**Assignment 1**

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.

\* is a mathematical operator, specifically the multiplication operator.

'hello' is a string value.

-87.8 is a floating-point value.

- is a mathematical operator, specifically the subtraction operator.

/ is a mathematical operator, specifically the division operator.

+ is a mathematical operator, specifically the addition operator.

6 is an integer value.

So, \*, -, /, and + are expressions, while 'hello', -87.8, and 6 are values.

2. What is the difference between string and variable?

In Python, a string is a sequence of characters enclosed within quotes (either single quotes or double quotes) and it represents textual data. For example, "Hello World" is a string.

On the other hand, a variable is a name that refers to a value or an object. In Python, you can assign a value to a variable using the assignment operator (=). For example, x = 42 assigns the integer value 42 to the variable x.

The main difference between a string and a variable is that a string is a specific type of data (textual), while a variable is a named reference to any type of data (including strings).

Here's an example to illustrate the difference:

# This is a string

message = "Hello World"

# This is a variable

x = 42

In this example, message is a variable that contains a string, and x is a variable that contains an integer. Note that the same variable can be assigned different types of data over time. For example:

# This is a variable that contains an integer

x = 42

# Now x contains a string

x = "Hello World"

In this case, x is first assigned an integer value, but later it is assigned a string value.

3. Describe three different data types.

In Python, there are many built-in data types, but here are three common ones:

Integer (int): This is a data type used to represent whole numbers (both positive and negative) without any fractional component. For example, 42 and -17 are both integers.

String (str): This is a data type used to represent textual data. It consists of a sequence of characters enclosed in quotes, either single quotes (') or double quotes ("). For example, "Hello World" and 'Python is fun' are both strings.

Boolean (bool): This is a data type used to represent true/false values. It has only two possible values: True and False. Booleans are often used in conditional statements and loops to control program flow. For example, the expression 5 > 3 evaluates to True, while the expression 5 < 3 evaluates to False.

There are many other data types in Python, including floating-point numbers (float), lists, tuples, dictionaries, and more. Each data type has its own properties and behaviors, and understanding them is an important part of programming in Python.

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

In Python, an expression is a combination of values, variables, and operators that can be evaluated to a single value. For example, 2 + 3 is an expression that evaluates to the integer value 5, and "Hello " + "World" is an expression that evaluates to the string value "Hello World".

Expressions can be made up of a variety of components, including:

Literals: These are fixed values such as numbers or strings, like 42 or "Hello".

Variables: These are names that represent values, like x or my\_variable.

Operators: These are symbols that perform operations on values, such as + for addition or - for subtraction.

Function calls: These are calls to built-in or user-defined functions that perform a specific task.

All expressions in Python have the purpose of computing a value. The value of an expression can be assigned to a variable, used as part of a larger expression, or printed to the screen. Expressions can be used in a variety of contexts in Python, such as mathematical calculations, string manipulation, and logical comparisons.

In general, expressions are a fundamental concept in programming and are used to perform a wide range of tasks, from simple calculations to complex algorithms.

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

In Python, an expression is a combination of values, variables, and operators that can be evaluated to a single value. For example, 2 + 3 is an expression that evaluates to the integer value 5, and "Hello " + "World" is an expression that evaluates to the string value "Hello World". Expressions can be used as part of a larger statement or on their own.

On the other hand, a statement is a complete line of code that performs a specific action. Assignment statements like spam = 10 are examples of statements, because they assign a value to a variable. Other examples of statements in Python include conditional statements (if statements), loops (for and while loops), and function definitions.

The main difference between expressions and statements is that expressions are evaluated to a single value, while statements perform some sort of action. Expressions are often used as part of larger statements, such as in the case of assigning a value to a variable using an expression.

Here's an example to illustrate the difference:

# This is an expression

x = 2 + 3

# This is a statement

if x == 5:

print("x is equal to 5")

In this example, 2 + 3 is an expression that is evaluated and the result is assigned to the variable x. The if statement is a complete statement that tests the value of x and performs an action (printing a message) based on the result of the test.

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

bacon = 22

bacon + 1

bacon = 22

bacon = bacon + 1

print(bacon) # Output: 23

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

'spam' + spamspam

'spam' \* 3

'spam' + 'spamspam': This expression will result in the string 'spamspamspam'. The + operator can be used with strings to concatenate them, i.e., join them end-to-end to create a longer string.

'spam' \* 3: This expression will result in the string 'spamspamspam'. The \* operator can be used with strings to repeat them a specified number of times. In this case, 'spam' \* 3 means "repeat the string 'spam' three times", resulting in the string 'spamspamspam'.

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

In Python, variable names must follow certain rules. Specifically, they must:

Start with a letter (either uppercase or lowercase) or an underscore (\_)

Be composed of letters, underscores, and/or digits

Therefore, eggs is a valid variable name because it starts with a lowercase letter and is composed only of letters. On the other hand, 100 is not a valid variable name because it starts with a digit, which is not allowed according to the rules.

Variable names starting with a digit can be confusing because they may be interpreted as numeric literals rather than variable names. To avoid this confusion, variable names in Python must start with a letter or an underscore.

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

In Python, the int(), float(), and str() functions can be used to convert a value to an integer, floating-point number, or string, respectively.

The int() function can be used to convert a value to an integer. For example, int(4.5) would return 4.

The float() function can be used to convert a value to a floating-point number. For example, float(3) would return 3.0.

The str() function can be used to convert a value to a string. For example, str(42) would return the string '42'.

It's important to note that not all types can be converted to every other type. For example, trying to convert the string 'hello' to an integer using int('hello') would result in a ValueError because the string cannot be interpreted as an integer. Similarly, trying to convert the string '3.14' to an integer using int('3.14') would also result in a ValueError.

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

'I have eaten' + 99 + 'burritos'

The expression 'I have eaten' + 99 + 'burritos' causes a TypeError because you are trying to concatenate a string ('I have eaten') with an integer (99) using the + operator. Python does not allow you to concatenate strings and non-string values directly.

To fix this error, you can convert the integer 99 to a string using the str() function before concatenating it with the other strings. Here's an example:

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

In this example, str(99) converts the integer 99 to the string '99', which can then be concatenated with the other strings using the + operator. The resulting string would be 'I have eaten 99 burritos'.