Syllabus
COSC 1010: Introduction to Programming

Description
Introduction to abstraction, algorithmic thinking, simulation and testing for computer-based problem solving. Students will learn a high-level programming language (Python) and use tools developed by computer scientists and software engineers to solve problems. 3 hours lecture, 2 hours lab.

Synopsis
Programming is about problem solving. Our tool to solve problems is a computing device that executes instructions. The art of software development consists in mapping real problems to the capabilities of devices controlled by software, and then solve the resulting programming problems. The software engineer will have to interact with all sorts of devices and data.

Rationale and Connection to the Creative and Technology and the Language, Cognition and Memory theme

A beginning programming class in computer science is not about learning the syntax of a language, but about learning how to use one's creativity in order to solve problems in an algorithmic manner. At the root of algorithmic design is creativity. One of the defining characteristics of a computer scientist is the desire to solve problems for people, problems that arise from human endeavors, whether it is the pursuit of beauty, subsistence, or intellectual achievements. Even in a beginning programming class, these almost universal aspirations enter, mainly through the example problems that are given to the students.

While we argue that this class is about creativity in problem solving, it is also an introduction to a technology that has, is, and will trans form the world.

The class is designed along the principles of "Learning in Teams," where collaboration and communication are integral problem-solving skills. Students will develop their creative problem-solving and collaboration skills through activities — problems to be solved in pairs —, group quizzes, individual weekly small projects, larger sets of activities in the laboratories, and group large projects.

These learning activities will ask students to process large texts and corpora of textual and numerical data, to create interactive command-line applications, and to develop graphical interfaces using event-driven programming. For the latter, aesthetic criteria need to be developed and applied.

This class is using course material developed jointly with Xavier College (Autonomous) in Ahmedabad, Gujarat, India and the Xavier Institute of Engineering in Mumbai, Maharashtra, India.

Prerequisites
No prior programming experience is assumed. Two years of college preparatory mathematics required.

Course Objectives
After successfully completing this class, students will possess
- the capability to solve problems

- the capability to think programmatically at a beginner's level
- the capability to program fluently in Python;
- the capability to plan a small software program;
- the capability to argue about programming tasks and document their work in an appropriate manner;
- the capability to program in the imperative and object-oriented style at a beginning level.

Learning Outcomes

After successfully completing the class, students will possess

- the capability to express an algorithm in Python using control structures and built-in data structures efficiently and correctly;
- the capability to decompose a more complex tasks into simpler subtasks and implement performance of these subtasks in individual functions or methods;
- the capability to use the imperative style of programming;
- the capability to interact programmatically with files and directories;
- the capability to program in an object-oriented manner;
- an understanding of event-based programming in a GUI application with TkInter;
- the capability to decompose simple tasks (such as processing a file with numerical data and derive statistical means, developing a simple desk calculator, writing a numerical integrator for functions) into individual processing steps and modules;
- the capability to use control structures in Python;
- the capability to interact with files and directories programmatically in Python;
- an appreciation of online and offline resources for successfully performing programming tasks;
- the capability to design pleasing and effective interfaces;
- the capability to use comments and doc-strings in an appropriate manner;
- an appreciation for the difficulty of working in small teams.

Contents

1. Examples of algorithms; arithmetic operations, values and types, interactive and programmatic python.
2. Variables, expressions and statements.
3. Functions, function calls, function definitions, locality.
4. Conditionals and recursions.
5. Iterations
6. Built-in data structures
    - Strings
    - Lists
    - Dictionaries
    - Tuples
    - Files
7. Elements of functional programming: List and dictionary comprehensions
8. Object oriented design
    - Programmer defined types
    - Classes and functions
    - Classes and methods
    - Overloading methods and operators (e) Inheritance
    - Exceptions
9. Introduction to graphic and event-oriented programming with Tkinter

Grading

Grading is based on four components given below

- Daily individual / group quizzes (25%)
- Small weekly programming projects (15%)
- Weekly laboratories (15%)
- Three larger programming projects (15%)
- Two midterm and one final examination (30%)

All activities will involve solving problems.

Each programming project has a deadline. For every week that you miss the deadline, your maximum grade will be lowered by 10%, similarly for missed labs that you can make up on your own. You can resubmit faulty projects. Each resubmission lowers your grade by 10%.

Discovery Class

This class is part of the discovery tier and contributes to the student’s problem solving skills.

(5) Leaders in Discovery: Marquette students will advance understanding of the world by identifying significant questions and then searching for answers based on a systematic process of discovery that is rooted in intellectual inquiry and the Jesuit liberal arts tradition.
(6) Global Problem Solvers:Marquette students will be well practiced in cooperative and cross-disciplinary problem-solving skills and they will be able to present innovative solutions that draw from theological, philosophical, qualitative and quantitative perspectives to address the increasingly blurred lines between local and global challenges.