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.

\* :- an expression

'hello' :- value

-87.8 :- value

- :- an expression

/ :- an expression

* :-- an expression

6 :- value

2. What is the difference between string and variable?

Answer:- Difference between string and variable is as follows:

1. Variables are references that you create to refer to other values in your program later and String is a datatype for a sequence of characters**.**
2. "This is a really lengthy string that no one wants to have to re-type ever, since it's so tedious and unpleasant to type out, it'd be fantastic to have a quick reference for this big, long string." When writing a program, you may assign the value to a variable (e.g., my\_string = "This is an really lengthy string...") and access it later by typing my\_string.

3. Describe three different data types.

Answer:-

1. Numeric

In Python, numeric data type represent the data which has numeric value. Numeric value can be integer, floating number or even complex numbers. These values are defined as int, float and complex class in Python.

* **Integers**– This value is represented by int class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.
* **Float**– This value is represented by float class. It is a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
* **Complex Numbers** – Complex number is represented by complex class. It is specified as (real part) + (imaginary part)j. For example – 2+3j

type() function is used to determine the type of data type.

**Program : -**

# Python program to

# demonstrate numeric datatype

a = 9

print("Type of a: ", type(a))

b = 9.0

print("\nType of b: ", type(b))

c = 2 + 3j

print("\nType of c: ", type(c))

**Output:**

Type of a: <class 'int'>

Type of b: <class 'float'>

Type of c: <class 'complex'>

1. Boolean

Data type with one of the two built-in values, True or False. Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false). But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.

- True and False with capital ‘T’ and ‘F’ are valid booleans otherwise python will throw an error.

**Program:**

# Python program to

# demonstrate boolean type

print(type(True))

print(type(False))

print(type(true))

**Output:**

<class 'bool'>

<class 'bool'>

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

[<ipython-input-1-41b100f19035>](https://localhost:8080/) in <module>

**2** print(type(False))

**3**

----> 4 print(type(true))

NameError: name 'true' is not defined

3)Mapping Types

Dictionary:-

[Dictionary](https://www.geeksforgeeks.org/python-dictionary/) in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair. Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a ‘comma’.

-Creating Dictionary

In Python, a Dictionary can be created by placing a sequence of elements within curly {} braces, separated by ‘comma’. Values in a dictionary can be of any datatype and can be duplicated, whereas keys can’t be repeated and must be immutable. Dictionary can also be created by the built-in function dict(). An empty dictionary can be created by just placing it to curly braces{}.

**Note**– Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.

**Program:-**

# Creating an empty Dictionary

Dict = {}

print("Empty Dictionary: ")

print(Dict)

# Creating a Dictionary

# with Integer Keys

Dict = {1: 'Sagarnanda', 2: 'Panditrao', 3: 'Wagholikar'}

print("\nDictionary with the use of Integer Keys: ")

print(Dict)

# Creating a Dictionary

# with Mixed keys

Dict = {'City': 'Pune', 1: [1, 2, 3, 4]}

print("\nDictionary with the use of Mixed Keys: ")

print(Dict)

# Creating a Dictionary

# with dict() method

Dict = dict({1: 'God', 2: 'is', 3:'Eternal'})

print("\nDictionary with the use of dict(): ")

print(Dict)

# Creating a Dictionary

# with each item as a Pair

Dict = dict([(1, 'Mumbai'), (2, 'Indians')])

print("\nDictionary with each item as a pair: ")

print(Dict)

**Output:**

Empty Dictionary:

{}

Dictionary with the use of Integer Keys:

{1: 'Sagarnanda', 2: 'Panditrao', 3: 'Wagholikar'}

Dictionary with the use of Mixed Keys:

{'City': 'Pune', 1: [1, 2, 3, 4]}

Dictionary with the use of dict():

{1: 'God', 2: 'is', 3: 'Eternal'}

Dictionary with each item as a pair:

{1: 'Mumbai', 2: 'Indians'}

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

Answer:-

* An expression is made up of operators and operands.
* The expression produces some values or result after being interpreted by Python interpreter.
* It can be also considered as a logical line of code that is evaluated to obtain some result.
* Example of expression:
* **Program :-**

a = 5

a = a + 10

print(a)

**Output:** 15

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

Answer:-

* A statement is an instruction that the Python interpreter can execute.
* spam = 10
* A statement that assigns a value to a name (variable). To the left of the assignment operator, =, is a name of the variable i.e. spam. To the right of the assignment token is a value 10 assigned to the spam variable. That is if we print a spam variable it will show 10 as an output.
* The differene between statement and expression in Python is as follows:

| **Statement** | **Expression** |
| --- | --- |
| A statement in Python is used for creating variables or for displaying values. | The expression in Python produces some value or result after being interpreted by the Python interpreter. |
| A statement in Python is not evaluated for some results. | An expression in Python is evaluated for some results. |
| The execution of a statement changes the state of the variable. | The expression evaluation does not result in any state change. |
| A statement can be an expression. | An expression is not a statement. |
| Program : x = 3. Output : 3 | Program: x = 3 + 6. Output : 9 |

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

bacon = 22

bacon + 1

Answer: -

After running the following code, the variable bacon contains 22 as bacon is not assigned. bacon + 1 is not reassigned to bacon.

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

'spam' + 'spamspam'

'spam' \* 3

Answer:-

1st term – 'spamspamspam'

2nd term - 'spamspamspam'

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

Answer:-

Eggs is a valid variable because it starts with letter e and doesn’t contain any special characters. While 100 is invalid because it’s start with the digit.

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

Answer:-

The three functions that can be used to get the integer, floating-point number, or string version of a value are int(), float() and string() respectively.

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

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

Answer:-

This expression causes an error because we cannot concatenate integer(99) to strings with + operator, we can only concatenate strings to other strings.

To fix this, we can convert integer 99 to string ‘99’ or we can use str(99).