and now I will perform the sample scripts on the following previous topic first is the python data types. Python supports various data types, including

integers, floats, strings, and booleans. Variables are used to store data, and their type is determined by the value assigned to them.

* **Python Data Types:**

integer = 10

print("Integer:", integer)

float = 10.5

print("Float:", float)

string = "Hello, World!"

print("String:", string)

boolean = True

print("Boolean:", boolean)  
  
**integer = 10**  
- This line creates a variable named integer and assigns it the value 10, which is a whole number without a decimal point in Python.  
  
**float\_num = 10.5**  
- This line assigns float to 20.5, a decimal point, representing real numbers, fractions, and irrational numbers, as a variable.

**string = "Hello, World!"**  
- This line assigns a string variable, "Hello, World!", which is an immutable sequence of characters, including letters, numbers, and special characters.  
  
**boolean = True**  
- This line creates a boolean variable in Python, assigning its value True, a special data type representing truth values, which can only be True or False.

**Second is the Python Numbers**: Python includes a wide range of operators for performing arithmetic, comparison, and logical operations. These operators allow you to manipulate data and make decisions in your code.

* **Python Numbers:**

a = 5

b = 10

print(a + b) # Addition

print(a - b) # Subtraction

print(a \* b) # Multiplication

print(a / b) # Division

**a = 5**

**b = 10**

**-** These lines create two variables, a and b, with values 5 and 10, respectively, using the = operator in Python, representing numerical data.

**print(a + b) # Addition  
-** This line adds a and b using the + operator in Python, resulting in 15 and printing it to the console.

**print(a - b) # Subtraction  
-** This line performs a subtraction operation between a and b in Python, resulting in -5, which is then printed to the console.

**print(a \* b) # Multiplication  
-** This line performs a multiplication operation between a and b using the \* operator in Python, resulting in 50, which is then printed to the console.

**print(a / b) # Division  
-** This line performs a division operation between a and b values in Python, resulting in a / b value of 0.5, which is then printed to the console.  
 **# -** The lines ending with # are comments; they are ignored by Python interpreters but provide explanations or notes within the code, making it more readable and understandable.

**Next is the Variable Assignments**: Variables in Python are used to store data, and their type is determined by the value assigned to them. Python is a dynamically typed language, meaning you don't have to declare the type of a variable when you create it.

* **Variable Assignments:**x = 5  
  y = "Hello"  
  z = x  
  print(z)

**x = 5  
-** This line assigns the value 5 to a variable named x, which is an integer in Python, which can be positive, negative, or zero.  
 **y = "Hello"  
-** This line assigns a string value to a variable named y, which is immutable and cannot be changed once assigned.  
 **z = x  
-** This line assigns a variable named z to the value of x, creating a reference to the same object. In Python, variables are references to objects, and changes made to one variable will be reflected when accessing the other.  
 **print(z)  
-** This line uses the print(round bracket) function to output the value of z to the console, assigning it to x, which is 5.

**Then next is the Strings**: Strings are sequences of characters enclosed in quotes. They are immutable, meaning their value cannot be changed once assigned. Strings can contain letters, numbers, and special characters.

* **Strings:**greeting = "Hello, World!"  
  print(greeting[0]) # Accessing the first character

**greeting = "Hello, World!"  
-** This line assigns a string value to a variable named greeting, which is immutable and cannot be changed once assigned.  
 **print(greeting[0]) # Accessing the first character  
-** The `print()` function in Python outputs the first character of the string 'greeting' to the console, using the index number to access individual characters in a string.

**Next is the Indexing and Slicing**: Python allows you to access individual characters in a string using indexing, and you can access a range of characters using slicing. This feature is useful for manipulating strings, such as extracting specific characters or substrings.

* **Indexing and Slicing:**list\_example = [1, 2, 3, 4, 5]  
  print(list\_example[1:4]) # Slicing from index 1 to 3

**list\_example = [1, 2, 3, 4, 5]  
-** This line assigns a list of integers to a variable named list\_example, a mutable list in Python.   
 **print(list\_example[1:4]) # Slicing from index 1 to 3  
-** The print() function outputs a portion of list\_example to the console using slicing, accessing items from index 1 to 3, and returning a new list.

**Next is the Print Formatting with Strings**: Python provides a way to format strings using f-strings or the format() method. This allows you to insert variables into strings and control how they are displayed.

* **Print Formatting with Strings:**name = "John"  
  age = 30  
  print(f"My name is {name} and I am {age} years old.")

**name = "John"  
-** This line assigns a variable named name to a string, "John", which is an immutable sequence of characters, including letters, numbers, and special characters.  
 **age = 30  
-** This line assigns the integer value 30 to the variable age, which can be positive, negative, or zero in Python.

**print(f"My name is {name} and I am {age} years old.")  
-** This line uses the print() function to output a formatted string, defining it using an f-string and curly braces. The expressions inside the curly braces are evaluated at runtime, resulting in the string "My name is John and I am 30 years old."

**Next is the Lists**: Lists are ordered collections of items that can contain elements of different types. They are mutable, meaning you can change their content. Lists are used for storing and manipulating sequences of data.

* **Lists:**  
  my\_list = [1, 2, 3, 4, 5]  
  my\_list.append(6) # Adding an element to the end of the list  
  print(my\_list)

**my\_list = [1, 2, 3, 4, 5]**  
- This line assigns a mutable list to a variable named my\_list, which contains integers 1, 2, 3, 4, and 5.  
  
**my\_list.append(6) # Adding an element to the end of the list**  
- The append() method in Python adds an integer 6 to the end of a list, creating a new list with the values 1, 2, 3, 4, 5, and 6.

**print(my\_list)**  
- The print() function is used to output the contents of the list my\_list to the console, converting it to a string representation.

Question 2: What happens when you create a list named my\_list with the elements 1, 2, 3, 4, and 5, and then use the append() method to add the integer 6 to the end of my\_list?

Answer: The list my\_list will now contain the elements 1, 2, 3, 4, 5, and 6. The append() method modifies the list in place, adding the new element to the end of the list.

**Next is the Dictionaries**: Dictionaries are unordered collections of key-value pairs. They are mutable and can contain items of different types. Dictionaries are used for storing and accessing data in a key-value format.

* **Dictionaries:**  
  my\_dict = {"name": "John", "age": 30}  
  print(my\_dict["name"]) # Accessing a value by its key

**my\_dict = {"name": "John", "age": 30}**  
- This line assigns a unique dictionary to a variable named my\_dict, which contains two key-value pairs, "name" and "age", respectively.  
  
**print(my\_dict["name"]) # Accessing a value by its key**  
- The print() function in Python outputs the value of the key "name" in a dictionary, "John", to the console.

**Next is the Tuples**: Tuples are similar to lists but are immutable, meaning their content cannot be changed after they are created. Tuples are used for storing sequences of data that should not be modified.

* **Tuples:**  
  my\_tuple = (1, 2, 3)  
  print(my\_tuple[0]) # Accessing the first element

**my\_tuple = (1, 2, 3)**  
- This line assigns a tuple containing integers 1, 2, and 3 to a variable named my\_tuple, creating an immutable, ordered collection of items.  
  
**print(my\_tuple[0]) # Accessing the first element**  
- The print() function in Python outputs the first element of the tuple my\_tuple to the console, using the index number to access individual elements in the tuple.

**Next is the If Elif and Else**: Control flow statements, such as if, elif, and else, allow you to execute different blocks of code based on certain conditions. This is crucial for creating dynamic and interactive programs.

* **If Elif and Else:**  
  x = 20  
  if x > 20:  
   print("x is greater than 20")  
  elif x == 20:  
   print("x is equal to 20")  
  else:  
   print("x is less than 20")

**x = 20**  
- This line assigns a whole number, 20 to a variable named x, which can be positive, negative, or zero in Python.  
  
**if x > 20:  
 print("x is greater than 20")**  
- The statement checks if x is greater than 20, executes the code block, and prints the message "x is greater than 20" to the console.

**elif x == 20:  
 print("x is equal to 20")**  
- The elif statement checks if x equals 20, executing the code block and printing the message "x is equal to 20" to the console.

**else:  
 print("x is less than 20")**  
- The else block is executed if none of the previous conditions are true, like x equals 20. If not, the console displays "x is less than 20".

Question 1: What is the output when you use a variable named x with a value of 20 in a conditional statement that checks if x is greater than 20, equal to 20, or less than 20?

Answer: The output will be a message indicating that x is equal to 20. This is because the condition that checks if x is equal to 20 is true, and the corresponding message is printed.

**And the last is For Loops**: Loops, including for and while loops, enable you to execute a block of code multiple times. This is useful for iterating over sequences, performing repetitive tasks, and more.

* **For Loops:**  
  for i in range(5):  
   print(i)

**for i in range(5):**- This line initiates a for loop, iterating over a sequence of numbers from 0 to 4, using the range(5) function. The loop assigns the current number to variable i in each iteration.  
  
 **print(i)**  
- The loop uses the print() function to output the value of i to the console, with numbers 0, 1, 2, 3, and 4 being printed one per line.