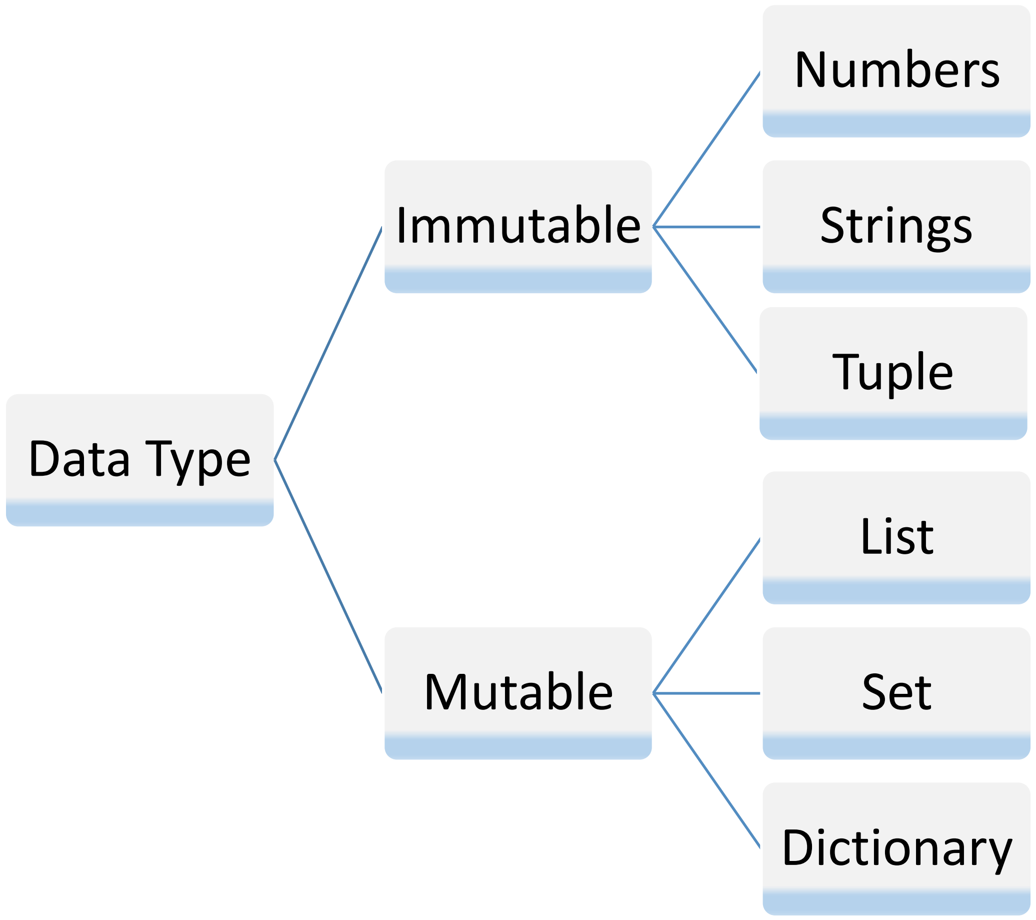
Data Types/Data Structures (Tuples and Lists)

Aegis

SCHOOL OF DATA SCIENCE

Data Types/ Inbuilt Data Structures in Python



Tuple

Tuple

- Tuple is a collection of like/unlike items similar to list.
- Tuple is immutable.
- In Tuple objects are separated by commas and are placed in parentheses ().

Syntax:

```
tuple = (value1, value2, value3,...valueN)
```

Tuple (contd...)

- Tuples can also represented without any parentheses.

For Example: -

```
tup1=56,
  Or
tup2 =12,16,15,18
```

- To access the elements of the tuples, indexing can be used.

```
print(tup2[3])
```

- Negative Indexing can be done.

```
print(tup2[-1])
```

Tuple(contd...)

As they are immutable any modification if done will lead to an error in tuple.

For eg: -

```
orderItem=(1, "Jeff", "Computer", 75.50, True)
orderItem[2]="Laptop“
```

```
TypeError: 'tuple' object does not support item assignment
```

Tuple (contd...)

- Use the del keyword to delete the tuple object.

```
del student
```

Aegis
SCHOOL OF DATA SCIENCE

Operations on Tuples

- Like string, tuple objects are also a sequence. Hence, the operators used with strings are also available for tuple.

Operator	Description	Example

+ Concatenation	Returns a tuple containing all the elements of the first and the second tuple object.	`>>> t1=(1,2,3)`
		 `>>> t2=(4,5,6)`
		 `>>> t1+t2`
		 `(1, 2, 3, 4, 5, 6)`
		 `>>> t2+(7,)`
		 `(4, 5, 6, 7)`

* Repetition	Concatenates multiple copies of the same tuple.	`>>> t1=(1,2,3)`
		 `>>> t1*4`
		 `(1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)`

Aegis
SCHOOL OF DATA SCIENCE

Operations on Tuples

[] slice	Returns the item at the given index. A negative index counts the position from the right side.	`>>> t1=(1,2,3,4,5,6)`
		`>>> t1[3]`
		`4`
		`>>> t1[-2]`
		`5`

[:] - Range slice	Fetches the items in the range specified by two index operands separated by the : symbol.
	If the first operand is omitted, the range starts at zero index. If the second operand is omitted, the range goes up to the end of tuple.
		`>>> t1=(1,2,3,4,5,6)`
		`>>> t1[1:3]`
		`(2, 3)`
		`>>> t1[3:]`
		`(4, 5, 6)`
		`>>> t1[:3]`
		`(1, 2, 3)`

in	Returns true if an item exists in the given tuple.	`>>> t1=(1,2,3,4,5,6)`
		`>>> 5 in t1`
		`True`
		`>>> 10 in t1`
		`False`

not in	Returns true if an item does not exist in the given tuple.	`>>> t1=(1,2,3,4,5,6)`
		`>>> 4 not in t1`
		`False`
		`>>> 10 not in t1`
		`True`

Aegis
SCHOOL OF DATA SCIENCE

Built-in Tuple Methods

- **len()**

Returns the number of elements in the tuple.

```
 t1=(12,45,43,8,35)
 len(t1)
5
```

- **max()**

If the tuple contains numbers, the highest number will be returned. If the tuple contains strings, the one that comes last in alphabetical order will be returned.

```
 t1=(12, 45, 43, 8, 35)
 max(t1)
45
 t2=('python', 'java', 'C++')
 max(t2)
'python'
```

Built-in Tuple Methods

- **min()**

If the tuple contains numbers, the lowest number will be returned. If the tuple contains strings, the one that comes first in alphabetical order will be returned.

```
 t1=(12,45,43,8,35)
 min(t1)
8
 t2=('python', 'java', 'C++')
 min(t2)
'C++'
```

Mutable Data Types

Lists

Lists

- Usually lists are homogenous collection of the data, but python support both heterogeneous as well as homogenous.
- It is an ordered sequence of items.
- Values in the lists are separated by comma and enclosed in square brackets [ ].

Syntax:

```
list = [value1, value2, value3,...valueN]
```

For example:-

```
names=["Jeff", "Bill", "Steve", "Mohan"]
orderItem=[1, "Jeff", "Computer", 75.50, True]
```

Aegis
SCHOOL OF DATA SCIENCE

Lists (Contd...)

- Each individual element in the sequence is accessed by the index in the square brackets []. An index starts with zero, as shown below.

```
orderItem=[1, "Jeff", "Computer", 75.50, True]
orderItem[0]
    #Output: - 1
orderItem[1]
   #Output: - 'Jeff'
orderItem[2]
   #Output: - 'Computer'
```

Aegis
SCHOOL OF DATA SCIENCE

Lists (Contd...)

- The list object is mutable. It is possible to modify its contents, which will modify the value in the memory.
- For instance, item at index 2 in orderItem can be modified as shown below.

```
orderItem=[1, "Jeff", "Computer", 75.50, True]
orderItem[2]="Laptop“
orderItem
        #Output: - [1, "Jeff", "Laptop", 75.50, True]
```

Lists (Contd...)

del keywords :- Use the *del* keyword to delete the list object.

del languages

```
languages
 Traceback (most recent call last):
 File "<stdin>", line 1, in <module>
 NameError: name 'languages' is not defined
```

Aegis
SCHOOL OF DATA SCIENCE

Operations on Lists

- Like the string, the list is also a sequence. Hence, the operators used with strings are also available for use with the list (and tuple also).

Operator	Description	Example

+ Concatenation	Returns a list containing all the elements of the first and the second list.	>>> L1=[1,2,3]
		>>> L2=[4,5,6]
		>>> L1+L2
		[1, 2, 3, 4, 5, 6]

* Repetition	Concatenates multiple copies of the same list.	>>> L1*4 [1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3]

[] slice	Returns the item at the given index. A negative index counts the position from the right side.	>>> L1=[1, 2, 3, 4, 5, 6]
		>>> L1[3]
		4
		>>> L1[-2]
		5

Aegis SCHOOL OF DATA SCIENCE

Operations on Lists

[:] - Range slice	Fetches items in the range specified by the two index operands separated by : symbol. If the first operand is omitted, the range starts from the zero index. If the second operand is omitted, the range goes up to the end of the list.	>>> L1=[1, 2, 3, 4, 5, 6]
		>>> L1[1:4]
		[2, 3, 4]
		>>> L1[3:]
		[4, 5, 6]
		>>> L1[:3]
		[1, 2, 3]

in	Returns true if an item exists in the given list.	>>> L1=[1, 2, 3, 4, 5, 6]
		>>> 4 in L1
		True
		>>> 10 in L1
		False

not in	Returns true if an item does not exist in the given list.	>>> L1=[1, 2, 3, 4, 5, 6]
		>>> 5 not in L1
		False
		>>> 10 not in L1
		True

Aegis
SCHOOL OF DATA SCIENCE

Built-in List Methods

- **len()**

The len() method returns the number of elements in the list/tuple.

```
L1=[12,45,43,8,35]
len(L1)
  5
```

- **max()**

The max() method returns the largest number, if the list contains numbers. If the list contains strings, the one that comes last in alphabetical order will be returned.

```
L1=[12,45,43,8,35]
max(L1)
45
L2=['Python', 'Java', 'C++']
max(L2)
'Python'
```

Built-in List Methods

- **min()**

The min() method returns the smallest number, if the list contains numbers. If the list contains strings, the one that comes first in alphabetical order will be returned.

```
 L1=[12, 45, 43, 8, 35]
 min(L1)
8
 L2=['Python', 'Java', 'C++']
 min(L2)
'C++'
```

Aegis

SCHOOL OF DATA SCIENCE

Built-in List Methods

- **append()**

Adds an item at the end of the list.

```
L2=['Python', 'Java', 'C++']
L2.append('PHP')
L2
['Python', 'Java', 'C++', 'PHP']
```

- **insert()**

Inserts an item in a list at the specified index.

```
L2=['Python', 'Java', 'C++']
L2.insert(1,'Perl')
L2
['Python', 'Perl', 'Java', 'C++']
```

Aegis
SCHOOL OF DATA SCIENCE

Built-in List Methods

- **remove()**

Removes a specified object from the list.

```
 L2=['Python', 'Perl', 'Java', 'C++']
 L2.remove('Java')
 L2
['Python', 'Perl', 'C++']
```

- **pop()**

Removes and returns the last object in the list.

```
 L2=['Python', 'Perl', 'Java', 'C++']
 L2.pop()
'C++‘
L2
['Python', 'Perl', 'Java']
```

Built-in List Methods

- **reverse()**

Reverses the order of the items in a list.

```
 L2=['Python', 'Perl', 'Java', 'C++']
 L2.reverse()
 L2
['C++', 'Java', 'Perl', 'Python']
```

Aegis
SCHOOL OF DATA SCIENCE

Built-in List Methods

- **sort()**

Rearranges the items in the list according to the alphabetical order. Default is the ascending order. For descending order, put reverse=True as an argument in the function bracket.

```
L2=['Python', 'C++', 'Java', 'Ruby']
 L2.sort()
 L2
['C++', 'Java', 'Python', 'Ruby']
 L2.sort(reverse=True)
 L2
['Ruby', 'Python', 'Java', 'C++']
```

Built-in List Methods

The following utility functions help in converting one sequence data type to another.

- **list()**

Converts a tuple or string to a list object.

```
t2=('python', 'java', 'C++')
list(t2)
['python', 'java', 'C++']
s1="Tutorials“
list(s1)
 ['T', 'u', 't', 'o', 'r', 'i', 'a', 'l', 's']
```

Built-in List Methods

- **tuple()**

Converts a list or string to a tuple object.

```
 L2=['C++', 'Java', 'Python', 'Ruby']
 tuple(L2)
('C++', 'Java', 'Python', 'Ruby')
 s1="Tutorials“
 tuple(s1)
('T', 'u', 't', 'o', 'r', 'i', 'a', 'l', 's')
```

Aegis
SCHOOL OF DATA SCIENCE

List Comprehensions

Let us consider a list of squares of even numbers:

In Normal way:

```
squares_of_even=[]
for n in range(10):
   if n%2==0:
        squares_of_even.append(n*n)
print(squares_of_even)
```

Aegis
SCHOOL OF DATA SCIENCE

List Comprehensions

- General syntax of List Comprehensions

new_list=[new_item for item in input_list]

new_list=[new_item for item in input_list if some_condition]

List Comprehensions

- Squares of even numbers:

```
squares_of_even=[n*n for n in range(10) if (n%2==0)
print(squares_of_even)
```