**IOT – Assignment on Python**

***Python Data Types & Data Structures:***  
I. *Introduction to Python*

- Python is a high-level, interpreted programming language that emphasizes code readability and simplicity.

- It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

- Python has a large standard library and many third-party modules that make it useful for a wide range of applications.

II. *Main characteristics*

- Multi-paradigm programming language: Python supports more than one programming paradigm, including object-oriented programming and structured programming.

- Interpreted Language: Python is an interpreted language and does not require an explicit compilation step. The Python interpreter executes the program source code directly, statement by statement, as a processor or scripting engine does.

- Minimalistic: Due to its minimalistic nature, Python provides a manageable structure for large programs.

- Portable: Since Python is an interpreted language, programmers do not have to worry about compilation, linking, and loading of programs. Python programs can be directly executed from source.

- Broad Library Support: Python has a broad library support and works on various platforms such as Windows, Linux, Mac, etc.

- Dynamically Typed: In Python, you don't need to declare the data type of a variable explicitly. The interpreter automatically assigns the data type based on the value assigned to it.

- Easy to Learn: Python has a simple syntax that is easy to learn and understand. It also has a large community of developers who contribute to its development and provide support.

III. *Python Data Types*

- Numbers: integers, floating-point numbers, and complex numbers

- Strings: sequences of characters enclosed in quotes

- Booleans: True or False values used for logical operations

IV. *Python Data Structures*

- Lists: ordered collections of items that can be modified (e.g., [1, 2, 3])

- Tuples: ordered collections of items that cannot be modified (e.g., (1, 2, 3))

- Dictionaries: unordered collections of key-value pairs (e.g., {'name': 'Alice', 'age': 30})

V. *Type Conversions*

- You can convert between data types using built-in functions like int(), float(), str(), and bool().

- For example, you can convert a string to an integer using int('42').

VI. *Examples*

Here are some examples of how you might use Python data types and structures:

- You could use a list to store the results of a survey or poll.

- You could use a tuple to represent the coordinates of a point in space.

- You could use a dictionary to store information about customers in an e-commerce website.

***Introduction:***

Python is a versatile and powerful programming language that supports multiple programming paradigms and easy to learn, read, and maintain. Additionally, Python offers an interactive mode that allows users to submit commands and interact directly with interpreter. This feature is particularly useful for beginners and experienced programmers alike.

*Interactive Mode in Python:*

Python provides an interactive mode that allows users to submit commands and interact with the interpreter directly. Once inside the Python interpreter, users can type in commands at will, and relevant output is displayed on subsequent lines without the >>> symbol. Quantities stored in memory are not displayed by default, but users can display them by typing their names. To exit the interpreter, users simply need to type ctrl - D.

*Python Libraries and Packages:*

Python has a vast library support and works on various platforms, including Windows, Linux, and Mac. Some popular libraries and packages in Python include NumPy for scientific computing, Pandas for data analysis, Matplotlib for data visualization, Scikit-learn for machine learning, Flask for web development, and Django for building web applications. There are many more libraries and packages available in Python that can be used to solve a wide range of problems.

*Resources for Learning Python:*

There are several resources available for learning Python, including books like "Learning Python" by Mark Lutz & David Ascher, "Programming Python" by Mark Lutz, and "Core Python Programming (2nd Edition)" by Wesley J. Chun. In addition, several websites provide helpful resources for learning Python, such as an online version of built-in Python function documentation, quick reference cards for Python, and extensive Python forums.

*Conclusion:*

Python is an easy-to-learn, read, and maintain programming language that is powerful and versatile enough to handle a wide range of applications. Its interactive mode and extensive library support make it an excellent choice for beginners and experienced programmers alike. With a solid foundation in Python programming, users can take their skills to the next level and solve complex problems across various platforms.

***Types & operators:***

Python is a versatile programming language that supports different numeric types and operators for performing calculations and manipulating data. It allows users to work with integers, long integers, floating-point numbers, octal constants, hex constants, and complex numbers. The language also offers a variety of operators for working with strings, including concatenation and repetition. Python supports mixed-type math and follows the PEMDAS order of operations, and the final answer will be of the most complicated type used.

*Types of Numbers :*

Python supports different numeric types, including integers, long integers, floating-point numbers, octal constants, hex constants, and complex numbers. Integers are positive or negative whole numbers, while long integers are similar but can be arbitrarily long and must end in either "l" or "L." Floating-point numbers can represent decimal fractions or scientific notation. Octal constants must start with a leading 0, while hex constants must start with a leading 0x or 0X. Complex numbers must end in "j" or "J." Typing in the imaginary part first will return the complex number in the order Re+ImJ.

Examples:

* Integers: 0, 1, 1234, -56
* Long integers: 999999999999999999999L
* Floating-point numbers: 0., 1.0, 1e10, 3.14e-2, 6.99E4
* Octal constants: 0177, -01234
* Hex constants: 0x9ff, 0X7AE
* Complex numbers: 3+4j, 3.0+4.0j, 2J

*Strings and Operations:*

Strings are ordered blocks of text enclosed in single or double quotation marks. Double quotation marks allow users to extend strings over multiple lines without backslashes, which signal the continuation of an expression. Python offers concatenation and repetition operators for manipulating strings. Concatenation is done with the + sign, while repetition is done with the \* sign.

Examples:

* Concatenation: 'abc '+'def' yields 'abcdef'
* Repetition: 'abc '\*3 yields 'abcabcabc'

*Operations on Numbers:*

Python provides various operators for performing calculations on both integers and floating-point numbers, including basic algebraic operations, exponentiation, comparison operators, bitwise operators, and shift operators.

Examples:

* Basic algebraic operations: a+b, a-b, a\*b, a/b
* Exponentiation: a\*\*b
* Comparison operators: a < b, a > b, a <= b, a >= b, a == b, a != b
* Bitwise operators: a | b, a ^ b, a & b
* Shift left or right by b bits: a << b, a >> b

***Python Basics:***

Python is a popular programming language known for its simplicity and ease of use. It provides various features such as control flow statements, functions, and methods that can be used to perform different operations on data.

*Control Flow Statements :*

Control flow statements are used to execute different blocks of code based on conditions and iterate over sequences. Python provides three main control flow statements: if, while, and for.

*If Statement:*

The if statement is used to execute a block of code if a condition is true.

Multiple conditions can be checked using elif statements, and a default action can be specified using else.

Example:

x = 5

if x > 5:

print("x is greater than 5")

elif x == 5:

print("x is equal to 5")

else:

print("x is less than 5")

*While Statement:*

The while statement is used to execute a block of code repeatedly as long as a condition is true.

Example:

x = 1

while x < 4:

print(x\*\*2)

x = x + 1

*For Statement:*

The for statement is used to iterate over a sequence (such as a list or string) and execute a block of code for each item in the sequence.

Example:

fruits = ['apple', 'banana', 'cherry']

for fruit in fruits:

print(fruit)

can also be used for general looping.

Example:

for i in range(5):

print(i)

This code will output the numbers 0 through 4, since range(5) generates a sequence of integers from 0 up to (but not including) 5.

*Functions:*

Functions are blocks of code that can be called repeatedly from different parts of a program. Functions can take arguments (inputs) and return values (outputs).

*Defining a Function:*

The basic structure of a function is:

def function\_name(arguments):

action

return value

Example:

def square(x):

return x\*\*2

print(square(3))

*Default Arguments:*

Functions can have default arguments, which are used if no value is provided for that argument when the function is called.

*Defining a Function with Default Arguments:*

The basic structure of a function with default arguments is:

def function\_name(argument1, argument2=default\_value):

action

return value

Example:

def greet(name, greeting='Hello'):

print(greeting + ', ' + name + '!')

greet('Alice')

greet('Bob', 'Hi')

*Methods:*

Methods are built-in functions that can be used to perform common operations on data. Python provides several built-in methods for strings and lists.

*String Methods:*

Strings have several built-in methods that can be used to manipulate them.

Example:

my\_string = ' hello, world '

print(my\_string.upper())

print(my\_string.strip())

print(my\_string.replace('l', 'L'))

print(my\_string.split(','))

*List Methods:*

Lists also have several built-in methods that can be used to manipulate them.

Example:

my\_list = [1, 2, 3]

my\_list.append(4)

my\_list.insert(0, 0)

my\_list.remove(2)

print(my\_list.pop())

*Conclusion:*

Python provides various features that can be used to perform different operations on data. Control flow statements can be used to execute different blocks of code based on conditions and iterate over sequences. Functions can be used to encapsulate blocks of code and make them reusable. Methods can be used to perform common operations on data such as strings and lists. Python's simplicity and ease of use make it a popular choice for various applications such as data analysis, scientific computing, and web development.

***Running Python Code:***

Python scripts can be written in text files with the suffix .py. The scripts can be read into the interpreter in several ways, including running a script from the command line or running Python interactively from the command line. Python can also be run using an Integrated Development Environment (IDE).

***Input and Output:***

Python provides several built-in functions for input and output, including print() and input(). Python can also read from and write to files using file objects. To open a file for reading, use the open() function with mode 'r'. To open a file for writing, use mode 'w'. Python also provides various modules such as os, sys, and shutil that can be used to perform operations on files and directories.

***Indentation:***

In Python, indentation is used to indicate blocks of code. Indentation is typically done using four spaces (not tabs). Incorrect indentation can lead to syntax errors or unexpected behavior.

Proper indentation is important in Python, as it determines which statements are part of a block and which are not.

Examples:

if x > 0:

print('x is positive')

else:

print('x is not positive')