**PYTHON 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.**

**\***

**&#39;hello&#39;**

**-87.8**

**-**

**/**

**+**

**6**

**Ans.**

1. '\*' is an operator (specifically, the multiplication operator).
2. 'hello' is a string value.
3. -87.8 is a floating-point number value.
4. '-' is an operator (specifically, the subtraction operator).
5. '/' is an operator (specifically, the division operator).
6. '+' is an operator (specifically, the addition operator).
7. 6 is an integer value.

So, to summarize

:

**Values: 'hello', -87.8, 6**

**Expressions (Operators): '\*', '-', '/', '+'**

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

A string and a variable are two distinct concepts in programming, and they serve different purposes:

**String:**

* A string is a data type used to represent text or a sequence of characters.
* It can contain letters, numbers, symbols, spaces, and other characters.
* Strings are typically enclosed in quotation marks (single or double), e.g., "Hello, World!" or '12345'.
* Strings are used to store and manipulate textual data in a program.
* Examples of operations on strings include concatenation (combining strings), searching for substrings, and extracting portions of text.

**Variable:**

* A variable is a symbolic name or identifier that is used to store and manipulate data in a program.
* Variables can hold various types of data, including strings, numbers, booleans, and more.
* Unlike strings, variables are not limited to storing text; they can store different types of data depending on how they are declared and used.
* Variables are used to store values that may change during the execution of a program, and they allow programmers to work with and manipulate data in a flexible way.

1. **Describe three different data types**.

**Ans**. Certainly! Data types are a fundamental concept in programming and are used to define the kind of data that can be stored and manipulated in a program. Here are three different data types:

* ***Integer (int):***
  + An integer data type represents whole numbers without any fractional or decimal parts.
  + It can be either positive or negative, or zero.
  + Example in Python: **age = 25**
* ***Floating-Point (float):***
  + A floating-point data type represents numbers with decimal points or fractional parts.
  + It is used to store real numbers, including both rational and irrational numbers.
  + Example in JavaScript: **price = 19.99**
* ***String (str):***
  + A string data type is used to represent sequences of characters, such as text or words.
  + Strings are typically enclosed in quotation marks (single or double) in most programming languages.
  + Example in Java: **name = "John"**

These are just a few examples of data types, and there are many more, including boolean (for representing true or false values), arrays (for storing collections of data), and custom data types created by programmers to suit specific needs. The choice of data type depends on the kind of data you need to work with and the operations you want to perform on that data.

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

**Ans.** An expression in the context of computer programming and mathematics is a combination of symbols, operators, and variables that represents a value. Expressions are used to perform calculations, manipulate data, and produce results in programming. Here's what expressions are made up of:

1. **Literals:** These are fixed values that do not change. For example, in the expression **5 + 3**, both **5** and **3** are literals.
2. **Variables:** Variables are symbols that represent values that can vary or change during program execution. For example, in the expression **x + 3**, **x** is a variable, and its value can vary.
3. **Operators:** Operators are symbols or keywords that perform operations on operands. Common operators include **+** (addition), **-** (subtraction), **\*** (multiplication), **/** (division), and many more.
4. **Functions:** In some programming languages, you can use functions as part of expressions. Functions take inputs (arguments) and return a value based on those inputs. For example, in the expression **sin(x)**, **sin** is a function.
5. **Parentheses:** Parentheses are used to control the order of operations in an expression. They specify which parts of the expression should be evaluated first. For example, in the expression **2 \* (3 + 4)**, the parentheses ensure that the addition inside them is performed before the multiplication.

The primary purpose of expressions is to compute a value. When you evaluate an expression, it produces a single value. Expressions can be used in various ways in programming, such as assigning values to variables, making decisions in conditional statements, or providing arguments to functions. Additionally, expressions can be part of larger statements or formulas, enabling you to create complex calculations and logic within a program.

In summary, expressions are fundamental building blocks in programming that combine literals, variables, operators, and sometimes functions to produce a value or result, which can then be used for further computation or decision-making in a program.

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

**Ans.** In programming, expressions and statements serve different purposes, and they have distinct characteristics:

1. Expression:
   * An expression is a combination of values (constants or variables), operators, and function calls that can be evaluated to produce a single value.
   * Expressions can be as simple as a single constant (e.g., 5) or as complex as a mathematical formula or function call (e.g., x + y, math.sqrt(x)).
   * They usually produce a result, which can be assigned to a variable, used in a computation, or returned as a value from a function.
   * Expressions can often be used within statements to compute values or make decisions.
2. Statement:
   * A statement is a complete line of code that performs a specific action or task in a program. Statements are used to control the flow of a program, define variables, and carry out operations.
   * Unlike expressions, statements do not necessarily produce a value as a result.
   * Examples of statements include assignment statements, conditional statements (e.g., if-else), loops (e.g., for, while), function declarations, and more.
   * Statements are executed sequentially, one after the other, in a program.
3. **After running the following code, what does the variable bacon contain?**

**bacon = 22**

**bacon + 1**

**Ans.** After running the provided code, the variable **bacon** still contains the value **22**.

Here's a breakdown of what happens in the code:

1. **bacon = 22**: This line assigns the value **22** to the variable **bacon**, so **bacon** now holds the value **22**.
2. **bacon + 1**: This line adds **1** to the value of **bacon** (which is **22**), but the result of this addition (**23**) is not stored in any variable or used in any way. So, while the expression **bacon + 1** does equal **23**, it doesn't change the value of the **bacon** variable, which still remains **22** after this line is executed.
3. **What should the values of the following two terms be?**

**‘spam’ + ‘spamspam’**

**‘spam’ \*3**

**Ans.** In Python, when you use the **+** operator with two strings, it concatenates (combines) them together. When you use the **\*** operator with a string and a number, it repeats the string multiple times based on the number.

Let's evaluate the two expressions:

1. **'spam' + 'spamspam'**: When you use the **+** operator between these two strings, it will concatenate them together, resulting in the string **'spamspamspam'**.
2. **'spam' \* 3**: When you use the **\*** operator with the string **'spam'** and the number **3**, it repeats the string **'spam'** three times, resulting in the string **'spamspamspam'**.

So, both expressions yield the same result: **'spamspamspam'**.

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

**Ans.** In most programming languages, variable names have rules and conventions that dictate what is considered a valid variable name. These rules are in place to ensure that variable names are clear, unambiguous, and do not conflict with the syntax or keywords of the language. Here are some common rules and conventions for variable names:

1. Variable names must start with a letter (a-z or A-Z) or an underscore (\_) in many programming languages. Some languages may allow certain non-letter characters as well.
2. Variable names can include letters, numbers, and underscores after the first character. For example, "egg123" is a valid variable name.
3. Variable names are case-sensitive in many programming languages, meaning "Egg" and "egg" would be considered two different variables.

Now, let's apply these rules to your examples:

* "eggs" is a valid variable name because it starts with a letter ("e") and is followed by letters only.
* "100" is considered invalid as a variable name in most programming languages because it starts with a number, violating rule #1. Variable names typically cannot start with a digit.

However, some programming languages have their own rules and conventions for variable names, so it's essential to consult the documentation or guidelines specific to the language you are using. In some languages, "100" might be a valid variable name or part of a valid variable name if it follows their specific rules.

For example, in Python:

* Variable names must start with a letter (a-z or A-Z) or an underscore (\_).
* They can contain letters, numbers, and underscores.

So, "eggs" is a valid variable name in Python, while "100" is not because it starts with a number. However, "eggs100" would be a valid variable name in Python.

1. **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 convert values to integers, floating-point numbers, or strings:

1. To get the integer version of a value, use the **int()** function. For example:

pythonCopy code

value = 42.5 integer\_value = int(value)

1. To get the floating-point number version of a value, use the **float()** function. This function can also be used to convert integers to floats. For example:

pythonCopy code

value = 42 float\_value = float(value)

1. To get the string version of a value, use the **str()** function. For example:

pythonCopy code

value = 42.5 string\_value = str(value)

These functions are built-in to Python and are commonly used for type conversions. They allow you to work with different data types and perform operations accordingly.

1. **Why does this expression cause an error? How can you fix it?I have eaten ; + 99 + &; burritos.**

**Ans.** The expression you provided:

pythonCopy code

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

causes an error because you are trying to concatenate a string ('I have eaten ') with an integer (99) directly using the **+** operator. In Python, you can only concatenate strings with other strings using the **+** operator.

To fix this error, you need to convert the integer (99) into a string before concatenating it with the other strings. You can do this by using the **str()** function to explicitly convert the integer to a string:

pythonCopy code

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

Now, this expression will work without any errors, and it will result in the following string:

"I have eaten 99 burritos."