COMP 364 (3 Credits) – Computer Tools for Life Science
Fall 2017
McGill University

Syllabus

Course description:
Introduction to computer programming in a high level language: variables, expressions, types, functions, conditionals, loops, objects and classes. Introduction to algorithms, data structures (lists, strings), modular software design, libraries, file input/output, debugging. Emphasis on applications in the life sciences.

List of life science topics used as examples:
- Central dogma of molecular biology
- RNA and/or protein structure prediction
- Genome sequencing and analysis
- Biological networks
- Evolution
- Epigenetics
- Biomarker discovery
- Biosystems dynamics
- Cell and biomedical imaging
- Modeling

Objectives:
By the end of this course, students will be able:
1) Design and describe precise, unambiguous instructions that can be used [by a computer] to solve a problem or perform a task;
2) Translate these instructions into a language that a computer can understand (Python);
3) Write programs that solve complex problems (especially those arising in Life Sciences) by decomposing them into simpler subproblems;
4) Apply programming-style and structure conventions to make your programs easy to understand, debug and modify;
5) Learn independently about new programming-language features and libraries by reading documentation and by experimenting.

Prerequisites: BIOL 112 and A CEGEP level mathematics course

Restrictions: Only one of COMP 204, COMP 202 and COMP 208 can be taken for credit. COMP 204 cannot be taken for credit with or after COMP 250, COMP 206, or COMP 364.

Instructors:

Christopher John Frederick Cameron : christopher.cameron@mail.mcgill.ca
Carlos Oliver: carlos.gonzalezoliver@mail.mcgill.ca

Course Webpage: http://cs.mcgill.ca/~cgonza11/COMP_364/

Teaching Assistants: No TAs are available for this course.

Location: Education Building, Room 216

Lecture format: 3 x 50 minutes of lectures per week (MWF).

Additional tutorials:

1. Installation of Python on laptop computers
2. Unix environment: File system organization and basic commands

Textbook: How to Think Like a Computer Scientist: Interactive Edition (Python)
http://interactivepython.org/courselib/static/thinkcspy/index.html#

Student evaluation:
Assignments: 35% (5 assignments worth 7% each)
Quizzes: 5% (5 quizzes worth 1% each)
Midterm exam: 20%
Final exam: 40%

Assignments:
5 Python programming assignments, each aiming at addressing a specific biological question.

Lecture schedule

Lecture	Computer science topic
1	Thinking algorithmically
2	What is a computer: CPU, RAM, storage, communication. Binary numbers, instructions
3	Python programming: What is code? How write code? How to run it? Basic programs
4	Python programming: variables and types. Program execution.
5	Python programming: Control flow: conditionals, boolean expressions
6	Python programming: Lists
7	Python programming: Control flow: for loops, while loops
8	Putting it together: Nested loops and conditionals
9	Python programming: Tracing program, debugging
10	Python programming: Debugging using a debugger (pdb)
11	Program organization: functions, encapsulation, execution flow, variable scope
12	Program organization: function parameters and return values. Composition.
13	Python programming: Dictionaries
14	Putting it together – part I: Combined use of functions, loops, and lists.
15	Putting it together – part II: Advanced examples; Commenting code
16	Algorithm design: Linear and Binary Search
17	Algorithm design: Selection Sort and InsertionSort
18	Review session
19	Midterm exam
20	Python programming: File IO, internet IO
21	Object-oriented programming: Modules, Classes, and Objects
22	Object-oriented programming: Modules, Classes, and Objects
23	Object-oriented programming: Modules, Classes, and Objects

24	Python libraries; How to read and use an API
25	Using libraries: Data visualization with MatPlotLib
26	Using libraries: Data visualization with MatPlotLib
27	Building complex programs using BioPython
28	Biological data analysis with BioPython: Sequence analysis
29	Biological data analysis with BioPython: molecular structures
30	Notions of machine learning
31	Intro to machine learning with scikit-learn
32	Intro to machine learning with scikit-learn
33	Notions of image processing
34	Introduction to image analysis with scikit-image
35	Introduction to image analysis with scikit-image
36	Toward other programming languages. Compiled vs interpreted languages
37	Toward other programming languages. Static vs dynamic typing
38	Review session

Right to submit in English or French written work that is to be graded:
In accord with McGill University's Charter of Students' Rights, students in this course have the right to submit in English or in French any written work that is to be graded.

Academic Integrity statement:
McGill University values academic integrity. Therefore all students must understand the meaning and consequences of cheating, plagiarism and other academic offences under the Code of Student Conduct and Disciplinary Procedures (see www.mcgill.ca/students/srr/honest/ for more information).