Syllabus and Course Information

BU MET CS-521 A2 (Fall 2022)

Information Structures with Python

Welcome to CS-521!!!

This course presents an effective approach to learn Python. With extensive use of graphical illustrations, it will build understanding of Python and its capabilities by learning through many simple examples and analogies. The class will involve active student participation, discussions, and programming exercises. This approach will help build a strong foundation in Python programming that can be used effectively in real-job situations and future courses.

Instructor: Professor Eugene Pinsky
Computer Science Department,
Metropolitan College, Boston University
1010 Commonwealth Avenue Room 327
Boston, MA 02215
email: epinsky@bu.edu

Course Times: Wed 8:00 – 10:45 am
Room: CAS 426
Office Hours: TBA

Teaching Assistants:
Gaurav Tiwari: gtiwari@bu.edu
Snigda Dubey: sndubey@bu.edu

Prerequisites

Familiarity with at least one programming language. Understanding of key language constructs and methods. Ability to formulate quantitative information symbolically and numerically.

BU Community COVID-19 Public Health Policies

All students returning to campus will be required to be vaccinated against COVID-19, and upload information about their status (including applications for a medical or religious exemption or an extension) to the Patient Connect portal. In addition to the vaccine requirement, students must follow all other safety protocols, including the face covering policy, and screening, contact tracing, and testing requirements. At the beginning of each class you will be asked to show a green Healthway compliance badge on your mobile device to the instructor, and wear your face mask over your mouth and nose at all times.

Course Learning Outcomes

1. learn to use Python programming language constructs to implement a variety of analytical and computational methods (searching and sorting)
2. understand trade-offs of different Python methods and data structures in computation
3. apply acquired skills in diverse settings by completing a course project on a topic chosen by a student
4. present both symbolic and visual results on the project
5. learn advantages and limitations of using Python

To accomplish this goal, course materials are divided into a set of mini-modules corresponding to particular topic(s). These mini-modules will typically include the following:

(1) course material with many examples
(2) self-test questions
(3) sample programming problems including typical Python job interview questions (collected from various sources in the internet)

HUB Learning Outcomes

I. Quantitative Reasoning II

1. The focus of this course is to learn a range of analytical and computational method using Python to be able to frame and formulate quantitative problems using Python.

2. Students apply quantitative tools using many of the Python built-in objects and methods.

3. Students test arguments by learning unit testing and extensive visualization.

4. Students express quantitative information using user-created objects and methods tailored for specific tasks

5. Students engage in quantitative reasoning and in problem solving through readings, tutorials and exercises, lab work, and project-based learning (for example, implementing text editor using different Python objects to understand the trade-offs and limitations both in terms of resource utilization and complexity of implementation.

II. Creativity/Innovation

1. promote creativity by showing the use of Python in both technical and non-technical setting.

2.in-class discussion on multiple strategies to common algorithms

III. Critical Thinking

1. learn to formulate formal algorithm description in Python

2.in-class discussion on different algorithms for problem solving

Course Materials:

(1) Required Textbook: The Practice of Computing Using Python, by W. Punch and R. Enbody, 3-rd edition, Pearson Publishing, ISBN 978-0-13-437976-0

(2) Course notes (from the course website)
(3) Pre-recorded mini-module
(4) Python Programming Environment – we will be using Spyder IDE (Integrated Development Environment) and Anaconda Python Distribution

Additional Resources:

There are many on-line resources available. This is a partial list:

1. http://www.pythontutor.com/visualize.html - this website is very useful and allows to run simple Python programs and visualize the execution. Many of the illustrations in the course notes were generated using this website.
2. https://docs.python.org/2/tutorial - an official Python tutorial
3. https://www.tutorialspoint.com/python - a detailed tutorial with many simple examples
4. https://www.learnpython.org - free, interactive tutorial
5. https://www.python.org/community/sigs/current/edu-sig/ - contains links to learning resources, including free books

Teaching Approach and Goals

I am a strong believer in learning by using many illustrated examples. These examples will help us build the fundamental understanding of Python and how to use it to solve real problems. Many exercises presented in the course will help you develop skills that are needed to use Python effectively in your workplace and more advanced courses.

Homework, Grading and Exams:

Final	30%
Project	20%
Homework	35%
Quizzes	15%

There are six 30 minute quizzes. The final is 120 minutes. All exams are multiple choice and will be done in the blackboard.

This is a programming class and it is essential that students have practice. Most homework assignments will consist both programming problems from the textbook.

Quizzes and the final are closed book and will consist of typical Python questions that one can expect at a job interview

The project is open ended and the topics can be chosen by students. In this project, students will frame and solve problems using quantitative capabilities of Python. Students will present their projects on the last day of the course.

The goal of the course to learn Python programming and understand its limitations and capabilities. This will be accomplished by many illustrated examples, active student participation and discussion.

Course Outline:

The course consists of 6 modules. Each module is one week and consists of a related set of topics (mini-modules). All mini-modules are pre-recorded and available in the blackboard. All (weekly) exercises are from the textbook. Due dates for the homework will be indicated explicitly. No late homework will be accepted.

Module 1

Topics: introduction to computing, program structure, programming environment, input/output, variable scopes and modules

Module 2

Topics: data types, expressions, types and type casting, numerical data types, arithmetic, logical, assignment and relational operators, Boolean expressions, operator precedence, hashing, mutability, Python ranges, object copying in Python.

Module 3

Topics: strings and text manipulation, indexing and slicing, collections, control flow, iterations, files and files manipulation.

Module 4

Topics: collections: lists, tuples, sets, dictionaries, comprehension, indexing and slicing, comprehension in mutable collections, applications (stacks, ques and linked lists), searching and sorting

Module 5

Topics: exception handling, functions, parameter passing, recursive functions, lambda function and functional programming

Module 6

Topics: objects and classes, class constructors, attributes and methods, instance and static variables, data encapsulation, overloading, inheritance and polymorphism, abstract classes

Module 7

Project presentations and review

About the Instructor:

Eugene Pinsky received his B.A. in Mathematics from Harvard University and his Ph.D. in Computer Science from Columbia University. He has taught extensively both in academia and industry. His research interests are in performance analysis and computational algorithms in data science and machine learning with emphasis on computational finance and programmatic advertising.