Foundations of Programming Study Guide

From Simple Studies, https://simplestudies.edublogs.org & @simplestudiesinc on Instagram

Intro to Programming

Intro to Programming

- Programmer: writes millions of lines of code
  - They are inventors, fluent, visionaries, gatekeepers, and researchers.
- Code: instructions in a computer program
- Program: A set of instructions written for a computer to execute
- Commands: Instructions given to a computer for a task
- Debugging: The process of finding and fixing errors in a computer program
- Software: Programs written for computers to perform specific tasks
- Interface: Allows communication between humans and computers
- Jobs that programmers can do:
  - Healthcare industry
  - Education
  - Military
  - Technology
  - Entertainment
  - Agriculture
  - Construction
  - Adaptive Technology
- Programming Now and Then
  - 1982: In the beginning, programming mostly existed on paper. The reality of programming was realized when Ada wrote an algorithm to compute numbers.
  - 1946: Programming was labor intensive and prone to errors.
  - 1957: Programmers developed new languages so they could translate commands into binary, or machine code, so computers could understand.
  - 1968 – 1969: Niklaus Wirth designed the famous programming language, Pascal, in his thirties.
  - 1981: A computer programmer’s job expanded from developing specialized programs for specific purposes, to building software for everyday users.

- Today: There are thousands of different programming languages today, all developed to meet different needs. Mobile devices and constant need to access information, means a higher demand for smarter and more complex programs.

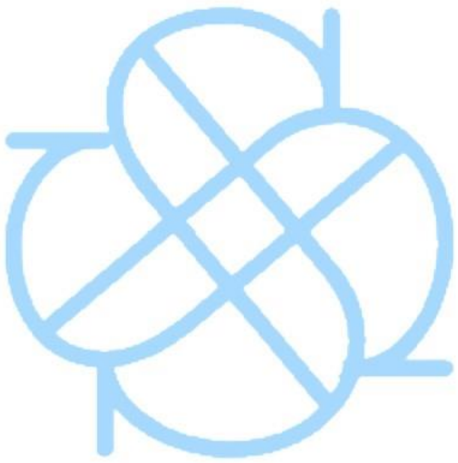
Introduction to Python

- Programming Languages: a language used to write instructions that can be executed by a computer
- Machine Code: a set of instructions composed of 1s and 0s the computer can execute without any translation
- High-Level Languages: translate human messages into machine code that the computer can understand
- Source Codes: program code written in a high-level language before being translated into machine code

	Low-Level Languages	High-Level Languages
Languages	Assembly Language, Machine Languages	Python, Java, C, HTML
Description	Less readable by humans. They are written in the computer's native machine code.	Closer to human languages making it easier for programmers to learn and use.
Uses	Write code which controls the computer's hardware.	Write programs for anything.
Pros	Controls the hardware and executes tasks more quickly.	User friendly and able to use on different kinds of hardware.
Cons	Written in machine code (hard for humans to understand).	Programs run slower than the low-level languages.
Limits	Programs work only on specific computers.	Must be translated to machine code.

- Interpreted Languages:
  - Characteristics
    - Translates one line of code and then executes it before moving to the next line.
    - Ex: Python
- Compiled Languages
  - Characteristics
    - Translates all lines of code together and executes them at once.

        - Ex: Java
- Most common Computer Languages
    - Java: Used for web applications and software development
    - Python: Used in web, applications, network programming, Android devices
    - C: Used in computer game development, cross-platform programming, and more
    - Assembly language: Uses letters and numbers to represent machine language
    - Machine language: Hard for humans to understand and write
    - HTML: Used to present information



Printing Strings

- Print: to display something on the screen
- String: Sequence of letters, numbers, spaces, and symbols, or alphanumeric info.
- Alphanumeric info: Any combination of letters, numbers, or symbols.
- String literal: Series of keyboard characters included within quotation marks
    - Ex: print (“**This will print!**”) – bolded part is a string literal
- Always make sure that you type print followed by parentheses. Inside the parentheses, you can type what you want Python to print. Don’t forget the quotes when printing a string literal because this will confuse the computer.
- Types of Errors
    - Syntax: Occurs when a programming language rule is broken. Programming languages have rules for their structure.
        - Error: print I live in Florida.
        - Correction: print (“I live in Florida.”)
    - Runtime:
        - Error: print (11 / 0)
        - Correction: print (11 / 17)
    - Logic:
        - Error: print (“Grass if blue.”)
        - Correction: print (“Grass is green.”)
- Troubleshooting Tips
    - If you are stuck, use these tips.
        - Check your spelling and capitalization – Python is case sensitive
        - Check punctuation
        - Check spacing
        - Check before and after – An error may be hiding the line above or below
        - Talk it out – Read the code to yourself and you may hear the bug/error
- Example
    - Print statement print (“Are we having fun yet?”)
    - Resulting Output Are we having fun yet?

Processing String Values

- String value: the characters (numbers, symbols, letters) that make a string
- Variable: a group of characters whose value can be changed as needed
- Assignment Statement: statement which assigns values to variables
- Concatenation: combining 2 or more string values to form a single string (using the + symbol)
- Index Position: position of a character in a string or an element (first index always starts with zero)
    - Index of “b” in umbrella – 2
- Slice: part of a string
- Rules for naming Variables
    - Do
        - Begin variable names with a letter or underscore. (Ex: song, songTitle)
        - After the first letter, the variable name can consist of additional letters or digits (0 to 9).
    - Do Not
        - Variable names should not be a Python keyword.
        - Variable names can’t have spaces.
        - Variable names can’t have any punctuation.
- Remember these rules to follow while naming a variable
    - Camelcase
    - Meaningful name
    - Reasonable length
    - Use nouns

String Input

- Interactive: programs which prompt the user for an input/response
- Prompt: communicates to the user to enter an input value
- Chatbot: program which simulates conversation between computer and person/user
- Pseudocode: outline/blueprint of your code, includes steps of the program which are easy for humans to understand
    - Be sure to include input, calculations (if needed), and output.
- Tips for writing good pseudocode:
    - Use it to plan your code
    - Outline the steps so it is very clear for when you write the code
    - Use simple, everyday words
    - Short phrases to describe the input and output
    - Review and make any changes if necessary

Computing Numerical Data

Processing Numerical Information

- Non-numeric data: a string which consists of a combination of letters, numbers, and/or symbols. This type of data can't be used in calculations.
- Numeric data: numerical value that can be used in calculations.
- int: refers to an integer
  - positive, negative whole numbers (including 0)
- float: refers to "floating point numbers". These numbers have a decimal point.
  - positive, negative numbers, including 0.0
- Three main reasons why we categorize numbers in Python
  - Memory: Programs usually take up a lot of storage space and numbers categorized as floating-point numbers take up more space than integers.
  - Precision: You may need a specific number with decimals.
  - Arithmetic: Helps you with correct math. Calculations with integers and floating-point numbers don't always end with the same results.
- Be very careful when inputting equations into your program, Python does follow the order of operations.
- The equal sign (=) has a different meaning in Python. It means to assign a value to a variable.
  - Example: x = 12, x = x + 1, print (x)
  - The program would print 13 for x.

Getting Numerical Input

- Three data types
    - String (str)
        - A value which consists of letters or numbers which can't perform a math function.
        - Example: "Hello, world!"
    - Float (float)
        - A numeric value which consists of a decimal point (positive or negative).
        - Example: 8.45, 0.0
    - Integer (int)
        - A whole number which is either positive or negative, also includes 0.
        - Example: -2, 3, 0
- The `input()` function
    - Python assumes that all values you enter are strings.
    - Ex: `box = input("What color do you want us to paint your box?")`
- Using the int and float data types with the `input()` function
    - `age = int(input("How old are you?"))`
    - `gpa = float(input("What is your GPA?"))`

Math Module

- Python has built-in functions which make programming easier.
- We use this import statement to start our programs: `from math import *`
- You can use these two functions from the math module to make it easier to solve problems.
    - To find the square root of x
        - sqrt()
    - To raise x to the power of y
        - pow(num, exp)
- Arguments: a value provided when a function is called

The Software Development Life Cycle

- Planning & Analysis
    - Describe the purpose of your project, requirements for the project, and analyze the “scope of the work”.
- Design
    - Write pseudocode to help design your project.
- Coding
    - Write the code.
- Testing
    - Identify defects and debug so that the software runs smoothly.
- Maintenance
    - Update and make any changes based on user feedback.
    - Resolve any new issues that arise.
- Two major concerns of programmers in a SLDC
    - Version control: good organization in the different versions of the software development.
        - Important because programs can go through numerous iterations during the life cycle.
    - Change management: documentation regarding any changes needed and made throughout the software development life cycle.
        - Important for when the team is reviewing the progress of their project.

Loops and Functions

Turtle Graphics

- Turtle graphics: set of functions in Python which allows the user or programmer to draw or create images on the screen
- The turtle graphic functions let you code your program to have visuals by using your cursor and moving it across the screen
- **EXAMPLE**

  -
    ```
    #The turtle module
    import turtle
    def main( ):
        tom = turtle.Turtle()
        tom.forward(50)
        tom.left(20)
        tom.forward(60)
    main( )
    ```
- Using Turtle Graphics
  - Step 1: Import Turtle Graphics module
  - Step 2: Name the turtle
  - Step 3: Use a function (forward, backward, left, and right)
- Turtle Graphic Module Functions
  - `forward(x)`
  - `backward(x)`
  - `left(x)`
  - `right(x)`
- `forward(x)` function requires an integer parameter for x so that Python knows how many pixels the turtle should move forward.
- You can use the turtle graphics module to connect lines or create shapes and patterns.

For Loops

- Loop: a program statement which can make a segment of code repeat itself until the terminating condition is met.
- **EXAMPLE (without loops)**

  -
    ```
    import turtle
    def main( ):
        mike = turtle.Turtle()
        mike.forward(50)
        mike.right(25)
        mike.forward(50)
        mike.right(25)
        mike.forward(50)
        mike.right(25)
        mike.forward(50)
        mike.right(25)
    main( )
    ```
- **EXAMPLE (with loops)**

  -
    ```
    import turtle
    def main( ):
        for side in range(4):
            mike.forward(50)
            mike.right(25)
    main( )
    ```
- `for` loop: tells Python to repeat an action, make coding more manageable
  - `for n in range(9, 40, 3):`
    - "n": counter variable
    - "in range (9": 9 is the start value for the counter variable
    - 3 in the above example is the change value. After each iteration of the loop, the counter variable (n) increases by 3.

While loops

- This type of loop is used when you want to repeat code but don’t know the number of iterations.
- Boolean: a logic which evaluates whether a condition is true or false
- Relational operators: operators that compare two values, expressions, or variables
    - Less than or equal to: <=
    - Less than: <
    - Equal to: ==
    - Not equal to: !=
    - Greater than: >
    - Greater than or equal to: >=
- `while` loop
    - This command waits for a condition/event to happen.
    - Needs to know what action it should do and under what conditions.
    -
      ```
      x = 0
      while (x < 10):
          x = x + 1
      ```
- Syntax points to remember:
    - Statement should begin with the keyword `while`
    - Test condition goes inside the parentheses after the keyword `while`
    - Statement ends with a colon
    - Code in the body of the loop should be indented

Similarities and differences between `for` loop and `while` loop

- `for` loop vs. `while` loop
  - `for` loops repeat for a range of values
  - `while` loops repeat when a condition is true/met
  - `for` loops have a start value for the loop variables
  - `while` loops repeat when a condition is true/met
  - `for` loop can be best when you know how many times to repeat the code
  - `while` loops can be best when you don't know how many times to repeat the code
- Similarities between `for` loops and `while` loops
  - Repeats code
  - Loop stops once the condition is false

Creating Functions

- Function: a segment of code which performs a specific task
- How to define a new function
  - def sayHello (name)

    print ("Hello," + name + "!")
- When you write your own functions, you will:
  - Identify the code you would want to put into the function
  - Define the function
  - Define the parameters in the parenthesis

Logic and Programming

Decisions (if Statements)

- An `if` statement tells the computer that if a condition is true, then it should execute the block of code within the function.
    - If it’s false, the computer should skip the code and continue with the rest of the program.
- Boolean logic: answers can be true or false (no middle ground)
- George Boole
    - Inventor of the concept of Boolean logic
    - He also developed the idea of what computers can use to make decisions – condition statements.
- `if-else` statements
    - If the condition is true, the computer executes the block of code but if the condition is false, the computer executes the else action.
    - Will always take some action because the condition has to be either true or false (not in the middle).
- `elif` statements
    - These statements allow you to check multiple conditions.
    - Only one condition needs to be true for the action to be taken.

    -
      ```
      age = input (“What is your age?”)
      if (age <= 15) :
            print (“I am older than you.”)
      if (age == 16) :
            print (“We are both 16!”)
      if (age >= 15) :
            print (“I am younger than you.”)
      ```

The logic of and, or, and not

- Compound conditions: comparing 2 or more Boolean statements using logic expressions such as and, or, and not.
- Boolean conditions
    - `and`: both conditions have to be true for the entire condition to be true
    - `or`: one of the conditions must be true
    - `not`: condition after the keyword `not` must be false for the entire condition to be true
- When you use multiple Boolean conditions, each condition is checked to see whether it is true or false.

Lists

- List: a structure which can hold multiple values
- Index position: position of a character or an element
    - The first index position is ZERO. Does not start with one.
- How to create lists in Python!
    - Give the list a name.
    - Make sure it can add or remove as many items as you want.
    - You can add or delete any elements in the list.
    - Replace any elements with new ones.
    - Refer to any element you need in the list.
- Three main types of lists in Python:
    - List of things you need to accomplish
    - List of numbers
    - Empty list (if you are not sure what kind of list you want)
- List of things… you need for school

    -
      ```
      materialsForSchool = [“Glue sticks”, “Binders”,
      “Pencils”, “Notebooks”, “Electronic Device”, “Pens”,
      “Lunch Bag”, “Pencil Pouch”, “Markers”, “Colored
      Pencils”, “Notebook Paper”]
      ```
- List of numbers

    - `memberShipsPerDayList = [2000, 1204, 9380, 2034, 6547, 3482, 7642]`
- Empty list
    - `emptyList = [ ]`
- Traversing through a list means to refer to every element in a list.
- How to make a proper list!
    - Make sure it has a meaningful name (doesn’t have to have a “list” in it).
    - Name = [list items]
    - How long is my list?
        - `length = len(materials)`
    - Accessing a specific element in the list
        - `print(materials[1])`
    - Traversing through a list

        -
          ```
          for n in range (0, len (materials)):
          print (str(n) + “ ” + materials [n] + “!”)
          ```
    - Appending a list item
        - `materials.append(“Calculator”)`
    - Deleting a list item
        - `del materials[3]`
    - Changing a list item
        - `Materials [3] = “Planner”`

Codes and Objects

Number Systems

- Decimal System:
  - Most frequently used number system by humans
  - Based on 10 digits
  - From 0-9
- Hexadecimal System:
  - They are common in the computer world.
  - Usually used in relation with memory locations and colors.
  - Based on 16 digits
  - From 0-9 and A-F
- Binary System
  - This is what computer programming languages are made of.
  - Based on 2 digits (0s and 1s)
  - The combination of these 2 digits can be converted into words or numbers.
- Bit: short for a binary digit
  - It is the smallest unit of data in the software world.
- How do you convert between number systems in Python?
  - `example`

    ```
    age = 16
    binaryAge = bin(age)
    print(binaryAge)
    ```

Decoding ASCII

- ASCII: American Standard Code for Information Interchange
  - It's a special code where all machines and computers use to receive and deliver data.
- Unicode
  - Covers more characters than ASCII does
- Ordinal means a numerical order.

    - `ord()` function decodes characters or words back into their numerical value.

Procedural and Object-Oriented Programming

- Procedural programming: A programming design approach which divides tasks into simpler tasks such as functions or procedures.
    - Known as the “Top-Down Approach”
- Object-oriented programming: a programming design approach which focuses on objects and not functions.
    - This is often referred to as OOP.
    - Classes define the objects and variables for a program.
- Objects
    - An instance of a class
    - It has its own variables and attributes
    - They are defined using a class.
- Class
    - Behaviors which describe a part of an object
- Languages that use procedural programming:
    - Fortran
    - COBOL
    - C
    - BASIC
- Languages that use object-oriented programming:
    - Python
    - Java
    - C++
- OOP requires more planning than procedural programming.
- Constructor: Refers to the code which creates a new object.
- Behaviors: Also known as methods.
- Advantages of object-oriented programming:
    - Reusable code: Programmers can use the code for a programming project for other similar projects.

        - A class for a deck of cards can be used to program any game that involves a deck of cards such as Go Fish or Solitaire.
    - Inherited traits: New objects can be created to inherit traits from other existing classes.
    - Flexibility: Programs made with OOP can be used for different purposes and behaviors making it easier to use with different programs.
    - Easy to maintain: They are easier to maintain because programs written with OOP are written in modules or objects.
        - It’s also easy to understand once the code is fully developed.

Classes and methods

- How to create a class in Python!
    - Define your class (used the Python keyword `class` to define it).
    - Initialize your variables.
    - Set any parameters necessary
    - Name your new instances.
    - Retrieve all your object values.
- Classes often represent real-life objects.

Testing and Security

Testing

- Topics involved:
    - Types, limits of testing, uses and benefits which ensure program quality
    - When these types of testing are used in programs
    - Program testing
    - Customer satisfaction and their role when releasing a new software
- Testing levels
    - Numerous testing practices which happen in different stages of the Software Development Life Cycle to help identify any errors and also prevent any bugs.
    - Unit Testing: verify sections in a single program.
    - Integration Testing: check the behavior and functionality of a program and that the different parts of a program are working together as expected.
    - System Testing: developers verify whether the entire system is working or not.
    - User acceptance Testing: confirming the readiness of a product and customer's satisfaction.
    - Automated Testing: Testing programs overnight (doesn't require humans).
    - Usability Testing: Users testing the software.
    - Regression Testing: Checking new and old versions of a software.
    - Quality Testing: Many different practices to test the quality of the final product.
        - Separation of duties: Ensuring the members of the programming team knows their responsibilities.
        - ISO 9000: International students try to maintain the quality.
        - Six Sigma: Management techniques which reduce the likelihood of the errors that have occurred.
        - Sandboxing: Trying to run software without risking harming the computer that is running the code.

- Different types of usability testing which allows for better customer experience.
    - Explorative: Used really early in the software development life cycle to evaluate the approach on the design of the product.
        - Prototype of the product
        - Tested with the user's expectations in mind
        - Tries to find important bugs
    - Assessment: Occurs during the software development life cycle process so that it can evaluate real-time performance of the software being tested.
        - Tells us the usability and effectiveness of the product.
        - Uses debuggers
    - Comparative: This testing compares designs, models, or products and evaluates the strengths and weaknesses.
        - Tested on multiple devices to identify what is working and to see if there any improvements needed for the software.
- Two very infamous bugs that have been caught with testing are Apple's SSL bug and a security bug which was found to encrypt private information (Google, Instagram, and Netflix are one of the many that had used this).

The Ethics of Programming

- A programmer’s job isn’t done as soon as they write the program code. They also have to be able to maintain the program by keeping an eye out for accuracy of the code and security.
- The Code of Ethics and Professional Conduct is written by Association for Computing Machinery (ACM).
    - “Contribute to Society and human well-being.”
    - “Avoid harm to others.”
    - “Respect the privacy of others.”
    - “Give proper credit for intellectual property.”
    - “Access computing and communication resources only when authorized to do so.”
- These are some common ways of getting your information stolen.
    - Email scams: Scammers send you emails which make you respond with personal information or click on certain dangerous links.
    - Privacy settings: Privacy settings and the complexity of your passwords determines whether you are volunteering your personal information to be out there or keep your information safe.
    - Sharing passwords: DO NOT share your password (even with your closest friends or partner).
    - Dummy sites: Clicking on random pop-ups that are attractive might seem like a good idea until your personal information is stolen. Make sure you check for consistent copyright tags on each page if you are asked to navigate to any external links.
- Enterprise software: it’s software that is not designed for individuals, but for big organizations or companies.
    - They help organizations be more effective because of the software that is built especially for businesses.

    - Many companies end up with unlimited access to a lot of personal information due to this idea of enterprise software.
    - They still hold the responsibility of keeping this information safe.
- Disaster recovery plan
    - A document which outlines the procedures and protects the business’s data in case a disaster occurs.
    - Every company is required to have some form of a disaster recovery plan.
- Parts of a Disaster Recovery Plan
    - Backup: Have a backup of your data to make sure that you can restore files and data and be able to return to normal ASAP.
    - Mitigation: This is known as the effort to reduce the impact of a disaster by being prepared with backups, checklists, and an emergency plan.
    - Monitoring: This can help you prevent any potential risks and keep any eye out for anything that might go wrong.
    - Redundancy: Having redundant backups is very useful because it reduces the downtime for your company and also allows for a faster recovery period.
    - Response: This is known as the emergency response plan. It outlines how the business will respond and what actions they should prioritize to get it up and running again.
- Few Types of backups
    - Full backup
    - Incremental Backup
    - Differential Backup
    - Mirror Backup

Coding and Careers

- Being fluent in multiple programming languages has many benefits.
    - Some businesses or organizations use software which requires knowledge of certain programming languages.
- Few potential paths anyone can take to continue their education in programming.
    - Online programs
    - Industry credentials and certifications
    - Two-Year College
        - Associate’s degree in computer science
    - Four-Year College
    - Advanced Degree
    - Military
- Careers which work closely with programming.
    - Database Administrator
    - Help Desk Technician
    - Web Architect
    - Web Marketing Professional
    - Website Analyst
    - Security Analyst
    - Server Administrator
    - Web Application Developer
    - Website Manager
    - Website Designer
    - Network Engineer
    - PC Repair Technician