**Q1. 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.

Ans - \* , - , / , and + these are expression.

**Q2.What is the difference between string and variable?**

Ans - In Python, a string and a variable are different concepts.

A string is a data type in Python used to represent a sequence of characters. It is enclosed in either single quotes ('') or double quotes ("").

A variable, on the other hand, is a named container that can hold different types of values, including strings. It is a way to store and refer to data in memory. Variables can be assigned values and can be updated or changed as needed. Here's an example:

```python

my\_variable = 42

```

**Q3. Describe three different data types.**

Ans - Python provides several built-in data types. Here are three commonly used data types:

A. Integer (int): Integers represent whole numbers, both positive and negative, without any fractional part. For example: 0, 5, -10. Integer values can be used in mathematical operations like addition, subtraction, multiplication, and division.

B. String (str): Strings represent sequences of characters enclosed in either single quotes ('') or double quotes (""). They are used to store and manipulate text-based data. For example: "Hello, World!", 'Python'.

1. List: A list is an ordered collection of items enclosed in square brackets ([]). It can store elements of different data types, including integers, strings, and even other lists. For example: [1, 2, 3], ['apple', 'banana', 'cherry'].

**Q4. What is an expression made up of? What do all expressions do?**

Ans - In Python, an expression is a combination of values, variables, operators, and function calls that evaluates to a single value. Expressions are the building blocks of Python programs and are used to perform calculations, produce results, and make decisions.

An expression can consist of:

1. Literals: Literal values such as numbers (e.g., 42, 3.14) or strings (e.g., "Hello").

2. Variables: Named containers that hold values. Variables can be used in expressions to represent data.

3. Operators: Symbols that perform operations on one or more operands. Examples include arithmetic operators (+, -, \*, /), comparison operators (==, !=, <, >), logical operators (and, or, not), and more.

4. Function calls: Invocations of functions that perform specific tasks. Functions can be called with arguments, and their return values can be used within expressions.

When an expression is evaluated, Python interprets and executes the expression according to the precedence and associativity of the operators. The result of the expression is a single value of a specific data type.

Expressions in Python can be used for various purposes, including:

1. Performing mathematical calculations: Expressions can combine numeric values and arithmetic operators to perform addition, subtraction, multiplication, division, and more.

2. Assigning values: Expressions can be used to assign a value to a variable. For example, `x = 10` is an expression that assigns the value 10 to the variable `x`.

3. Making decisions: Expressions can be used in conditional statements (e.g., if-else) to determine the flow of a program based on certain conditions.

4. Creating complex data structures: Expressions can be used to create lists, dictionaries, and other data structures by combining literals, variables, and other expressions.

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

Ans - In Python, expressions and statements are two distinct concepts.

An expression is a combination of values, variables, operators, and function calls that evaluates to a single value. It produces a result. For example, `2 + 3` is an expression that evaluates to `5`. Expressions can be used within larger expressions, as part of assignments, or as arguments to functions.

A statement, on the other hand, is a complete unit of code that performs an action or task. It carries out a specific operation and does not produce a value. An assignment statement, such as `spam = 10`, is an example of a statement. It assigns the value `10` to the variable `spam`. Other types of statements in Python include control flow statements (if-else, for loop, while loop), function and class definitions, import statements, and more.

Here are some key differences between expressions and statements:

1. Result: Expressions produce a value as a result, while statements do not. Statements are executed for their side effects or to control the flow of a program.

2. Composition: Expressions can be composed together to form larger expressions, whereas statements are standalone and do not combine with other statements.

3. Usage: Expressions can be used within statements. For example, an expression can be used as an argument to a function call within a statement. However, statements cannot be used within expressions.

4. Evaluation: Expressions are evaluated to produce a value, whereas statements are executed or carried out to perform a specific action or task.

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

**bacon = 22**

**bacon + 1**

Ans - The variable bacon will still contain the value 22

**Q7. What should the values of the following two terms be**?

Ans - The values of the two terms will be as follows:

1. `'spam' + 'spamspam'`: This expression performs string concatenation. It combines the strings `'spam'` and `'spamspam'` together. The result will be `'spamspamspam'`.

2. `'spam' \* 3`: This expression performs string repetition. It repeats the string `'spam'` three times. The result will be `'spamspamspam'`.

**Q8. Why is eggs a valid variable name while 100 is invalid?**

In Python, variable names must adhere to certain rules and conventions. Here are the reasons why `eggs` is a valid variable name while `100` is invalid:

1. Starting Character: Python variable names must start with a letter (a-z, A-Z) or an underscore (\_). They cannot begin with a number. Since `eggs` starts with a letter 'e', it satisfies this requirement and is a valid variable name. On the other hand, `100` starts with a number '1', violating this rule and making it an invalid variable name.

2. Valid Characters: Apart from the starting character, variable names in Python can contain letters (a-z, A-Z), numbers (0-9), and underscores (\_). They cannot contain spaces or special characters. Since `eggs` only contains letters, it meets this criterion and is a valid variable name. However, `100` contains only numbers and no letters or underscores, which makes it invalid.

**Q9. What three functions can be used to get the integer, floating-point number, or string version of a value?**

Ans - In Python, you can use the following three functions to obtain the integer, floating-point number, or string version of a value:

1. int()
2. float()
3. Str()

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

**“I have eaten” + 99 + “ burritos”**

Ans - The given expression `'I have eaten ' + 99 + ' burritos.'` causes an error because it is trying to concatenate a string with an integer (`99`) directly. In Python, concatenation requires all the operands to be of the same data type.

To fix the error and perform the concatenation correctly, you need to ensure that all operands are strings. One way to do this is by converting the integer `99` into a string using the `str()` function.

By using `str(99)`, we convert the integer `99` into the string `'99'`. Now all the operands are strings, and the concatenation can be performed without an error.