Week 1 - Python Basics Assignment (Theory)

1. **What is Python, and why is it popular?**
    - Python is a high-level, general-purpose programming language known for its readability and versatility. It's popular due to its simplicity, a large community, extensive libraries, and diverse applications across web development, data science, and machine learning.

2. **What is an interpreter in Python?**
    - In Python, an interpreter is a program that reads and executes Python code line by line. It translates the human-readable Python code into machine-executable bytecode, which is then run by the computer. Essentially, the interpreter acts as a bridge between the programmer's code and the hardware.

3. **What are pre-defined keywords in Python?**
    - Keywords in Python are predefined reserved words with special meanings so the interpreter can understand them. They perform specific tasks in Python programming and are a part of its syntax.

4. **Can keywords be used as variable names?**
    - No, keywords cannot be used as variable names in most programming languages. Keywords are reserved words that have predefined meanings within the language and are used to define the structure and syntax of the code. Attempting to use a keyword as a variable name will result in a syntax error.

5. **What is mutability in Python?**
    - In Python, mutability refers to an object's ability to be changed after it's created. If an object is mutable, its contents can be modified without creating a new object. Immutable objects, on the other hand, cannot be changed after creation, and any modification will result in a new object being created.

6. **Why are lists mutable, but tuples are immutable?**
    - Tuples and lists are the same in every way except two: tuples use parentheses instead of square brackets, and the items in tuples cannot be modified (but the items in lists can be modified). We often call lists mutable (meaning they can be changed) and tuples immutable (meaning they cannot be changed).

7. **What is the difference between “==” and “is” operators in Python?**

== (Equality Operator)	is (Identity Operator)
1. Compares the values of two objects.	1. Checks if two variables point to the same object in memory
2. Returns True if the objects have the same value, even if they are different objects (i.e., have different memory addresses).	2. Returns True if both variables refer to the exact same object. Returns False if the variables refer to different objects, even if they have the same value.
Example: 1 "==" 1.0 returns True because 1 and 1.0 have the same value, but they are of different types (integer and float).	Example: x = [1, 2, 3] and y = x. Then x is y returns True because both x and y are pointing to the same list object in memory.

8. **What are logical operators in Python?**
   - In Python, logical operators are used to perform logical operations on boolean values, returning a boolean result (True or False). Python has three logical operators: and, or, and not. They are crucial for combining and manipulating conditions within conditional statements and expressions.
   - Here's a breakdown of each operator:
     1) and: Returns True if both operands are True, otherwise returns False.
     2) or: Returns True if at least one of the operands is True, otherwise returns False.
     3) not: Inverts the truth value of the operand. If the operand is True, not returns False, and vice versa.

9. **What is type casting in Python?**
   - Type casting in Python is the process of changing the data type of a variable. It's also known as type conversion.

10. What is the difference between implicit and explicit type casting?

Implicit Type Casting	Explicit Type Casting
The Python interpreter automatically performs type conversion on some operations without any user involvement. This is known as implicit type conversion.	When the Python Interpreter cannot implicitly convert between types, you can use explicit type casting.

Example:
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))
	Example:
	# num_one of type integer
	nun_one = 50
	# num_two of type string
	num_two_str = "150.75"
	# num_two explicitly converted from string to number
	num_two = float(num_two_str)
	print(type(num one))
	print(type(num_two))
	<class 'int'>
	<class 'float'>
	# add 'num_one' of type int to 'num_two' of type string
	result = num_one + num_two
	print(result)
	print(type(result))
	200.75
	<class 'float'>

11. What is the purpose of conditional statements in Python?

- Conditional statements in Python control the flow of program execution by allowing different code blocks to be executed based on whether specific conditions are true or false. They enable programs to make decisions, respond to different inputs, and handle various scenarios dynamically.

12. How does the elif statement work?

- The “elif” keyword in Python, stands for “else if”. It can be used in conditional statements to check for multiple conditions. For example, if the first condition is false, it moves on to the next “elif” statement to check if that condition is true.

13. What is the difference between for and while loops?

For Loop	While Loop
1. Iterate over a sequence of elements, such as an array, list, or string.	1. Repeat a block of code as long as a specified condition remains true.
2. The number of iterations is predetermined or can be easily calculated based on the size of the sequence.	2. The number of iterations is not known in advance and depends on the condition.

Example 1:
for i in range(10):
print(i)
#(This loop will print numbers from 0 to 9).
	Example 1:
	x = 0;
	while x < 10:
	print(x);
	x += 1
	
	#(This loop will print numbers from 0 to 9, but the number of iterations is determined by the condition x < 10).

Example 2: for char in "hello": print(char) # (This loop will print each character of the string "hello").	Example 2: while True: print("This loop will run indefinitely unless broken") # (This loop will continue indefinitely until a break statement is encountered).

14. Describe a scenario where a while loop is more suitable than a for loop?

- In programming, for loops are best used when you know the number of iterations ahead of time, whereas a while loop is best used when you don't know the number of iterations in advance. Both methods can help you iterate through your code.