List

List

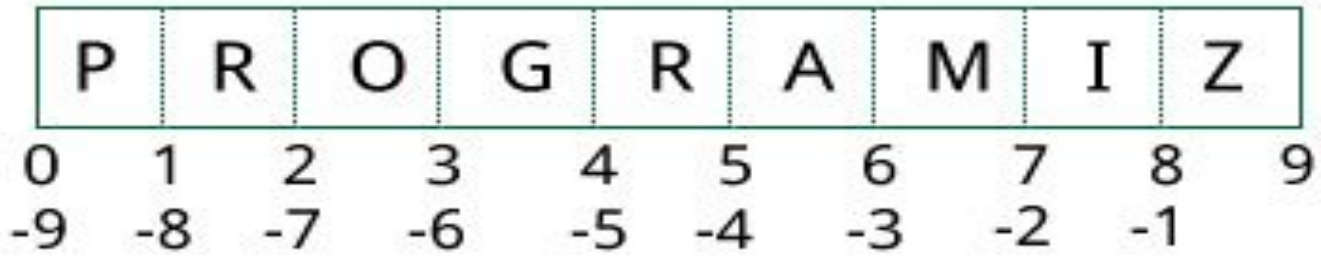
- A list is a collection which is ordered and changeable. In Python lists are written with square brackets.

For example –

```
list1 = ['physics', 'chemistry', 1997, 2000];

list2 = [1, 2, 3, 4, 5 ];

list3 = ["a", "b", "c", "d"]
```



Slices in detail

A slice is a part of a sequence,returned as a new sequence.

The original sequence remains unchanged.

The [] brackets are used as a slice operator.

b=a[1] #b is a single element slice of a

b=a[2:5] #slice is a[2].a[3] and a[4]

b=a[2:7:2] #slice is a[2],a[4],a[6] the index gap is 2 for elements a[start:end:stride]

Updating Lists

For example –

```
list = ['physics', 'chemistry', 1997, 2000];
print("Value available at index 2 : ")
print(list[2])
list[2] = 2001;
print("New value available at index 2 : ")
print(list[2])
```

output::

```
Value available at index 2 :
1997
New value available at index 2 :
2001
```

Delete List Elements

```
list1 = ['physics', 'chemistry', 1997, 2000];
print(list1)
del list1[2];
print("After deleting value at index 2 : ")
print(list1)
```

Built-in List Functions & Methods

Sr.No.	Function with Description
1	len(list) Gives the total length of the list.
2	max(list) Returns item from the list with max value.
3	min(list) Returns item from the list with min value.
4	list(seq) Converts a tuple into list.

Sr.No.	Methods with Description
1	list.append(obj) Appends object obj to list
2	list.count(obj) Returns count of how many times obj occurs in list
3	list.extend(seq) Appends the contents of seq to list
4	list.index(obj) Returns the lowest index in list that obj appears
5	list.insert(index, obj) Inserts object obj into list at offset index
6	list.pop(obj=list[-1]) Removes and returns last object or obj from list
7	list.remove(obj) Removes object obj from list
8	list.reverse() Reverses objects of list in place
9	list.sort([func]) Sorts objects of list, use compare func if given

List Comprehensions

- List comprehensions provide a concise way to create lists. Common applications are to make new lists where each element is the result of some operations applied to each member of another sequence or iterable, or to create a subsequence of those elements that satisfy a certain condition.
- **E.g.**

```
odd_square = []
for x in range(1, 11):
    if x % 2 == 1:
        odd_square.append(x**2)
print(odd_square)
```

```
[1, 9, 25, 49, 81]
```

```
# below list contains square of all # odd numbers from range 1 to 10
odd_square = [x ** 2 for x in range(1, 11) if x % 2 == 1]
print(odd_square)
```

Tuple

Tuples are ordered immutable sequences

a=10,20,30,40

a=(10,20,30,40)

In both the above, a tuple is assigned to the variable a.

a[0] is 10.

a[0]=12 is not allowed since a tuple is immutable.

Tuples can be added to get a new tuple.

a=(10,20,30,40)

b=(15,16)

c=a+b

print(c)

This would print:

(10,20,30,40,15,16)

Accessing Values in Tuples

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

output –

tup1[0]: physics

tup2[1:5]: (2, 3, 4, 5)

Updating Tuples

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# Following action is not valid for tuples
# tup1[0] = 100;

# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

output : –

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

del tup;

Basic Tuples Operations

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Built-in Tuple Functions

Sr. No.	Function with Description
1	cmp(tuple1, tuple2) Compares elements of both tuples.
2	len(tuple) Gives the total length of the tuple.
3	max(tuple) Returns item from the tuple with max value.
4	min(tuple) Returns item from the tuple with min value.
5	tuple(seq) Converts a list into tuple.
6	count() Returns the number of times a specified value occurs in a tuple
7	index() Searches the tuple for a specified value and returns the position of where it was found

set

A set is a collection which is unordered and unindexed.

In Python sets are written with curly brackets

sets are unordered immutable sequences of UNIQUE elements.

a={10,20,30,40}

b={20,50,60}

c=set() #makes an empty set

If a duplicate element was provided during assignment,

it would be discarded automatically.

Python Built-in set methods

Method	Description
add(item)	It adds an item to the set. It has no effect if the item is already present in the set.
clear()	It deletes all the items from the set.
copy()	It returns a shallow copy of the set.
difference() Symbol : -	It modifies this set by removing all the items that are also present in the specified sets.
intersection() Symbol : &	It returns a new set that contains only the common elements of both the sets.
Isdisjoint(....)	Return True if two sets have a null intersection.
Issubset(....)	Report whether another set contains this set.
Issuperset(....)	Report whether this set contains another set.
pop()	Remove and return an arbitrary set element that is the last element of the set. Raises KeyError if the set is empty.
remove(item)	Remove an element from a set; it must be a member. If the element is not a member, raise a KeyError.
symmetric_difference(....) Symbol : ^	Remove an element from a set; it must be a member. If the element is not a member, raise a KeyError.
union(....) Symbol : |	Return the union of sets as a new set. (i.e. all elements that are in either set.)
update()	Update a set with the union of itself and others.