PLEASE REVIEW THE EXTENSIVE SUMMARIZED POINTS IN RESPECT TO PROGRAMMING IN PYTHON OR OTHER LANGUAGES

VARIABLES

- A variable is a place to store values.
- Another name for a variable is an identifier.
- A variable may contain numbers, letters, and underscore.
- When declaring a variable, the variable names are case-sensitive.
- Avoid starting a variable with a number.
- It is better to start a variable with a LETTER.
- You may use more than one word when declaring a variable in most programming languages.
- Use Camel case for variables.
- Camel casing helps to identify a variable name.
- Do not use a Python keyword such as “print” to create a variable.
- Some program based words are also called “Reserved” words.
- Avoid starting a variable with a number as they are often considered as a keyword.
- Special characters such as punctuation marks are captured. In a string variable, letters from the alphabet.
- Avoid using spaces between words as white space is considered to be a character or series of characters.
- When you assign a value to a variable, it becomes a container.
- You can assign number to a variable.
- Do not confuse a variable with a constant.
- Visit the website for more detailed examples to accelerate your learning. Python Variable Types -

http://www.tutorialspoint.com/python/python_variable_types.htm

DATA TYPES

- Data type tells the compiler and the interpreter how to manage the data.
- We have different types of data to keep the program organized.
- Two basic types of data are text (or string) and numeric.
- Text (or string) is written within quotations marks.
- Common data types are: integers, Booleans, characters, floating-point numbers, and alphanumeric strings.
- Python data types: Numbers, String, List, Tuple and Dictionary.
- A string data type is enclosed in double quotes or single quote—it does not matter.
- An integer (int) is a whole number such as 5 or -5.
- A float is a number with a decimal point such as 5.5 or 2.666.
- List is an array that contains multiple objects; a list may contain multiple data types such as a string, float, and int. For example >>> sampleList = ["I like programming", 5, "times a week."] will execute if you >>> print (sampleList).

COMMENTS

- Use "#" for single comments.
- Use ‘’’ three single quotes for multiline comments.

OPERATORS

- Mathematical Arithmetic operators such as (+, -, *, /).
- Assignment operators such as (=, +=, -=).
- Comparison (Relational) operators such as (>, <, >=, <=, ==).

- Logical operators such as (AND, OR, NOT).

You may read more about operators at: Python – Basic Operators: https://www.tutorialspoint.com/python/python_basic_operators.htm

METHODS

- There are **different methods** you may use with a data type; you may type dir(str) to get a list of methods to use with a string data type; or dir(int) to get a list of methods to use with an integer as examples.
- For example if you type “Programming” . you will get a list of methods to execute after the dot operator as noted below. Type >>> "Programming".lower()
- Notice the results—the word “Programming “ becomes lower cased ‘programming’ with this execution.

HOW PYTHON EXECUTES A SCRIPT

- When Python executes a script, it executes it from TOP to BOTTOM.
- Python3.6 uses parentheses as brackets. Please remember to include equal number of brackets in your coding.

COMMON PROGRAMMER ERRORS

- You will get a **Syntax Error** if you start a variable with a number, particularly in Python.
- **Syntax** represents the programming rules and coding structure.
- **Syntax Error** is a coding misstep in respect to the programming language you are using.
- **Semantic Error** different from a Syntax Error in that the programming structure and rules might be correct but the output makes no sense.
- **Rules of Precedence** determines the order of operations (**PEMDAS**), which you may view at this video to learn

more: https://www.khanacademy.org/math/pre-algebra/pre-algebra-arith-prop/pre-algebra-order-of-operations/v/order-of-operations

CLASS BELOW IS A LIST OF RESOURCES OFFERED BY FORMER STUDENT AND HIS OPINIONS

- The first one is a book that a student has been working with for a couple weeks. The Author is very condescending and uses sarcasm at every point he can. The student noted he is not mean in his writing it is just not a book I would recommend for anyone. But, if you can get past his writing style, if it is not for you, then the book is a fantastic read. The book is:
- *"Learn Python 3 the Hard Way"* by: Zed A. Shaw
- ISBN-13: 978-0134692883
- ISBN-10: 0134692888
- https://www.amazon.com/dp/0134692888/ref=cm_sw_em_r_mt_dp_U_xB6yCb9A73B83
- As the student noted, his writing is not for everyone; however, the book cannot be beat. He creates a great learning path. He wants the reader to destroy the code, look up other functions that could be used, create their own program, and so much more!
- The second resource the author is more down to Earth with his explanations. They are still great to go over and learn about practical uses with Python.
- https://www.youtube.com/user/schafer5
- His YouTube channel is well constructed with playlists that hit great uses for the Python programming language. I wish I could credit all the resources that I have learned from. The hundreds of videos I have gone through have been a God send. But, for now these two resources are what I would credit my learning too.
- Oh! Another great resource I think would be applicable to this course would be:
- *"Automate the Boring Stuff"*
- https://automatetheboringstuff.com
- This book can be purchased from retail. But!! The book is also entirely free online. The author provides a great amount of

information that is written to help even the most basic user understand how Python, and programming languages in general, operate and why they are important.

Class, remember to practice using Python in your IDLE as you address the different items in the Discussion and in the assignments. Python will tell you when you are doing something wrong. You must explicitly tell Python what you want done. **Repetition is the key to retention!**

Note the following:

- Develop an appreciation for proper indentation and use of structure to avoid syntax errors.
- Practice examining MULTIPLE CONDITIONS using "If Statements" at the same time, particularly if you need two conditions to result in TRUE. In this case, the focus is on Logical operators such as (AND, OR, NOT), which are further discussed at the website: Python - Basic Operators - https://www.tutorialspoint.com/python/python_basic_operators.htm.
- Try not to confuse decision statements and operators.
- You will notice the *if statement* uses a *conditional test* via the *values True and False* to assess if the code should get executed or get ignored.
- Visit the website for more detailed examples to accelerate your learning. Python Variable Types - http://www.tutorialspoint.com/python/python_variable_types.htm.

Remember, when writing a program, you may simply save and modify it,

- Go to your IDLE, clicking **File** and **New File**.
- You will get an untitled edit window where you may write your codes and run your program.

- Write your codes.
- Save your file with a **.py** extension by going to **File, Save As.**
- Go to the **Run** option in your menu.
- Click **Run Module** and your codes will execute your sequence of codes.
- These codes will get executed as if you were in the Python shell.
- The results will show in the Python shell.
- You may edit your codes, save it, and exit.
- Open an editor and run your program in the Python shell IDLE environment.
- The video may help you: Using Python: The Python shell and IDLE: https://www.youtube.com/watch?v=kXbpB5_ywDw

Common programmer errors:

- You will get a **Syntax Error** if you start a variable with a number, particularly in Python.
- **Syntax** represents the programming rules and coding structure.
- **Syntax Error** is a coding misstep in respect to the programming language you are using.
- **Semantic Error** different from a Syntax Error in that the programming structure and rules might be correct but the output makes no sense.
- **Rules of Precedence** determines the order of operations (**PEMDAS**), which you may view at this video to learn more: https://www.khanacademy.org/math/pre-algebra/pre-algebra-arith-prop/pre-algebra-order-of-operations/v/order-of-operations

Keep the following definitions in mind:

- **Cascading if statements** – A series of nested if statements.

- **Compound condition** - Asking multiple questions before an outcome is determined.
- **If-then** - A single-alternative selection in which action is required for only one outcome of the question.
- **Nested decisions** (Nested if) - A decision “inside of ” another decision. (Farrell, 2012, p. Chapter 3).

Class, remember to behave as a responsible programming professional by using comments. They are helpful, especially for more complex and long programs. Adding notes, allows you to describe the purpose of the program and answer the “what, why, etc.” of a program. When using the (#), Python keeps the comment to first line and executes the codes on the lines below.

ADDITIONAL RESOURCES:
In respect to List, take some time to note the following:

- Since there is no native array data structure, use List in Python.
- It may contain more than one element.
- Use a plural to name your List.
- Lists are ordered collections.
- Elements are used to build a List.
- Individual elements are separated by commas.
- Index and positions are used to access any element in the List.
- Python considers the first element in a List as zero or 0, 1 represents the second position in the List, etc.
- Use a simple counting system to decide what to add, subtract, multiply or divide.
- You may use different methods.

Using methods improves your coding efficiency. Below are some commonly used methods:

- append() - Add Single Element to The List

- extend() - Add Elements of a List to Another List
- insert() - Inserts Element to The List
- remove() - Removes Element from the List
- index() - returns smallest index of element in list
- count() - returns occurrences of element in a list
- pop() - Removes Element at Given Index
- reverse() - Reverses a List
- sort() - sorts elements of a list
- copy() - Returns Shallow Copy of a List
- clear() - Removes all Items from the List

Explore these methods in your coding while practicing. You may read more about these codes by going to – Python List Methods - https://www.programiz.com/python-programming/methods/list. Remember, much like using loops serve as a foundation for programming, using arrays also serve as a basic expectation for programming.