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

2. What is the difference between string and variable?

3. Describe three different data types.

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

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

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

bacon = 22

bacon + 1

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

'spam' + 'spamspam'

'spam' \* 3

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

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

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

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

Answer:

1. the given values are :

‘hello’ (string)

-87.8 (float)

6 (integer)

The given operators expressions are:

/

\*

+

-

2. the difference between the string and variables are:

String: A string is a data type that represents a sequence of characters. It is used to store and manipulate textual data. In most programming languages, strings are enclosed in single quotes ('') or double quotes (""). For example, 'hello' and "world" are strings.

Variable: A variable is a named container or memory location that holds a value. It is used to store and refer to different types of data, including strings, numbers, or other data types. Variables allow you to assign a value to a specific name or identifier, making it easier to manipulate and reuse the data throughout your program. You can think of a variable as a label that represents a value or an object. For example, in Python, you can define a variable like this: `name = 'John'`. Here, `name` is the variable, and `'John'` is the string value assigned to it.

3. Certainly! Here are descriptions of three different data types commonly found in programming:

Integer (int):

- The integer data type represents whole numbers without decimal points.

- Integers can be positive, negative, or zero.

- They are used to perform mathematical operations like addition, subtraction, multiplication, and division.

- Examples of integers: -5, 0, 42.

String (str):

- The string data type represents a sequence of characters.

- Strings are typically used to store and manipulate textual data.

- They can contain letters, numbers, symbols, and spaces.

- Strings are enclosed in single quotes ('') or double quotes ("").

- Examples of strings: 'Hello', "World", '123'.

Boolean (bool):

- The boolean data type represents logical values, either true or false.

- Booleans are used in decision-making and control flow statements.

- They are the result of logical operations and comparisons.

- Examples of booleans: True, False.

4. An expression is made up of operators, operands, and sometimes function calls or method invocations. It combines these elements to produce a new value.

Operators: Operators are symbols or keywords that perform specific operations on one or more operands. Examples of operators include arithmetic operators (+, -, \*, /), comparison operators (==, <, >), logical operators (and, or, not), assignment operators (=), and more. Operators define the behavior of the expression and determine how the operands are manipulated.

Operands: Operands are the values or variables that are used in an expression. They can be literals (explicit values) or variables (symbols representing values stored in memory). For example, in the expression `3 + 5`, the operands are the numbers 3 and 5.

Function calls or method invocations: Expressions can also include function calls or method invocations. These are operations that involve calling a function or method and passing arguments to it. The function or method returns a value that becomes part of the expression. For example, in the expression `len("hello")`, the function `len()` is called with the argument "hello", and the returned value is used as part of the expression.

5. The difference between an expression and a statement lies in their purpose and behavior within a programming language:

Expression: An expression is a combination of values, variables, operators, and function calls that can be evaluated to produce a resulting value. Expressions are usually used to perform calculations, retrieve values, or determine conditions. They can be as simple as a single value or as complex as a combination of nested operations. Examples of expressions: `3 + 5`, `x \* y`, `len("hello")`.

Statement: A statement is a complete instruction or command that performs a specific action or control flow within a program. Statements are used to define the sequence of operations, control the execution flow, and modify the program's state. Statements can include expressions as part of their execution or assign values to variables. Examples of statements: assignment statements (`spam = 10`), control flow statements (if-else, loops), function declarations, class definitions.

6. After running the given code:

bacon = 22

bacon + 1

The variable `bacon` will still contain the value 22.

The expression `bacon + 1` evaluates to 23, but since it is not assigned to any variable or used in any way, the result is not stored or saved. Therefore, the value of `bacon` remains unchanged at 22. If you want to update the value of `bacon` to the result of the expression, you need to assign it explicitly, like this:

bacon = bacon + 1

After executing the above line, the variable `bacon` will contain the value 23.

7. The values of the following two terms would be:

`'spam' + 'spamspam'`: This expression performs string concatenation. It combines the string `'spam'` with the string `'spamspam'`. The result is the string `'spamspamspam'`. So, the value of this expression is `'**spamspamspam**'`.

`'spam' \* 3`: This expression performs string repetition. It repeats the string `'spam'` three times. The result is the string `'spamspamspam'`. So, the value of this expression is also `'**spamspamspam**'`.

8. In most programming languages, including Python, variable names need to follow certain rules and conventions. The difference between the variable names "eggs" and "100" lies in these rules:

**Valid variable names:** Variable names can start with a letter (a-z or A-Z) or an underscore (\_). After the first character, variable names can contain letters, digits (0-9), or underscores. Variable names are case-sensitive, meaning that uppercase and lowercase letters are considered different. Examples of valid variable names: eggs, \_count, myVariable, spam123.

**Invalid variable names:** Variable names cannot start with a digit. They should not contain special characters like spaces, punctuation marks, or operators. Variable names should not be reserved keywords in the programming language. Examples of invalid variable names: 100, 3count, my-variable, spam$.

9. In Python, you can use the following three built-in functions to convert a value to different data types:

`int()`: This function converts a value to an integer. It can be used to convert a string or a floating-point number to an integer. If the value cannot be converted to an integer, it will raise a `ValueError`.

Example :

int\_value = int("10") # Converts the string "10" to an integer

`float()`: This function converts a value to a floating-point number. It can be used to convert a string or an integer to a float. If the value cannot be converted to a float, it will raise a `ValueError`.

Example:

float\_value = float("3.14") # Converts the string "3.14" to a float

`str()`: This function converts a value to a string. It can be used to convert an integer, floating-point number, or any other data type to a string.

For Example:

str\_value = str(42) # Converts the integer 42 to a string

10. The expression `'I have eaten ' + 99 + ' burritos.'` causes an error because it attempts to concatenate a string with an integer directly, which is not supported in Python. The `+` operator for string concatenation expects both operands to be strings. To fix the error, you can convert the integer value `99` to a string before concatenating it with the other strings. Here's an updated version of the expression:

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