Python Assignment 01

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.

Ans.

* : As this is an asterisk which is a mathematical operator for multiplication, it's an **expression**.

'hello' : As the word hello lies in between single quotes, it's a string. So hence it's a **value.**

-87.8: The question doesn't mention about floats specifically but -87.8 does fall under a **value**

- : As this is a negative or minus sign which is a mathematical operator for subtraction, it's an **expression**

/ : As this is a forward slash which is mathematical operator for division, it's an **expression**

+ : As this is a positive or plus sign which is a mathematical operator for addition, it's an **expression**

6 : As this is an integer, it's a **value**.

2. What is the difference between string and variable?

Ans.

String: It's a data type that holds or stores sequences of character data which could be a noun or just random letters or numbers. A string can be declared or created by typing characters in either single quotes **('….')** or double quotes **("….")**.

Variable: It can be defined as a symbol or a location which can be used to store data in a program or memory. They can be declared by typing the variable name and giving it a value using the equal sign **(=)**.

Now for the comparison between these two we can think of variable as an empty box that you could fill with some data or value, which in this case you can use a string.

Following is an example of a variable being declared with its data being a string value:

V = 'Peace'

Here the empty box/variable is 'V' and the string value assigned to it is Peace.

3. Describe three different data types.

Ans.

Some of the basic data types are integers, floats and strings.

1. **Integer (int):** An integer is a whole number without a fractional component. It can be a positive or negative number or zero. For example: -3, 0, 42. In Python, integers are represented by the **int** data type.
2. **String (str):** A string is a sequence of characters enclosed in single quotes (' ') or double quotes (" "). Strings are used to represent text or any sequence of characters. For example: "Hello, World!", 'Python'. In Python, strings are represented by the **str** data type.
3. **Float (float):** A float is a number with a decimal point. It represents both whole and fractional numbers. For example: 3.14, -0.5, 1.0e-5. In Python, floating-point numbers are represented by the **float** data type.

Here's an example demonstrating the usage of these data types:

```
# Integer

age = 25

print(age)  # Output: 25


# String

name = "John"

print(name)  # Output: John


# Float

pi = 3.14

print(pi)  # Output: 3.14
```

Also there are other data types like lists, tuples and dictionaries which are discussed below.

Lists: They are sequence data type that ordered collection of elements and are similar to arrays compared to other languages. Lists can contain heterogeneous elements, meaning it can store elements irrespective of the data type as illustrated in the below example:

Elem = ["Apples", 1, 1.32, ["Apples", 1 , 1.32]]

Lists are mutable, meaning the elements stored in it can changed, like add, delete or modify them. To declare a list you need to place the elements between square brackets [….], with commas between the elements.

Tuples: They are also sequence data type like lists that are ordered collection of elements. Tuples can contain heterogeneous elements, meaning being to store different data types as illustrated in the below example:

Elem = (1, 2, 4, 3, "Oranges", (234, 342))

The feature that makes tuples different from lists is that tuples are immutable, meaning the stored elements cannot be altered or changed after they're declared. And to declare them you need to place the elements between or inside parenthesis (….), with commas between the elements.

Dictionaries: Unlike lists or tuples these are mapping data type and are collection of unordered elements and each element stored in a dictionary has a key and a value, making it a key-value pair. Similar to lists and tuples even dictionaries can contain heterogeneous elements.

They are mutable, meaning the key-value pairs can be changed i.e. we can modify, add or delete them from a dictionary. It can be declared by placing the key-value pairs inside curly braces {…..} and commas between the pairs. Also the key and value are separated by a colon (:) as illustrated in the below example

A_dict = { "name" : "James", "score" : 88 }

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

Ans.

Generally an expression can contain two or more operands and one or more operators as shown below, where 5, 3, x and y are the operands and +, * are the mathematical operators.

5 + 3

x * y

But it is also a combination of values, variables, operators and function calls that evaluates to a single value. It can be as simple as a single variable or value, or it can be a more complex combination of these elements.

It's important to note that expressions are different from statements in Python. Expressions produce a value when evaluated, while statements perform actions or control the flow of the program.

Now to answer the question of what expressions do, in Python they are used to compute or evaluate values. Based on the operations performed and the input values used an expression produces results when executed and this result can be stored in a variable, passed as an argument to a function.

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

Ans.

As discussed earlier in the above question, an expression is basically a combination of values, variables, operators and function calls that compute to a single value. Expression produces a value when computed as illustrated below:

```
X = 10 + 15                # Evaluates to 25
```

On the other hand, a statement is a task or an instruction that performs an action or control the flow of a program. Statements can contain expressions, but they aren't evaluated to produce a value. Instead they execute specific tasks or operations. Example

```
x = 5 + 3     # Assignment statement: Assigns the value 8 to the variable x

if x > 10:    # Control flow statement: Executes a code conditionally

        print("x is greater than 10")
```

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

```
bacon = 22

bacon + 1
```

Ans.

After running the code, I got the output as 23 but to answer the question of what value does the variable bacon contain, it would be 22 because the second expression does not reassign the value in bacon, to do so it should have been

bacon = bacon + 1

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

'spam' + 'spamspam'

'spam' * 3

Ans.

Both of the given expressions evaluate to a string as the output which is 'spamspamspam'.

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

Ans.

100 is an invalid variable name or an identifier because an identifier must never start with a digit or a number which is like a rule to give a variable name. That said it doesn't mean that you cannot use numbers in identifiers, you just cannot begin with them but you can include them in between or by starting the variable name with an underscore like

_100 or num100 which are completely valid to use as identifiers.

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

Ans.

As dicussed earlier in the question number 3, the different data types, the functions you can use to get the integer, float or string version of a value are discussed briefly below:

1. **'int()':** We can use this int() function to convert a value to an integer as shown below.
   ```
   number = "42"
   integer_value = int(number)
   print(integer_value)  # Output: 42
   ```

2. **‘float()’:** We can use this float() function to convert a value to a floating point number or a number that has a fractional value like 3.14 as shown below.
   number = "3.14"
   float_value = float(number)
   print(float_value) # Output: 3.14

3. **‘str()’:** This str() function can be used to convert a value to a string or collection of characters as illustrated below.
   number = 42
   string_value = str(number)
   print(string_value) # Output: "42"

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

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

Ans.

The above expression is causing an error because you are trying to add or concatenate (link) a string with an integer 99. In Python it’s only possible when the data you are trying to link are all of string type.

Now to fix the error there are 2 simple ways

First is to enclose the integer 99 in single or double quotes, which makes python see it as a string rather than an integer.

'i have eaten ' + "99" + " burritos"

And second is to use the str() function and write the 99 between the parenthesis.

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