NEW JERSEY CENTER
FOR TEACHING & LEARNING

Progressive Science Initiative® (PSI®)
CSCI4333: Programming in Python

Primary Student Contact:	Maria Surace	maria@njctl.org
Faculty Team:	Dr. Bob Goodman	bob@njctl.org
	Maria Surace	maria@njctl.org
	Katy Goodman	katy@njctl.org

Course Credit: 3.0 NJCTL credits

COURSE DESCRIPTION:

This course is for teachers to learn the content of *PSI Programming in Python* and how to teach that course to students. It focuses on fundamental programming skills and thought processes required for successful programming in any language while integrating components of Python. Topics include an Introduction to Programming; Operators & Logic; Algorithmic & Control Structures; Lists; Functions; Classes; Inheritance; and Libraries & APIs.

STUDENT LEARNING OUTCOMES:

Upon completion of the course, the student will be able to:

1. Demonstrate proficiency in programming in Python, as listed in the module learning outcomes below.
2. Integrate PSI materials (including presentations, labs, practice problems, etc.) to support student learning and deliver effective instruction.
3. Create a social constructivist learning environment through the use of formative assessment questions, interpreting the results of this assessment to effectively facilitate student-led discussions that support deeper understanding of the content.
4. Integrate multiple attempts to demonstrate student mastery of content knowledge, as encouraged/fostered by the PSI pedagogy.
5. Implement learning plans that are aligned to secondary computer science standards and allow for differentiation based on the needs of learners.

PROGRESSIVE SCIENCE INITIATIVE®, PSI®, PROGRESSIVE MATHEMATICS INITIATIVE® and PMI® are registered trademarks and PROGRESSIVE TEACHING INITIATIVE and PTI are trademarks of Dr. Robert Goodman and the New Jersey Center for Teaching and Learning is the exclusive Licensee of these marks

TEXTS, READINGS, INSTRUCTIONAL RESOURCES:

Required Texts:

- PSI Programming in Python uses a free digital textbook accessible at: https://njctl.org/materials/courses/programming-in-python/
- Participants will download SMART Notebook presentations, homework files, labs, and teacher resources from the PSI Programming in Python course

COURSE REQUIREMENTS:

In order to receive a Passing grade, the participant must complete the following course requirements:

1. Activities: A number of different learning activities will ensure participant engagement and learning in the course. These include:
   - Engage in video module lessons which demonstrate minimized direct instruction followed by frequent formative assessment
   - Completion of formative assessments aligned to learning objectives which include detailed analysis when answered incorrectly.
   - Interaction with module discussion boards that allow conversation with peers and course instructors about the module's content, delivering that content to students. Discussion boards also serve as a place to ask and answer questions related to the module's content.
2. Mastery Exercises: For each module, these multiple-choice question quizzes assess the content knowledge gained in a module. Participants have the opportunity to retake; random questions are pulled from a larger question bank on each attempt ensuring varied questions.
3. Programming Assignments/Labs: In each module, a Python file is submitted to demonstrate an understanding of the graded programming assignment. These assignments are
   problem-based questions that require the writing of a program. These promote a deeper understanding of logical reasoning and the applications of programming. A culminating lab activity occurs in the final module.
4. Module Exam: One is completed at the end of each module. It is a culminating exam consisting of multiple choice and free response questions aligned to the standards and objectives of the module.
5. Final Exam: At the end of the course, a comprehensive exam consisting of Multiple Choice and Free Response questions assesses the content knowledge learned throughout the course and aligns to the AP College Board Exams.

GRADE DISTRIBUTION AND SCALE:

Grade Distribution:

Module Exams	65%
Final Exam	15%

Programming Assignments/Labs	15%
Mastery Exercises	5%

Grade Scale:

A	93 – 100
A-	90 – 92
B+	86 – 89
B	83 – 86
B-	80 – 82
C+	77 – 79
C	73 – 76
C-	70 – 72
D	60.0 – 69.9
F	59.9 or below

ACADEMIC STANDING:
NJCTL has established standards for academic good standing within a student's academic program. Students enrolled in any NJCTL online course must receive an 80 or higher to successfully complete a course and receive credit for that course. An 80 is equivalent to a GPA of 2.7 or B-. Additionally, students in an endorsement program must receive a cumulative GPA of 3.0 for all courses combined in order to successfully complete the program.

ACADEMIC INTEGRITY:
Students must assume responsibility for maintaining honesty in all work submitted for credit and in any other work designated by the instructor of the course. Academic dishonesty includes cheating, fabrication, facilitating academic dishonesty, plagiarism, reusing /repurposing your own work, unauthorized possession of academic materials, and unauthorized collaboration.

CITING SOURCES WITH APA STYLE:
All students are expected to follow proper writing and APA requirements when citing in APA (based on the APA Style Manual, 6th edition) for all assignments.

DISABILITY SERVICES STATEMENT:
We are committed to providing reasonable accommodations for all persons with disabilities. Any student with a documented disability requesting academic accommodations should contact the Dean of Students, Melissa Axelsson, for additional information to coordinate reasonable accommodations for students with documented disabilities (melissa@njctl.org).

NETIQUETTE:
Respect the diversity of opinions among the instructor and classmates and engage with them in a courteous, respectful, and professional manner. All posts and classroom communication must be conducted in

according with the student code of conduct. Think before you push the send button. Did you say just what you meant? How will the person on the other end read the words?

Maintain an environment free of harassment, stalking, threats, abuse, insults or humiliation toward the instructor and classmates. This includes, but is not limited to, demeaning written or oral comments of an ethnic, religious, age, disability, sexist (or sexual orientation), or racist nature; and the unwanted sexual advances or intimidations by email, or on discussion boards and other postings within or connected to the online classroom.

If you have concerns about something that has been said, please let your instructor know.

CLASS SCHEDULE:

Module	Module Learning Outcomes	Assignments
1 – Introduction to Programming	• Differentiate between data types, strings and variables. • Concatenate strings. • Understand coding input and output. • Correctly use comments in Python to annotate your programs. • Debug Python code when basic errors are identified.	• Graded Programming Assignment • Mastery Exercise • Module Exam
2 – Operators & Logic	• Write comparisons in Python using math operators, including modulus. • Apply order of operations correctly, as it pertains to writing simple programs. • Write programs using comparison and logical operators. • Write a program using `Random()`.	• Graded Programming Assignment • Mastery Exercise • Module Exam
3 – Algorithms & Control Structure	• Differentiate between what is and what is not an algorithm. • Differentiate between the three different types of control structures; sequence, iteration and selection. • Write efficient programs using the most appropriate control structure as an organizational planning structure. • Write code using nesting to repeat specific portions of a program.	• Graded Programming Assignment • Mastery Exercise • Module Exam
4 - Lists	• Create lists in Python. • Traverse and search lists. • Compare strings and lists to find the commonality of each.	• Graded Programming Assignment • Mastery Exercise • Module Exam
5 – Functions	• Utilize basic functions in programs. • Create lists using functions. • Return values and vary parameters.	• Graded Programming Assignment • Mastery Exercise • Module Exam
6 - Libraries & APIs	• Select and implement appropriate built in ArrayLists to perform key tasks when writing programs. • Create array lists to increase the versatility and complexity when writing programs. • Understand the elements of an ArrayLists and how to create, modify or refer to them. • Understand the differences of arrays and ArrayLists and when to integrate them accordingly when writing programs. • Compare and contrast simple data structures to program with object orientation in mind. • Understand the existence and scope of variables and how they are affected through various actions throughout the structure of a program. • Examine and debug compiler and runtime errors within programs.	• Graded Programming Assignment • Mastery Exercise • Module Exam

	• Understand how content is implemented in a student-centered classroom as it applies to logic and algorithmic design.	
7 - Classes	• Create classes to model and define programs. • Write user-defined classes to divide a complex problem into simpler parts. • Describe how various data types and objects are stored. • Write code segments to access user-defined methods and variables within classes. • Differentiate between the use of public, private and static types when writing programs. • Understand the scope of a variable within a class through the construction of object-oriented programs. • Compare and contrast simple data structures to program with object orientation in mind. • Examine and debug compiler and runtime errors within programs. • Understand how content is implemented in a student-centered classroom as it applies to logic and algorithmic design.	• Graded Programming Assignment • Mastery Exercise • Module Exam
8 - Inheritance	• Understand how inheritance works between classes and implement appropriately when writing a program. • Define subclasses from superclasses using inheritance. • Understand polymorphism and dynamic binding through the constructs of a program. • Use downcasting when appropriate in program writing. • Invoke subclasses and superclasses appropriately within the context of a program. • Use appropriate data structures when writing programs to analyze and interpret massive data collections. • Understand the existence and scope of variables and how they are affected through various actions throughout the structure of a program. • Examine and debug compiler and runtime errors within programs. • Understand how content is implemented in a student-centered classroom as it applies to logic and algorithmic design..	• Graded Programming Assignment • Mastery Exercise • Module Exam
9 - Final Exam	• Review module topics • Discussion board posts and Zoom meeting with instructor, as needed	• Final Exam