Introduction: A 14 Part Series

Part 0: Introduction
Part 1: What is Programming?
Part 2: How do we write Code?
Part 3: How do we get information from computers?
Part 4: What can computer's do?
Part 5: What are variables?
Part 6: How do we manipulate varaibles?
Part 7: What are conditional statements?
Part 8: What are Arrays?
Part 9: What are loops?
Part 10: What are errors?
Part 11: How do we Debug code?
Part 12: What are functions?
Part 13: How can we import functions?
Part 14: How do we make our own functions?
Part 15: What are arraylists and Dictionaries?
Part 16: How can we use Data Structures?
Part 17: What is Recursion?
Part 18: What is Pseudocode
Part 19: Choosing the right language
Part 20: Applications of Programming

What is programming?

Programming :

1. The process of preparing an instructional program for a device.
2. Attempting to get a computer to complete a specific task without making mistakes.

Programming languages serve as a middle-man of sorts
Translate your instructions into machine code:

1. The series of 0's and 1's that the computer can understand
2. Very useful for programmers.
3. Programming languages serve as interpreters for converting languages into other languages.
    a. Faster than converting by hand.

What is Programming: High vs Low level

1. Each language also has an attribute known as power or level
    a. Basically how similar it is to machine code
2. Low-level programming languages: Assembly or C
3. High-level programming languages: Java or Python.
4. Lower the level -> More similar to machine code.

How do we write code: Sending Instructions

1. To properly send instructions to the computer we need programming languages
2. But we also can't type in a certain language and expect the same thing.

How do we write code: IDE's

1. We use IDE's (Integrated development Environments) to write code
    a. A place to write, run and debug code and also convert it to machine code.
2. IDE's are likely any other program on your computer except used for the facilities of code.
    a. Such as NetBeans, IntelliJ, Visual Studios.

How do we write code: IDE Functionality

1. In addition to a place to write code, IDE's also include
    a. Built in Error-Checking
        i. For when code doesn't go right
    b. Auto-fill for frequently used words
    c. Project Hierarchy
2. A big step up from previously used methods of programming.

How do we get information from Computers: How to use the Console?

1. The main use of the console is to output text from the program using code
    a. More specifically a print statement
        i. Prints text to the console for the programmer viewing pleasure

How do we get information from computers - Using the print statement:

1. To use the print statement, simply instruct the console to print, and then include whatever you want to be printed inside the parenthesis
    a. Using Python, we can print to the console like so.

How do we get Information from Computers: Variations on the Print Statement

The print statement will vary depending on the language.

How do we get information from computers: A Background Tool?

1. The console is mainly a developer tool
    a. Not usually meant to be used and interacted with by the end user except in very abstract cases.
        i. Text-based games/simple programs
    b. Tends to be hidden away behind the scenes

What can computer's do: Strings Continued ?

print("Game over, 4 was your final score.") X wrong
print("Game over, " + 4 + " was your final score.")

What are the variables : Integers -"

1. Integer's
    a. A variable that can store and integer value such as -2, 147, 483, 648 to 2,147, 483, 648
    b. Can't and will not hold any decimal values.

What are Variables: Booleans -

1. It can store a value of either true or false.
2. It can only hold true or false
    a. No other types of information.

What are Variables: Floats and Doubles

1. Both are types of floating point data types
    a. Can store number with decimal places
2. Float variables
    a. Can store up to 32 bits of information
3. Double Variables
    a. Can store up to 64 bits of information

What are variables: Strings -

1. It is useful for displaying text and storing input information
    a. Information the user inputs into our program
2. Also useful for outputting information in a readable format for the user.

What are variables: Why are variables so useful -

1. Often times you're going to want to keep track of things such as a user's name or score
    a. By creating a variable you can store this information in that variable and then reference, add to, or modify it.

What are variables: Why are variables so useful -

1. Other important uses for variables
    a. Taking input from the user
    b. Making our programs have variability
        i. Have it change based on certain factors
    c. Manipulating variables is necessary for many tasks in programming

How do we manipulate variables?

How do we change the variables we have, and how do we even make them?

1. Creates a little space in memory that stores your variable name and its contents.

    Int x = 10;

How do we manipulate variables: Warehouse example -

{ Number, age and score} → Name → NullPointerException.

How do we manipulate variables : Changing variable example -

{ age } → 18

Age = 18;

How do we manipulate variables: RPG example -

1. We want to make an RPG Game which has different stats.

How do we manipulate variables : Other ways of manipulating variables

1. Integer, float, and double variables can be

    a. Added
    b. Subtracted
    c. Multiplied
    d. Divided
    e. Modulused
2. String variables can be
    a. Added
3. Char's and boolean's can't be operated on

How do we manipulate variables: Naming variables:

1. There is one big rule when naming variables
    a. They must be one continuous string
2. Most programmers name variables according to camelCase
    a. Don't capitalize the first word, but capitalize the first letter of all words after it.

Int playerScore = 20; → GOOD

What are Conditional Statements?

1. Depending on certain conditions, we want our code to do different things.

What are conditional statements: The If Statement -

- The most basic conditional statement is the If statement
    - If something is True do this, otherwise do something else.
- The single most important thing to note about arrays is how we reference each element inside of them
    - In programming we use indexes
        - That numbers 'place' in the array

Numbers	1	2	3	4	5	6
Index	0	1	2	3	4	5

What are Array's: Creating Arrays -

1. Populate First
    a. Insert the elements you would like in the array immediately
2. Popular Later
    a. Create an array with a specific size, but choose to add elements later.

What are array's: Array sizes -

1. When we create array their sizes are FINAL
    a. Cannot be increased or decreased in size through conventional methods
    b. This is what allows us to easily access their indexes.
2. What are Array's: Bookshelf Example -
    a. We cannot change the size of an array within the code
3. What are array's : Resizing Arrays
    a. We cannot change the size of an array within the code
4. When we create array their sizes are final
    a. Cannot be increased or decreased in size through conventional methods
    b. This is what allows us to easily access their indexes.
5. We cannot change the size of an array within the code

6. When initializing an array, you must determine its type then and there:
    a. Example (String array, integer array, double array, etc)
7. They have to all be the same type.
8. Putting an array inside an array is known as a 2-D dimensional array
    a. Similar to matrices in math/physics classes
9. To index 2D arrays we use 2 numbers
    a. First number is the row
    b. Second number is the column

What are loop's?

How can we save code repeat segments without repeating lines of code?

1. A loop is a statement that is used to run certain instructions repeatedly
2. Very useful for repeated sections of code.

If we wanted to print something 15 times ….

1. We could use 15 print statements or we could use a loop
2. The For loop
    a. Used when you would like to carry out a certain set of instructions numerous times.
3. Consisting of 3 parts
    a. An integer value
    b. A conditional statement the integer must reach to exit the loop
    c. An operation to modify the integer value after the instructions inside of the loop are completed.
4. What are loops: for loop example -
   When using a for loop, you must make sure you set up a condition that, given the initial integer value and the operation, will at some point be met.
   Otherwise an infinite loop will occur.

   What is for each loop?
    a. Used for iterating through entire arrays or lists
    b. The loop will go through each element in the array and carry out a set of instructions for each value.
    c. Useful for performing operations across an entire collection of data.

   What are loops: The while loop -

1. The while loop
    a. Will continually carry out its instructions while a conditional statement given to it is true
    b. Similar to a for loop, but broken apart.
    c. Can sometimes be used to purposely create an infinite loop.

   What are loops: Purposely creating an infinite loop:

   What are loops: The Do-While Loop
   The Do-While loop
    a. Functionally similar to while loops
    b. However, will always carry our instructions at least once

c. Instructions inside the loop will run once before checking the conditional statement.

What are loops: Benefits of loops -

1. Benefits of loops
    a. Performs operations many times in a row
    b. Able to iterate through arrays and lists
    c. Decrease clutter of your code.

2. What are errors?
    a. What happens when our code doesn't work the way that we want it too?
    b. What are errors: errors introduction
        i. Code does not always work as expected
        ii. These are known as errors
            1. Come up often in computer science
    c. Three different types
        i. Syntax errors
        ii. Runtime errors
        iii. Logic errors
3. What are errors: Syntax errors
    a. Syntax errors:
        i. Parts in your program where you fail to meet the programming rules, resulting in an error.
    b. Usually the easiest of the 3 to solve
    c. Highlighted by the IDE in most cases.

What are Errors: Debugging Syntax Errors -

1. IDE's underline syntax errors and usually provide helpful hints
2. Syntax errors are like small misspellings or grammatical errors
3. Some IDE's will restrict you from running the code unless all syntax errors are cleared.

What are errors: Runtime Errors -
Runtime Errors

a. Don't appear until you actually 'run' the code

Caused by a part of your code not being able to be computed in a reasonable amount of time

b. Most common form → The infinite loop

What are Errors: Infinite Loops

1. Essentially what happens with the computer
    a. It is given some condition with no feasible way to finish
2. Puts the computer in error mode
3. Will never reach the ending condition, causing a crash.

How do we debug code: commenting-

1. Commentingg
    a. Allows us to mark-up the code without the computer reading it as actual code

    b. Essentially a documentational tool for programmers

How do we debug code : Preventing errors

1. Backup code frequently
2. Version managers such as Github or subversion can help
    a. Backup code to an online cloud service in which you can easily pull previous versions of the program
3. Useful for backtracking to find when the error was written
4. Run your program frequently
    a. Prevents you from saving a backup that doesn't work
    b. Makes it easier to figure out when you wrote the error

What are functions?

1. We've actually been using functions this whole video
    a. Print statements
    b. For loops
    c. Basic Math Operators
2. A function
    a. A segment of code that can be easily run by 'calling' the function name
    b. Depending on the type of function, may do something in return
3. Can be called numerous times, and in numerous places
4. Like wrapping code into a present and giving it a name.
5. We "call" the print function, and enter in what we want to be printed to the console inside the parenthesis
    a. Computer does it for us
    b. Behind the scenes there is more code, which takes care of printing our message to the console.
    c. Abstracts all that code down to a single line.
    d. Functions serve many purposes
        i. Used to recycle sections of code which serve the same purpose
        ii. Used for equations you want to allow multiple inputs of
        iii. Used to save space within your program
    e. Extremely powerful
6. There are thousands of types of functions
    a. Often times you will just import the ones you need in your program
7. There are 4 different types of functions
    a. Separated by whether or not they take in arguments
    b. Separated by whether or not they return values
        i. Functions
            1. Takes in arguments
                a. Return values
                b. Return no values
            2. Doesn't take in arguments
                a. Return values
                b. Return no values
        ii. Arguments
            1. Variables we pass into a function in order to be manipulated and then either

    a. Returned back to us
    b. Printed to the console
    c. Used in another operation
2. Arguments are a way for programmers to have one function that can do many different things
3. Add variability to programming
4. Helps diversify your code
5. A function which takes no arguments and returns no values is similar to the printStats() functions
    a. Cannot be set to any variable since it returns no value
6. Libraries
    a. Collection of functions that all have the same theme
        i. Math library, data analysis library,etc.
7. Importing libraries is as simple as using an import statement
    a. Usually consists of the word “Import” followed by the library you would like to import
    b. Searching algorithms
        i. Ways in which we can look through a list of values stored in an array and find a particular piece of data
        ii. Lists can either be sorted or unsorted
        iii. Efficiency is based on two values
            1. Worst case scenario
            2. Average number of items
        iv. Known as Big O notation
    c. Linear search
        i. Start at the beginning and systematically check each data point until you find what you are looking for
        ii. The linear search can work on both sorted and unsorted lists
        iii. Uses a recursive process
            1. Breaks down the list into smaller and smaller parts to find the item we are looking for faster
        iv. Takes advantage of the fact the list is sorted
    d. Recursion
        i. The practice of using functions that repeatedly call themselves
    e. In the instructions that occur within a function
        i. One of the instructions will be a call to that same function
    f. What is recursion: Summative Example
        i. Public int recursiveSum(int n)
            {
                If (n>100) {
                    return n;
                }

```
        }
        Else {
        Return (n+ recursiveSum(n-1));
        }
}
```

8. What is Recursion: The Stack
    a. Stack
        i. A data structure which contains all of the tasks you instruct your program to complete
        ii. Based on a certain method, your program will then carry out the tasks you give it.
9. What is Recursion: LIFO structure
    a. Our programs will follow the LIFO data structure
        i. Last in first out
    b. This means the last item put on the stack will be the first one removed from it.
10. Recursion is so useful because it breaks large problems into much simpler pieces to compute
11. How can we plan our code before-hand to reduce errors and make our software development process run smoothly?
12. Pseudocode can be compared to an outline for a paper
13. Explain the main topics of the essay and plan out the major talking points
14. Don’t worry about the nitty gritty details
    a. Word Choice
    b. Grammar Conventions
    c. Proper formatting
15. Pseudocode can take many different forms based on the type of programmer you are
    a. Hundreds of different pseudocode methods
    b. Flowcharts can be used to think through the process of a particular function
    c. Another strategy is writing what you want your code to do chronologically
    d. What is the program you are writing trying to accomplish?
        i. Step by step
    e. Flowchart method
        i. Good for thinking through the flow of a function
    f. Write-up method
        i. Good for getting the general idea of a program down
    g. Functionality planning method
        i. Good for listening out the functions of a certain program
    h. Scripting languages

        i. Language with many commands for you to use that can be run without being complied
        ii. Easier to port between operating system
    i. Many times also used in web development

What is the software development lifecycle and software development?

1. SDLC
    a. Planning
    b. Requirement Analysis
    c. Designing
    d. Implementation
    e. Testing
    f. Deployment and maintaining

Planning:Discuss Requirements

1. The purpose of the application
2. The details about the end-user of the product
3. Key elements like format and attributes of the application for designing
4. The user interfaces for all the software

Requirement Analysis

1. Information about each element to design
2. Validating the installation
3. Calibrating the security protocols and risk analysis
4. Software requirement specification document (SRS)

Design:

1. Devise the system design following
2. Check feasibility with requirements
3. Design document specification (DDS Document) → share → analysis and stakeholders

Implementation:

1. Developers start writing the code using the languages
2. Actual implementation of the software product
3. Development tools to implement the code

Testing:

1. Deployed in multiple test environments to check the functioning
2. Testing team may find errors or bugs
3. Continued till the system

Deployment:

1. Ready for deployment and consumers
2. Development team will set up an installation link for the users.'
3. Fix bugs: Debugging of the application is done on a regular basis.
4. Regular updates with enhanced features

The concept of recursion: is solved by a very specific problem.It can be divided into smaller, similar subproblems.

```
⇒ def  countdown(val):
        ⇒ if val ==0: return
        ⇒ print(val)
        # recursion
        ⇒ countdown(val -1)
⇒ countdown(5)
```

Regex: Regular expressions are sequences of characters that define a search pattern.

⇒ /[A-Z][A-Za-z]+ [A-Z][A-Za-z]+/

Object Oriented Programming: is a programming paradigm that organizes and structures code around objects rather than functions and logic.

Threading: is a programming concept used to achieve multitasking by dividing a program into multiple lightweight threads of execution.

Process: an independent unit of execution in a computer.

Multiprocessing: the ability of a system to run multiple processes simultaneously. It takes advantage of multiple CPU cores for parallel execution, improving performance for CPU intensive tasks.

Abstraction: a process of hiding programming complexity and logic and making something simpler to you.

Compiler: converts a program source code into some other source code which means that it turns human readable source code into machine code that the CPU can understand.

High/Low level programming languages:
⇒ High level programming languages: JS, Python, Java, C# ,C ,C++
⇒ Low level programming languages: Assembly