Assignment-1

1. In the below elements which of them are values or an expression? e.g.:- values can be integer or string and expressions will be mathematical operators.

* :- Mathematical Operator

'hello' :- String(value)

-87.8 :- integer(value)

- :- Mathematical Operator

/ :- Mathematical Operator

+ :- Mathematical Operator

6 :- integer(value)

2. What is the difference between string and variable?

String	**Variable**
A string is a specific type of data representing text.	A variable is a general programming concept used to store and manage data, including strings.
A string is a data type used to represent text or a sequence of characters.	A variable is a container or a symbolic name that stores data, including strings.
It is enclosed within quotation marks (single or double) in many programming languages.	Variables are used to store and manipulate data within a program.
Strings can contain letters, numbers, symbols, and whitespace characters.	You can assign a string to a variable, and that variable will hold the value of the string.
Examples of strings: • "Hello, World!"	Examples of variables holding strings in Python: my_string = "Hello, World!"

• "12345" • "This is a string."	name = "John"

3. Describe three different data types.

Three common and fundamental data types:

1. Integer (int) :-
    - Integer data type represents whole numbers, either positive or negative, without a fractional or decimal component.
    - Examples: -3, 0, 42, 1001
    - In many programming languages, integers can have different sizes (e.g., int, long, short) based on the platform or language.
2. Floating-Point (float or double):-
    - Floating-point data types are used to represent numbers with fractional or decimal parts.
    - Examples: -3.14, 0.5, 3.14159
    - Floating-point numbers are approximations and may have limited precision due to the way they are stored in binary.
3. String (str):-
    - The string data type is used to represent text or sequences of characters.
    - Strings are typically enclosed in single (' '), double (" "), or triple (''' ' ' ' or """ """ for multi-line) quotation marks, depending on the programming language.
    - Examples: "Hello, World!", 'Python', "12345"
    - Strings can contain letters, numbers, symbols, and whitespace characters.

4. **What is an expression made up of? What do all expressions do?**
   An expression in computer programming is made up of one or more of the following elements:

1.Values:- Expressions often include specific values, which can be literals (e.g., numbers, strings) or variables (e.g., a named container holding data).

2.Operators:- Operators are symbols or keywords that perform operations on values. These operations can be mathematical (e.g., + for addition, - for subtraction), logical (e.g., && for logical AND, || for logical OR), or string-related (e.g., + for string concatenation).

3.Functions:- In some programming languages, expressions can include function calls. Functions are named blocks of code that perform specific tasks and return a value.

4.Parentheses:- Parentheses are used to control the order of operations and can be used to group parts of an expression.

An expression's primary purpose is to compute a value. What an expression does depends on its composition and the programming language in which it is used. Here are some common things expressions do:

1.Evaluate to a Value:- Expressions, when executed, result in a single value. For example, the expression `2 + 3` evaluates to `5`.

2.Perform Operations: Expressions can perform various operations like addition, subtraction, multiplication, division, comparison, and logical operations. For example, the expression `x * (y + 3)` performs multiplication and addition.

3. Combine and Transform Data:- Expressions can combine and transform data, such as concatenating strings or applying mathematical functions.

4.Assign Values:- In some cases, expressions can be used to assign values to variables. For example, `x = 10` assigns the value `10` to the variable `x`.

5. Control Flow:- In conditional expressions, such as in an `if` statement, expressions are used to determine whether certain blocks of code should be executed based on the result of the expression.

6.Return Values:- In function calls, expressions are used to return values computed within the function. For example, `result = add(3, 4)` returns the result of the `add` function.

In summary, expressions are fundamental building blocks in programming that combine values, operators, functions, and other elements to compute a result. They are used for calculations, data manipulation, decision-making, and many other tasks within a program.

5. **This assignment statements, like spam = 10. What is the difference between an expression and a statement?**

Expressions and statements are fundamental concepts in programming, and they serve different purposes:

Expression:-An expression is a combination of values, variables, operators, and function calls that can be evaluated to produce a single value.

Expressions are often used to perform calculations, manipulate data, and return results.

They can appear within statements and are often the part of a statement that produces a value.

Examples of expressions:-

2 + 3 (evaluates to 5)

x * 10 (evaluates to the result of multiplying x by 10)

len("hello") (evaluates to the length of the string "hello")

Statement:-A statement is a complete instruction that performs an action or task. It does not necessarily produce a value.

Statements are used to control the flow of a program, define functions, assign values to variables, loop, and make decisions.

Assignment statements, such as spam = 10, are a common type of statement. They assign a value to a variable.

Other types of statements include if statements, while loops, function definitions, and more.

In the case of an assignment statement like spam = 10, it's a statement because it doesn't produce a value (unlike an expression), but it has an effect – it assigns the value 10 to the variable spam. You can't use this statement in a mathematical operation or as part of a larger expression, but you can use the variable spam in expressions once it's been assigned a value.

In summary, expressions are used to produce values, while statements are used to perform actions or control the flow of a program. Assignment statements, like spam = 10, are statements used to assign values to variables.

6. After running the following code, what does the variable bacon contain?

bacon = 22

bacon + 1

The variable **bacon** still contains the value **22**.

The second line of code, **bacon + 1**, does not modify the value of **bacon**. It performs the addition operation **bacon + 1**, but the result is not assigned back to the **bacon** variable. If you want to update the **bacon** variable with the result of the addition, you should use an assignment statement like this:

bacon = bacon + 1

Now, the variable **bacon** would contain **23**.

7. What should the values of the following two terms be?

'spam' + 'spamspam'

'spam' * 3

The values of the two terms you provided are as follows:

1. `'spam' + 'spamspam'`:

   - This expression performs string concatenation, so it combines the two strings `'spam'` and `'spamspam'` to create a single string.

   - The result is `'spamspamspam'`.

2. `'spam' * 3`:

   - This expression performs string repetition. It takes the string `'spam'` and repeats it three times.

   - The result is `'spamspamspam'`.

In both cases, you end up with the same string `'spamspamspam'`.

8. Why is eggs a valid variable name while 100 is invalid?

In most programming languages, variable names have rules and conventions that determine their validity. Here's why "eggs" is a valid variable name while "100" is generally considered invalid:

1. **Variable Naming Rules:**

    - Variable names typically must start with a letter (a-z or A-Z) or an underscore (_) in many programming languages.

    - After the initial character, variable names can contain letters, numbers, and underscores.

- Digits (0-9) are generally not allowed as the first character of a variable name in most programming languages.

2. **Avoiding Confusion:**

   - Allowing variable names to start with digits could lead to confusion because it might be unclear whether you're referring to a variable or a numeric literal.
   - For example, if "100" were a valid variable name, it would be difficult to distinguish between the variable named "100" and the numeric value 100.

So, "eggs" is a valid variable name because it starts with a letter and doesn't cause ambiguity, whereas "100" is often invalid because it starts with a digit and might be confused with a numeric value. However, some programming languages have their own specific rules or conventions, so it's essential to check the rules of the language you're using, as they can vary.

9. What three functions can be used to get the integer, floating-point number, or string version of a value?

In Python, you can use the following functions to convert values between different data types:

1. To get the integer version of a value, you can use the `int()` function. This function takes a value and attempts to convert it into an integer. If the conversion is not possible, it raises a `ValueError` exception.

```python
  value = "42"

  integer_value = int(value)  # Converts the string "42" to an integer 42
```

2. To get the floating-point (decimal) version of a value, you can use the `float()` function. This function takes a value and attempts to convert it into a floating-point number. If the conversion is not possible, it raises a `ValueError` exception.

```python
  value = "3.14"

  float_value = float(value)  # Converts the string "3.14" to a float 3.14
```

3. To get the string version of a value (i.e., convert a value to a string), you can use the `str()` function. This function takes a value of any data type and converts it into its string representation.

```python
value = 42

string_value = str(value)  # Converts the integer 42 to the string "42"
```

These functions are helpful for data type conversions in Python, allowing you to work with different data types as needed in your programs.

10. Why does this expression cause an error? How can you fix it?

'I have eaten ' + 99 + ' burritos.'

The expression `'I have eaten ' + 99 + ' burritos.'` causes an error because you are attempting to concatenate a string with an integer without explicitly converting the integer to a string. In many programming languages, including Python, you cannot directly concatenate a string and an integer without performing the conversion.

To fix this, you can convert the integer to a string using the `str()` function before concatenating it with the other strings:

```python
'I have eaten ' + str(99) + ' burritos.'
```

Now, this expression will work correctly, and it will produce the following string:

"I have eaten 99 burritos."

By using `str(99)`, you've converted the integer `99` to a string, allowing it to be combined with the other strings without causing an error.