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.

\* **: Expression**

'hello' **: Values**

-87.8 **: Values**

- **: Expression**

/ **: Expression**

* **: Expression**

6 **: Values**

2. What is the difference between string and variable?

|  |  |  |
| --- | --- | --- |
| **Sl. No** | **String** | **Variable** |
| 1 | Strings are arrays of bytes representing Unicode characters | A variable is a name given to a memory location. It is the basic unit of storage in a program |
| 2 | It’s a value representing text | It is a name that can refer to any value |
| 3 | Strings in Python can be created using single quotes or double quotes or even triple quotes.  Ex: var= “Welcome”  Here “Welcome” is a string | A variable is created the moment we first assign a value to it  Ex: var = ”Welcome”  Here var is variable |
| 4 | Strings are immutable; hence elements of a String cannot be changed once it has been assigned. Only new strings can be reassigned to the same name | We can re-declare the python variable once we have declared the variable already.  Var=12  Var=35  We can use same name for different type.  Ex:  Var=”hi”  Var=14.4 |
|  | Updation or deletion of characters from a String is not allowed. This will cause an error because item assignment or item deletion from a String is not supported. Although deletion of the entire String is possible with the use of a built-in del keyword  To access a range of characters in the String, the method of slicing is used | A variable name must start with a letter or the underscore character.  A variable name cannot start with a number.  A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ ).  Variable names are case-sensitive (name, Name and NAME are three different variables).  The reserved words(keywords) cannot be used naming the variable. |

3. Describe three different data types.

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

Following are the standard or built-in data type of Python:

* Numeric
  + Integer
  + Float
  + Complex number
* Sequence Type
  + String
  + List
  + tuple
* Boolean
* Set
* Dictionary

**Numeric:**

In Python, numeric data type represents 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

**Sequence Type:**

In Python, sequence is the ordered collection of similar or different data types. Sequences allows to store multiple values in an organized and efficient fashion. There are several sequence types in Python –

* String:
* In Python, Strings are arrays of bytes representing Unicode characters. A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by str class.
* Strings in Python can be created using single quotes or double quotes or even triple quotes
* List:
  + Lists are just like the arrays, declared in other languages which is a ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.
  + Lists in Python can be created by just placing the sequence inside the square brackets [].
* Tuple:
  + Tuple is also an ordered collection of Python objects. The only difference between tuple and list is that tuples are immutable i.e., tuples cannot be modified after it is created. It is represented by tuple class.
  + In Python, tuples are created by placing a sequence of values separated by ‘comma’ with or without the use of parentheses for grouping of the data sequence. Tuples can contain any number of elements and of any datatype (like strings, integers, list, etc.).

**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.

**Set:**

* Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements. The order of elements in a set is undefined though it may consist of various elements.
* Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, separated by ‘comma’. Type of elements in a set need not be the same, various mixed-up data type values can also be passed to the set.

**Dictionary:**

* 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’.
* 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 {}.
* Dictionary keys are case sensitive, same name but different cases of Key will be treated distinctly.

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

* An *expression* is a combination of **operators** and **operands** that is interpreted to produce some other value.
* In any programming language, an expression is evaluated as per the precedence of its operators. So that if there is more than one operator in an expression, their precedence decides which operation will be performed first. We have many different types of expressions in Python.
  + **Constant Expressions:** These are the expressions that have constant values only.

x = 15 + 1.3

* + **Arithmetic Expressions:** An arithmetic expression is a combination of numeric values, operators, and sometimes parenthesis. The result of this type of expression is also a numeric value. The operators used in these expressions are arithmetic operators like addition, subtraction, etc.

x = 40

y = 12

add = x + y

sub = x - y

pro = x \* y

div = x / y

* + **Integral Expressions:** These are the kind of expressions that produce only integer results after all computations and type conversions.

a = 13

b = 12.0

c = a + int(b)

* + **Floating Expressions:** These are the kind of expressions which produce floating point numbers as result after all computations and type conversions.

a = 13

b = 5

c = a / b

* + **Relational Expressions:** In these types of expressions, arithmetic expressions are written on both sides of relational operator (> , < , >= , <=). Those arithmetic expressions are evaluated first, and then compared as per relational operator and produce a Boolean output in the end. These expressions are also called Boolean expressions.

a = 21

b = 13

c = 40

d = 37

p = (a + b) >= (c - d)

* + **Logical Expressions:** These are kinds of expressions that result in either *True* or *False.* It basically specifies one or more conditions. For example, (10 == 9) is a condition if 10 is equal to 9. As we know it is not correct, so it will return False. Studying logical expressions, we also come across some logical operators which can be seen in logical expressions most often.

P = (10 == 9)

Q = (7 > 5)

R = P and Q

S = P or Q

T = not P

* + **Bitwise Expressions:** These are the kind of expressions in which computations are performed at bit level.

a = 12

x = a >> 2

y = a << 1

* + **Combinational Expressions:** We can also use different types of expressions in a single expression, and that will be termed as combinational expressions.

a = 16

b = 12

c = a + (b >> 1)

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

|  |  |
| --- | --- |
| **Expression** | **Statement** |
| An expression evaluates to a value | A statement executes something |
| The evaluation of a statement does not change state | The execution of a statement changes state |
| Evaluation of an expression always Produces or returns a result value. | Execution of a statement may or may not produces or displays a result value, it only does whatever the statement says. |
| Every expression can’t be a statement. | Every statement can be an expression. |
| **Example:** >>> a + 16 >>> 20 | **Example:** >>> x = 3 >>> print(x) **Output:** 3 |

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

bacon = 22

bacon + 1

After execution of code bacon contains **22**. Since bacon+1 is not assigned to variable bacon again after increment operation

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

'spam' + 'spamspam'

'spam' \* 3

Output of two terms is same ***spamspamspam*** first term does string concatenation and second term does multiplication of string

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

Rules for a valid variable name is

1. A variable name must start with a letter or the underscore character.
2. A variable name cannot start with a number.
3. A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_).
4. Variable names are case-sensitive (name, Name and NAME are three different variables).
5. The reserved words(keywords) cannot be used naming the variable.

Eggs satisfies all the above rules, whereas 100 is an invalid variable because it fails to satisfy rule number 2.

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

The int(), float(), and str() functions will evaluate to the integer, floating-point number, and string versions of the value passed to them.

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

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

This expression throws an error because integer value is tried to concatenate with string. Only strings can be concatenated to other strings with the + operator. It could be fixed by converted integer to string either using str or placing integer within quotes. i.e.

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

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