Course #: Practical Programming with Python

Course Description & Syllabus

Name of College or University
Semester Year

1. Overview

Title: Practical Programming With Python
Credits/Units: 3-credit hours or 4-credit hours with a lab.
Pre-requisites: Introduction to Computing.

Institution LMS access: https://lms.univ-or-college.edu

Recitation:

1. **Tuesday, 8:00 AM – 8:50 AM ET, BLDG XYZ (Videotaped)**

Teaching Staff:

Prof. Best Faculty
faculty-email@xyz.edu
Office Number, +1-555-555-5555
Office hours: Tuesday, 3-4 pm (time zone)

TAs typically hold office hours in building/room. The TA office hours are posted on the LMS:

- TA Name<ta-email@xyz.edu>
- TA Name<ta-email@xyz.edu>

2. Course Description

Students learn the concepts, techniques, skills, and tools needed for developing programs in Python. Core topics include types, variables, functions, iteration, conditionals, data structures, classes, objects, modules, and I/O operations. Students get an introductory experience with several development environments, including Jupyter Notebook, as well as selected software development practices, such as test-driven development, debugging, and style. Course projects include real-life applications on enterprise data and document manipulation, web scraping, and data analysis.

3. Course Goals

In this course, students gain hands-on experience solving real-world problems by completing programming projects in Python focused on the language fundamentals, software development practices, data manipulation and analysis, web scraping, and automatic document processing. Specifically, students are exposed to realistic software development projects, real-world data, and scenarios in order to learn how to:

1. Explain and use fundamental constituents of the Python programming language to solve a variety of computational problems.
2. Read data stored in different data formats, such as CSV, JSON, and XML, or database systems, such as MySQL or MongoDB, transform it, and persist the transformed data in a different file format or a database system using Python.
3. Recognize and utilize well-established software development practices.
4. Process common office document formats and obtain information from the web or publicly available APIs using Python.
5. Analyze and visualize real-world data using Python data science libraries.

Through this process, we aspire for our students to become sophisticated, independent, and resilient problem solvers who are able to overcome challenges and learn.

4. Learning Outcomes

In this project-based course, we have project and conceptual learning objectives.

The conceptual learning objectives (LOs) are the following. Students will be able to:

1. Explain the core building blocks of a programming language, such as variables, user-defined, and built-in functions.

2. Explain the flow control techniques (iteration, conditionals).

3. Describe the fundamental programming data structures and their implementation in Python (lists, sets, tuples, dictionaries).

4. Discuss the object-oriented programming paradigm and its purpose.

5. Discuss fundamental software development lifecycle and practices (e.g., top-down design, test-driven development, object-oriented analysis, and design).

6. Describe common document formats and considerations related to web scraping and office document processing and manipulation.

7. Explain the fundamental principles of database design, common data formats, and programming interfaces.

8. Discuss and compare different data storage, access solutions (flat files, SQL, NoSQL, key-value stores), and application programming interfaces (APIs).

9. Discuss data loading, cleaning, manipulation, analysis, and visualization.

4.1. Project Learning Objectives

The project learning objectives (LOs) are the following. Students will be able to:

1. Write small Python scripts using variables, built-in and user-defined functions.

2. Write more advanced scripts using conditionals and iteration (control flow).

3. Employ Python data structures, such as lists, dictionaries, sets, and tuples.

4. Use libraries implemented in an object-oriented fashion and interact with classes and objects imported from those libraries.

5. Employ fundamental software development practices (e.g., top-down design, test-driven development, style, linting, pep8, documentation).

6. Scrape information from the web or query publicly available APIs and extract data from common office document formats, transform it and present it using a different appropriate format.

7. Contrast among and experiment with different data storage and access solutions (CSV, JSON, XML, SQL, NoSQL, key-value stores).

8. Analyze and visualize real-world data using Python data science libraries.

4.2. Course organization

Your participation in the course will involve several forms of activity:

1. Reading the conceptual content for each unit.
2. Completing the graded weekly assessments after each unit.
3. Complete projects, which are hands-on training and automated feedback.

5. Getting help

Students are encouraged to ask questions about content and projects through the Q & A forum. The course link for the forum is:

http://Q&A-forum.edu/

6. Policies

Working Alone on Projects

Projects that are assigned to single students should be performed individually.

Handing in Projects

All assessments are due at 11:59 PM ET (one minute before midnight) on the due dates specified on the Sail() Platform. All hand-ins are electronic.

Appealing Grades

After each project module is graded, you have seven calendar days to appeal your grade. All your appeals should be provided by email to the professor.

7. Assessment

Inline activities ("Learn by Doing" and "Did I Get This"), which are available in most pages in the OLI course, are simple, non-graded activities to assess your comprehension of the material as you read through the course material. You are advised to complete all of the inline activities before proceeding through to the next page or module. If you missed many of the activities, it is recommended that you review the material again.

The conceptual units consist of modules of content on OLI, each week has a Checkpoint Quiz that you must complete before the deadline posted on OLI. Each weekly Checkpoint Quiz will be worth N% of your total grade. It is your responsibility to ensure that the quiz is submitted prior to the deadline. You will have only a single attempt to complete each Checkpoint Quiz on OLI.

This course includes several individual projects. Each project module has to be completed based on the deadlines posted on Sail(). The write-up required to complete each project module is available on Sail(). Each module has a submission process that is specific to the project module that is due. It is the students' responsibility to make sure that all project work is completed and that the project module is submitted prior to the deadline. Students typically have multiple attempts to submit the project module on Sail().

Type	Number	Weight
Conceptual Content Quizzes	8	XX%
Projects	8	YY%
Total Grade		100%

8. Cheating

We urge each student to carefully abide by the course policy on academic integrity, which outlines the policy on cheating, plagiarism, or unauthorized assistance. It is the responsibility of each student to produce her/his own original academic work. Collaboration or assistance on academic work to be graded is not permitted unless explicitly authorized by the course instructor. Each unit checkpoint quiz or project module submitted must be the sole work of the student turning it in. Student work on the cloud is logged, submitted work will be closely monitored by automatic cheat checkers, and students may be asked to explain any suspicious similarities with any piece of code available. The following are guidelines on what collaboration is authorized and what is not:

What is cheating?

1. Sharing a solution, code, or other electronic files by either copying, retyping, looking at, or supplying a copy of any file. Copying any solution, code from the internet (stackoverflow.com or GitHub or others). Do not use other students' solutions or code to "test" the auto-grader. Anything you submit to the auto-grader must be your work.
2. Copying answers to any checkpoint quiz from another individual, published or unpublished written sources, and electronic sources.
3. Collaborating with another student or another individual on checkpoint quizzes or project modules.

4. Sharing written work, looking at, copying, or supplying work from another individual, published or unpublished written sources, and electronic sources.
5. Collaboration in team projects is strictly limited to the members of the team.

What is **not** cheating?

1. Clarifying ambiguities or vague points in-class handouts.
2. Helping others use computer systems, networks, compilers, debuggers, profilers, or system facilities.
3. Helping others with high-level design issues.
4. Guiding others through code debugging but not debugging for them.

Cheating in projects will also be strictly monitored and penalized. Be aware of what constitutes cheating (and what does not) while interacting with students. You cannot share or use solutions, written code, and other electronic files from students. If you are unsure, ask the teaching staff.

Be sure to store your work in protected directories. The penalty for cheating is severe, and might jeopardize your career – cheating is simply not worth the trouble. By cheating in the course, you are cheating yourself; the worst outcome of cheating is missing an opportunity to learn. In addition, you will be removed from the course with a failing grade. We also place a record of the incident in the student's permanent record.

9. Conceptual Topics

The course content will be structured into the following modules:

Module	Title	Learning Objectives
1	Fundamental Programming Constructs	● Discuss the concepts of expressions and statements. ● Describe selected Python built-in functions. ● Discuss the concept of user-defined functions. ● Describe the usefulness of variables and functions for code organization and readability.
2	Flow Control	● Explain how iteration is used to solve computational problems involving repetition. ● Distinguish the use cases where for loop is appropriate from the use cases where while loop is a better solution. ● Trace execution of a program involving iteration. ● Explain how conditional statements are used to solve computational problems involving decisions.
3	Data Structures	● Contrast the use cases where a list, a set, a tuple, or a dictionary is the appropriate solution in Python. ● Contrast the behavior of a list to the behavior of a set when it comes to the ordering of elements and their inclusion. ● Contrast the behavior of a list to the behavior of a tuple (immutability). ● Compare the behavior of assignment statements when the object of assignment is a primitive type versus a complex data structure, such as the ones covered in this module.
4	Object-oriented Programming	● Explain the concept of an object as a logical grouping of data and functions/methods that operate on them. ● Explain the concept of a class as a blueprint for objects. ● Identify classes that are an appropriate solution in some cases compared to a collection of loose data containers and functions. ● Explain the concept of a module as a logical grouping of related classes, functions, and data.

		● Demonstrate the usefulness of classes, objects, and modules for code organization and readability. ● Recognize the type of instance and its corresponding class. ● Select appropriate special methods for a class. ● Identify the appropriate cases for static methods. ● Translate similar classes into subclasses by using inheritance.
5	Software Development	● Discuss the importance of writing comments and how to write correct comments. ● Identify application requirements from an informal and incomplete description. ● Identify trivial and non-trivial application code in a programming file. (variable names, unused code) ● Discuss how to avoid repetitive code and magic numbers making code robust. ● Report the application design based on the requirements using top-down design and OOAD. ● Report a complex application consisting of user-defined as well as pre-existing modules. ● Appraise the benefits of a high-quality codebase.
6	Data Manipulation	● Identify the common pitfalls in storing data as {TC}SV flat files (quote and line-break chars, size). ● Describe the format of common flat files (CSV, TSV). ● Report types and data structures available in Python to types and data structures available in JSON. ● Discuss the concept of a well-formed XML document and a relationship between an XML schema and a valid XML document. ● Recognize the differences between XML document and HTML document. ● Identify the necessity of ACID transactions during database operation. ● Understand CAP theorem when designing a database. ● Design entity-relationship diagrams to view data and define primary and foreign keys. ● Understand the SQL and NoSQL database design (Database, table, row, column...) ● Compare the database design and structure of Redis and MongoDB and learn how to define the primary key for both. ● Report the loading, manipulation, and storage with MySQL, Redis, and MongoDB. ● Recognize proper use cases and limitations for table-oriented data stores (MySQL), document-oriented data stores (MongoDB), and in-memory key-value stores (Redis).
7	Web Scraping	● Describe HTML format. ● Explain the utility and typical use cases of HTML documents. ● Explain how to automatically download HTML files from the Web using common Python libraries. ● Explain how to extract information from HTML files using common Python libraries. ● Discuss some concerns related to parsing HTML files.
8	Data Analysis	● Understand loading data function in Pandas module for different sources of structured data (CSV, TSV) and semi-structured data (JSON, XML). ● Recognize appropriate ways to realize the data cleaning process. (dropNA, fill missing data)

		● Discuss the manipulation functions in the Pandas module to do analysis for Pandas data frame. (Summarize, Group, Combination) ● Select appropriate data visualization types with different data. ● Select different parameters for an appropriate data visualization type.

10. Projects

The projects are geared towards providing hands-on experience. Students will learn to develop all projects using various public cloud services. For each project, students are expected to work within a specified budget otherwise they risk being penalized, and fulfill the following learning objectives.

10.1. Project 1: Types, Variables, Functions

- Utilize built-in functions and primitive data types.
- Distinguish Python code from comments and Python docstrings.
- Import build-in modules, as well as functions and constants defined in those modules.
- Write your own (user-defined) functions.
- Demonstrate the usefulness of variables and functions for code organization and readability.
- Write a simple Python program consisting of several statements and functions using a simple IDE called IDLE.
- Write simple statements to test the correctness of your code.

10.2. Project 2: Flow Control (Iteration, Conditionals, Strings, Basic I/O)

- Use iteration to solve computational problems involving repetition.
- Implement for loops and while loops in Python.
- Trace execution of a program involving iteration.
- Design and implement algorithms using nested iteration.
- Use conditional statements to solve computational problems involving decisions.
- Apply relational and logical operators to form Boolean expressions.
- Recognize the effects of flow control mechanisms (iteration and conditionals) on code organization and readability.
- Create, manipulate, and format strings in Python.
- Read, write, and modify text files using Python.

10.3. Project 3: Lists, Sets, Tuples, Dictionaries

- Perform basic operations on a list, a set, a tuple, and a dict (dictionary) in Python.
- Contrast the use of a list, a set, a tuple, and a dict.
- Create a list in Python, insert, access, modify, and delete its elements.
- Create a set in Python, insert and remove its elements, and test for element membership.
- Create a tuple and experiment with immutability.
- Create a dict, insert, access, modify and delete its elements, and test for the presence of a key.
- Utilize list and dict in interactive applications to store, read, and manage the application's state.
- Utilize list and dict in data analysis where data points need to be organized and compared.

10.4. Project 4: Classes, Objects, Modules

- Implement classes in Python, with attributes and methods.

- Experiment with class inheritance for code reuse.
- Implement and use custom Error types.
- Re-organize flat code from a single file into several files (modules) and classes.
- Demonstrate the usefulness of classes and modules for code organization and readability.

10.5. Project 5: Style, Testing, Packaging

- Fix style mistakes in a Python program to improve the code quality and maintainability using automatic style checkers and linting.
- Improve the quality of a complex code written by someone else.
- Design and implement complete test suites achieving 100% test coverage using Python test and coverage libraries.
- Generate a built distribution and source archive from a custom Python project.
- Install and uninstall a custom module from a local package using pip.

10.6. Project 6: Files and Data Stores

- Utilize common Python libraries to load, manipulate, and store data in CSV, XML, and JSON formats.
- Load data in one format and store them in another.
- Filter data based on different criteria.
- Create a MySQL database and a table.
- Load data from a CSV file into the MySQL table.
- Query the MySQL table from Python.
- Create a MongoDB database and a collection.
- Load data from a JSON file to MongoDB collection.
- Query the MongoDB collection from Python.

10.7. Project 7: Web Scraping

- Iteratively download information from multiple websites using the Python requests package.
- Extract relevant information from an HTML document using the Python beautifulsoup4 package.
- Automatically create an Excel spreadsheet using the Python openpyxl package.
- Read data from an Excel spreadsheet using the Python pandas package.
- Automatically generate a batch of Word documents using the Python python-docx package.

10.8. Project 8: Data Analysis

- Prepare data stored in various formats so that they can be conveniently loaded into a pandas DataFrame.
- Use several pandas loading functions in order to conveniently ingest data stored in various formats (and having various shapes) into a pandas DataFrame with a desired shape (i.e., index and columns/fields).
- Merge multiple pandas DataFrame objects into a single one and thereby achieve integration of data from multiple widely different sources into a unified data set.
- Persist pandas DataFrame using Python pickle module.
- Examine the created data set stored in pandas DataFrame to obtain basic descriptive statistics.
- Assess the data set on its usability and basic properties of the captured statistics.
- Generate systematic reports on the usability and basic properties of the statistics and store it as an Excel spreadsheet.
- Utilize regular expression to extract patterns (email addresses) from a large data set of text documents (email messages).
- Compare the expressiveness of regular expressions to that of simple text matching.

11. Schedule

The tentative schedule is as follows (specific deadlines are posted on OLI and Sail()):

15-week option:

Week	Conceptual Content on OLI	Quiz	Hands-on Projects on Sail()
1	M1: Fundamental Programming Constructs	Quiz 1	P1: Types, Variables, Functions
2-3	M2: Flow Control	Quiz 2	P2: Iteration, Conditionals, Strings, Basic I/O
4-5	M3: Data Structures	Quiz 3	P3: Lists, Sets, Tuples, Dictionaries
6-7	M4: Object-oriented Programming	Quiz 4	P4: Classes, Objects, Modules
8-9	M5: Software Development	Quiz 5	P5: Style, Testing, Packaging
10-11	M6: Data Manipulation	Quiz 6	P6: Files and Datastores
12-13	M7: Web Scraping	Quiz 7	P7: Web Scraping
14-15	M8: Data Analysis	Quiz 8	P8: Data Analysis

8-week option:

Week	Conceptual Content on OLI	Quiz	Hands-on Projects on Sail()
1	M1: Fundamental Programming Constructs	Quiz 1	P1: Types, Variables, Functions
2	M2: Flow Control	Quiz 2	P2: Iteration, Conditionals, Strings, Basic I/O
3	M3: Data Structures	Quiz 3	P3: Lists, Sets, Tuples, Dictionaries
4	M4: Object-oriented Programming	Quiz 4	P4: Classes, Objects, Modules
5	M5: Software Development	Quiz 5	P5: Style, Testing, Packaging
6	M6: Data Manipulation	Quiz 6	P6: Files and Datastores
7	M7: Web Scraping	Quiz 7	P7: Web Scraping
8	M8: Data Analysis	Quiz 8	P8: Data Analysis

12. Accommodations for Students with Disabilities

If you have a disability and have an accommodations letter from the Disability Resources office, I encourage you to discuss your accommodations and needs with me as early in the semester as possible. I will work with you to ensure that accommodations are provided as appropriate. If you suspect that you may have a disability and would benefit from accommodations but are not yet registered with the Office of Disability Resources, I encourage you to contact them at access@xyz.edu.

13. Take care of yourself

Do your best to maintain a healthy lifestyle this semester by eating well, exercising, avoiding drugs and alcohol, getting enough sleep, and taking some time to relax. This will help you achieve your goals and cope with stress.

All of us benefit from support during times of struggle. You are not alone. There are many helpful resources available on campus and an important part of the college experience is learning how to ask for help. Asking for support sooner rather than later is often helpful.