COURSE TITLE

Computer Programming 1

LENGTH

One Semester
Grades 9-12

DEPARTMENT

Computer Department
Barbara O'Donnell, Supervisor

SCHOOL

Rutherford High School

DATE

Spring 2017

COMPUTER PROGRAMMING 1

I. Introduction/Overview/Philosophy

Computer Programming 1 is designed for students who wish to gain an understanding of computer coding or who plan to further their knowledge of computer design and computer programming in upper level high school computer classes, college, business, or vocational schools. This course provides an introduction to the Python programming language. Students will learn the fundamental concepts of programming—concepts that can be applied in the study of any programming language. Students will also dive into specific features of the Python programming language. The course utilizes a blended classroom approach. The content is fully web-based, with students writing and running code in the browser. Teachers utilize tools and resources provided by CodeHS to leverage time in the classroom and give focused 1-on-1attention to students. This course will provide a foundation for more advanced computer programming languages and computer design techniques

II. Objectives

Course Outline

I. Getting Started with Python
   - A. Introduction to Python / Computer Science
   - B. Introduction to OOP (Object-Oriented Programming) with graphics
   - C. Introduction to use of basic commands

II. Basic Python and Console Interaction
   - A. Printing to the console
   - B. Learn debugging techniques
   - C. Declaration, initialization, and assignment of variables
   - D. Data types (int, double, string, boolean)
   - E. Translation of arithmetic expressions
   - F. Using and converting keyboard input

III. Control Flow
   - A. Use of comments for program understanding
   - B. Use of Decision Statements
     1. If statements
     2. Conditional operators
     3. Boolean values
   - C. Use of Iteration
     1. While loops
     2. For loops
     3. Nested loops
   - D. Introduction to Functions
     1. User defined functions
     2. Built-in functions
   - E. Exception handling and identification

IV. Strings
   - A. Manipulation of strings
   - B. Math operators on strings
   - C. Traverse through a string

If time allows and based upon student needs the following topics can/will be covered:

V. Data Structures
   A. Tuples
   B. Lists
   C. Dictionaries
   D. 2D Lists
   E. List comprehensions
   F. Packing and unpacking
   G. Mutable vs. immutable
   H. Equivalence vs. identity
VI. Classes and objects
   A. Classes
   B. Attributes
   C. Class variables vs. instance variables
   D. Methods
   E. Built-in methods
   F. Composition, inheritance and polymorphism
   G. Private attributes

Student Outcomes:

Upon completion of the course, students will demonstrate the ability to:

- Code in the Python language fluently in a well-structured fashion
- Understand the concepts of object-oriented programming as used in Python including control statements, loops, functions, and lists.
- Develop an intuition for syntax in program creation
- Analyze and design strategies for solving basic programming problems by using logical reasoning

NEW JERSEY STUDENT LEARNING STANDARDS

TECHNOLOGY

Standard 8.1: Educational Technology: All students will use digital tools to access, manage, evaluate, and synthesize information in order to solve problems individually and collaboratively and to create and communicate knowledge.

Standard 8.2: Technology Education, Engineering, and Design: All students will develop an understanding of the nature and impact of technology, engineering, technological design, and the designed world, as they relate to the individual, global society, and the environment.

Strand E. Computational Thinking: Programming: Computational thinking builds and enhances problem solving, allowing students to move beyond using knowledge to creating knowledge.

21ST CENTURY LIFE AND CAREERS

Standard 9.2: Career Awareness, Exploration, and Preparation
Standard 9.3 – Career & Technical Education (CTE)

Pathway: Programming & Software Development (IT-PRG)

III. Proficiency Levels

This course is open to grades 9 -12.

IV. Methods of Assessment

Student Assessment

The teacher will provide a variety of assessments during the course of the year. Among these are: homework, laboratory exercises, weekly projects, teacher-made tests and quizzes, and long-term projects.

Curriculum/Teacher Assessment

The teacher will provide the subject area supervisor with suggestions for changes on an ongoing basis.

V. Grouping

There is no prerequisite for Computer Programming 1. Students completing this course will have sufficient knowledge and experience with data types and structures to be successful in advanced programming and design courses such as C++, Java, etc.

VI. Articulation/Scope & Sequence/Time Frame

Course length is one semester and is offered to students in grades 9-12.

VII. Resources

CodeHS: Introduction to Computer Science in Python https://codehs.com/info/curriculum/intropython

How to Think Like a Computer Scientist: Interactive Edition
http://interactivepython.org/runestone/static/thinkcspy/index.html

Automate the Boring Stuff with Python https://automatetheboringstuff.com/

https://codeacademy.com - an online interactive platform that offers free coding classes

VIII. Methodologies

The course utilizes a blended classroom approach offered through CodeHS. The content is fully web-based, with students writing and running code in the browser. Teachers utilize tools and resources provided by CodeHS to leverage time in the classroom and give focused 1-on-1 attention to students. Each unit of the course is broken down into lessons. Lessons consist of video tutorials, short quizzes, example programs to explore, and written programming exercises, adding up to over 100 hours of hands-on programming practice in total. Each unit ends with a comprehensive unit test that assesses student's mastery of the material from that unit. Much of the class time is spent in lab work with required hands-on exercises to be completed.

IX. Suggested Activities

- Laboratory programming problems
- Game simulated programs
- Cooperative programming projects

X. Interdisciplinary Connections

Connections are made to mathematics by using a variety of arithmetic formulas, as well as higher mathematical concepts. Connections are also made to the disciplines of business, art and English, by means of incorporation of theses ideas into programming projects.

XI. Differentiating Instruction for Students with Special Needs: Students with Disabilities, English Language Learners, and Gifted & Talented Students

Differentiating instruction is a flexible process that includes the planning and design of instruction, how that instruction is delivered, and how student progress is measured. Teachers recognize that students can learn in multiple ways as they celebrate students' prior knowledge. By providing appropriately challenging learning, teachers can maximize success for all students.

Examples of Strategies and Practices that Support:

Students with Disabilities

- Use of visual and multi-sensory formats
- Use of assisted technology
- Use of prompts
- Modification of content and student products
- Testing accommodations
- Authentic assessments

Gifted & Talented Students

- Adjusting the pace of lessons
- Curriculum compacting
- Inquiry-based instruction
- Independent study
- Higher-order thinking skills
- Interest-based content
- Student-driven
- Real-world problems and scenarios

English Language Learners

- Pre-teaching of vocabulary and concepts
- Visual learning, including graphic organizers
- Use of cognates to increase comprehension
- Teacher modeling

- Pairing students with beginning English language skills with students who have more advanced English language skills
- Scaffolding
    - •word walls
    - •sentence frames
    - •think-pair-share
    - •cooperative learning groups

XII. Professional Development

The teacher will continue to improve expertise through participation in a variety of professional development opportunities.

XIII. Curriculum Map

Month	Topics
September/February	Introduction to Python / Computer Science Introduction to OOP with graphics Introduction to use of basic commands Printing to the console Learn debugging techniques Declaration, initialization, and assignment of variables
October/March	Data types (int, double, string, boolean) Translation of arithmetic expressions Using and converting keyboard input Use of comments for program understanding BENCHMARK PROJECT
November/April	Use of Decision Statements Applications of Loops (While, For, Nested)
December/May	Introduction to Functions (Built-in vs. User Defined) Exception handling and identification
January/June	Manipulation of strings Math operators on strings Transverse through a string BENCHMARK PROJECT
Other Topics (That May Be Covered)	Data Structures: ● Tuples ● Lists ● Dictionaries ● 2D lists ● List comprehensions ● Packing and unpacking

	● Mutable vs. immutable ● Equivalence vs. identity Project: ●Allow students to combine a variety of topics (strings, loops, booleans, user input, etc.) in a single program ● Introduce students to incremental development ● Strengthen debugging skills by having students develop a larger project Classes and Objects: ● Classes ● Attributes ● Class variables vs. instance variables ● Methods ● Built-in methods ● Composition, inheritance, and polymorphism ● Private attributes