**Instructions**: You must read the material and create an outline of the topics in your OWN words.  Do not copy the text from the tutorials into your notes. Make sure your outline contains notes for each subsection of the reading assignment. Thoroughly cover each topic to show you have a firm understanding of the programming concept or construct.

| **Ques** | **NOTES:** |
| --- | --- |
| What is a programming language?  Low-Level vs High-Level Programming Languages  Most Popular Programming Languages  Overview of Python  How to Use a Python Comment: Block, Inline, and Multiline | Simply put, it is the means to translate our intentions to computers in a language they understand. Programming languages are divided into two high-level and low-level. Humans write in the former and have it changed to the latter for machines (computers) to understand and carry out our instructions. Examples of high-level languages include Java, Python, JavaScript, etc., and low-level languages are binary, machine code (assembly language), etc. Each type has its philosophy and use case but are used based on these criteria.  Low-level programming languages are described as closer to machine code, or binary and are considered to be more difficult to read by humans. They are faster and offer precise control over how any computer will function.  High-level programming languages are more akin to how humans communicate. It uses words like object, order, run, class, request, etc.) that are human friendly. They are generally easier to program and though they take more time to translate into machine code for the computer to understand with the advances in computer science the time gap is gradually decreasing.  Programming languages are considered as popular based on the number of times it is search for (using the Github’s PYPL Index) but some of these include:   * Python: seen to be very versatile, general-purpose programming language. It is used in a variety of fields from data science and machine learning to web development. * Java: it can be used for many things, including mobile applications, software development, and large systems development. * JavaScript: this is a front-end and back-end friendly language that enables web applications, game development, and mobile applications. * C#: Microsoft’s popular language that can be used for a wide variety of applications, including game development, enterprise software, video games, mobile apps, and much more. * C++: is one of the most powerful programming languages and is used in a wide variety of industries. * PHP: is a widely used server-side language. It’s a great choice for building dynamic web applications and works well with databases and HTML. * R: is a statistical programming language popular among data scientists. * Swift: Apple’s programming language, and it is required if you plan to develop applications for iOS and MacOS. * Kotlin: do JetBrains develop an open-source programming language. It is popularly used for web development, Android development, and more.   This is a clear and powerful object-oriented programming language comparable to languages like Perl, Ruby, Scheme, or Java.  Python’s Notable Features:   * The syntax is quite elegant making it easy to write. * It is an easy-to-use language as it is simple to get your written program working on all operating systems. * It is very maintainable. * It comes with a large standard library that supports many common programming tasks. * It has an interactive mode that makes it easy to test your code. * It can be easily extended with new modules from compiled languages like C and C++. * It can be embedded into applications to provide a programmable interface. * It runs anywhere, including MacOS, Windows, Linux, and Unix. * It is free in two senses: it costs nothing to download and use and can be modified and distributed.   Some Programming-Language Features of Python include:   * It includes a variety of data types like numbers, strings, lists, and dictionaries. * It supports object-oriented programming with classes and multiple inheritances. * Its code can be grouped into modules and packages. * It supports raising and catching exceptions, resulting in cleaner error handling. * Data types are strongly and dynamically typed. * It contains advanced programming features such as generators and list comprehensions. * It has automatic memory management that frees up memory.   Glossary:   |  |  | | --- | --- | | Hashmark | A hashmark, hash character, or hashtag (#) tells the interpreter to ignore the rest of the line of code. | | Indentation | This is the space at the beginning of each line of code and indicates a new line. | | Source code | The human-readable instructions a coder writes to develop programs. | | Interpreter | This is a computer program that translates source code into machine code that the computer can read and execute. | | Newline | The “\n” is used to create a new line. If inserted into a string, all characters after the \n will be added to the new line. | | Comment out | Commenting out code describes using the hashmark to turn code into a comment so that Python will ignore it. | | Docstring | This is a string literal used to document a specific class, module, method or function. |   How to Comment in Python:  To add a comment you follow these steps:   1. Make sure your comment begins at the same indent level as the code it’s about. 2. Begin with the hashtag (#) and a space. The hash character tells the interpreter to ignore the rest of the line of code. 3. Write your comment. 4. Close your comment with a newline.   Example:  1 print(5 + 10)  2 y = 20 \* 5  3 print(y)  4 # Hi, this is a comment  5 print("This is a print statement")  Inline Commenting in Python:  This is used to add comments to just one line in your code and you begin your comment with a hashtag (#).  Example:  1 def price\_adjust(price):  2 return price \* 0.13 #Price increase to adjust for inflation rate |
| How to Comment Out a Block of Code in Python  What is the difference between comments and docstrings in Python?  Why Comment in Python?  Key Takeaways | Guides for inline Python comments:   * Avoid explaining your code in comments. Rather, if your code is too complex, rewrite it for clarity. * Focus on the *why* and the *how*. * Comments should not restate what is obvious in your code.   Examples of Helpful vs Unhelpful Comments:  Unhelpful:  1 statetax = 1.0625 # Assigns the float 1.0625 to the variable 'statetax'  2 citytax = 1.01 # Assigns the float 1.01 to the variable 'citytax'  3 specialtax = 1.01 # Assigns the float 1.01 to the variable 'specialtax'  Helpful:  1 statetax = 1.0625 # State sales tax rate is 6.25% through Jan. 1  2 citytax = 1.01 # City sales tax rate is 1% through Jan. 1  3 specialtax = 1.01 # Special sales tax rate is 1% through Jan. 1  How to Comment Out Multiple Lines in Python:  Python has built-in methods for multiline commenting, but we can use the hash character to comment on several lines.  1 # You can create a multi-line comment  2 # by adding the hash symbol to each line  Indentation is used to identify a block of related code in Python and to add a comment(s) to a large block of code; the docstrings are recommended. The docstring is the string literals appearing directly after the definition of a function. It can be used to associate documentation with classes, functions, modules, and methods, but it can also be used to comment out code.  One-line docstrings:  To create this, do the following:   1. Make sure the docstring begins at the same indentation level as the scope. 2. Use a triple quote to start the string, followed by a phrase beginning with a capital letter and ending with a period. 3. Close the string with triple quotes.   No blank lines should exist before and after a single line docstring.  ''' This is a comment using string literals with triple quotes '''  Multiline docstrings:  Here we wrap the multiline comment inside a set of triple quotes. Technically, this is not a comment; it’s a string that isn’t assigned any variable.  '''  This is a multiline  comment using string literals  with triple quotes  '''  Note that the docstring method does give you multiline comment functionality but remember that they are not technically comments. They are strings not assigned to any variable allowing the program to ignore them at runtime.   |  |  | | --- | --- | | **Python comment** | **Python docstring** | | Used to leave notes about a segment of code | Used to document functions, classes, and modules | | Exists only in the source code | Will be embedded in code for some modules, functions, and classes | | Enables the programmer to provide additional information about the code | Enables the program to provide descriptions of modules, functions and classes | | Describes why and how, not what | Describes what, not how or why | | Can **not**be accessed with the built-in help function and doc attribute | Can be accessed with the built-in help function and doc attribute |   They are useful as reminders or as helpful documentation for you and others. They allow users to communicate logic, algorithms, and formulas in source code without affecting the program’s execution. They are also useful for testing and debugging.  Using Python Comments for Testing:  They may include the following:   * Commenting out new code to ensure smooth implementation of a previous code. * Trying out various programming methods. * Pinpointing the source of an error by commenting out and running parts of a program.   Example:  a = 2 + 5  #b = 7 + "four"  c = "eight" + "zero"  here, the second line causes an error and so is commented out.  The following key points should be the focus of creating comments:   * Comments and docstrings should be made at the same indentation level as the code they’re about. * Comments begin with a # and a space. * Docstrings begin and end with triple quotes. * Python has no built-in function for multiline comments, but you can use several lines beginning with hash characters or docstrings. * Comments should be about the why and the how. * Docstrings should be about the what. |
| Hello World  Variable Types  Composite Variables  Python Dictionaries  Python Type Conversion | To start off with Python, we use the hello world printing exercise, but it should be noted that we have two versions of Python 2 and 3, with the latter being more recent and preferred.  Python is completely object-oriented and not “statically typed.” Here, you do not need to declare variables before using them or declare their type, and every variable in Python is an object.  Numbers:  Two types of numbers are supported in Python – integers(whole numbers) and floating point numbers(decimals), and also support complex numbers.  Floating Point Numbers:  myfloat = 7.0  print(myfloat)  myfloat = float(7)  print(myfloat)  Strings:  These are defined either with a single quote or a double quote.  mystring = 'hello'  print(mystring)  mystring = "hello"  print(mystring)  Note that double quotes make it easy to include apostrophes while including this in single quoted strings would terminate the string.  Variables are Useful:  These can be described as containers used to store information (data) and made available to whoever or whatever processes need them. They are a useful tool to programmers.  A Need for Something More:  More complex data types are needed to hold more complex information, such as a shopping list or list of friends to be invited to a party.  Composite Variables:  These are present in Python and are designed to hold several things at once or a collection of items.  List:  This data type in Python holds a list of items with a square bracket [ ]. These items are separated by commas:  names = [“Ekpo”, “Akpan”, “Imabong”]  evens = [2,4,6,8,10]  Dictionaries:  This composite variable data is stored in key: value pairs. Individual values can be accessed using the keys.  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 }  Data is stored in key: value pairs.  Dictionary Creation:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } print(thisdict)  Dictionary Items:  These are ordered and changeable, and duplicates are not allowed.  Accessing an item:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } print(thisdict["brand"])  ordered or Unordered:  As of Python 3.7, dictionaries have been ordered. In Python 3.6 and earlier, dictionaries are unordered.  When we say that dictionaries are ordered, it means that the items have a defined order, and that order will not change.  Unordered means that the items do not have a defined order, you cannot refer to an item by using an index ( a number assigned to an item).  Changeable:  This means we can change, add, or remove items after creating the dictionary.  Duplicates Not Allowed:  Two items with the same key are not allowed:  thisdict = {   "brand": "Ford",   "model": "Mustang",   "year": 1964,   "year": 2020 }  Dictionary Length:  This is used to determine the number of items and uses the len() function.  Dictionary Items – Data Types  The value of dictionary items can be of any data type.  thisdict = {   "brand": "Ford",   "electric": False,   "year": 1964,   "colors": ["red", "white", "blue"] }  Type(): Dictionaries are defined as objects with the data type of ‘dict’.  Dictionary Constructor:  This is also used to create a dictionary:  thisdict = dict(name = "John", age = 36, country = "Norway") print(thisdict)  Python Collections (Arrays)  There are four collection data types in the Python programming language:   * List * Tuple * Set * Dictionary   This is a process in programming that involves the conversion of data of one type to another, e.g. converting int data to str.  They two types in Python:   1. Implicit Conversion – automatic type conversion. 2. Explicit Conversion – manual type conversion.   Implicit Type Conversion:  In this situation Python promotes the conversion of the lower data type (integer) to the higher data type (float) to avoid data loss.  In the example below int addition operation would round the number up but since in this case it is to be added to a floating point number the int is converted to float to prevent data loss.  integer\_number = 123  float\_number = 1.23  new\_number = integer\_number + float\_number  # display new value and resulting data type  print("Value:",new\_number)  print("Data Type:",type(new\_number))  output:  Value: 124.23  Data Type: <class 'float'>  Note: Python always converts smaller data types to larger data types to avoid the loss of data.  Note:  For the conversion of type str to int, we get TypeError as Python cannot use Implicit Conversion.  Explicit Type Conversion:  Here, users convert the data type of an object to the required data type using built-in functions like int(), float(), str(), etc., as the case may be. This type of conversion is also called typecasting because the user casts (changes) the data type of the objects. |
| User Input in Python  Python Arithmetic Operators  Reading From a File | Example:  num\_string = '12'  num\_integer = 23  print("Data type of num\_string before Type Casting:",type(num\_string))  # explicit type conversion  num\_string = int(num\_string)  print("Data type of num\_string after Type Casting:",type(num\_string))  num\_sum = num\_integer + num\_string  print("Sum:",num\_sum)  print("Data type of num\_sum:",type(num\_sum))  Output:  Data type of num\_string before Type Casting: <class 'str'>  Data type of num\_string after Type Casting: <class 'int'>  Sum: 35  Data type of num\_sum: <class 'int'>  Key Points to Consider:   1. Type Conversion involves the conversion of one object from one data type to another data type. 2. the Python interpreter automatically performs Implicit Type Conversion. 3. Python uses Implicit Type Conversion to prevent data loss. 4. Explicit Type Conversion is also known as Type Casting; here, conversion is done using predefined functions by the user. 5. Data loss may occur in Type Casting as we enforce the conversion from one data type to another.   A major aspect of computer programming is the ability and need to get input from a user, and so in Python, the inbuilt function input() is used to collect user input. It should be noted that the type returned by this function is always string (str), so the programmer must do an Implicit Conversion after collecting user input.  Example:  number = int(input(“Number: “))   |  |  |  | | --- | --- | --- | | **Operator** | **Name** | **Example** | | + | Addition | x + y | | - | Subtraction | x - y | | \* | Multiplication | x \* y | | / | Division | x / y | | % | Modulus | x % y | | \*\* | Exponentiation | x \*\* y | | // | Floor division | x // y |   In addition to using the input() function to get user input, Python can read data from files and use this data in your program. It is recommended to place the data file in the same directory as the Python code file (.py file) so as to reduce the chances of error, but this is not to say Python can handle files in different locations.  Get a Handle On It:  First, we need to create a “handle” for the file, then use the methods associated with this handle to read the contents of that file.  fh = open(“numbers.txt”, “r”)  Here, we use the open() function to access the file by passing two arguments: the exact file name and a character indicating the file mode, in this case, “r”. The “r” indicates we are opening the file for reading, and if the file is found, the method returns a reference to the file and stores it in a variable called “fh” defined by us.  Three Ways to Read:  Let’s use the above image to demonstrate how files are read.  The Python handle we created above can read the file with several methods:  read()  this method would read every line in the above file and return a list of such: ['2', '4', '6', '8', '10\n', '12', '14', '16', '18', '20\n', '22', '24', '26', '28', '30\n']. Note that each character starting from ‘2’ is given an index with two being zero (0) and each being of type string. Hence data[0] will return ‘2’ and data[1] will return a “comma” and yet again data[8] will return ‘1’ not ‘10’. It is read like this: It is read like this: 2-comma-4-comma-6-comma-8-comma-1-0. You will see index 8 (start counting with zero!) is '1'.  Notice the ‘\n’ this is the newline character that’s invisible to us as we read files line by line but programming languages can detect it and use it to determine the end of file.  readline():  This reads on line at a time and returns a list of the contents of the line: "2, 4, 6, 8, 10\n".  readlines():  This reads all the lines into a list where each element of the list is a line:  ['2,4,6,8,10\n', '12,14,16,18,20\n', '22,24,26,28,30\n']. Here data[1] will return "12,14,16,18,20\n'".  Processing the Data:  The data read is referred to as “comma separated values” or csv and make it easy to process as we use the string split(delimiter) function to split them into a list like so:  Data = “2,4,6,8,10”.split(“,”) 🡪 [‘2’, ‘4’, ‘6’, ‘8’, ‘10’]  We split the string at wherever we found a comma, with that we can iterate over the list and do whatever we want.  nums = [] for item in data:      nums.append(int(item))  for item in data:      print(item)  Shared Understanding  Based on the above data we can easily pick out information based on the presence of comma between the datum on each line and with that we can read in the data using readline() and then split it by the comma and add that to a dictionary like so:  president = {}  #empty dictionary fh = open("presidents.txt", "r")   #open the file for reading data = fh.readline()   #read a line presdata = data.split(",")   #split on the commas, create a list president['lastname'] = presdata[0]  #the first value--index 0-- is last name president['firstname'] = presdata[1]  #the second value--index 1-- is first name president['height'] = presdata[2]  #the third value--index 2-- is height in cm president['weight'] = presdata[3]  #the fourth value--index 3-- is weight in pounds  Then if we print out each item we get this:  {'lastname': 'Lincoln', 'firstname': 'Abraham', 'height': '193', 'weight': '185\n'} |

Summary:

I learned about programming and the different types of languages that are used both at the high level and low level. I also learned about Python, its use, and the basic language components. I learned about data types and how to do conversion between them. I learned about how Python reads data from a user and also from a file and the different methods used to carry out such operations.