**Question: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.**

**Answer:1.**

**'\*' -** mathematical operator (multiplication).

**'hello' -** this is a value, a string.

**-87.8 -** this is a value, a floating-point number.

**'-' -**This is a mathematical operator (subtraction).

**'/' -** This is a mathematical operator (division).

**'+' -** This is a mathematical operator (addition).

**'6' -** This is a value, specifically an integer.

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

**Answer:2.**

**String:** A string is a data type used to represent a text or characters. In python, string is typically enclosed within single quotes (') or double quotes ("). Example: "i" and 'neuron' are both strings.

**Variable:** A variable is a symbolic name that refers to a value stored in the computer's memory. Variables are used to store and manipulate data within a program. Unlike strings, variables can hold different types of data, i.e. numbers, strings, or objects.

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

**Answer:3.**

**(i) Boolean (bool):** Booleans represent truth values, True or False. They are often used in conditions and logical operations.

**(ii) Float (float):** Floats represent real numbers with a decimal point. They can be positive, negative, or zero. Example of floats: 1.2, -2.34, 5.0.

**(iii) Integer (int):** Integers are whole numbers without any decimal point. They can be positive, negative, or zero. Examples of integers: 1, -3.

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

**Answer:4.**

An expression is made up of one or more operands and operators that can be evaluated to produce a value. An operand can be a variable, a constant, a function call, or another expression. An operator specifies the operation to be performed on the operands.

In Python, expressions can include arithmetic operations, logical operations, comparisons, function calls, and more. Here are some examples:

* Arithmetic expression: **10\* (4 + 1)**
* String concatenation expression: **'i' + ' ' + 'neuron'**
* Function call expression: **len('iNeuron)**
* Comparison expression: **i > N**

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

**statement?**

**Answer:5.**

**Expression:**

* An expression is any valid combination of literals (constants), variables, operators, and function calls that evaluates to a single value.
* Expressions can be simple or complex, but they always produce a value.
* Examples of expressions include arithmetic expressions (2 + 3), string concatenation ('hello' + ' world'), function calls (len('spam')), and more.

**Statement**:

* A statement is a complete line of code that performs an action, represents a command, or declares a specific behaviour within a program.
* Statements do not necessarily produce a value, although some statements may include expressions.
* Examples of statements include assignment statements (**spam = 10**), conditional statements (**if**, **else**, **elif**), loop statements (**for**, **while**), function definitions, import statements, and more.

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

**Answer:6.** bacon = 22

bacon + 1

**code:**

**bacon = 22, bacon + 1**

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

The expression **bacon + 1** evaluates to **23**, but this value is not stored or assigned to any variable. If we want to update the value of **bacon** to be **23**, we need to assign the result of the expression **bacon + 1** back to the variable **bacon**, like this:

**bacon = bacon + 1**

OR **bacon += 1**

Then, **bacon** would contain the value **23**. But in the code provided, the value of **bacon** remains **22**.

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

**‘spam’ + ‘spamspam’**

**‘spam’\* 3**

1. **'spam' + 'spamspam'**:
   * This expression concatenates two strings **'spam'** and **'spamspam'**.
   * So, **'spam' + 'spamspam'** results in the string **'spamspamspam'**.
2. **'spam' \* 3**:
   * This expression repeats the string **'spam'** three times.
   * Multiplying a string by an integer **n** repeats the string **n** times.
   * So, **'spam' \* 3** results in the string **'spamspamspam'**.

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

**Answer:8.**

In Python, variable names must adhere to certain rules and conventions:

1. **Valid Characters:** Variable names can contain letters (both uppercase and lowercase), digits, and underscores (\_). However, they cannot start with a digit. So, ‘eggs’ is a valid variable name because it starts with a letter (e) and contains only letters.
2. **Cannot start with a digit:** Variable names cannot be start with a digit in Python. For example, 16\_eggs.

While eggs satisfy declaring variable rules, 100 is invalid as a variable name because it starts with a digit, violating the rule that variable names cannot start with a digit.

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

**Answer:9.**

1. **Integer Conversion: int()**
   * This function converts a value to an integer.
   * If the value is a floating-point number, it removes the decimal part.
   * Example: int(3.14) would return 3.
2. **Floating-Point Conversion: float()**
   * This function converts a value to a floating-point number.
   * If the value is an integer, it converts it to a floating-point number.
   * If the value is a string representing a floating-point number, it converts the string to a floating-point number.
   * Example: float('3.14') would return 3.14.
3. **String Conversion: str()**
   * This function converts a value to a string.
   * It returns the string representation of the value.
   * Example: str(42) would return '42'**.**

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

**‘I have eaten’ + 99 + ‘burritos.’**

**Answer:10.**

This expression throws error because Python doesn't allow concatenating strings with integers directly without explicitly converting the integer to a string first.

To fix this error, we need to convert the integer **99** to a string before concatenating it with the other strings.

We can do this using the **str()** function.

Code:

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

Output: **I have eaten 99 burritos.**

Top of Form