Python Programming

Muhammad Abdul-Mageed

Linguistics/Information/Computer Science

The University of British Columbia

- **Course title:** Python Programming: **LIBR559C/LING530G** (Cross-listed)
- **Year:** Winter Session I 2019
- **Time:** Wed 9:00am – 11:50am.
- **Location:** iSchool Terrace Lab.
- **Instructor:** Dr. Muhammad Abdul-Mageed
- **Office location:** Totem Field Studios 224
  (Department of Linguistics: 2613 West Mall V6T 1Z4)
- **Office phone:** (Apologies, I do not use office phone. Please email me)
- **E-mail address:** muhammad.mageed@ubc.ca
- **Office hours:** Tue. 12:00-14:00pm @Totem Field Studios 224, or by appointment. *(I can also handle inquiries via email or, in limited cases, Skype.)*
- **Portal:** http://canvas.ubc.ca

1. Course Rationale & Goal:

Rationale/Background: Python is a remarkably powerful dynamic programming language that is used for a very wide variety of application domains, including text processing, data mining and analytics, and web development. Python's

very clear, readable syntax (e.g., use of English keywords where other languages use punctuation) and strong introspection capabilities make it an ideal choice for novice programmers. Indeed, more and more instructors are adopting the language for introductory programming courses.

Goal: The goal of this course is to introduce programming concepts using the Python programming language. The course is designed to be self-standing and focuses on basic concepts, assuming no prior programming experience. For an IT professional, programming skills are almost indispensable: General understanding of programming and programming experience both improve students' overall understanding of information systems and help develop their general problem solving skills. Programming skills are also essential for analysis and mining of information users' behaviors and communities. Good programming skills significantly boost work productivity and students usually utilize them during their program of study and throughout their professional lives.

Potential audiences for this course are:

- People with no programming/IT experience who want to acquire an introductory level understanding of programming generally and Python specifically.
- People planning to take courses on text analytics, social media mining, information retrieval, information visualization, human-information interaction, natural language processing, machine learning, web and database design, or similar courses, who want to prepare for, or significantly advance, carrying out work and research in these areas.
- People interested in using Python as a toolkit for carrying out Linguistics (or other types of social sciences) research.

2. Course Objectives:

Upon completion of this course students will be able to:

- Identify, analyze, assess, and solve a problem or need using Python programming. [1.1]
- Apply Python programming knowledge to practical information access, extraction, retrieval, and data (e.g., text) mining tasks. [1.2]
- Enhance interpersonal and written communication skills through assignments and discussions with classmates and instructor. [2.1]

- Collaborate effectively with peers through course assignments, demonstrating the ability to work as part of a team, including initiative taking, integrity, dependability and co-operation. [3.1]

- Understand, develop, and apply relevant library and information science tools to address information needs, questions and issues. [4.1]

3. Course Topics:

- Overview of (Python) programming;

- Unix: *Directory navigation & control*, *File maintenance commands*, *Display commands*, *Text processing*

- Data types (numbers. Strings, lists, tuples, and dictionaries/hashes);

- Control flow & functions;

- File input and output;

- Exception handling;

- Iterators;

- Applications: Simple text processing

4. Prerequisites:

- MLIS and Dual MAS/MLIS: Completion of MLIS Core or permission of iSchool Graduate Advisor (**prerequisite for iSchool students only**)

- Access to a computer on a regular basis. You should make your own arrangements for this.

5. Format of the course:

• This course will involve lectures, class hands-on activities, individual and group work, and instructor-, peer-, and self-assessment.

6. Course syllabus:

Recommended book:

- Hetland, M. L. (2017). Beginning Python: from novice to professional (3rd ed.). Apress. ISBN-10: 1484200292. ISBN-13: 978-1484200292. [Amazon Link]. **Note:** Full text soft copy available via the UBC Library!

Suggested supplemental book:

- Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python. `O'Reilly Media, Inc.`. (ISBN-13: 063-6920516491; ISBN-10: 0596516495) nltk-book

Online Tutorials:

- You will have access to Jupyter notebook and video tutorials created by the instructor.
- Pandas

7. Calendar / Weekly schedule and readings (tentative)

- H= Hetland, M. L. Book
- BKL= Bird, S., Klein, E., & Loper, E. Book

See online syllabus: [Link].

8. Course Assignments/Grades:

- Professionalization & attendance: 15%
- ASSIGNMENT 0: In-Class & Video Tutorial Code: 15%
- ASSIGNMENT 1: 10%
- ASSIGNMENT 2: 10%
- ASSIGNMENT 3: 10%
- ASSIGNMENT 4: Term Project (GROUP): Proposal (5%), Report (40%)

See also online syllabus.

Notes about assignments:

- All written assignments must be submitted before 9:00 am of a respective assignment date;
- All coding assignments must be submitted **in print form** as well as the form of a Python script (either .py file or an `iPython/Jupyter notebook`). Support will be provided on how to create `.py` scripts and Jupyter notebooks;
- Project proposal/outline must be in a pdf format;
- All submitted files should be labeled with your last name(s) followed by an underscore and an assignment code. Assignment codes are two digit numbers, i.e., the assignment number preceeded by `0` for readability (e.g., "assignment01," "asignment02,"). *For example, `ASSIGNMENT 01` would be submitted as "abdul-mageed_assignment01.ipynb"*

Assignments/Requirements in Detail:

Professionalization & attendance: In an educational (or work) setting, each of us is expected to learn and demonstrate ability to interact with others professionally and collegially. This is important as it ensures we all work together efficiently. A total of 15% of your grade is devoted to the combination of attendance and encouraging professional communication related to the course both inside class (e.g., the way you interact with peers and the instructor) and outside class (responding to instructor communication timely). Practically, this means you should come to class in time, ensure your communication with everyone in class is civil, be attentive and do not engage in an external task or browse the social web during the class session, and respond to instructor email timely (e.g., not *consistently* responding after 3 days, but responding the same day or in 1 or 2 days is fine and an occasional delay within reason is also fine [e.g., if you are traveling]). Part of professionalization is clear communication. As such, you are encouraged to ask questions, freely and civilly agree/disagree with others, etc.

Assignment 0: In-Class & Video Tutorial Code:

Throughout weeks 2-11, you are required to practice coding inside the class session. There will be time in-class devoted to practice where you will be able to a) extend examples the instructor provides in his slides/notebooks and/or b) run code from the Hetland book. The resulting code may end up having some mistakes, look unorganized, lack depth and illustrative examples for a given task, or look identical to what the code instructor provides. To help you write better code and acquire

deeper understanding of programming concepts, you will be given the chance to spend some more time at home re-running the same code with more examples. To further enhance the process, the instructor will provide thematically-organized video tutorials that you are required to watch before working further on your in-class code. Once you watch the video relevant to a given week, you can re-organize and tidy your code, provide depth and run it with more examples, and submit it.

This assignment is meant to engage you both during the class time and at home (where you can involve in content explained in a different mode and engage at your chosen time and place). The goal is to ensure you are not only able to run code shared with you during a given session, but also extend it slightly. This assignment is only accepted if you attend class. If you miss a class, this automatically means you are not allowed to submit In-Class & Video Tutorial Code for the session/sessions you missed. As such, failure to attend class without excuse (e.g., a sickness or another emergency which you must explain to the instructor over email prior to a class whenever possible) will result in a lower class grade. *Note: The Video Tutorials are typically short (between 15-30 minutes long each).*

Deliverables: In-class code snippets A total of 10 code snippets in the form of `.py` or `.ipynb` files for the code you write based on In-Class practice + Watching Video Tutorials for weeks 2-11 of class.

Grading: Each code snippet (from each of the 10 weeks) will be worth 1.5 marks (total 15 marks). While students are required to submit the deliverables in time (as indicated above), the instructor will grade the deliverables together in two iterations (each 5 weeks' deliverables together. This is intended to simplify the process, while also sometimes allowing the instructor provide some flexibility in helping any students who may need additional help by allowing them to resubmit deliverables of specific weeks.

Assignments 1-3: Coding assignments:

These three assignments are coding exercises meant to support learning of the covered content. Each of these will be extensions of problems covered in class. The following criteria are required for each of them:

- You must run your code before submitting, and ensure it is working code. Also, you should provide evidence that your code runs well (e.g., the result of each snippet when run or first 5/few lines of output in the case of large amounts of output). The code should be submitted in the form of `.py` or `.ipynb` files.

- For each problem, in addition to your code, you should provide explanatory Python comments as appropriate. Your comments should be short and to-the-point. As a programmer, you should develop a sense as to when to use

comments and when not. This requirement is meant to teach you about using comments. Using comments will be discussed in class and feedback on your deliverables will be provided.

- In addition to the functionality of your code, you should care for algorithms and style. This means, over time, you are expected to develop a sense of what the best ways to solve a problem in a computationally efficient way and using the capabilities of the Python programming language. These are aspects that will be discussed in class over the course of the semester.

Deliverables: `.py` or `iPython notebooks` with code * For each of assignments the 1-3 assignments, you are required to submit your code an in the form of `.py` or `.ipynb` files, including your code and relevant narratives/comments, etc.

Assignment 4: Term Project:

The purposes of this assignment include:

- Identifying, analyzing, assessing, and solving a problem or need using Python programming;
- Applying acquired Python skills and knowledge to practical tasks;
- Developing oral and written communication skills through discussions with classmates and instructor;
- Demonstrating ability to work as part of a team, including initiative taking, integrity, dependability and co-operation, and effective collaboration.

For this assignment, each student is required to work as part of a group of of **3 students** on a project involving a significant amount of coding (e.g., coding at least equal to or more than the amount you wrote for assignments 1-3). Example projects include designing a system that generates language (e.g., the language of William Shakespeare), analyzes emotions in a dataset of literary work, visualizes concepts from a data source, summarizes texts, builds a simple website that delivers some service, builds a web interface or an app for a service (e.g., a library service), builds an interactive database with use case scenarios in meaningful contexts, analyzes a publicly available (e.g., governmental) dataset. You will have the chance to discuss your project with the instructor upon your request, and example projects will be discussed with class as well.

Deliverables Proposal (500 words)

- Who are the the group members?

- What are you designing or coding for?
- What motivates your work? Why is it important or useful? For which audience?
- How does your project compare to other projects people have articulated?
- How do you think the project will further your Python programming skills?
- What Python libraries do you expect to use?
- How does the work bread down and what each member of the team be contributing?
- Timeline for completing the project, including goals for each segment.

Final Report/Paper (3000-4000 words maximum):

The final deliverable should include:

- A detailed and clear description of your project, including the necessary sections, as appropriate. For system-oriented projects (e.g., building a website or an app), you will need to provide motivations, describe the audience, details of your implementation, and system functionality. For projects more focused on data analysis, you will need to include research questions, a brief literature review, a description of datasets, implementation details and methods employed, results, and a conclusion involving limitations and future directions;
- All relevant code;
- Pointers to live versions of your system, if any;
- Some of these components will be more relevant to some projects but not others. For example, if your project involves a system for analysis of a dataset, you will need to include standard sections as you would find in an academic journal;
- As appropriate, you should situate your work within the wider context of previous works and approaches, with supporting arguments (*5 sources*);
- Use case scenarios and screenshots from your system interface, if any;
- Employment of figures, tables, and visualizations as appropriate to enhance argument and facilitate communicating your findings/results;

9. Course Policies

Attendance

The UBC calendar states: "Regular attendance is expected of students in all their classes (including lectures, laboratories, tutorials, seminars, etc.). Students who neglect their academic work and assignments may be excluded from the final examinations. Students who are unavoidably absent because of illness or disability should report to their instructors on return to classes."

Evaluation

All assignments will be marked using the evaluative criteria given in this syllabus.

Access & Diversity

Access & Diversity works with the University to create an inclusive living and learning environment in which all students can thrive. The University accommodates students with disabilities who have registered with the Access and Diversity unit. You must register with the Disability Resource Centre to be granted special accommodations for any on-going conditions.

Religious Accommodation

The University accommodates students whose religious obligations conflict with attendance, submitting assignments, or completing scheduled tests and examinations. Please let your instructor know in advance, preferably in the first week of class, if you will require any accommodation on these grounds. Students who plan to be absent for varsity athletics, family obligations, or other similar commitments, cannot assume they will be accommodated, and should discuss their commitments with the instructor before the course drop date. UBC policy on Religious Holidays.

Academic Integrity

The following material about academic misconduct is copied and pasted from the UBC calendar:

Academic Misconduct

Students are responsible for informing themselves of the guidelines of acceptable and non-acceptable conduct for graded assignments established by their instructors

for specific courses and of the examples of academic misconduct set out below. Academic misconduct that is subject to disciplinary measures includes, but is not limited to, engaging in, attempting to engage in, or assisting others to engage, in any of the actions described below.

1. Cheating, which may include, but is not limited to:

    - falsification of any material subject to academic evaluation, including research data;
    - use of or participation in unauthorized collaborative work;
    - use or possession in an examination of any materials (including devices) other than those permitted by the examiner;
    - use, possession, or facilitation of unauthorized means to complete an examination (e.g., receiving unauthorized assistance from another person, or providing that assistance); and
    - dishonest practices that breach rules governing examinations or submissions for academic evaluation (see the Student Conduct during Examinations).

2. Plagiarism, which is intellectual theft, occurs where an individual submits or presents the oral or written work of another person as his or her own. Scholarship quite properly rests upon examining and referring to the thoughts and writings of others. However, when another person's words (i.e. phrases, sentences, or paragraphs), ideas, or entire works are used, the author must be acknowledged in the text, in footnotes, in endnotes, or in another accepted form of academic citation. Where direct quotations are made, they must be clearly delineated (for example, within quotation marks or separately indented). Failure to provide proper attribution is plagiarism because it represents someone else's work as one's own. Plagiarism should not occur in submitted drafts or final works. A student who seeks assistance from a tutor or other scholastic aids must ensure that the work submitted is the student's own. Students are responsible for ensuring that any work submitted does not constitute plagiarism. Students who are in any doubt as to what constitutes plagiarism should consult their instructor before handing in any assignments.

3. Submitting the same, or substantially the same, essay, presentation, or assignment more than once (whether the earlier submission was at this or another institution) unless prior approval has been obtained from the instructor(s) to whom the assignment is to be submitted.

4. Impersonating a candidate at an examination or other evaluation, facilitating the impersonation of a candidate, or availing oneself of the results of an impersonation.

5. Submitting false records or information, orally or in writing, or failing to provide relevant information when requested.

6. Falsifying or submitting false documents, transcripts, or other academic credentials. Failing to comply with any disciplinary measure imposed for academic misconduct.

Websites to help you understand plagiarism:

- UBC Learning Commons: Avoid Plagiarism
- UBC Wiki: How to avoid plagiarism

Additional UBC resources on plagiarism and academic integrity:

- UBC Learning Commons: Academic Integrity

If after reading these materials you still are unsure about how to properly use sources in your work, please ask me for clarification. Students are held responsible for knowing and following all University regulations regarding academic dishonesty. If a student does not know how to properly cite a source or what constitutes proper use of a source it is the student's personal responsibility to obtain the needed information and to apply it within University guidelines and policies. If evidence of academic dishonesty is found in a course assignment, previously submitted work in this course may be reviewed for possible academic dishonesty and grades modified as appropriate. UBC policy requires that all suspected cases of academic dishonesty must be forwarded to the Dean for possible action.