What is Data?
Data is a set of values of qualitative or quantitative variables.
Qualitative data is descriptive information (it describes something)
Quantitative data, is numerical information (numbers).

What is Information?
If we arrange some data in an appropriate sequence, then it forms a Structure and gives us meaning. We found two things in Information: One is Data and the other is Structure.

Define Data Structure?
A data structure is a specialized format for organizing and storing data.

A data structure is classified into two categories:
1 Linear data structures
2 Non-Linear data structures

Linear data structure:
A data structure which is sequential and continues in nature.
Example: array,linked list,stack and queue.

Non Linear data structures
These are arranged in random manner. It can be used to hierarchical relationship among data elements.
Example: Tree, Hash tree, Binary tree, Heap and graph.

Data Structures/Collections in PYTHON:
In PYTHON builtin Data Structures are Four types:
1 List is a collection which is ordered and changeable. Allows duplicate members. enclosed [ ]
2 Tuple is a collection which is ordered and unchangeable. Allows duplicate members. enclosed in parentheses()
3 Set is a collection which is unordered and unindexed. No duplicate members. enclosed in curly brackets {,}
4 Dictionary is a collection which is unordered, changeable and indexed. No duplicate members. Enclosed with curly brackets {}

For Data Structures in PYTHON:
Hashable is ==> Immutable (Readonly)
Unhashable is ==> Mutable (Read&Write)
LIST is Unhashable ==> Mutable (Read&Write)
TUPLE is Hashable => Immutable (Readonly)
SET is Unhashable ==> Mutable (Read&Write)
FROZENSET is Hashable=>Immutable(Readonly)
DICTIONARY :
KEY ==> Hashable(Readonly)
Value ==> Unhashable(Read&Write)

String operations in PYTHON:
In general, you can’t perform mathematical operations on strings, even if the strings look like numbers. The + operator performs string concatenation, which means it joins the strings by linking them end-to-end.

Example:

```
first="First"; second="Second"
fullstr=first+" "+second
print(fullstr)
```

Example:String Repetition

```
print('Spam'*3)
```

Example: Different Quotes in PYTHON:

```
PyStr="PYTHON"
print(type(PyStr))#<class 'str'>
PyStr='PYTHON'
print(type(PyStr))#<class 'str'>
PyStr="""PYTHON"""
print(type(PyStr))#<class 'str'>
PyStr='''PYTHON'''
print(type(PyStr))#<class 'str'>
```

Assigning and Re-Assigning:
To assign a value to Python variables, type the value after the equal sign(=).

Example:

```
MyStr="PYTHON"
print(MyStr)#PYTHON
MyStr="Machine Leaning"
print(MyStr)#Machine Leaning
```

Example:

```
PyStr="python"
print(PyStr)
print(PyStr+PyStr)
print(PyStr+" "+PyStr)
print(PyStr*3)
print(PyStr[0])
PyStr[0]='P'
print(PyStr)
```

Multiple Assignment
You can assign values to multiple python variables in one statement.

Example:

```
a=b=c=5
print(a);print(b);print(c)
```

Read Please?

Define String Slicing?
To cut a substring from a string is called string slicing.

Here two indices are used separated by a colon (:). A slice 3:7 means indices characters of 3rd, 4th, 5th and 6th positions. The second integer index i.e. 7 is not included. You can use negative indices for slicing.

Example:

```
PyStr="PYTHON IS FUN"
#P     Y        T      H      O    N       I     S          F      U
```

```
N
#0     1        2      3     4    5  6    7    8    9  10    11    12
#-13 -12  -11  -10   -9   -8 -7  -6   -5  -4  -3     -2     -1

print(PyStr[0:2])
print(PyStr[2:6])
print(PyStr[7:10])
print(PyStr[1:8])
```

String Indices or Indexing

Strings are arrays of characters and elements of an array can be accessed using indexing. Indices start with 0 from left side and -1 when starting from right side.

```
string1 ="PYTHON"
```

Character	P	Y	T	H	O	N
Index (from left)	0	1	2	3	4	5
Index (from right)	-6	-5	-4	-3	-2	-1

Example:

```
PyStr="Naresh i Technologies"
print(PyStr[0])
print(PyStr[-1])
print(PyStr[-4])
print(PyStr[4])
```