What is difference between list and set?

List	Set
List is ordered collection	Set is unordered collection
List is sequence data type	Set is non sequence data type
List allows duplicate elements	Set does not allows duplicate elements
List support indexing and slicing	Set does not support indexing and slicing
In list elements are organized in sequential order	In set elements are organized using hashing data structure
List allows any type of objects	Set allows only hashable objects (immutable objects)
List is create using []	Set is created using {}
“list” class or data type represents list object	“set” class or data type represents set object
In application development list is used to represent group of individual objects where duplicates are allowed.	In application development set is used to represent group of individual objects where duplicates are not allowed and perform mathematical set operations.

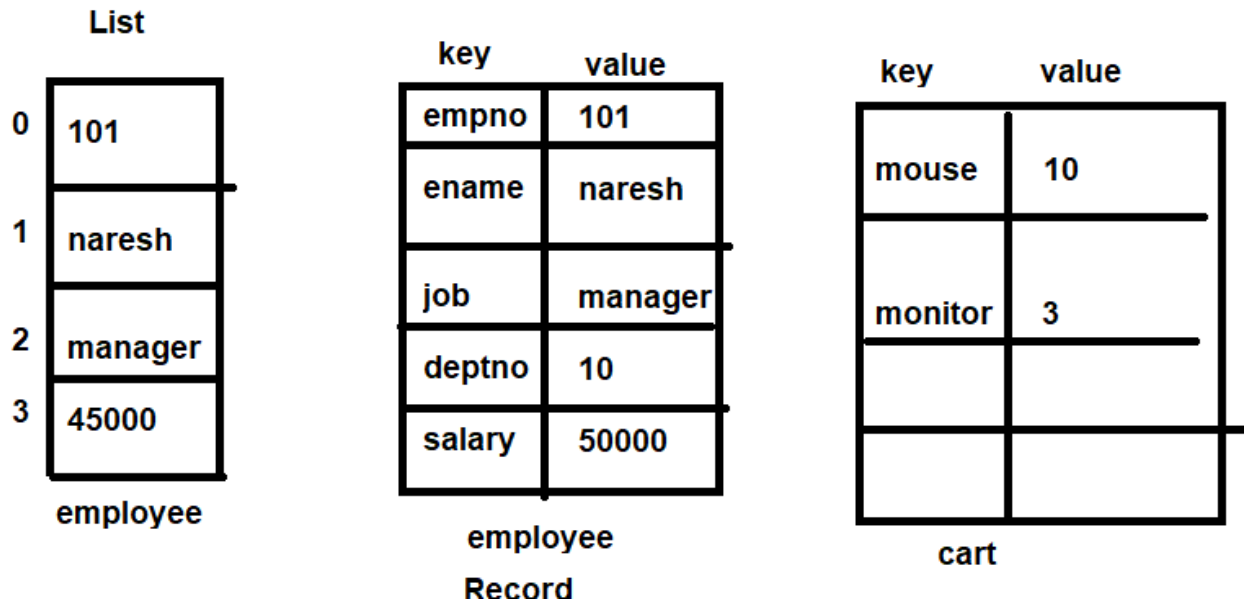
What is difference between set and frozenset?

Set	Frozenset
Set is a mutable collection	Frozenset is immutable collection
Set cannot be used as an element inside set	Frozenset can be used to represent element within set.

Dictionary
Mapping Types — dict

A mapping object maps hashable values to arbitrary objects. Mappings are mutable objects. There is currently only one standard mapping type, the *dictionary*.

In dictionary data is organized as pair of values (key,value). Each key is mapped with one or more than one value.
Duplicate keys are not allowed but duplicate values are allowed.
Keys are immutable and values are mutable.



How to create dictionary?

Dictionaries can be created by several means:

- Use a comma-separated list of key: value pairs within braces: {'jack': 4098, 'sjoerd': 4127} or {4098: 'jack', 4127: 'sjoerd'}
- Use a dict comprehension: {}, {x: x ** 2 for x in range(10)}
- Use the type constructor: dict(), dict([('foo', 100), ('bar', 200)]), dict(foo=100, bar=200)

Creating empty dictionary

```
>>> d1={}
>>> print(d1)
{}
>>> print(type(d1))
```

```
<class 'dict'>
>>> d2=dict()
>>> print(d2)
{}
>>> print(type(d2))
<class 'dict'>
```

Creating dictionary with items

```
>>> person={'naresh':50,'suresh':40,'kishore':30}
>>> course={'java':3000,'python':6000,'C':2000}
>>> emp={'empno':[1,2,3],'ename':['naresh','suresh','kishore']}
>>> print(person)
{'naresh': 50, 'suresh': 40, 'kishore': 30}
>>> print(course)
{'java': 3000, 'python': 6000, 'C': 2000}
>>> print(emp)
{'empno': [1, 2, 3], 'ename': ['naresh', 'suresh', 'kishore']}
>>> d3={1:10,2:20,3:30,4:40,5:50}
>>> print(d3)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> d4={1:10,1:20,1:30,1:40,1:50}
>>> print(d4)
{1: 50}
>>> d5={1:10,2:10,3:10}
>>> print(d5)
{1: 10, 2: 10, 3: 10}
```

Creating dictionary using dict() function

dict() is type conversion function, this function is used to convert other iterables into dictionary type.

Syntax-1: dict() : Create empty dictionary
Syntax-2: dict(iterable) : Create dictionary using existing iterable/collection. This iterable or collection must generate key and value.

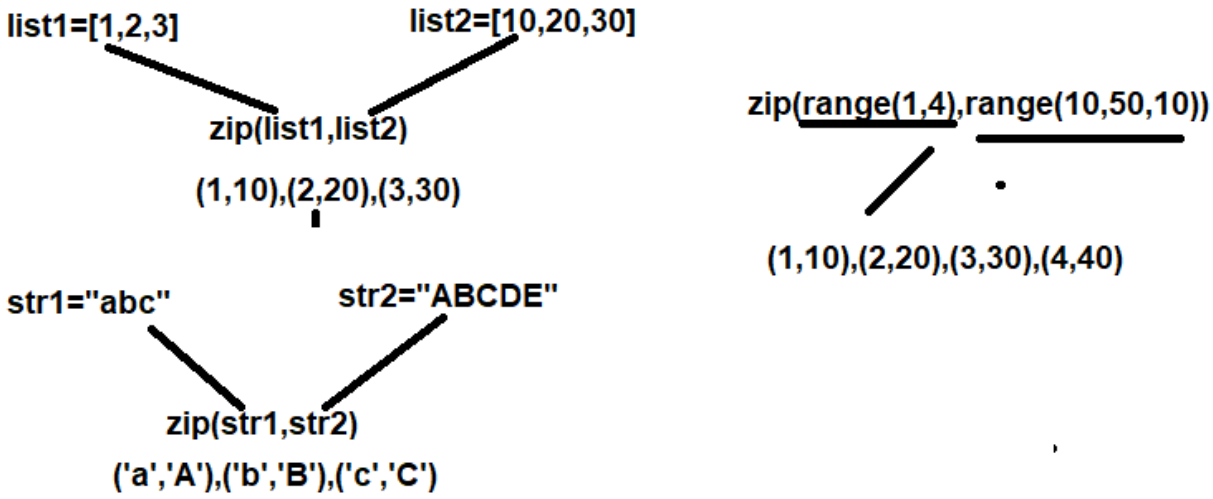
Example:

```
>>> d1=dict()
>>> print(d1)
{}
```

```
>>> print(type(d1))
<class 'dict'>
>>> list1=[1,2,3,4,5]
>>> d2=dict(list1)
Traceback (most recent call last):
  File "<pyshell#23>", line 1, in <module>
    d2=dict(list1)
TypeError: cannot convert dictionary update sequence element #0 to a
sequence
>>> list1=[[1,10],[2,20],[3,30],[4,40],[5,50]]
>>> d2=dict(list1)
>>> print(list1)
[[1, 10], [2, 20], [3, 30], [4, 40], [5, 50]]
>>> print(d2)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> t1=(('naresh',40),('suresh',35),('kishore',20))
>>> d3=dict(t1)
>>> print(d3)
{'naresh': 40, 'suresh': 35, 'kishore': 20}
>>> l1=[1,2,3]
>>> l2=[10,20,30]
>>> l3=[[l1[i],l2[i]] for i in range(3)]
>>> print(l1)
[1, 2, 3]
>>> print(l2)
[10, 20, 30]
>>> print(l3)
[[1, 10], [2, 20], [3, 30]]
>>> d4=dict(l3)
>>> print(d4)
{1: 10, 2: 20, 3: 30}
```

zip(*iterables)
Iterate over several iterables in parallel, producing tuples with an item from each one.

More formally: zip() returns an iterator of tuples, where the i-th tuple contains the i-th element from each of the argument iterables.




```
>>> d5=dict(zip(range(1,6),range(10,60,10)))
>>> print(d5)
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50}
>>> d6=dict(zip("abc","ABC"))
>>> print(d6)
{'a': 'A', 'b': 'B', 'c': 'C'}
>>> d7=dict(zip([1,2,3],[10,20,30]))
>>> print(d7)
{1: 10, 2: 20, 3: 30}
```

How to read content of dictionary?

Dictionary content can be read in different ways

1. Using key
2. Using for loop
3. Using dict functions
    a. Keys()
    b. Values()
    c. Items()
    d. get()