In [ ]:
```
# Datatypes :

# -> type of data the varaible has been assigned with.
# -> There are two types of data types :
#     1. Universal Datatype/ Basic DataType : int, float, char/str, bool
#     2. Collective Data Type : string, tuple, list, dictionary, set
```

In [ ]:
```
# Collective Datatype :

# -> Collection of values
# -> These collective datatypes can be further divided into two categories :
#     1. Mutable Datatype : values once assigned can be altered even after declaration. Eg: List, Dictionary, Set
#     2. Immutable Datatype : values once assigned can't be altered after declaration. Eg: String, Tuple
```

In [22]:
```
# Strings :

# -> collection of characters.
# SYNTAX :
#     varaiable_name = "string"
#     varaiable_name = 'string'

s1 = "Hello"
s2 = "Learners"
print(s1)
print(s2)

print(s1,s2)

# len() : return the number of elements present in the collective data type.

print(len(s1))
print(len(s2))

# type() : returns the type of data the variable is holding.

print(type(s1))
print(type(s2))

# str' object does not support item assignment
# s1[4] = "a"
```

```
Hello
Learners
Hello Learners
5
8
<class 'str'>
<class 'str'>
```

In [11]:

```python
# Indexing and Slicing :

# Indexing : process of accessing an element from the collection.
#     Index : positional value of an element.
# SYNTAX :
#     variable_name[index_number]

# There are two types of indexing :
# 1. Forward Indexing/ Regular Indexing : starts from left to right with a zero '0'
# 2. Reverse Indexing/ Negative Indexing : starts from right to left with a -1

s3 = "INTELLIPAAT"

# 0     1     2     3     4     5     6     7     8     9     10   -----> Forward Indexing
# I     N     T     E     L     L     I     P     A     A     T
# -11  -10   -9    -8    -7    -6    -5    -4    -3    -2    -1   ------> Reverse Indexing

print(s3[3])
print(s3[-8])

# Slicing : process of extracting multiple elements in a sequence from a collection
# SYNTAX:
#     variable_name[start_index : end_index + 1]

print(s3[0:5])
```

```
E
E
INTEL
```

In [20]:

```python
# Tuple :

# -> Its is a immutable collective datatype
# -> It is a collection of heterogenous element(elements with different datatypes).

# SYNTAX :
#     varaible_name = (elements separated by commas)
```

```
t1 = (1,2,3,"Python",23.45,'Avinash',4, True)
print(t1)

print(len(t1))

print(type(t1))

print(t1[2])
print(t1[0:4])

# 'tuple' object does not support item assignment
# t1[5] = "Intellipaat"
```

```
(1, 2, 3, 'Python', 23.45, 'Avinash', 4, True)
8
<class 'tuple'>
3
(1, 2, 3, 'Python')
```

In [35]:

```
# List :

# -> Its is a mutable collective datatype
# -> It is a collection of heterogenous element(elements with different datatypes).
# SYNTAX:
#     varaible_name = [elements separated by commas]

l1 = [1,2,3,"Python",23.45,'Avinash',4, True]
print(l1)

print(type(l1))
print(len(l1))

print(l1[3])
print(l1[2:5])

print(l1)

# Lists are mutable data type
l1[5] = "Intellipaat"
print(l1)

# Add a new value :

# append(): listname.append(value) -> adds the value at the end of the list
# insert(): listname.insert(index,value)
```

```
l1.append(67)
print(l1)
l1.append("Live")
print(l1)

l1.insert(3,"Java")
print(l1)
l1.insert(0,0)
print(l1)

# Remove a value :
# pop() : listname.pop(index)
# remove() : listname.remove(value)

l1.pop(4)
print(l1)
l1.remove("Live")
print(l1)
```

```
[1, 2, 3, 'Python', 23.45, 'Avinash', 4, True]
<class 'list'>
8
Python
[3, 'Python', 23.45]
[1, 2, 3, 'Python', 23.45, 'Avinash', 4, True]
[1, 2, 3, 'Python', 23.45, 'Intellipaat', 4, True]
[1, 2, 3, 'Python', 23.45, 'Intellipaat', 4, True, 67]
[1, 2, 3, 'Python', 23.45, 'Intellipaat', 4, True, 67, 'Live']
[1, 2, 3, 'Java', 'Python', 23.45, 'Intellipaat', 4, True, 67, 'Live']
[0, 1, 2, 3, 'Java', 'Python', 23.45, 'Intellipaat', 4, True, 67, 'Live']
[0, 1, 2, 3, 'Python', 23.45, 'Intellipaat', 4, True, 67, 'Live']
[0, 1, 2, 3, 'Python', 23.45, 'Intellipaat', 4, True, 67]
```

In [46]:

```
# Dictionary :

# -> Its is a mutable collective datatype
# -> It is a collection of heterogenous element(elements with different datatypes).
# -> It stores the data in the form of Key,Value pair.

# SYNTAX :
#     variable_name = {Key1:value1,key2:value2, ......... }

# No order is maintained (Indexing and Slicing is not applicable)
# Keys can't be of collective data type except for string.

d1 = {1:"Avinash", "Intellipaat":"Live Class",12.45:4000}
```

```
print(d1)

print(type(d1))

print(d1.keys())
print(d1.values())

d2 = {"Names": ["Harry", "Avinash", "Adi"], "Post" : ["CEO","Trainer","Intern"]}
print(d2)

d2["Names"][2] = "Ahaan"
print(d2)

print(d2["Post"])

d2["Designation"] = d2["Post"]
print(d2)

del d2["Post"]
print(d2)
```

```
{1: 'Avinash', 'Intellipaat': 'Live Class', 12.45: 4000}
<class 'dict'>
dict_keys([1, 'Intellipaat', 12.45])
dict_values(['Avinash', 'Live Class', 4000])
{'Names': ['Harry', 'Avinash', 'Adi'], 'Post': ['CEO', 'Trainer', 'Intern']}
{'Names': ['Harry', 'Avinash', 'Ahaan'], 'Post': ['CEO', 'Trainer', 'Intern']}
['CEO', 'Trainer', 'Intern']
{'Names': ['Harry', 'Avinash', 'Ahaan'], 'Post': ['CEO', 'Trainer', 'Intern'], 'Designation': ['CEO', 'Trainer', 'Intern']}
{'Names': ['Harry', 'Avinash', 'Ahaan'], 'Designation': ['CEO', 'Trainer', 'Intern']}
```

In [49]:

```
# Set :

# -> Unordered, unindexed list of elements
# -> It is a collection of heterogenous element(elements with different datatypes).
# -> It is a mutable data type

# SYNTAX :
#     variable_name = {elements separated by commas}


s1 = {0,1,2,3,"Python",23.55,"Avinash"}
print(s1)

s1.add("Adi")
print(s1)
```

```
s1.remove("Adi")
print(s1)
```

```
{0, 1, 2, 3, 'Python', 23.55, 'Avinash'}
{0, 1, 2, 3, 'Python', 'Adi', 23.55, 'Avinash'}
{0, 1, 2, 3, 'Python', 23.55, 'Avinash'}
```

In [ ]:
```
# Conditional Statements :

# -> Based on the conditions evaluted the statements are executed.
# There are three types of conditional statements :
# 1. simple if statement
# 2. if else statement
# 3. if elif else statement.
```

In [51]:
```
# simple if statement

# SYNTAX:
#     if(condition):
#         statements to be executed if the condition is true

# To check if a is greater than b

a = 30
b = 20
if(a>b):
    print("a is greater than b")

a = 20
b = 30
if(a>b):
    print("a is greater than b")
```

```
A is greater than B
```

In [52]:
```
# if else statement

# SYNTAX:
#     if(condition):
#         statements to be executed if the condition is true
#     else:
#         statements to be executed if the condition is false

# To check if a is greater than b
a = 30
```

```
b = 20
if(a>b):
    print("a is greater than b")
else:
    print("a is not greater than b")

a = 20
b = 30
if(a>b):
    print("a is greater than b")
else:
    print("a is not greater than b")
```

```
A is greater than B
A is not greater than B
```

In [59]:

```
# if elif else statement.

# SYNTAX:
#     if(condition):
#         statements to be executed if the condition is true
#     elif(condition2):
#         statements to be executed if the condition2 is true
#     elif(condition3):
#         statements to be executed if the condition3 is true
#     elif(condition4):
#         statements to be executed if the condition4 is true
#         .
#         .
#         .
#         .
#     else:
#         statements to be executed if the condition is false

# To check if a is greater than, less than or equal 50

a = 50
if(a > 50):
    print("a is greater than 50")
elif(a<50):
    print("a is less than 50")
else:
    print("a is equals to 50")
```

```
a is equals to 50
```

In [61]:
```
# Looping Statements :

# -> Statements that will allow us to perform the same set of tasks again and again

# There are two type of looping statement in python :
# 1. For Loop : works on range data.
# 2. While Loop : works on conditional data.
```

In [71]:
```
# For Loop:

# SYNTAX for for loop:
#     for iterating_varaible in range():
#         tasks to be performed

# range() : inbuilt functions that is used to create a range of defined values.
# SYNTAX for range() :
#     range(start, end+1)

# range(0,10) -> 0,1,2,3,4,5,6,7,8,9

# print(range(0,10))

# for i in range(0,10):
#     print(i)

# Print python 5 times

for i in range(100,105):
    print("Python")
```

```
Python
Python
Python
Python
Python
```

In [74]:
```
# While Loop :

# SYNTAX :
#     iterating_variable defination
#     while(condition):
#         tasks
#         increments

# Print python 5 times using while loop
```

```python
i = 0
while(i < 5):
    print("Python")
    i = i + 1

# i     i<5     print()     i=i+1
# 0     True    Python      1
# 1     True    Python      2
# 2     True    Python      3
# 3     True    Python      4
# 4     True    Python      5
# 5     Fasle -----> Exit the loop (Exit Condition)
```

```
Python
Python
Python
Python
Python
```

In [ ]: