1. Write a simple implementation of this model which clearly shows how the different classes are composed. Write some example code to show how you would use your classes to create an album and add all its songs to a playlist. Hint: if two objects are related to each other bidirectionally, you will have to decide how this link should be formed –one of the objects will have to be created before the other, so you can't link them to each other in both directions simultaneously!

Code:

```
class Song:


    def __init__(self, title, artist, album, track_number):
        self.title = title
        self.artist = artist
        self.album = album
        self.track_number = track_number


        artist.add_song(self)



class Album:


    def __init__(self, title, artist, year):
        self.title = title
        self.artist = artist
        self.year = year


        self.tracks = []


        artist.add_album(self)


    def add_track(self, title, artist=None):
        if artist is None:
            artist = self.artist
```

```
        track_number = len(self.tracks)

        song = Song(title, artist, self, track_number)

        self.tracks.append(song)


class Artist:
    def __init__(self, name):
        self.name = name

        self.albums = []
        self.songs = []

    def add_album(self, album):
        self.albums.append(album)

    def add_song(self, song):
        self.songs.append(song)


class Playlist:
    def __init__(self, name):
        self.name = name
        self.songs = []

    def add_song(self, song):
        self.songs.append(song)

band = Artist("Bob's Awesome Band")
album = Album("Bob's First Single", band, 2013)
album.add_track("A Ballad about Cheese")
```

```
album.add_track("A Ballad about Cheese (dance remix)")
album.add_track("A Third Song to Use Up the Rest of the Space")

playlist = Playlist("My Favourite Songs")

for song in album.tracks:
    playlist.add_song(song)
```

2. Write an "abstract" class, Box, and use it to define some methods which any box object should have: add, for adding any number of items to the box, empty, for taking all the items out of the box and returning them as a list, and count, for counting the items which are currently in the box. Write a simple Itemclass which has a name attribute and a value attribute –you can assume that all the items you will use will be Itemobjects. Now write two subclasses of Box which use different underlying collections to store items: ListBox should use a list, and DictBox should use a dict.

Code:

```
class Box:
    def add(self, *items):
        raise NotImplementedError()

    def empty(self):
        raise NotImplementedError()

    def count(self):
        raise NotImplementedError()


class Item:
    def __init__(self, name, value):
        self.name = name
        self.value = value
```

```
class ListBox(Box):
    def __init__(self):
        self._items = []

    def add(self, *items):
        self._items.extend(items)

    def empty(self):
        items = self._items
        self._items = []
        return items

    def count(self):
        return len(self._items)


class DictBox(Box):
    def __init__(self):
        self._items = {}

    def add(self, *items):
        self._items.update(dict((i.name, i) for i in items))

    def empty(self):
        items = list(self._items.values())
        self._items = {}
        return items
```

```
    def count(self):
        return len(self._items)
```

3. Write a program to PrintFibonacci series with iterator (special method) only.

Code:-

```
class Fibonacci:
    def _init_(self, n):
        self.n = n
        self.i = 0

    def _iter_(self):
        self.a = 0
        self.b = 1
        return self

    def _next_(self):
        if self.i == 0:
            self.i += 1
            return self.a
        elif self.i == 1:
            self.i += 1
            return self.b
        elif self.i < self.n:
            x = self.a + self.b
            self.a = self.b
            self.b = x
            self.i += 1
            return x
        else:
            raise StopIteration
```

```
fibo = Fibonacci(7)
myiter = iter(fibo)

for x in myiter:
    print(x)
```