# Q 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.

**' hello '**



**\***



**-87.8 - / + 6**

# let's consider all the given input as operator opr\_ = ['\*',

'hello',

-87.8,

'-',

'/',

'+', 6]

for opr in opr\_:

if(opr == '\*' or opr == '-' or opr == '/' or opr == '+'): print(f"{opr} is an operator\n")

else:

print(f"{opr} is an Expression\n")

 \* is an operator

hello is an Expression

-87.8 is an Expression

- is an operator

/ is an operator

+ is an operator

6 is an Expression

**Observation :** Hence, There are a total of 4 operators nad 3 expressions, They are:

**Operators :** \* , - , / ,



+

***Expressions :*** 'hello' , 87.8 ,



6

# Q2. What is the difference between string and variable?

**Ans : Variable :** It is used to store information or data. It is like a **box** or a **bucket**, like **for example :** if you want to store water for washing cloths where you will store amount of water inside a bucket so, you can **use it after a particular time** in similar way, if we want to **store a data or information for reusability purpose**. Variable is an empty space where we filled data for te using it after a particular time.

**String :**It is a kind of data type or data or information itself which is a **sequence of characters**

and can be easily stored inside a variable, it is usually enclosed by single or double quotes

" "

' '

# Declaring a variable

var = 'Some data has to be stored here'

# variable declared for useability purpose

print(f"This print() use this variable value like here we call it, var = {var}.")

# We can store String Data inside a Variable

print(f"This print() use to show the stored value data type: {type(var)}, str means it is

This print() use this variable value like here we call it, var = Some data has to be This print() use to show the stored value data type: <class 'str'>, str means it is

def error(err):

return "\033[38;2;{};{};{}m{} \033[38;2;255;255;255m".format(255, 0, 0, err)

# Q3. Describe three different Data Types ?

**Ans:** There are three fundamental Data types in python are int, float, complex.

**int data type:** We can use **int** data type to represent zero, positive or negative whole numbers (integral values) that are without any fractional part or unlimed precision

**float data type:** We can use float data type to represent floating point values (decimal values), he maximum size depends on your system. The float beyond its maximum size referred as "inf",

"Inf", "INFINITY", or "infinity".

**complex data type:** Complex number is represented by complex class. It is specified as (real part) + (imaginary part) j. Both real and imaginary component are there but Real part is not mandatory to be present and for imaginary part otherthan 'j' as suffix wouldnot work at all.

# int data example

int\_frac = 1/2 # fraction value int\_minus1 = -1

int\_1 = 1

int\_0 = 0 if True:

print(f"Interger of -1 value and type: {int\_minus1}, {type(int\_minus1)} \

\nInteger of 1 : {int\_1} and, {type(int\_1)}\

\nInteger of zeroth value : {int\_0}, {type(int\_0)}\

\nFor value = {int\_frac} (0r) 1/2, it is False because the data type is {type(int\_frac)}

Interger of -1 value and type: -1, <class 'int'> Integer of 1 : 1 and, <class 'int'>

Integer of zeroth value : 0, <class 'int'>

For value = 0.5 (0r) 1/2, it is False because the data type is <class 'float'>

#Declaration of complex number cn = 1 + 2j

print(f'{cn} is a {type(cn)} of data.')

#Withon any coefficient cn1 = 3j

print(f'{cn1} is a {type(cn1)} of data.')

(1+2j) is a <class 'complex'> of data. 3j is a <class 'complex'> of data.

cn2 = 3 + 4k # We can't use f

print(f'{cn2} is a {type(cn2)} of data.')

# OBSERVATION:

As we can see, complex number can be possible without real number or only with the imaginary part but it must have 'j' as suffix other than that gives SyntaxErrror.

# Q4. What is an expression made up of? What do all expressions do?

**Ans:** An expression is a combination of numeric data, values, variables, operators, and, calls to functions.

Expressions need to be evaluated, if we see in python for printing an expression, the python interpreter has to evaluate the expression and display the resulted output.

8\*5+40-80 # This ia an expression and after evaluation the python interpreter shows 0 as r

0

# Q5. This assignment statements, like spam = 10. What is the difference between an expression and a statement?

**Ans** An expression is a combination of numeric data, values, variables, operators, and, calls to functions.

It is a mathematical statement.

Expressions need to be evaluated, if we see in python for printing an expression, the python interpreter has to evaluate the expression and display the resulted output.

**Eg :** 8\*5+40-80, this is an expression which will return "0", as resultan output.

Statement is a unit of code that has an effect (or) side-effect, like creating variable or displaying a value.

When we type a statement , the python interpreter executes it without returning any resultant output or values.

**Eg** : Variable Declaration and Assignment, If, elif, else etc are statement because they **do not return** any value

# Example

8\*5+40-80 #Is an Expression

course\_name = 'iNeuron: Full Stack Data Science Job Guaranteed Program' # Is a Statement print("Hi, Sudhansu Sir :)") # Is Both an Expression and Statement

Hi, Sudhansu Sir :)

# Q6 .After running the following code, what does the variable bacon contain?

bacon = 22

bacon + 1

**Ans:** The variable

bacon

is set to 22 . The expression

doesnot re-assign the value in

bacon (that would the case if the expression is like instead of bacon + 1 )

bacon + 1

bacon = bacon + 1

# Case - 1:

bacon = 22

bacon + 1 # value is not stored print(bacon)

22

# Case - 2

bacon = 22

bacon += 1 # value is stored print(bacon)

23

# Q7. What should the values of the following two terms be?

'spam' + 'spamspam'

'spam' \* 3

**Ans:** Both expressions evaluate to the string where as the first expression

**'spam' + 'spamspam'**

follows and the second expression follows

String Concatenation

String Multiplication

print('spam' + 'spamspam') # Concatenation print('spam' \* 3) # multiplication

spamspamspam spamspamspam

# Why is eggs a valid variable name while 100 is invalid?

**Ans:** As per python, variable names cannot begin with a number. The python rules for naming a variable are :-

* 1. Variable name must start with a letter (or) Underscore (\_) character.
  2. Variable name cannot start with a number.
  3. Variable name can only contain alpha-numeric characters(A-z, 0-9) and underscore (\_).
  4. Variable names are case-sensitive.

Eg: both considered different variable by the python interpreter.

ineuron and INEURON

* 1. Reserved keywords cannot be used as variable naming.

# Example

egg = 'ovum' # Valid variable initialization print(egg)

ovum

100 = 'Indian currency' # Invalid variable initialization print(100)

\_SomeVar = 1 # Starting with Underscore(\_) --> Valid print(\_SomeVar)

1

@SomeVar=1 # Starting with Special Character --> Invalid print(@SomeVar)

var1 = 'ineuron' var2 = 'INEURON'

print(var1 == var2) # Case Sensitive print(var1 == var2.lower())

print(var1.upper() == var2)

False True True

global = 1 # Reserved Keyword as Variable name --> Invalid print(global)

# Q9. What three functions can be used to get the integer,floating-point number,or string version of a value?

**Ans:** The int() , float() ,and functions will evaluate to the integer,floating-point

str()

number,string version of the value passed to them.

# Examples:

print(f'int(10.0) -> {int(10.0)}') # int() function converts given input to int

print(f'float(10) -> {float(10)}') # float() function converts given input to float print(f'str(10) -> {str(10)}') # str() function converts given input to string

int(10.0) -> 10

float(10) -> 10.0

str(10) -> 10

# Q10. Why does this expression cause an error? how can you fix it?

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

**Ans:** This cause of Error is 99, because 99 is not a string.

99 must be typecasted to a string to fix this error. the correct way is:

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

Output: 'I have eaten 99 burritos.'

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

I have eaten 99 burritos

0s completed at 7:53 PM