**1.hello’**: This is a **string** value. Strings are sequences of characters, typically used for text.

**-87.8**: This is a **numeric value** (specifically a **floating-point number**). It represents a negative decimal value.

**-**: This is a **mathematical operator** (the subtraction symbol).

**/**: This is also a **mathematical operator** (the division symbol).

**+**: This is a **mathematical operator** (the addition symbol).

**6**: This is an **integer value**.

**2. String**:

* + A **string** is a data type used to represent text or character sequences.
  + It is enclosed in either single quotes (') or double quotes (").
  + Examples: 'hello', "world", "Python is fun".

**Variable**:

* + A **variable** is a named container that holds a value.
  + It can store different types of data, including strings, numbers, or other objects.
  + Variables allow us to assign a name to a value, making it easier to manipulate and reference.
  + Example: If we define name = 'Alice', then name is a variable containing the string value 'Alice'.

1. **Three Different Data Types**:
   1. **Integer (int)**: Represents whole numbers (positive, negative, or zero). Example: 42.
   2. **Float (float)**: Represents decimal numbers. Example: 3.14.
   3. **String (str)**: Represents text. Example: 'Hello, World!'.
2. **Expressions**:
   1. An **expression** is a combination of values, variables, and operators that can be evaluated to produce a result.
   2. Expressions can include arithmetic operations (addition, subtraction, multiplication, division), comparisons (greater than, less than), and function calls.
   3. All expressions have a value. For example, 2 + 3 is an expression with a value of 5.
3. **Difference Between Expression and Statement**:
   1. An **expression** produces a value. It can be part of a larger expression or assigned to a variable. Examples: 2 + 3, len('hello').
   2. A **statement** performs an action or changes the program’s state. It doesn’t necessarily produce a value. Examples: assignment (spam = 10), loops, conditionals.

After running the code bacon = 22; bacon + 1, the variable bacon remains unchanged. The expression bacon + 1 evaluates to 23, but it’s not assigned back to bacon. To update bacon, you’d need to reassign the result: bacon = bacon + 1.

6.After running the code, the variable **bacon** still contains the original value of **22**. The expression **bacon + 1** evaluates to **23**, but it is not assigned back to the variable. To update the value of **bacon**, you would need to reassign it explicitly:

bacon = 22

bacon = bacon + 1 # Now bacon contains 23

7.'spam' + 'spamspam':

* + The + operator concatenates the two strings.
  + Result: **‘spamspamspam’**

'spam' \* 3:

* + The \* operator repeats the string three times.
  + Result: **‘spamspamspam’**

Both expressions yield the same value: **‘spamspamspam’**.

8.eggs: This is a valid variable name because it starts with a letter and contains only letters.

100: This is invalid because it starts with a digit. Variable names cannot begin with a number.

**9.Conversion Functions**:

* + To convert values between different data types, you can use the following functions:
    - int(): Converts a value to an integer (whole number).
    - float(): Converts a value to a floating-point number (decimal).
    - str(): Converts a value to a string (text).

**10.Expression Error and Fix**:

* + The expression 'I have eaten' + 99 + 'burritos' causes an error because you are trying to concatenate a string ('I have eaten') with an integer (99).
  + To fix it, you need to ensure that all parts of the expression are strings. You can convert the integer 99 to a string using str(99):
    - 'I have eaten' + str(99) + 'burritos' will correctly concatenate the strings and produce the result: **‘I have eaten99burritos’**.