Python Basic Assignment – 1

1. **In the below elements which of them are values or an expression**

Given Elements are

* \*
* ‘hello’
* -87.8
* -
* /
* +
* 6

The values are :

* ‘hello’ ( A String Value)
* -87.8 ( A Floating Point Value )
* 6 ( An Integer Value )

The Expressions are :

* \* ( Multiplication Operator )
* / ( Division Operator)
* + ( Addition Operator )
* - ( Subtraction Operator)

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

A String can be defined as a datatype which stores a text or collection of characters. It can include letters, numbers, symbols, and spaces. They are enclosed in quotation marks( ‘’ , “ “ ).

Example: “hello” , “123456”, “hello@”

A Variable can be defined as a container used to store data values. It can store datatypes such as Strings, Integers, Characters and more. It allows you to assign values to them.

Example: name = “John”, age = 25.

Precisely, A string is a specific datatype used to represent text while variable is a named container that can hold various datatypes.

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

**Integer:**

* It represents Whole numbers without any fractional and decimal part. It includes both positive and negative values including zero.
* It is having specific range: -2,147,483,648 to 2,147,483,647.
* Operations performed on integers include addition (+), subtraction (-), multiplication (\*), division (/), and modulus (%).
* Examples: -23, 0, 142

**String:**

* The string data type represents a sequence of characters, such as text, words, or symbols. It can include letters, numbers, whitespace, and special characters.
* Strings are usually enclosed in single quotes ('') or double quotes (""), allows the interpreter or compiler to recognize them as a series of characters.
* Operations performed on strings include concatenation (combining two strings together), substring extraction, length calculation, and comparison.
* Examples: “hello” , “123456”, “hello@”

**Boolean:**

* The Boolean data type represents a binary value that can be either true or false.
* Booleans are primarily used for logical operations and conditional branching.
* They are particularly useful in control flow statements, such as if statements and while loops, where decisions are made based on the truth or falsity of a condition.
* Booleans can be derived from comparisons, logical operators, or the evaluation of conditions.
* Examples: True, False

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

An expression is a combination of one or more operands and operators that, when evaluated, produces a resulting value.

**Operands:** The values or variables that participate in an expression.

Examples of operands:

* Values: These can be literals like numbers (e.g., 5, 3.14) or strings (e.g., "hello", 'world').
* Variables: These are named storage locations that hold values. Variables can be assigned values, and those values can be used as operands in expressions.

**Operators:** The Symbols or functions that perform specific operations on the operands.

Examples of operators:

* Arithmetic operators: These perform mathematical calculations. Examples include addition (+), subtraction (-), multiplication (\*), division (/), and modulus (%).
* Comparison operators: These compare two operands and return a boolean value (true or false). Examples include equal to (==), not equal to (!=), greater than (>), less than (<), etc.
* Logical operators: These are used to evaluate conditions and perform logical operations. Examples include logical AND (&&), logical OR (||), and logical NOT (!).
* Assignment operator: The assignment operator (=) is used to assign a value to a variable.
* Function call operators: These are used to invoke functions and pass arguments.

When an expression is encountered in a program, it is evaluated to produce a resulting value. The evaluation involves applying the specified operators to the given operands, following the rules of operator precedence and associativity.

For example, in the expression 5 + 3 \* 2, the multiplication operator (\*) has higher precedence than the addition operator (+), so the multiplication is performed first, resulting in 6. Then, the addition is performed, resulting in 11.

The resulting value of an expression can be used in various ways, such as assigning it to a variable, passing it as an argument to a function, or using it in conditional statements. Expressions allow programmers to perform calculations, combine values, make comparisons, and evaluate conditions, enabling the creation of complex logic and algorithms in a program.

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

The main difference between an expression and a statement lies in their purpose and behaviour within a programming language:

An expression is a combination of operands and operators that, when evaluated, produces a resulting value. They are used to perform calculations, make comparisons, concatenate strings, and more.

A statement is a complete instruction or command that performs a specific action or sequence of actions. They are used to control the flow of execution, define behaviour, or modify program state.

In this case of an assignment statement like "spam = 10", it falls into the category of a statement. It assigns the value 10 to the variable "spam". The statement doesn't produce a value itself but rather changes the state of the program by assigning a value to a variable. In contrast, an expression like "5 + 3" evaluates to a resulting value of 8.

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

**bacon = 22**

**bacon + 1**

* bacon = 22

The above line assigns the value 22 to the variable bacon. So, after executing this line, the variable bacon will have the value 22.

* bacon + 1

This represents **“**bacon = bacon + 1”. After executing this line, the value in the bacon which is having value 22 is incremented by 1 value. This would update the value of bacon to 23.

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

**'spam' + 'spamspam'**

**'spam' \* 3**

* 'spam' + 'spamspam’ - This expression concatenates the strings 'spam' and 'spamspam'.

**Result:** 'spamspamspam'.

* 'spam' \* 3: This expression multiplies the string 'spam' by 3. It repeats the string three times.

**Result:** 'spamspamspam'.

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

In Python, variable names follow specific rules and restrictions which determine whether they are valid or not.

**Variable name Rules:**

* Variable name must starts with a letter or an underscore.
* A variable name contains letters, digits and an underscore.
* These are case-sensitive
* It can’t be a python keyword(reserved word).

In present context, according to the variable rules and restrictions, ‘eggs’ is a valid variable name because its starts with a lowercase letter and it contains letters only whereas ‘100’ is not a valid variable name because it violates the first rule i.e., a variable name must be starts with letter or an underscore. As it starts with a digit, it can’t be a valid variable.

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

In python, we can use the following three functions to get the integer, floating-point number or a string version of a value. The following functions are commonly used for type conversion allows you to switch between integer, floating point and string representation of data.

* **int() :** This function is used to convert a value to an integer as it takes a number or string argument and returns the corresponding integer representation

num = “42”

integer\_val = int(num)

print(integer\_val) #output: 42

* **float()** : This function is used to convert a value to a floating-point number as it takes a numeric or string argument and returns the corresponding floating-point representation.

num = “3.14”

float\_val = int(num)

print(float\_val) #output: 3.14

* **str() :** This function is used to convert a value to its string representation as it takes any value as an argument and returns its string representation.

num = 42

str\_val = str(num)

print(str\_val)

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

The following expression “**I have eaten ' + 99 + ' burritos”** causes an error because we are attempting to concatenate a string with an integer by directly using the ‘+’ operator. Generally, ‘+’ operator is used to add numeric values or concatenate strings.

**Fixing this error:**

1. We can explicitly convert the integer ‘99’ to a string before concatenating with the other string parts. We can use str() to convert the integer to string.

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

1. Otherwise, we can use string formatting to fix this error.

**f'I have eaten {99} burritos.'**