**Python Assignment 1**

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

\*

'hello'

-87.8

-

/

+

6

Ans. "\*" - is an expression representing multiplication operator.

"hello" - is a value representing a string.

"-87.8" - is a value representing a negative floating-point number.

"-" - is an expression representing subtraction operator.

"/" - is an expression representing division operator.

"+" - is an expression representing addition operator.

"6" - is a value representing a positive integer.

2. What is the difference between string and variable?

Ans. A string is a data type used to represent a sequence of characters. It can contain letters, numbers, symbols, and spaces. Strings are typically enclosed in quotation marks (either single or double quotes) to indicate that they are string literals. Example: “Hello World!"

Strings are commonly used to store and manipulate textual data, such as names, sentences, or any other sequence of characters.

A variable is a named storage location that holds a value. It is used to represent and store different kinds of data, including strings. Variables are used to store values that can change during the execution of a program. They provide a way to refer to and manipulate data without explicitly using the data itself. Variables are assigned a name, which is used to access and modify their values.

Example a variable that stores a string:

name = "John"

3. Describe three different data types.

Ans. There are three type of data types.

* Integer (int): The integer data type represents whole numbers without any fractional or decimal part. It can include both positive and negative numbers. For example, 5, -3, and 0 are integers.
* Floating (float): The floating data type represents numbers that can have a fractional or decimal part. It is used to store real numbers with a certain degree of decimal precision. For example, 3.14, -0.5, and 2.71828 are floating numbers.
* Boolean (bool): The Boolean data type represents a binary value that can be either true or false. It is used to express logical conditions or states. Booleans are particularly useful in decision-making and control structures. For example, True (1) and False (0).

4. What is an expression made up of? What do all expressions do?

Ans. An expression is a combination of values, variables, and operators that can be evaluated to produce a result. An expression can be as simple as a single value, such as 42, or as complex as a nested combination of operators and operands.

In general, all expressions in programming serve the purpose of computing a value. They can be used to perform mathematical operations, manipulate strings, evaluate conditions, or perform other types of computations depending on the programming language and the context in which they are used.

An expression can contain the following components:

* Values: These are the actual data or literals that are used in the expression. Examples include numbers, strings, and boolean values.
* Variables: These are named storage locations that hold values or references to values. Variables can be used in expressions to represent data or to store the result of a computation.
* Operators: These are symbols or keywords that perform operations on values or variables. Examples of operators include arithmetic operators like +, -, \*, and /, comparison operators like <, >, and ==, and logical operators like and, or, and not.
* Functions: These are named procedures or subroutines that perform specific tasks. Functions can be used in expressions to manipulate data or to perform more complex computations.

All expressions in programming are evaluated based on the order of precedence of the operators and the data types involved. The result of an expression can be a value of any data type, depending on the nature of the expression and the types of the operands involved.

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

Ans. In programming, an expression is a combination of values, variables, and operators that can be evaluated to produce a value. Examples of expressions include:

3 + 4

"hello" + "world"

x \* y

In contrast, a statement is a complete line of code that performs a specific action. Examples of statements include:

spam = 10

print("Hello, World!")

if x > y: print("x is greater than y")

In other words, an expression is a part of a statement that produces a value, whereas a statement is a complete line of code that performs an action or a sequence of actions.

In the example spam = 10, spam = 10 is a statement that assigns the value 10 to the variable spam. The expression here is just the value 10.

In summary, expressions produce values, while statements perform actions.

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

bacon = 22 bacon + 1

Ans. After running the following code, the variable bacon contains the value 22. The expression bacon + 1 evaluates to 23, but this value is not assigned to any variable, so it is not stored anywhere.

If you want to update the value of bacon to the result of the expression bacon + 1, you need to assign it back to bacon as follows:

bacon = 22

bacon = bacon + 1

print(bacon) # Output: 23

Here, we first assign the value 22 to the variable bacon. Then, we update the value of bacon to the result of the expression bacon + 1, which is 23. Finally, we print the value of bacon, which is now 23.

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

'spam' + 'spamspam' 'spam' \* 3

Ans. The value of 'spam' + 'spamspam' would be the string 'spamspamspam', which is the concatenation of the two strings 'spam' and 'spamspam'.

The value of spam \* 3 would depend on the value of the variable spam. If spam is a string, then the expression spam \* 3 would concatenate three copies of the string 'spam' together, resulting in the string 'spamspamspam'. If spam is a numeric variable, then the expression spam \* 3 would multiply the value of spam by 3.

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

Ans. In Python, variable names can consist of letters, numbers, and underscores, but they must follow certain rules. Specifically, variable names must start with a letter or an underscore, and they cannot start with a number. This is why "eggs" is a valid variable name - it starts with a letter.

On the other hand, "100" is not a valid variable name because it starts with a number, not a letter or an underscore. Variable names cannot start with a number in Python, which is why "100" is not a valid variable name. However, it is possible to include numbers within a variable name, as long as the name starts with a letter or an underscore. For example, "eggs100" would be a valid variable name in Python.

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

Ans. The following functions can be used to convert a value to a specific type:

* int(): This function can be used to convert a value to an integer. If the value is a string, the function tries to parse the string and convert it to an integer. If the value is a floating-point number, the function truncates it to an integer.
* float(): This function can be used to convert a value to a floating-point number. If the value is a string, the function tries to parse the string and convert it to a floating-point number. If the value is an integer, the function converts it to a floating-point number.
* str(): This function can be used to convert a value to a string. If the value is an integer or a floating-point number, the function converts it to a string representation of the number. If the value is already a string, the function returns the string as is.0

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

'I have eaten’ + 99 + 'burritos'

Ans. The expression causes an error because it is trying to concatenate a string and a number using the + operator. In Python, the + operator is used for both string concatenation and addition, depending on the types of the operands. When one operand is a string and the other is a number, Python does not know how to handle the operation and raises a Type Error exception.

To fix this error, we need to ensure that all operands are of the same type. One way to do this is to convert the number to a string using the str() function before concatenating it with the other strings. For example, we can rewrite the expression as follows:

'I have eaten ' + str(99) + ' burritos'

The expression, the number 99 is converted to a string using the str() function before being concatenated with the other strings. This ensures that all operands are strings, and the expression should execute without error.