Course Name: Python
Course Number(s) DS D 590
Instructor: Vel Malbasa
Course Date: Fall 2017

Instructor Information:
Name: Vel Malbasa
Title: Visiting Professor
Address: Computer Science, School of Informatics and Computing, Indiana University, Lindley Hall, Room 401C, 150 S. Woodlawn Ave. Bloomington, IN 47405
Email: vmalbasa@indiana.edu
Webpage: http://www.cs.indiana.edu/
Phone: (812) 856 - 1872
Office Hours: Wednesday, 5:30 pm to 8 pm EST, or by appointment

Lead AI Information:
Name: Rakshesh Rajeshkumar Shah
Title: Master student, Data Science
Email: rakshah@umail.iu.edu
Webpage: https://www.linkedin.com/in/raksheshshah
Phone: (812) 325 3642
Social Media Handle: rakshesh
Office Hours: Thursday, 4:30 to 7:30 PM EST

AI Information:
Name: Akshada More
Title: Master student, Computer Science
Email: akmore@umail.iu.edu
Phone: (812) 955-1681
Office Hours: Friday, 4:30 to 7:30 PM EST

Course Description/Overview:
This course provides a gentle, yet intense, introduction to programming in Python for students with little or no prior experience in programming. Python is an open-source language that allows rapid application development of scalable software systems, is object-oriented by design and provides an excellent platform to learn the basics of programming. The course will focus on planning and organizing programs, and developing high quality, working software that solves real world problems.
Course Goals:
Students will:

- Learn how to design and implement scalable Python programs that solve real world problems, with focus on Data Science applications.
- Learn top-down and object oriented approaches to software design.
- Learn data structures and algorithms used in numeric and text data processing.

Course Learning Objectives:

- Design, code, and test small and medium sized Python programs that meet requirements expressed in English. This includes design and use of procedural statements — assignments, conditional statements, loops, method calls — and sequences. Use Python packages for data science applications and real world data analysis.
- Apply the concepts of top-down design and object-oriented software design as used in Python: classes, subclasses, properties, inheritance and overriding
- Demonstrate understanding and use of basic data structures (including strings, lists, dictionaries, and tuples) and algorithms (including searching and sorting).

Course Prerequisites:
Basic high school mathematics (no calculus) and no programming experience.

Course Books/Readings/Additional Resources:

Downey, Allen B., *Think Python: How to Think Like a Computer Scientist*, O'Reilly, 2012

You can obtain a free copy of the textbook in PDF at: [http://www.greenteapress.com/thinkpython/](http://www.greenteapress.com/thinkpython/)

Additional Resources:


Course Communication Channel(s):
Canvas Discussion Forum; Email listserve; Zoom for live discussion sessions and office hours.

Course Policies:

- This course follows Indiana University’s academic calendar for summer 2017. The course ends with submission of final grades on July 28, 2017.
- Students are expected to carry out all activities within each unit. Each unit has some small number of activities associated with it.
• Students are expected to maintain more than one copy of their assignments in case of equipment failures (disk crash).
• Students are expected to do the lesson readings and watch lesson videos before the discussion of the lesson.
• Students are expected to attend and participate in weekly live discussions via Zoom.
• It is expected that a student will put in 6-7 hours a week every week into the course which includes time spent in readings, reflections, and engaging with instructional content.

Course Assignments/Assessments and Grade Percentages:
There will be weekly programming assignments and quizzes and a final project.

Grade percentages:

<table>
<thead>
<tr>
<th>Assignment Type</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignments</td>
<td>70%</td>
</tr>
<tr>
<td>Final Project</td>
<td>15%</td>
</tr>
<tr>
<td>Quizzes</td>
<td>10%</td>
</tr>
<tr>
<td>Discussion participation</td>
<td>5%</td>
</tr>
</tbody>
</table>

Grading Policies - Late Assignments:
All assignments must be turned in by the due date and time. Late assignment reports will be penalized:

- 1 day late: 10%
- 2 days late: 20%
- 3 days late: 30%

No reports accepted if late 4 or more days. Exceptions will be granted if a student has a prior arrangement with the instructor based on a valid excuse, as outlined by Indiana University policy.

Incomplete Policy:

Circumstances Permitting Incompletes:
(Approved: Faculty Council 11/5/52, amended 2/19/63)

The grade of Incomplete used on the final grade reports indicates that the work is satisfactory as of the end of the semester but has not been completed. The grade of Incomplete may be given only when the completed portion of a student’s work in the course is of passing quality.

Instructors may award the grade of Incomplete upon a showing of such hardship to a student as would render it unjust to hold the student to the time limits previously fixed for the completion of his/her work.

Departmental Records:
Each academic unit shall maintain a record of Incomplete grades recorded in its courses. This record, completed by the instructor, should include (I) the name of the student and the student’s identification number, (2) the course number, section number, and hours of credit, (3) semester and year of enrollment, (4) the signature of the instructor, (5) a brief statement of the reason for recording the Incomplete, and (6) an adequate guide for removal of the Incomplete grade (with a
suggested final grade) in the event of the departure or extended absence of the instructor from the campus.

**Removal of Incompletes:**

*Methods.* A grade of Incomplete may be removed (a) by the student completing the course within the time limit and the instructor sending the appropriate Removal- of-Incomplete form to the Office of the Registrar, and (b) by the dean of the student's school authorizing the change of Incomplete to W.

*Limits.* The time allowed for the removal of an Incomplete is one calendar year from the date of its recording, except that the dean of the student's college or school may authorize adjustment of this period in exceptional circumstances. By assigning an Incomplete an instructor implicitly authorizes and requires the "I" to be changed to an "F" at the end of the appropriate time period, if that instructor does not otherwise act to remove the "I". The Registrar will automatically change the "I" to "F" at the end of the appropriate time period except when an adjustment of the period has been authorized or the student has received a degree since that date. Both the student and the instructor in whose course the student received the Incomplete will be notified of this change of grade.

A student may not re-enroll in a course in which a grade of Incomplete has been recorded. The student may be denied the right to make up an Incomplete if it seems to the unit dean and the instructor that it is impractical for the student to complete the course. In this case, the student should be given the opportunity to withdraw from the course.

**Absence from Final Examinations:**

(Approved: Faculty Council 11/5/52)

Where the grade of Incomplete is given because the student missed the final examination, he or she shall be allowed to remove the Incomplete by taking the examination only if he or she has followed the regular procedure to have his/her absence excused and the Committee on Absence has notified the instructor that the student may be permitted to take the examination. If the Committee on Absence, under the Dean for Student Services, determines that the reason for the student's absence is not satisfactory, it should inform the instructor that the grade of Incomplete should be changed to a grade of "F". Where the Incomplete was received because of absence from the final examination, students may prefer to receive a grade of W instead of taking the examination. In such cases, the Dean will not approve the grade of W unless the Committee on Absence has approved the reason for absence.

**Drop Policy:**

Before you drop or add a class, make sure you understand the rules. Your timing makes all the difference when it comes to whether or not you’re hit with extra fees or a bad grade.

Dropping or adding classes can affect your financial aid, scholarship status, or tuition cost. If you do decide to drop or add a class, be sure to talk with your academic advisor first. If you’re worried about what will happen with your finances, contact us. We’ll talk you through it.
Making changes during the continuous drop/add period:

Continuous drop/add is available from the beginning of early registration through the Sunday following the first week of classes. If you drop a class during this time, it won’t show up on your transcript or grade reports. So if you want to avoid a W (“Withdrawn”) grade, make sure to drop by this deadline.

Schedule adjustment fees:

You'll be charged a drop/add access fee of $8.50 each day you make a schedule change after two business days from the date you register for the term.

If you drop courses after the first week of classes, you’ll be charged an additional fee for each course you drop.

Student Integrity Policy:

Students are expected to conduct themselves in a manner befitting their status as a student of a respected and distinguished institution of higher education. In college courses, we are continually engaged with other people’s ideas: we read them in texts, hear them in lecture, discuss them in class, and incorporate them into our own writing. As a result, it is very important that we give credit where it is due. Plagiarism is using others’ ideas and words without clearly acknowledging the source of that information.

See http://www.indiana.edu/~wts/pamphlets/plagiarism.shtml for help in addressing plagiarism in your own work.

Work you submit must be your own. Doing your own work means that you must turn in your own, original work. It means you do not turn in a solution copied from somebody else or found on the web. In programming projects, working together may extend to figuring out overall strategies for solution, but you may not work together to write the actual code that you submit.

Articles and data posted to Canvas for this course are for the convenience of students only and should not be shared outside the course.

Students with Disabilities Accommodations:

Arranging Accommodations:

Once you have submitted a request for service, provided appropriate documentation verify your disability, and met with your DSS Coordinator, it is time to arrange your reasonable accommodations with your instructors.

What is a Reasonable Accommodation?

A reasonable accommodation is any modification or adjustment that will enable a qualified student with a disability to participate in a course, program, activity or service. Reasonable accommodations assure that individuals with a disability have rights and privileges equal to students without disabilities. Reasonable accommodations may include academic adjustments, auxiliary aids or adaptive technology, services, or modifications for facilities.
**Academic Accommodation Memorandum (Memo):**

The DSS office prepares an Academic Accommodation Memorandum (Memo) for each student receiving accommodations. This document explains the reasonable accommodations to be provided to you as a student of IU to your instructors. It is your responsibility to give a copy of our memo to each of your instructors at the beginning of each academic term.

[Request an Academic Accommodation Memo.](#)

**Submitting Memos to Your Instructors:**

As a student receiving accommodations, you are required to meet with each of your professors in person either during office hours or by individual appointment. **Electronic web meeting may be substituted for online students.** It’s necessary for you to meet your professor early in the semester - within the first two weeks if possible. By the time you leave your meetings with professors, you should have an understanding of how the professor will handle your accommodations during the semester. If you have any questions about the memo or any part of the DSS system and process, don’t hesitate to reach out to your coordinator.

**Course Roadmap - Topic Outline:**

**Module 1: Python Tutorial**

**Topics:**
- Introduction to the course, syllabus
- Installing Python
- Python tutorial

**Key Concepts:**
- Hardware and software, programming languages, Python.

**Learning objectives:**
- Students will learn how to install Python interpreter on their computers, write and execute simple programs in Python in both interpretive and script modes.

**Learning Activities:**
- Read the syllabus
- Read pages 1 - 8 from “Think Python”
- Watch lecture videos
- Practice programming examples presented in the lectures.
- Participate in discussion, question #01:
  - Course instructor and assistant instructors will present their backgrounds and what they currently do with what they will be teaching and what we hope the students will get out of the class. The students will present themselves, their motivation to enroll and what they expect to learn in the course.

**Assessment:**
- Online Quiz #01 contains five questions related to the topics covered in the Module. Due date to complete the quiz is May 13, 2017.
- Programming assignment #01 will be used to measure students’ progress in writing simple Python programs. Due date for submitting the assignment report is May 14, 2017.
Module 2: Python syntax

Topic Overview:
- Software development process
- Python syntax: names, expressions, assignment statements, input, output, eval.
- Keyboard input
- Definite loops.

Key Concepts:
- Software development, Python syntax, definite loops.

Learning objectives:
Students will learn how to design an algorithm according to the specifications and develop a program in Python to get information entered from the keyboard, perform computation, output information to the screen, and perform a counted loop.

Learning Activities:
- Watch lecture videos.
- Read Chapter 2 and Sections 4.2, 5.11, 7.1 and 7.2 from the textbook “Think Python.”
- Practice programming examples presented in the lectures.
- Participate in discussions, question #02:
  - What are the key differences between natural and programming languages? What are the key differences between a recipe and an algorithm?
  - What kind of problems can be solved by repeating the same small set of programming commands over a list of numbers? It’s possible to find the count and average using a single pass. Multiple passes can be used for sorting.
  - How would you draw a box using the character # on the console? How would you draw a diamond inside the box? Can you parameterize your program so that it can draw diamonds and boxes of different sizes?

Assessment:
- Online Quiz #02 contains five questions related to the topics covered in the Module. Due date to complete Quiz is May 20, 2017.
- Programming assignment #02 will be used to assess and measure students’ progress in writing Python programs, which include loops, based on the specifications. Due date for submitting the assignment report is May 21, 2017.

Module 3: Numeric Data Types

Topic Overview:
- Numeric data types
- Math library
- Accumulator and counter patterns
- Limitations of computer arithmetic, handling large numbers
- Conditional execution

Key Concepts:
- Data types, computer arithmetic, conditional execution.
Learning objectives:
Students will learn how to develop and implement a program that uses the math library and accumulator and counter patterns to perform numeric computations.

Learning Activities:
Watch lecture videos.
Read sections 1.5, 2.5, 3.2, 3.3, 5.1, 5.4, 5.5 and 10.7 from the textbook “Think Python.”
Practice programming examples presented in the lectures.
Participate in discussions, question #03:
Why can’t computers accurately represent numbers like 0.1 even though they are not irrational like π (pi)? How do errors like this accumulate if many operations are performed using imprecise floating point representation? What could be done to alleviate this problem? How could a precise representation like int be used to approximate floats at a fixed decimal length?

Log-representation is a technique where instead of storing a number x we store log(x), simplifying multiplication and allowing us to represent an exceptionally large range of numbers at the cost of accuracy. What are the limitations of an approach like this if we only use integers to represent the log10 of numbers? What is the range and granularity of numbers that can be represented?

Assessment:
Online Quiz #03 contains five questions related to the topics covered in the Module. Due date to complete Quiz is May 27, 2017.

We will use programming assignment #03 to assess and measure students’ progress in developing Python programs to perform numeric computations based on math library functions and accumulator and counter patterns. Due date for submitting the assignment report is May 28, 2017.

Module 4: String and Lists

Topic Overview:
String data type, string processing, string methods.
Lists, list methods.
Input/output as string manipulation.
Files and file processing.

Key Concepts:
Strings, lists, input/output, files.

Learning objectives:
Students will learn how to design and implement Python programs based on string and list data types to process textual information.

Learning Activities:
Watch lecture videos.
Read Chapter 8 (Strings) and 10 (Lists) from the textbook “Think Python.”
Practice programming examples presented in the lectures.
Participate in discussions, question #04:
How can you draw an approximate circle on the console? Hint: Create a 2D array of characters and draw into it before printing it to console.

What kind of data types would we need to describe an email? A friend’s contact information? Song? Picture? Video?

What is the computational complexity of indexing an element of a list? What is the computational complexity of appending an element to a list? If a finite amount of memory is allocated at some point to the array then adding elements to it iteratively will in finite time fill that space.

How and why does having the memory size double every time it’s filled optimize the computational time of growing an array by iteratively appending to it (amortized analysis)?

Assessment:

Online Quiz #04 contains five questions related to the topics covered in the Module. Due date to complete Quiz is June 3, 2017.

We will use programming assignment #4 to assess and measure students’ progress in developing and implementing Python programs based on string and list data structures. Due date for submitting the assignment report is June 4, 2017.

Module 5: Functions

Topic Overview:
Functions, function calls.
Parameter passing, pass-by-value, pass-by-reference.
Local variables, scope of variables.
Program structure with cooperating functions

Key Concepts:
Functions, parameter passing, program structure.

Learning objectives:
Students will learn how to define new functions, and develop programs with modular structure based on functions.

Learning Activities:
Watch lecture videos.
Read Chapter 3, Functions, from the textbook “Think Python”.
Practice programming examples presented in the lectures.
Participate in discussions, question #05:

What functions are performed by a salesperson in a convenience store? How could you deconstruct a hierarchy of functions from higher level jobs, over intermediate procedures to lower level decisions, operations and movements? Provide a natural language recipe. Specify inputs and outputs of each function.

Functions should exhibit loose coupling and tight cohesion. At which levels do the described convenience store job functions exhibit these attributes?
Try to describe the high level functions involved in a computer program which plays checkers or chess. How would these differ and which would be similar if the task instead was to program a robot hand and a camera to perform the same task in the real world?

Assessment:

Online Quiz #5 contains five questions related to the topics covered in the Module. Due date to complete Quiz is June 10, 2017.

We will use programming assignment #5 to assess and measure students’ progress in designing and implementing Python programs based on functions. Due date for submitting the assignment report is June 11, 2017.

Module 6: Decision structures

Topic Overview:
Boolean data type and Boolean expressions.
Decision structures, conditions, multi-way decisions.
Exceptions and exception handling.
Good programming practice.

Key Concepts:
Boolean data type, conditional statements, exceptions.

Learning objectives:
Students will learn how to read, write, and implement algorithms and programs that employ decision structures, including those that employ sequences of decisions and nested decision structures.

Learning Activities:
Watch lecture videos.
Read pages 39-43 from the textbook “Think Python.”
Practice programming examples presented in the lectures.
Participate in discussions, question #06:
How would you design an algorithm which based on scores between 0 and 100 assigns letter grades? What if you wanted curve the grades to have the same amount of B’s (25%), C’s (25%) and D’s (25%) and half as many A’s (12.5%) and F’s (12.5%)?

Can you describe the decision process you make when you wake up as a series of if then else statements? Identify the variables upon which decisions are made and a data type which would be useful in the descriptive sense as well as in the comparison. For example the weather may be (very cold), (cold), (mild), (warm), (hot) as it is easier to interpret by humans but may be more easily compared if represented in degrees Fahrenheit.

Assessment:
Online Quiz #06 contains five questions related to the topics covered in the Module. Due date to complete Quiz is June 17, 2017.

Programming assignment #06 will be used to assess and measure students’ progress in developing Python programs with decision and nested decision structures. Due date for submitting the assignment report is June 18, 2017.
Module 7: Loops

Topic Overview:
- Definite and indefinite loops
- Interactive and sentinel loops
- Nested and post-test loops

Key Concepts:
- Program Loops

Learning objectives:
Students will learn how to design and implement solutions to problems involving decision and loop patterns including nested loop structures.

Learning Activities:
- Watch lecture videos.
- Read pages 63-68 from “Think Python”.
- Practice programming examples presented in the lectures.
- Participate in discussions, question #07:

  Every “for” loop can be rewritten as a “while” loop but the opposite is not true without special commands. What makes these two loops so significantly different, what makes them similar? Why is the while loop not always used?

  How would you go about writing a program which draws a right angled triangle filled with #, whose longest side is diagonal on the console, while the shorter sides are vertical and horizontal? Given the length of the vertical side are the other parameters fixed?

  A “for” loop at the start of its execution knows how many times the block of operations indented under it need to be executed. Is it possible for the loop to be executed out of order and still provide the correct result? What are some of the necessary conditions necessary for this to be true? For example adding all the elements in a list may be done out of order.

Assessment:
- Online Quiz #07 contains five questions related to the topics covered in the Module. Due date to complete Quiz is June 24, 2017.

  Programming assignment #07 will be used to assess and measure students’ progress in writing Python program based on loops and nested loops. Due date for submitting the assignment report is June 25, 2017.

Module 8: Objects and Graphics

Topic Overview:
- Concept of objects.
- Computer graphics, graphics library.
- Mouse and text based input in a graphical programming context.

Key Concepts:
- Objects, computer graphics, mouse and text inputs.

Learning objectives:
- Students will learn to write interactive graphic programs using objects from the graphics library.

Learning Activities:
Watch lecture videos.
Read Chapter 15, Classes and Objects, from the textbook “Think Python”.
References to on-line documentation:
https://docs.python.org/3/tutorial/classes.html
http://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml/graphics.html
Practice programming examples presented in the lectures.
Participate in discussions, question #08:

How can the apparent quality of graphics be improved by anti-aliasing? Anti-aliasing is a sub-pixel precision heuristic which attempts to smooth the jagged edges of pixels by introducing shades of color to indicate how filled a pixel is with a particular shape. Remember drawing the circle. How would you go about anti-aliasing the graphics functions in python? What computations would need to be performed to compute the ultimate value of pixels?

How would you go about simulating the filled circle function using only the line function? How would you draw a spiral?

Assessment:

Online Quiz #08 contains five questions related to the topics covered in the Module. Due date to complete Quiz is July 1, 2017.

Programming assignment #08 will be used to assess and measure students’ progress in developing interactive graphics programs using objects from the graphics library. Due date for submitting the assignment report is July 2, 2017.

Module 9: Numerical Python

Topic Overview:
Multidimensional arrays for vectorized arithmetic operations
Standard mathematical functions for fast operations on entire arrays of data
Linear algebra operations

Key Concepts:
Numerical Python, vectorized computation, linear algebra operations

Learning objectives:
Students will learn how to use Numerical Python package in writing data analysis programs involving multidimensional arrays and vectorized numeric operations.

Learning Activities:
Watch lecture videos.
Read short numpy tutorial:
https://engineering.ucsb.edu/~shell/che210d/numpy.pdf
Practice programming examples presented in the lectures.
Discussion Question #09

Why are visualizations important to data science?

How do human visual abilities compare to that of a machine? What are some examples of imitating the human visual ability (Facebook, how-old.net, …)

For a given set of tabular data how can we use visualization to discover correlations between attributes?
A graphical line can be described as a linear function $w_1 \cdot x + w_2 = y$. Using the linear function to describe the line, how can we find out where the line cross the $x$- and $y$-axis? Hint: for the $x$ axis the values of $x = 0$ and for the $y$ axis $y = 0$.

Assessment:

Online Quiz #09 contains five questions related to the topics covered in the Module. Due date to complete Quiz is July 8, 2017.

Programming assignment #09 will be used to assess and measure students’ progress in developing data analysis programs based on Python NumPy package. Due date for submitting the assignment report is July 9, 2017.

Module 10: Python Package pandas

Topic Overview:

- Series and DataFrame data structures
- Indexing and slicing pandas objects
- Operations and function applications on pandas objects

Key Concepts:

- pandas, Series, DataFrame

Learning objectives:

- Students will learn how to use Python package pandas in writing data analysis programs based on Series and DataFrame.

Learning Activities:

- Watch lecture videos.
- Read Chapter 6, “10 minutes to pandas” from “pandas: powerful Python data analysis toolkit” by Wes McKinney (available online)
- Read pages 111 - 142 from “Python for Data Analysis.”
- Practice programming examples presented in the lectures.

Participate in discussions, question #10:

- In the 3D Euclidean space we live in, distance is an important measure. It however loses relevance in higher dimensional spaces

To illustrate this concept compute and display the all the distances between a set of 100 multivariate Gaussian points with mean 0 and variance unity in 2, 5, 10, 20, 50 and 100 dimensions. How does increasing dimensionality affect distances?

Another illustration of this concept is computing the volume of a hyper-cube with an edge of size 1 and a hyper-cube of edge length 0.9. Derive the formula for the volume of a hyper-cube given its edge length and the number of dimensions. How does increasing dimensionality affect the ratio of volumes of cubes of edge length 1 and 0.9?

What approaches do you think are useful to visualize high dimensional data (beyond 3 dimensions)?

Assessment:
Online Quiz #10 contains five questions related to the topics covered in the Module. Due date to complete Quiz is July 15, 2017.

Students will start working on the Final project, in which they will use Python to implement k-means algorithm. Phase 1 of the Final project will be used to assess student progress in writing programs to impute missing values, plot basic graphics and compute description of data. Due date for submitting the Phase 1 report is July 16, 2017.

Module 11: Plotting

Topic Overview:
- Python package "matplotlib"
- Figure object and subplots
- Plotting Series and DataFrames
- Handling missing data

Key Concepts:
- Matplotlib, plotting, missing data.

Learning objectives:
- Understand and use matplotlib in making publication-quality plots.
- Plot Series and DataFrame objects as lines, bars and scatter plots.
- Filter and fill missing data in Series and DataFrame.

Learning Activities:
- Watch lecture videos.
- Read matplotlib tutorial, available at: https://www.labri.fr/perso/nrougier/teaching/matplotlib/
- Learn how to use matplotlib online documentation: http://matplotlib.org/
- Practice programming examples presented in the lectures.
- Participate in discussions, question #11:
  - Why is it important to have a solid specification before coding to ensure adequate testing?
  - How can we assess the length of time it will take to complete a programming task? What are the important attributes related to the task and related to the programmer?
  - Describe the hierarchy of testing that occurs during the development of software, from programmers, to QA, to end-users. What are your experiences with error reporting by the end-user?

Assessment:

Online Quiz #11 contains five questions related to the topics covered in the Module. Due date to complete Quiz is July 22, 2017.

Students will continue working on the Final project, in which they will use Python to implement k-means algorithm. Phase 2 of the Final project will be used to assess student progress in writing program to implement k-means algorithm, including initialization, assignment and recalculation steps. Due date for submitting the Phase 2 report is July 23, 2017.
Module 12: Object Oriented Concepts

Topic Overview:
- Objects, classes, encapsulation
- Data processing with classes
- Object oriented graphical user interface

Key Concepts:
- Objects, classes, graphical user interface

Learning objectives:
- Students will learn how to write object oriented programs for graphical user interface.

Learning Activities:
- Watch lecture videos.
- Read Chapters 15, 16 and 17 from the textbook "Think Python."
- Practice programming examples presented in the lectures.
- Participate in discussions, question #12:
  Strong cohesion relates to how a single module’s components should be directly related to one another, while loose coupling relates to how modules should provide a generic interface to using them from other modules. Why is strong cohesion and loose coupling important in software development?

Assessment:
- Online Quiz #12 contains five questions related to the topics covered in the Module. Due date to complete the quiz is July 26, 2017.

Students will start working on the Final project, in which they will use Python to implement k-means algorithm. Phase 1 of the Final project will be used to assess student progress in writing programs to impute missing values, plot basic graphics and compute description of data. Due date for submitting the Phase 1 report is July 26, 2017.