BRONX COMMUNITY COLLEGE
of City University of New York
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

SYLLABUS: CSI32 Introduction to Computer Programming II 3 credits / 4 hours

PREREQUISITE: CSI31 or departmental permission, ENG02, RDL02 if required
TEXT: Object-Oriented Programming in Python, by Goldwasser and Letscher,
Pearson/Prentice Hall, 1st Edition, 2008
Please note that the book is out of print, however you can download it from here:
http://cs.slu.edu/~goldwasser/oopp/
Software: Python 3.4 or later; DIA v0.97 (recommended structured diagram software)

Content:
In this class we will study basics of object-oriented design (OOD) and object-oriented programming (OOP) using the Python language. Learn to use Unified Modeling Language (UML) diagrams (class, sequence, activity and state diagrams) as a design tool. Discuss modules, types, classes, inheritance, methods, constructors, and recursion.

Objectives:
1. To deepen the student's understanding of Python as an OO language to a level where other OO languages such as C++ or Java can be easily assimilated.
2. To regard every variable as an object of some class, and to review the built-in types from this perspective.
3. To provide the student with opportunities to use OOD/OOP to design correctly and to implement a programming project.

Students will complete 8 to 10 small programming assignments selected from the list of suggested exercises or comparable assignments developed by the instructor, and one big (final) project.

Attendance Policy :

- Students who miss more than 5 classes, without a genuine and documented reason will be assigned an academic grade of F.
- Students who come to the class late by 20 minutes or more for the first time should consider themselves as warned. Continuous or habitual late-coming or early departure will be considered as absence.

	Section(s)	**Homework Assignment(s)**
Lecture 1 Data and Types Operations, Functions, and Algorithms Conditional Statements (review)	 1.1, 1.2, 4.4	
Lecture 2 For loops (review), Case Studies: DNA to RNA, While loops (review), Flowcharts, Dia editor	 4.1, 4.2, 5.1	
Lecture 3 Object-Oriented Paradigm,	 1.4	

UML (activity diagram, class diagram, sequence diagram)		
Lecture 4 Good Software Practices Using objects: the **list** class, Other Sequence Classes: **str** and **tuple,** Numeric Types: **int**, **long**, and **float,** Type Conversions	Chapter 7 2.2, 2.3, 2.4, 2.5	
Lecture 5 Exercise 2.37: DNA mutation List comprehension Calling Functions, Python Modules, Expressions	p. 87, exercise 2.37 4.5, 2.6, 2.7, 2.8	
Lecture 6 Functions (review), Case Study: Computing the Square Root Error Checking and Exceptions	5.2, 5.4, 5.5	
Lecture 7 The Canvas, Drawable Objects	3.1, 3.2	
Lecture 8 Rotating, Scaling, and Flipping; Cloning (optional)	3.3, 3.4	
Lecture 9 A Fraction Class	6.4	
Lecture 10 Set Class BinaryNumber Class	p. 234, exercise 6.15 p. 235, exercise 6.18	
Lecture 11 Game design and implementation	Chapter 7	
Lecture 12 Inheritance: Augmentation, Specialization, When Should Inheritance (Not) Be Used	9.1, 9.2, 9.3	
Lecture 13 Class Hierarchies and cs1graphics	9.4	
Midterm Examination (10/19)		
Lecture 14 Basics of Event-Driven Programming Event Handling in out Graphics Module	15.1, 15.2	
Lecture 15 The Event Class Programming Using Events	15.3, 15.4	

Lecture 16 Standard Input and Output, Formatted Strings, Working with Files, Handling Newline Characters, Case Studies	 8.1, 8.2, 8.3, 8.4, 8.5	
Lecture 17 A Bullseye Class Case Study: Drawing a Pyramid	 11.1, 4.3	
Lecture 18 Functional Recursion Binary Search	 11.3, 11.4	
Lecture 19 Two Familiar Containers: **list** and **tuple** **Dictionaries** **Containers of containers** **Arrays**	 12.1, 12.2, 12.3, 12.5	
Lecture 20 A Network Primer Writing a Basic Client	 16.1, 16.2	
Lecture 21 Basic Network Servers Case Study: Network Chat Room	 16.3, 16.4	
Lecture 22 Peer-to-Peer: Instant Messenger	 16.5	

Fall 2016 / NN