# Computer Programming

**2020-2021**
**Grades 11-12**
Prerequisite(s): Intro to IT
& Computer Hardware & System Support

## Course Description

Computer Programming uses the Python programming language as well as Java and HTML to introduce students to basic programming skills and the programming languages. Students learn the principles of programming. The course begins with the foundation of all programs: algorithms; then it lays a foundation for programming syntax, variables, operators, and control structures. Students use models and mini programs as a way to quickly learn the basics. After this foundation is established, students will design programs and write functions from scratch. In addition to learning the foundational skills for computer programming, students learn program design, documentation, formal debugging, and testing.

## Course Units/Objectives

**This course is broken into 3 units:**



Back to Top

## UNIT 1: PROGRAMMING BASICS (PYTHON):

In this unit the focus is on programming fundamentals using the most syntax friendly Python language. Basics include: syntax, errors, input/output statements, variables, data types and comments. Scholars will create simple lines of code that demonstrate. As they learn the logic behind programming code they will also gain a solid understanding of syntax, variables, types, understanding how to read error messages and debugging code.

**UNIT 1 UNDERSTANDINGS:**

**U1**: Programming is the "software" side of IT, and how programs and programmers differ based on both the hardware they interact with and the platform "purpose" of the programs.

**U2:** Programming uses mathematical and logical concepts or Algorithms; and how this mathematical concept has become the foundation of programming.

**U3:** Python is one of the most used because of the structure and simplicity of the language and the many platforms it can be used on to develop software programs.

**U4**: The fundamental programming concepts in the Python programming language and how they are used to create simple programs, graphics and games.

| Skills | Knowledge |
|---|---|
| 1. Explaining ways in which programs are being used today<br>2. Recognizing and defining the structure and components of a Python program IDE or platform that make it useful to software developers<br>3. Utilize more than one Python interface to determine common features and to determine user friendliness<br>4. Name elements of their code by following specific guidelines and rules in order to create readable and working programs | ❖ Why programming is a useful skill<br>❖ Why Python is a useful language for software development and what industries use it.<br>❖ How to use various **Python interfaces** available on the market today<br>❖ What **Top Down Design** is and how it is used in programming |
| 5. Define/Describe basic programming terms | ❖ **Programming Vocabulary** Basics:<br>➢ Comments, Debugging, Errors, Floats, Input, Integer, Mathematical Operators, Output, Strings, Statements, Syntax, Types, Variables |

Back to Top

| | |
|---|---|
| 1. Discuss why variables are used in our programs<br>2. Create variables using assignment statements and naming rules<br>3. Assign and convert variable types | ❖ What **variables** are and how they are used in programming. |
| 1. Incorporate user input into their code in order to customize their programs<br>2. Create programs that take in user input, do simple computations with the input, and produce useful output<br>3. Use mouse click events to create programs that respond to user clicks | ❖ How programs take in user i**nput/and produce output** to the screen<br>❖ How user input can be taken from the user's mouse using the **mouse clicked method.** |
| 1. Describe why comments are helpful for both themselves and anyone else looking at their code; use comments throughout their program<br>2. Incorporate comments into their programs in order to make them more readable | ❖ How and why **comments** are used by programmers inside of a set of code. |
| 1. Describe the different mathematical operators that can be used in their programs<br>2. Create programs that use basic math to compute useful things | ❖ How **mathematical operators** are used in programs. |
| 1. Use indexing in order to find a specific character in a string<br>2.<br>3. Use various string methods to alter string values<br>4. Explain what immutability is and how this applies to strings in Python<br>5. Use the in keyword to check if a character is in a string | ❖ What strings are and how string indexing, string slicing, string methods are used to find or alter characters in a string<br>❖ In Python, strings have the property of **"immutability**" which means they cannot be mutated or changed. |

Back to Top

| | |
|---|---|
| 6. Use mathematical operators with strings<br>7. Perform string operations in order to concatenate values together | ❖ how the **in keyword** can be used in an if statement to see if a particular letter or substring is in a string. |
| 1. Create graphical Python programs that draw shapes on the canvas<br>2. Using basic Tracy- graphics commands: movement, angles, shapes, pen, fill<br>3. Locate points on the graphics canvas using (x, y) coordinates | ❖ How using geometric concepts, multiple **graphic objects** can be created in Python<br>❖ **Graphic creation** relies on setting the type, shape, size, position, and color on the artist's canvas before adding to the screen. |

**UNIT 1 - PERFORMANCE TASK:**

Scholars will be using the turtle graphics program to demonstrate what they have learned about basic coding in Python in the form of movement, colors, speed, grids and screen dimension, if/ then and loops; they will also use print statements for strings and numbers as well as asking for input from the user.

**Goal**: To create a simple program that allows the user to place their name on the screen, and to choose either the colors of the shape that the turtle will be drawing. Simple means that they were able to parse down their code into FOR loops to reduce the need for massive lines of code.

**Role**: High School Computer Programming student

**Audience**: Pre-K/ Kindergarten age students

**Situation**: Create a simple program that allows for the input of the user's name, allows them to choose a color for their pen and then draws a shape or series of shapes on the screen

**Product**: Simple graphics program that can be used to teach shapes and colors to pre-k or kindergarten level students.

Back to Top

**UNIT 2: SETTING STRUCTURE IN A PROGRAM (PYTHON):**

In this unit, we are still in Python but move from basic syntax and coding to how programs are structured to get the results needed also known as *structure*. In this unit students will begin to identify different decision structures that control program flow, i.e. control structures. Manipulate and output data using strings, conditionals, looping, and operators

**UNIT 2- UNDERSTANDINGS:**

**U1** Why all computer languages have common structures and how these structures are used to help computers do their work.

**U2** How programmers utilize lists and dictionaries to structure the store and manipulate of data; and which one is an appropriate data structure for a given application.

**U3** How Sequential( Operators), Selection(Booleans) and Repetition (Loops) are used by programmers to improve both the efficiency and effectiveness in the structure of written code.

**U4** The role functions and classes play in Object Oriented Programming for structure and problem solving.

| Skills | Knowledge |
|---|---|
| 1. create and store information in tuples<br>2. explain the characteristics of a tuple<br>3. understand and explain the characteristics of a list<br>4. use lists to store and recall information<br>5. use for loops to go through items in a list<br>6. apply useful list methods to alter and access information about a list | ❖ How to create and alter **data structures** using tuples, lists, and list methods |
| 1. Use dictionaries to structure data | ❖ How **dictionarie**s differ from other data structures and why they are useful. |
| 1. Identify the different control structures we can use to modify the flow of control through a program<br>2. Combine control structures to solve | ❖ How different **control structures** we can use to modify the flow of control through a program |

Back to Top

| | |
|---|---|
| complicated problems<br>3. Choose the proper control structure for a given problem | |
| 1. Create boolean variables to represent meaningful yes/no values<br>2. Print out the value of a boolean variable<br>3. Predict the boolean result of comparing two values<br>4. Print out the boolean result of comparing values | ❖ What **Booleans** are and how they are used to test whether a condition is true or false. |
| 1. Explain the meaning of each of the comparison operators (<, <=, >, >=, ==, !=)<br>2. Create programs using the comparison operators to compare values | ❖ How **comparison operators** give the ability to compare two values which allow programs to make decisions. |
| 1. Describe the meaning and usage of each logical operator: or, and, and NOT (!)<br>2. Construct logical statements using boolean variables and logical operators | ❖ How **Logical operators** give the ability to connect or modify Boolean expressions. |
| 1. Use if statements for control flow in their programs<br>2. Use if/else statements in order to make decisions between multiple scenarios | ❖ How **If Statements** (Selection) allow the use of conditions to determine how their code should run |
| 1. Effectively use while loops in their programs<br>2. Identify and resolve infinite loops<br>3. Explain the critical difference between break and continue<br>4. Describe why a break or continue | ❖ How **Iterations** - **while loops/for loops** allow code to be executed repeatedly based on a condition<br>❖ How loops can be written in a way that causes an error or **infinite loop**<br>❖ How break and continue statements are |

Back to Top

| | |
|---|---|
| statement would be needed in a coding scenario | used in a while loop |
| 1. Create for loops to repeat code a fixed number of times<br>2. Explain when a for loop would be a useful tool<br>3. Utilize for loops to write programs that would be difficult / impossible without loops<br>4. Use i as a variable inside their for loop to control different commands | ❖ How **For loops** execute the same lines of code a given number of times |
| 1. Identify the different control structures that can be used to modify the flow of control through a program<br>2. Combine control structures to solve complicated problems<br>3. Choose the proper control structure for a given problem | ❖ That **nested control structures** are control structures within control structures. |
| 1. Explain why functions are used<br>2. Define a function/Call a function<br>3. Incorporate parameters into their functions in order to adapt their functions to multiple situations<br>4. Chain functions together using return values | ❖ **Function**s as the main building block of complex Python programs.<br>❖ How Functions let break programs into different parts that can be organized and reused |
| 1. Explore Python's way of handling errors with exceptions. | ❖ That **Exception**s are run-time errors that stop the program from running |
| 1. Define class and object as well as create them inside their programs.<br>2. Create an init method to give attributes to objects. | ❖ What **classes and objects** are<br>❖ How to break code down in this structure.<br>❖ How to use the **init method** to give |

Back to Top

| | |
|---|---|
| 3. Create methods inside class definitions and call them on objects.<br><br>4. Describe the differences between class and instance variables.<br><br>5. Use inheritance to make classes that are given attributes by other classes.<br><br>6. Describe the different namespaces with regards to classes and objects. | attributes to objects.<br><br>❖ How to define and call **methods** on objects.<br><br>❖ What difference is between **class and instance variables** and how they are located.<br><br>❖ How to use **inheritance** to make classes that are given attributes by other classes.<br><br>❖ How **attribute**s are linked between objects and classes and the path they follow based on namespaces |
| 1. Import and use modules in their programs. | ❖ What are **modules** and what are the different ways to import modules to be used in our programs. |

**UNIT 2 - PERFORMANCE TASK:**

Scholars will be writing the code to simulate the popular game of Mastermind. This code will allow them to practice data structures, control structures.

**Goal:** To create a program that allows the user to play against the computer is a simple Guess the Number game called Mastermind.

**Role:** You are a college freshman in Software Engineering

**Audience:** You will be sharing your program with the other students in the class - playing their programs to make sure they are working and error free.

**Situation:** Your professor has given you an outline for creating an online game called Mastermind - you must do so using Python as your coding platform.

**Product**: Code that demonstrates the following:

1. Generate A List Of Numbers That The User Has To Try To Guess.
2. Create A Guess List From The User.
3. Create A Function To Check If The Numbers In The User List Match The Computer List. The Output Of The Function Should Be A List Of Red, White And Black.
4. Need To Shuffle The Array So The Order Of The Return Value Does Not Correspond To The Correct Position.

Back to Top

5. Create A Check_win() Function That Will Take The Response List As A Parameter, And Will Print A Message To The User That They Have Won
6. Create A Play_game Function That Uses A While Loop To Track The Number Of Guesses That A User Still Has. You Should Print The Total Number Of Guesses Left, And Display A Message If The User Loses After 5 Guesses. Your Function Should Break If The User Wins!
7. Add An Explanation For How To Play The Game So That The User Understands How The Game Works.
8. Create Comments Throughout Explaining Each Step

**UNIT 3 INTRODUCTION TO JAVA PROGRAMMING - (JAVA/JAVA SCRIPT LANGUAGES)**

In this unit students learn the basics of the Java and Javascript programming languages. These modules introduce printing, variables, types, as well as how to use the basic control structures in the Java languages. The Javascript module has an additional video game component which is the final project for the course.

**UNIT 3- UNDERSTANDINGS:**

**U1** How Java can and is used for a large number of things, including software development, mobile applications, and more.

**U2** How Java is like Python as an Object Oriented approach to programming.

**U3** What the purpose of the JavaScript language is and how it is used to make web pages more dynamic, user-friendly and more interactive.

**U4** Why JavaScript is considered the default language of the internet, and how it is being used outside of the internet today and into the future.

**UNIT 3- JAVA BASICS**

| Skills | Knowledge |
|---|---|
| 1. Explaining ways in which programs are being used today<br>2. Recognizing and defining the structure and components of a Java program IDE or platform that make it useful to software developers<br>3. Review elements of this code verses Python by following specific guidelines and rules in | ❖ **Why Java** is a popular object-oriented programming language and what industries use it.<br>❖ Object-oriented programming in Java including basic syntax for defining classes and creating instances<br>❖ Different types of **errors in Java** and practice finding them. |

Back to Top

| | |
|---|---|
| order to create readable and working programs | |
| 1. Explain what the Java console program is and its function<br>2. Explain the starter syntax for a Java program<br>3. Use the console by creating a simple hello world program | ❖ The **general format for a Java console** program writing a Java class: public, class, classname, {} |
| 1. Explain the purpose of the Run Method<br>2. Explain the first thing that happens in your program when you click the Run button.<br>3. Write a fully-formed Java program by including a class and a run method. | 1. The **run method** is where a Java program starts its execution. |
| 2. Explain the input method of read to get user input.<br>3. Explain the output method of system.out to produce output to the screen<br>4. Write a program that takes in user input then prints out the answers. | ❖ How programs take in user input **(read)** /and produce output**(System.out)** to the screen |
| 1. Explain the proper way to declare and initialize a variable in Java | ❖ What **variables** are and how they are used in Java programming. |
| 1. Explain what a primitive data type is in Java and list those types<br>2. Write a program that declares data then print it out. | ❖ **Datatypes in Java** and how we use them: *int, char, boolean, double* |
| 1. Interpret a Java math expression to determine the results<br>2. Create a java programs that perform math functions and prints out results | ❖ How mathematical operators are used in arithmetic expressions |

Back to Top

| | |
|---|---|
| 1. Explain what casting is<br>2. Evaluate a expression using casting for an *int* and *double*<br>3. Use casting to write a program that prints the division of the integers | ❖ What casting is in Java and how it is used |
| 1. Explain what a Boolean is in Java<br>2. Create a program that asks the user whether something is true or false that prints to the screen | ❖ What **Booleans** are and how they are used to test whether a condition is true or false |
| 1. Recognize and define logical operators in Java<br>2. Evaluate a Java expression that uses logical operators<br>3. Use a logical operator to decide whether the user is eligible | ❖ How **Logical operators** give the ability to connect or modify Boolean expressions. |
| 1. Evaluate a Java expression that uses comparison operators<br>2. Use comparison operators to decide a range | ❖ How **comparison operators** give the ability to compare two values which allow programs to make decisions |
| 1. Explain why we use for loops in Java<br>2. Create programs that use for loops with output | ❖ How **For loops** execute the same lines of code a given number of times |
| 1. Effectively use while loops in their programs<br>2. Identify and resolve infinite loops | ❖ How **Iterations** - **while loops/for loops** allow code to be executed repeatedly based on a condition<br>❖ How loops can be written in a way that causes an error or **infinite loop** |
| 1. Use if statements for control flow in their programs<br>2. Use if/else statements in order to make | ❖ How **If Statements** (Selection) allow the use of conditions to determine how their code should run |

Back to Top

| | |
|---|---|
| decisions between multiple scenarios | |
| 1. Create a program that computes 2 boolean expressions in order to determine what the user is allowed to do<br>2. Convert these two lines into their equivalent De Morgan style boolean expressions | ❖ What **DeMorgans Law** is and how it is used in Java coding |
| 1. Perform string operations in order to concatenate values together | ❖ What strings are and what is the proper way to compare String values in Java<br>❖ How to add and combine strings |

**UNIT 3- JAVASCRIPT BASICS**

| Skills | Knowledge |
|---|---|
| 1. Explaining ways in which JavaScript programs are being used today<br>2. Recognizing and defining the structure and components of a JavaScript program platform that make it useful to software developers<br>3. Review elements of this code verses Python by following specific guidelines and rules in order to create readable and working programs | ❖ Why **JavaScript** is a popular web development language and what industries use it.<br>❖ What are the **platform(s)** used by JavaScript programmers |
| 1. Explain what the JavaScript console program is and its function<br>2. Explain the starter syntax for a JavaScript program<br>3. Use the console by creating a simple hello world program | ❖ The **general format for a JavaScript console** program |
| 1. Explain the input method of read to get user input.<br>2. Explain the output method of system.out to | ❖ How programs take in user input **(readLine)** /and produce output**(println)** to the screen |

Back to Top

| | |
|---|---|
| produce output to the screen<br>3. Write a program that takes in user input then prints out the answers. | |
| 1. Explain the proper way to declare and initialize a variable in JavaScript<br>2. Identify the scope of a variable<br>3. Identify which variables are in scope at a given point in a program | ❖ What **variables** are and how they are used in JavaScript programming<br>❖ Scoping of a variable which is where the variable is “defined” or where it exists |
| 1. Interpret a JavaScript math expression to determine the results<br>2. Create JavaScript programs that perform math functions and prints out results | ❖ How mathematical operators are used in arithmetic expressions |
| 1. Create graphical programs that draw shapes on the canvas<br>2. Using basic graphics commands: width, height, add, new, top, bottom, mid, color<br>3. Locate points on the graphics canvas using (x, y) coordinates | ❖ How using geometric concepts, multiple graphic objects can be created in JavaScript<br>❖ Graphic creation relies on setting the type, shape, size, position, and color on the artist’s canvas before adding to the screen. |
| 1. Explain how Booleans are used in JavaScript<br>2. Create a program that represents yes/no options | ❖ What **Booleans** are and how they are used to test whether a condition is true or false |
| 1. Recognize and define logical operators in JavaScript:<br>2. Evaluate a JavaScript expression that uses logical operators! = NOT, \|\| = OR, && = AND<br>3. Use a logical operator to decide whether the user is eligible to graduate | ❖ How **Logical operators** give the ability to connect or modify Boolean expressions. |
| 1. Explain the meaning of each of the comparison operators (<, <=, >, >=, ==, !=) | ❖ How **comparison operators** give the ability to compare two values which allow |

Back to Top

| | |
|---|---|
| 2. Evaluate a JavaScript expression that uses comparison operators<br>3. Use comparison operators to decide a range | programs to make decisions |
| 1. Explain why we use for loops in JavaScript<br>2. Create for loops to solve increasingly challenging problems<br>3. Create nested for loops: for loops inside of for loops | ❖ How **For loops** execute the same lines of code a given number of times |
| 1. Explain the purpose of a while loop<br>2. Create while loops to repeat code while a condition is true<br>3. Utilize while loops to solve new types of problems | ❖ How **Iterations** - **while loops/for loops** allow code to be executed repeatedly based on a condition<br>❖ How loops can be written in a way that causes an error or **infinite loop** |
| 1. Explain the purpose of functions<br>2. Create their own JavaScript functions<br>3. Utilize their JavaScript functions to solve simple problems<br>4. Create functions that take in parameters as input<br>5. Create functions that take in multiple parameters as input, and use print statements for output<br>6. Utilize their JavaScript functions to simplify their graphics programs<br>7. Create programs that call functions with return values and stores the result | ❖ **Functions** as the main building block of complex programs.<br>❖ How Functions let break programs into different parts that can be organized and reused with **parameters**<br>❖ Functions with multiple parameters can be used to creating several different graphical objects<br>❖ Parameters are like inputs into the function, and the return value is output. |

**UNIT 3 - PERFORMANCE TASK:**

Use your knowledge of basic Javascript to create some fun programs! Students will create their own Ghost

Back to Top

drawings from Pac-Man and a drawing of their own choosing. This will allow students to get creative with their code to show what they have learned.

**Goal:** create their own Ghost drawings from Pac-Man and a drawing of their own choosing

**Role:** You are a college freshman in New Media programming class and this is your mid-term project

**Audience:** You will be sharing your program with the other students in the class - playing their programs to make sure they are working and error free.

**Situation:** Your professor has given you two challenges - one to draw the ghosts for a Pac-Man game, the next is a to draw a shape of your choosing

**Product**: Code that demonstrates the following:

**Pac-Man Ghost**
Write a program to draw 5 ghosts on the screen. You must do this by writing a function called drawGhost, which takes three parameters, the center x location of the ghost, the center y location of the ghost and the color of the ghost.

- Use the drawCircle function from previous programs.
- The constants for all of the ghost dimensions are given.
- Start off with something simpler.
- Try just drawing the general ghost shape, which is a circle and then a rectangle right on it.

**Your Own Drawing:**
- You must use at least one line, at least one rectangle, and at least one circle. All of these shapes must be visible on the canvas.
- You must use shapes of at least two different colors.
- You must use shapes of at least two different sizes.
- You must have at least five shapes total.
- You must leave a comment at the top of your program explaining what you are trying to do.
- Otherwise, there are no other requirements!

Back to Top