	**Group or Individual**	**Learning Objectives**	**Weight**
Assignment	Individual	1, 2, 3, 4	30%
Attendance and Participation	Individual	1, 3	15%
Course project	Group	1, 2, 3, 4	25%
Final Exam	Individual	1, 2, 3, 4	30%
Total			**100%**

Assignment
Every week students will have an individual programming assignment. These weekly assignments targeting using python in different business contexts are used to solidify each week's learnings and prepare students for the next week's class. Each assignment is graded based on the following criteria:

Program correctness and delivery: 40%
Program pragmatics: 40%
Addressing and analyzing the question: 20%

Class participation:
Students are expected to come to class fully prepared and ready to participate in class discussions. Pre-recorded videos, problem descriptions, case studies, and book chapters are assigned in advance. Cold call is used if necessary. The class participation grade will depend on the student's contribution to group learning, which can take the form of good questions, answers, observations, correcting coding mistakes, and/or shared experiences. Class participation is graded based on frequency and quality of participation.

Course Project
Students are required to form groups of 3-4. Students are required to submit the project report along with the project code by the end of week 8. Projects must be students own work. Your code will be compared to datasets

and codes available on platforms such as github. So please do your own work and submit your own work. More details will be provided in a separate document.

Final Exam

There are two types of questions in the final exam:

Python syntax questions: These questions test students understanding of Python built-in capabilities and syntax.

Scripting questions: Students will be given a set of problems with accompanying dataset. They are expected to write and submit their script along with their analysis. Each submitted script is graded based on the following criteria:

Program correctness and delivery: 40%
Program pragmatics: 40%
Addressing and analyzing the question: 20%

Code Assessment

Every submission will be assessed on the following criteria:

- **Program Correctness and delivery (40% of the grade):**
  This covers on-time delivery and program *syntax* and *semantics*.
  - **On-time delivery**
    No late submission will be accepted
  - **Program Syntax**
    You are expected to use python language *correctly*; i.e. follow python's programming language rules (called syntax) properly and have a synthetically correct program that runs free of any warnings/ errors.
  - **Program Semantics**
    It is not enough to have a program that will run. You want a program that will run and produce the correct result; i.e. the meaning of the program must be right and it must do what you want it to. The meaning of a program is called semantics.
- **Program Pragmatics (40% of the grade)**
  As Guido Van Rossum, the creator of Python, has noted "*Code is read more often than it is written*". A well-written program has "style" and is coded in a manner that will make it easier for others to read, to understand, and to reuse. Program pragmatics (style) has two related components:
  - **Program Documentation (20% of the grade)**
    Documenting your program will not only help you in understanding and reusing your own code, but also it will help others understand your code. At the very minimum, a well-documented code:
    - Is well-commented
    - Follows python's PEP-8 coding standard
    - Lists all the references used
  - **Program Readability (20% of the grade):**
    You are expected to use python language features *well*. A well-written program is easier for people to read and understand. The computer is completely ignorant to such things, but to a human reader, program readability is extremely important.
- **Business Problem Analysis (20% of the grade)**
  In this course we use python codes for the sake of analyzing business situations. A correct python code will provide an output; it is your responsibility to turn the computer-generated outputs into presentable reports leading to managerial insights. Effective communication of the coding outputs is equally as important as writing the code itself.

The following rubric will be used in grading your submissions

	Unsatisfactory (0%) Note: Any Unsatisfactory criteria will result in a zero grade for the assignment, and no further grading will be done	Satisfactory (70%)	Good (90%) Must also include all “Satisfactory” criteria, and no “Unsatisfactory” criteria	Excellent (100%) Must also include all “Good” criteria, and no “Unsatisfactory” criteria
Program Correctness and delivery	• Completed less than 75% of the requirements. • Not delivered on time or not in correct submission format specified in the assignment • Does not execute due to syntax errors. • Does not execute due to runtime errors (endless loop, crashes, etc.) • Does not have correct semantics (does something other than the requirements)	• Completed at least 75% of the requirements. • Delivered on time, and in correct format • Executes without any errors. • Uses proper libraries	• Completed between 80-99% of the requirements. • Executes without any warnings.	• Completed 100% of requirements
Program Pragmatics	• No programmer name, JHED ID, email, or assignment title included • Poor use of white space (indentation, blank lines) making code hard to read. • Disorganized and messy • Not following PEP 8 standards • Not including references • No documentation included. • A difficult and inefficient solution.	• Includes name, JHED ID, and email and assignment title. • White space makes program fairly easy to read. • Organized work. • Good use of variables. • Basic documentation has been completed including a summary of requirements. • Purpose is commented for each function. • A logical solution that is easy to follow but it is not the most efficient.	• Good use of white space. • Organized work. • Good use of variables and constants • Purpose is noted for each function and control structure. • Solution is efficient and easy to follow (i.e. no confusing tricks).	• Excellent use of white space. • Creatively organized work. • Excellent use of variables and constants. • Specific purpose is noted for each function, control structure, input requirements, and output results • Solution is efficient, easy to understand, and maintain
Business Problem Analysis	• No business analysis provided. • Python output have submitted without any further analysis • Poor report formatting	• A basic business analysis was provided • Fair managerial insights offered	• A good business analysis leading to meaningful managerial insights provided • Organized formatted report submitted	• An excellent analysis done • Report complemented by fact/figures provided from other references

Here is an example of how one assignment grading might be done:

	Unsatisfactory (0%)	Satisfactory (70%)	Good (90%)	Excellent (100%)	Weight	Score	
Program Correctness and delivery			☒		40%	90%x40%=36%	
Program Pragmatics				☒	40%	100%x40%=40%	
Business Problem Analysis	☒				20%	0%x20%=0%	
						Grade:	36%+40%+0%=76%

Students with limited programming background should expect to spend 12-15 hours per week on this course. This time commitment may reduce as semester progress and you get more familiar with coding.

Grading
The grade of A is reserved for those who demonstrate extraordinarily excellent performance as determined by the instructor. The grade of A- is awarded only for excellent performance. The grades of B+ and B are awarded for good performance. The grades of B-, C+, C, and C- are awarded for adequate but substandard performance. The grades of D+, D, and D- are not awarded at the graduate level (undergraduate only). The grade of F indicates the student failure to satisfactorily complete the course work.

Tentative Course Calendar
Instructors reserve the right to alter course content and/or adjust the pace to accommodate class progress. Students are responsible for keeping up with all adjustments to the course calendar.

Week	Topic	Reading	Due

1	**Python fundamentals I**
	• Why Learn Python?
	• Basics of *JupyterLab*
	• Python Basic Data types
	  ○ int, float, str, Boolean data types
	  ○ variables and identifiers
	• Control Statements
	  ○ Conditionals
	  ○ Iterations
	• In-class coding exercise		

2	**Python fundamentals II** • Data types ○ Sequential and non-sequential data types • List comprehensions • In-class coding exercise		Homework 1
3	**Python data analysis capabilities & Intro to Pandas** • Python Functions • An overview of data analytics libraries • Intro to Pandas library • Course project introduction • In-class coding exercise		Homework 2
4	**Data exploration and cleaning using Pandas library** • Data aggregation/summarization using Pandas • Cleaning and handling missing data • Using lambda functions in pandas • In-class coding exercise		Homework 3

Week	Topic	Reading	Due
5	**Advanced Operations using Pandas** • Merge types and operations using Pandas • Str and dt operations using Pandas • In-class coding exercise		Homework 4

6	**Python data visualization capabilities**
	• Introduction to Data Visualization
	  ○ Matplotlib
	  ○ Seaborn
	• A comprehensive example of data exploration and visualization exercise
			Homework 5

7	**Select topics in Python** A subset of topics selected from the following list • Numpy • Application Programming Interface (API) • Class and object-oriented programming in Python • Git and Github • Openpyxl library • Interactive plots with Plotly • Outlier detection • Introductory Machine-Learning using Python • Introductory optimization using Python		Homework 6
8	Final Exam		Final Project