**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:** |
| --- | --- |
| Leveraging Knowledge and Understanding | No Point in Reinventing the Wheel  This paradigm promotes using already established tools or methods to solve an existing problem. It is common for those programming in Python to attempt to write Python functions for which an equivalent function already exists because they don’t know that such a function exists or may not know where to find them. The intention here is to show us how to find these pre-existing modules and functions so that we can leverage or multiply the skills that we have as a Python programmer.  Module and Libraries  Code that carries out similar functions, like calculating various statistics or generating random numbers, for example, can be collectively called modules and modules grouped together called libraries. Python language actually contains over 200 modules as part of the Standard Library. In addition to the built-in libraries, many third-party vendors provide free or low-cost modules to help you accomplish your complex programming goals. We can use search engines to find them, and commercial repositories of helpful libraries, like Github, provide a central supply hub that you can tap into.  What’s in a Namespace  In computer programming, we write blocks of code we call functions, and each of these functions has a name. When we import modules or libraries into our program, we bring in groups of new functions; two or more functions in our combined code may have the same name. To avoid this, we include the name of the module or library referenced. We do this in three different ways:   1. The standard way: “import statistics”   We use this in our code by prepending the name of the library onto any functions we call.  med = statistics.median([2,3,4])   1. Using alias:   Import statistics as stat  The alias follows the word “as” and is used like this  med = stat.median([2,4,3,5])   1. Import the module or library into the built-in (i.e. default) namespace of Python like this:   from statistics import \*  with the last option, we don’t need to prepend a method call by a namespace call by a namespace or an alias. Here, you are saying there are no overlapping function names; metaphorically, no two people have the same name. This method is discouraged by many programmers as it will cause the default of the built-in namespace to be polluted, and a clash of function names could inadvertently occur; others see no problem using it.  Knowing What is Available and How to Use it  How do we use these modules or libraries; we search online documentation using the name of the module followed by a reference phrase “reference.” An example of online documentation would be: |
|  |  |
|  | Additionally, the reference also provides examples of how to use the functions: |
| Modules vs Libraries in Python | Modules are different from libraries and often confuse new comers to Python programming.  Module and Libraries in Python:  Real-world programs are complex, and small programs can use thousands of lines of code. for this reason, programmers prefer to modularize their code to make it easy to understand and use. Python focuses on modularity to make its use and flow easy.  Modules:  These are a collection of related codes that are packed under a Python program and may contain functions, classes, or variables. It is also used to accommodate runnable codes within modules. There are normal files with the suffix .py when created. The types include:  Predefined Modules:  These are also called built-in modules, and Python caters to many built-in modules. They can be used directly in Python programs using their name and the keyword “import.” E.g., import math. Most are written in C and combined with the Python interpreter to make the interpreter work with these.  Examples include math, datetime, statistics, random, os, sys, etc.  Program:  from math import sqrt, factorial  print('Square root of 64: ',sqrt(**64**))  print('Factorial of 4: ',factorial(**4**))  Output:  Square root of 64: **8.0**  Factorial of 4: **24**  User-defined Modules:  Python makes it possible and easy to create your own modules. This allows the programmer to customize their operations and control their hands. Within these modules, a user can create their own set of functions, variables, and classes and the importing mechanism is the same the built-in modules.  Program:  newModule.py  def **funcName**(val):  print("You have passed the", val, "value as the parameter")  #funcName()  AnotherProgram:  import newModule  newModule.funcName(**6**)  Libraries:  This umbrella term comprises a reusable set of Python code/instructions. This is a collection of related modules bundled together under one single name. It is popularly used by developers to share reusable code with the community. Developers and community researchers can create their own set of useful functions related to the same domain. Standard libraries come bundled with the Python interpreter when programmers and developers install it in their systems. Some common examples of Python libraries are matplotlib, Pygame, Pytorch, Requests, Beautifulsoap, etc.  Program:  import matplotlib.pyplot as mpl  a = [**6**, **4**, **9**]  b = [**3**, **5**, **8**]  mpl.plot(a, b)  mpl.xlabel('x\_axis')  mpl.ylabel('y\_axis')  mpl.title('Data Visualization')  mpl.show()  Output:  Difference Between Modules and Libraries:   |  |  | | --- | --- | | **Modules** | **Libraries** | | A module is a collection of code or functions that uses the .py extension. | A Python library is a set of related modules or packages bundled together. | | It is used by the programmers as well as the developers. | It is mostly used by the community members, developers, and researchers. | | Use of modules makes reading the code easier. | Libraries do not contribute in better readability. | | Modules logically cluster the functionality that programmers can import to reuse their code or set of statements. | Libraries make the collection of logically related code reusable for the programming language users, developers, and other researchers. | | Whenever a programmer imports a module in a Python program, the interpreter scans for several locations to look for the module's definition or body. | We have to install the libraries in our Python project before using its modules or packages. Usually, we use the pip install command. | | When a module is not found by Python's import statement, it searches for each directory within the shell variable i.e., PYTHONPATH. The PYTHONPATH is Python's environment variable consisting of a list of directories. | Whenever the Python interpreter do not found the Python library associated with the project, it shows an error message and the program ends abruptly. | | Modules are mostly written using valid Python statements or codes. | Libraries, mainly standard libraries, are mostly written using C language or Python. | | The main focus of creating modules is to avoid DRY (Don’t Repeat Yourself). | Libraries do not have any such objective. | | We can use the Python’s built-in dir() function to return a sorted list of strings holding the function names defined within a module. | There is not such explicit function that can return the number of modules a library contain. Still, programmers can use the help() to extract some information. | | Example of popular built-in Python modules are os, sys, math, random, etc. | Example of popular built-in Python libraries are Pygame, Pytorch, matplotlib, etc. | |
| Libraries in Python  Pattern Recognition  A Peek at Databases | Conclusion:  As a programmer doing a normal project using Python, he/she should try to cluster the code into small chunks of functions and bring them under one module, and as a developer or a researcher working for the Python community or working for any company that wants a large bundle of similar modules, bring the modules under one library.  Compared to the traditional libraries where books are kept in programming, a library is a collection of precompiled codes that can be used later in a program for specific, well-defined operations. It can also contain documentation, configuration data, message templates, classes, values, etc.  In Python, a library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes it simpler and more convenient for the programmer, and we don’t need to write the same code repeatedly for different programs. Libraries play a vital role in the fields of Machine Learning, Data Science, Data Visualization, etc.  Working of Python Library  We use libraries so that we don’t need to write the code again in our program that is already available and works. The library files in the MS Windows environment have a DLL extension (Dynamic Load Libraries). When the library is linked with our program and runs that program, the linker automatically searches for that library, extracts the functionalities of that library, and interprets the program accordingly. This is how we use the library methods in our program.  Python Standard Library  This contains the exact syntax, semantics, and Python token. It contains built-in modules that provide access to basic system functionality like I/O and some other core modules. Most are written in C programming language and contain over 200 core modules. All these works together to make Python a high-level programming language. The Python Standard Library plays an important role, and without it, programmers can’t have access to the functionalities of Python. These are some of the commonly used libraries:   1. TensorFlow: developed by Google in collaboration with the Brain Team. It is open-sourced and used for high-level computations. It is used in machine learning and deep learning algorithms. 2. Matplotlib: this is open-sourced and used in the plotting of numerical data. 3. Pandas, used by data scientists and open-sourced, are important in machine learning. It provides flexible high-level data structures and a variety of analysis tools. Pandas support operations like Soring, RE-indexing, Iteration, Concatenation, Data Conversion, Visualization, Aggregation, etc. 4. Numpy: the name stands for “Numerical Python.” It is the commonly used library for mathematical operations and contains mathematical functions for easy computations. Even libraries like TensorFlow use Numpy internally to perform several operations on tensors. The Array Interface is one of the key features of the library. 5. SciPy: the name stands for “Scientific Python” and is an open-source library for high-level scientific computations. It is built as an extension of Numpy. It works with Numpy to handle complex computations and is widely used by application developers and engineers. 6. Scrapy: an open-source library used to extract data from websites. 7. Scikit-learn: is a famous Python library that works with complex data. It is open-source and also supports machine learning. It supports various supervised and unsupervised algorithms like linear regression, classification, clustering, etc. It works in association with Numpy and SciPy. 8. PyGame: provides an easy interface to the Standard Directmedia Library (SDL) platform-independent graphics, audio, and input libraries. It is used to develop video games using computer graphics, audio libraries, and Python programming language. 9. PyTorch: this is the largest machine-learning library that optimizes tensor computations. It has a rich API to perform tensor computations with strong GPU acceleration. It is used in applications built for neural networks. 10. PyBrain: the name stands for Python-based Reinforcement Learning, Artificial Intelligence, and Neural Networks library. It is open-source and built for beginners in the field of Machine Learning. Provides fast and easy-to-use algorithms for machine learning tasks.   Python has many more libraries and plays a crucial role in helping developers.  Use of Libraries in Python Program  As we write large programs in Python, we want to reuse parts of the code, so we strive to make our code modular. We do this in Python by creating modules that can be imported and reused. When we have multiple related modules, we store them in a library and import them from that library for use in our code.  # Importing math library  **import** math    A **=** 16  print(math.sqrt(A))  Output:  4.0  Here in the above code, we imported the math library and used one of its methods, i.e., sqrt (square root), without writing the actual code to calculate the square root of a number. That’s how a library makes the programmers’ job easier. But here, we needed only the sqrt method of the math library, but we imported the whole library. Instead of this, we can also import specific items from a library module.  Importing Specific Items from a Library Module  Sometimes, we may need to import one or two methods from a library, and this can be done as seen below:  # Importing specific items  **from** math **import** sqrt, sin    A **=** 16  B **=** 3.14  print(sqrt(A))  print(sin(B))  Output:  4.0  0.0015926529164868282  Patterns can be found in ideas, words, symbols, and images. Pattern recognition is the ability to see order in a chaotic environment, which is the primary condition for life. This type of intelligence is said to have the highest correlation with the general intelligence factor, g. Although pattern recognition is mostly unlearned, it is still possible to train!  Databases Hold Data Tables  These are containers that hold tables. Tables can hold reams of information that are managed by a database. Databases provide a host of tasks such as managing access to data, providing a means of creating, changing, and destroying tables, and extracting specific data from those tables in response to commands from the user. These commands are written in a language called Structured Query Language, or SQL, and pronounced “Ese-cue-ell” or “Sequel.”  CRUD  One is considered proficient in SQL if you can write and execute four types of SQL statements: Create, Read, Update and Delete. CRUD is an acronym that takes each of its letters from those tasks.  An example table: |
|  | From above, it shows that a row is a record, and the vertical columns are fields, or the name describing the information stored there.  So, to perform CRUD on this data table, you would need to be able to:   * + 1. Add a new record-- a new person-- to this contacts table     2. Extract any record, or any subset of records, from this table to read them     3. Update any piece of information in the table that needs to be changed, and     4. Delete any record you no longer need.   Getting Started: The Three Cs  Connect  To use a database, we must either create it or locate it using Python. To do this, we use the following commands:  import sqlite3  conn = sqlite3.connect(‘myinfo.db’)  Here, line one brings in the SQLite3 module, and the second line calls the connect method from that module. A connection is made if a database exists in the current directory. If it does not exist, the database is created, and a connection to it is established.  Create a Cursor  After creating a database, we need to add to it, and to do this, we use the Cursor object that acts as a go-between Python and the database.  cursor = conn.cursor()  it communicates directly with the database through the connection object created earlier. Any operations done with the database are done through it, and it holds on to whatever information it has been asked to retrieve from the database.  Create a table  After creating the database to hold and manage data tables, we use the cursor object to add a table to it, and it does this using the SQL language.  From above, in line 1, we import the SQLite3 module; in line 2, we create the connection object; in line 3, we create the cursor object from the connection object; then, in line 4, we state the SQL command to use, and the general format is CREATE TABLE CONTACTS( ). Inside the parenthesis, we have the names and datatypes of the eight fields we will have in our table: contacted, last, First, Address, City, State, PostalCode, and Phone. Spaces are not allowed.  A data type follows the column name; here, it is either an int (integer) or a varchar(n) where n is some whole number. Varchar(n) means a variable number of characters up to the number in the parenthesis. Note that the term PRIMARY KEY also follows the int data type for the *contactid*field.  The only modifier here is the PRIMARY KEY, and this is our guarantee to the database that this field will be unique – no duplicates.  Line 13 is an insurance against crashing: if we try to make something that already exists, the program will crash. So we use line 13 to tell our program to drop (destroy) the table if it already exists, allowing us to create a fresh one.  Line 14 executes the SQL created in lines 4-12.  Inserting Data  Here, we instruct the cursor to a) insert a record in the contacts table and b) have the following sequence of fields matched up to the sequence of data that follows:  sql = ''' INSERT INTO CONTACTS (contactid, Last, First, Address, City, State, PostalCode, Phone)   VALUES (1, 'Roosevelt', 'Franklin', '4097 Albany Post Rd', 'Hyde Park', 'NY', '12538','845 229-5320') ''' cursor.execute(sql)  Reading Data  After inserting a record, we can now instruct the cursor to retrieve that record using the SQL language:  sql = "SELECT \* from CONTACTS" cursor.execute(sql)  A data set is returned by the cursor and we access the data by asking for it (fetchall()) and assigning it to a variable:  result = cursor.fetchall()  for row in results:  print(row)  for our current record only one will be returned but it is also possible to return zero records. In a situation where we have a database with thousands of records we can use the WHERE clause to select a subset of records from the table:  sql = "SELECT \* from CONTACTS WHERE state = 'NY'"  Updating Records  Records change, and it’s possible to update an existing record using the WHERE clause to pinpoint where the change should be made:  sql = "UPDATE contacts set Phone = '845 229-5302' WHERE contactid = 1" cursor.execute(sql)  Deleting Records  When deleting records, we must use the WHERE clause to make sure we don’t inadvertently delete the wrong records. In using this key it’s best to use it with unique keys in the table helping to ensure only the desired records are deleted.  sql = "DELETE from CONTACTS WHERE contactid = 1" cursor.execute(sql) |