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.

ANS-

* a.\* = expression
* ‘Hello’ = expression
* -87.8 = values
* - = expression
* / = expression
* + = expression
* 6 = values

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

ANS- explain string and variable given below :-

1. **String**

A **string i**s a sequence of one or more characters (letters, numbers, symbols) that can be either a constant or a variable. Made up of Unicode, strings are immutable sequences, meaning they are unchanging.

For eg- String concatenation =

Concatenation means joining strings together end-to-end to create a new string. To concatenate strings, we use the + operator. Keep in mind that when we work with numbers, + will be an operator for addition, but when used with strings it is a joining operator.

Let’s combine the strings "Sammy" and "Shark" together with concatenation through a print() statement:

print("Sammy" + "Shark")

output= Sammyshark

1. **Variable**

A variable is something that holds a value that may change. In simplest terms, a variable is just a box that you can put stuff in. You can use variables to store all kinds of stuff, but for now, we are just going to look at storing numbers in variables.

For eg-

Va riable is my\_var= 3 or “hello’’

3. **Describe three different data types.**

ANS-

In computer programming, data types specify the type of data that can be stored inside a variable. For example,

num = 24

1. **Python Numeric Data type**

In Python, numeric data type is used to hold numeric values.

Integers, floating-point numbers and complex numbers fall under python number category. They are defined as int, float and complex classes in Python.

* int - holds signed integers of non-limited length.
* float - holds floating decimal points and it's accurate up to 15 decimal places.
* complex - holds complex numbers.

For eg- num1 = 5

print(num1, 'is of type', type(num1))

num2 = 2.0

print(num2, 'is of type', type(num2))

num3 = 1+2j

print(num3, 'is of type', type(num3))

output = 5 is of type <class 'int'>

2.0 is of type <class 'float'>

(1+2j) is of type <class 'complex' >

## Python List Data Type

## List is an ordered collection of similar or different types of items separated by commas and enclosed within brackets [ ].

## For eg-

## languages = ["Swift", "Java", "Python"]

## # access element at index 0

## print(languages[0]) # Swift

## # access element at index 2

## print(languages[2]) # Python

## Python String Data Type

## String is a sequence of characters represented by either single or double quotes.

## For eg-

## name = 'Python'

## print(name)

## message = 'Python for beginners'

## print(message)

## output = Python

## Python for beginners

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

## ANS-

## An expression is any sequence of literals (strings, numbers, lists, sets, tuples), objects, function calls, identifiers combined with a variety of operators which results in a value - regardless of whether of the resulting value is an object, or a number, or even a None. So examples of expressions are :

## ‘Hello World’ - this is a string literal

## 23.6 - this is a floating point literal

## {1,2,3,4} - a set literal

## my\_func(1,2,3) - a function call (even if the function returns None)

## 2 + the\_number \* math.factorial(7) - a combination of literals, operators, identifiers and function calls

## Expressions aren’t things like for loops, while loops, with statements, try/except, if/else - they are statements.

## This definition doesn’t just a apply in Python - it is normal across most languages: expressions are things that are intended to result in a value.

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

## Ans-

## An expression evaluates to a single value. A statement does not.

## In above question, spam = 10 is the value .

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

## bacon = 22

## bacon + 1

## Ans-

## After running the following code, the variable bacon contains the value 22. The expression “Bacon+1” does not update the value of bacon , it just evaluates to 23. If you want to update the value of bacon to 23,you need to assign the result of the expression to the variable bacon, like this: “bacon =bacon + 1“ or “bacon += 1”.

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

**‘spam’ + ‘spamspam’**

**‘spam’ \* 3**

Ans-

The value of the first term 'spam' + 'spamspam' should be 'spamspamspam', since the + operator concatenates the two strings.

The value of the second term 'spam' \* 3 should be 'spamspamspam', since the \* operator repeats the string 'spam' three times.

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

Ans-

In Python, variable names must follow certain rules. The first character of a variable name must be a letter or an underscore (\_). Subsequent characters can be letters, underscores, or digits. Therefore, 'eggs' is a valid variable name because it starts with a letter, while '100' is an invalid variable name because it starts with a digit. Variable names cannot start with digits in Python.

**9. 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 functions to get the integer, floating-point number, or string version of a value:

1. int() - This function can be used to get the integer version of a value. If the value is a floating-point number, it will be truncated towards zero.

2. float() - This function can be used to get the floating-point number version of a value. If the value is an integer, it will be converted to a floating-point number.

3. str() - This function can be used to get the string version of a value. It will convert the value to a string.

For example:

x = 10

y = 3.14

z = "hello"

# get integer version of y

a = int(y)

# get floating-point number version of x

b = float(x)

# get string version of x

c = str(x)

In the above code, `a` will be `3`, `b` will be `10.0`, and `c` will be `"10"`.

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

**‘I have eaten’ + 99 + ‘burritos’.**

Ans-

This expression causes an error because you are trying to concatenate a string (`' I have eaten'`) with an integer (`99`) without converting the integer to a string first. In Python, you cannot concatenate a string and an integer directly.

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

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

This will result in the string `' I have eaten 99 burritos'`.