**Assignment\_Basic\_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.**

1. **\***
2. **‘hello’**
3. **-87.8**
4. **–**
5. **/**
6. **+**
7. **6**

Ans:

**\*** - Mathematical Operator for Multiplication

**‘hello’** - String Value

**-87.8** – Floating Point Number Value

**–** - Mathematical Operator for Subtraction

**/** - Mathematical Operator for Division

**+** - Mathematical Operator for Addition

**6** - Integer Value

**2. What is the difference between string and variable?**

Ans:

* A string is a sequence of characters enclosed within quotation marks, used to represent text in programming. It can contain letters, numbers, symbols, or spaces. For example, "Hello, World!" is a string.
* A variable, on the other hand, is a symbolic name used to store and refer to data, which can be of any type, including a string.
* Variables allow you to manipulate and store different values throughout a program. For instance, in `name = "Alice"`, `name` is the variable holding the string "Alice."
* While a string is a specific data type, a variable is a container that can hold various types of data, such as strings, numbers, or objects.

**3. Describe three different data types.**

Ans: - Here are descriptions of three common data types:

1. **Integer (int):** An integer is a whole number, either positive or negative, without any decimal point. It is used to represent numeric values for operations like counting, arithmetic, or indexing. For example, ***`5`***, ***`-10`,*** and **`0`** are all integers.

2. **String (str):** A string is a sequence of characters used to represent text. Strings can include letters, numbers, punctuation, and special characters, and are usually enclosed in quotation marks. For instance, ***`"Hello, world!"`*** or ***`"12345"`*** are strings.

3. **Boolean (bool):** A Boolean represents one of two possible values: `True` or `False`. It is often used in conditions or comparisons to control the flow of a program, like in decision-making processes (`if` statements). For example, the expression ***`5 > 3`*** returns ***`True`,*** while ***`5 == 3`*** returns ***`False`***.

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

Ans:

* An expression is made up of ***values*** like ***numbers***, ***strings***, or ***variables*** and ***operators*** such as ***`+`, `-`,*** ***`\*`, `/`***.
* It can also include function calls, parentheses, and other constructs depending on the programming language.
* All expressions are evaluated to produce a result or a value. The main purpose of expressions is to compute or return a value after performing operations. For example, in the expression ***`3 + 5`,*** the values ***`3`***and ***`5`*** are combined using the ***`+`*** operator, and the expression evaluates to the result ***`8`.***

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

Ans:

* An expression is a combination of values, variables, and operators that, when evaluated, produces a result. For example, ***`3 + 5`*** or ***`x \* 2`*** are expressions because they perform a computation and return a value.
* A statement, on the other hand, is an instruction or action in a program that performs a task but does not necessarily produce a value.
* For example, ***`spam = 10***` is a statement because it assigns the value ***`10`*** to the variable ***`spam`,*** but it does not return anything. In summary, the main difference is that an expression evaluates to a value, while a statement performs an action such as assignment, without returning a value.

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

***bacon = 22***

***bacon + 1***

Ans: After running the code in Python:

***bacon = 22***

***bacon + 1***

The variable ***bacon*** will still contain ***22***. The expression ***bacon + 1*** evaluates to ***23***, but it doesn't change the value of ***bacon*** itself. To update ***bacon*** to ***23***, you would need to assign the result back to ***bacon***, like this:

***bacon = bacon + 1***

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

**‘spam’ + ‘spamspam’**

**‘spam’ \* 3**

Ans: For the given expressions:

1. ***'spam'*** + ***'spamspam'*** results in the concatenation of the two strings, so the value will be ***'spamspamspam'***.

2. ***'spam'*** \* 3 results in the repetition of the string ***'spam'*** three times, so the value will be ***'spamspamspam'***.

3. In both cases, the result is ***'spamspamspam'***.

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

Ans: In most programming languages, variable names (also known as identifiers) must follow specific rules. Here’s why ***‘eggs’*** is a valid variable name and ‘***100’*** is not:

1. **Starting Character:** Variable names generally must start with a letter ***(a-z, A-Z)*** or an underscore **(\_)**. ‘eggs’ starts with a letter, which makes it valid. However, ‘100’ starts with a number, which is not allowed.

2. **Subsequent Characters:** After the first character, variable names can include letters, digits (0-9), and sometimes underscores. ***‘eggs’*** contains only letters, which is acceptable. ***‘100’*** only contains digits, which is not allowed for variable names.

3. **Reserved Words:** While this isn’t the issue here, it's also worth noting that variable names cannot be reserved words or keywords in the language.

**Conclusion:** ‘eggs’ is valid because it follows the naming conventions of starting with a letter and using only allowed characters. ‘100’ is invalid because it starts with a digit.

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

Ans:To convert a value to an integer, floating-point number, or string in Python, you can use the following three functions:

1. **`int () `:** Converts a value to an integer.

- Example: ***`int (3.5) `*** will return ***`3`***.

2. **`float () `:** Converts a value to a floating-point number.

- Example: ***`float (3) `*** will return ***`3.0`***.

3. **`str () `:** Converts a value to a string.

- Example: ***`str (3.5) `*** will return ***`"3.5"`***.

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

*‘I have eaten’ + 99 + ‘burritos.’*

Ans:The error occurs because Python does not allow the concatenation of strings and integers directly.

In the expression:

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

You're trying to concatenate the string `'I have eaten'` with the integer `99` and the string `'burritos.'`. Python expects all parts of the concatenation to be of the same type, but `99` is an integer.

Fix:

Convert the integer ***`99`*** to a string using the ***`str () `*** function. Here's the corrected version:

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

This will output:

***'I have eaten 99 burritos.'***