```python
#QUESTION 1
# Write a function in Python with a string such that it accepts a
parameter- “stringsplit”.
# This encoded string will contain your name, domain name and register
number.
#  You can separate the values in the string by any number of
underscores.
# [The string should not contain any other underscore symbols in your
name, domain name and register number].
#  The function should return a Python dictionary with your name,
domain name and register number.

def stringsplit(x):
    x = x.replace('_', ' ')
    name, domain, regNo = x.split()
    myDict = {"Name": name, "Domain": domain, "Registration Number":
regNo}
    return myDict

encode_string = input("Enter 'Name', 'Domain', 'Registration No.' with
any number of underscores in between: ")
newDict = stringsplit(encode_string)
print(newDict)
```

```
{'Name': 'alwin', 'Domain': 'blockchain', 'Registration Number':
'123'}
```

```python
#QUESTION 2
# Write a Python program to implement the object-oriented concepts of
multiple,
#  Multilevel and Hierarchical Inheritances using your domain
applications.
#Using Multiple Inheritence.
class User:
    def __init__(self, username):
        self.username = username
        self.playlists = []

    def create_playlist(self, name):
        playlist = Playlist(name)
        self.playlists.append(playlist)
        return playlist

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

    def upload_song(self, title, genre):
        song = Song(title, genre, self)
```

```
        self.songs.append(song)
        return song

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

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

class Song:
    def __init__(self, title, genre, artist):
        self.title = title
        self.genre = genre
        self.artist = artist
        self.likes = 0

    def like(self):
        self.likes += 1

user1 = User("user1")
artist1 = Artist("Artist 1")
genre1 = Genre("Pop")

song1 = artist1.upload_song("Song 1", genre1)
genre1.add_song(song1)

print(f"Song: {song1.title}")
print(f"Artist: {song1.artist.name}")
print(f"Genre: {song1.genre.name}")

playlist1 = user1.create_playlist("My Playlist")
playlist1.add_song(song1)

print(f"{user1.username}'s Playlists:")
for playlist in user1.playlists:
    print(playlist.name)
```

```
Song: Song 1
Artist: Artist 1
Genre: Pop
user1's Playlists:
My Playlist
```

```
#Using Multilevel Inheritence.
class Entity:
    def __init__(self, name):
        self.name = name

class Artist(Entity):
```

```
    def __init__(self, name):
        super().__init__(name)
        self.songs = []

    def upload_song(self, title, genre):
        song = Song(title, genre, self)
        self.songs.append(song)
        return song

class Genre(Entity):
    def __init__(self, name):
        super().__init__(name)
        self.songs = []

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

class Song:
    def __init__(self, title, genre, artist):
        self.title = title
        self.genre = genre
        self.artist = artist
        self.likes = 0

    def like(self):
        self.likes += 1

class User(Entity):
    def __init__(self, username):
        super().__init__(username)
        self.playlists = []

    def create_playlist(self, name):
        playlist = Playlist(name)
        self.playlists.append(playlist)
        return playlist

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

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

user1 = User("user1")
artist1 = Artist("Artist 1")
genre1 = Genre("Pop")

song1 = artist1.upload_song("Song 1", genre1)
```

```python
genre1.add_song(song1)

print(f"Song: {song1.title}")
print(f"Artist: {song1.artist.name}")
print(f"Genre: {song1.genre.name}")

playlist1 = user1.create_playlist("My Playlist")
playlist1.add_song(song1)

print(f"{user1.name}'s Playlists:")
for playlist in user1.playlists:
    print(playlist.name)
```

```
Song: Song 1
Artist: Artist 1
Genre: Pop
user1's Playlists:
My Playlist
```

```python
# Using Heirarchical Inheritence
class Entity:
    def __init__(self, name):
        self.name = name

class Artist(Entity):
    def __init__(self, name):
        super().__init__(name)
        self.songs = []

    def upload_song(self, title, genre):
        song = Song(title, genre, self)
        self.songs.append(song)
        return song

class Genre(Entity):
    def __init__(self, name):
        super().__init__(name)
        self.songs = []

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

class Song:
    def __init__(self, title, genre, artist):
        self.title = title
        self.genre = genre
        self.artist = artist
        self.likes = 0

    def like(self):
        self.likes += 1
```

```python
class User(Entity):
    def __init__(self, username):
        super().__init__(username)
        self.playlists = []

    def create_playlist(self, name):
        playlist = Playlist(name)
        self.playlists.append(playlist)
        return playlist

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

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

user1 = User("user1")
artist1 = Artist("Artist 1")
genre1 = Genre("Pop")

song1 = artist1.upload_song("Song 1", genre1)
genre1.add_song(song1)

print(f"Song: {song1.title}")
print(f"Artist: {song1.artist.name}")
print(f"Genre: {song1.genre.name}")

playlist1 = user1.create_playlist("My Playlist")
playlist1.add_song(song1)

print(f"{user1.name}'s Playlists:")
for playlist in user1.playlists:
    print(playlist.name)
```

```
Song: Song 1
Artist: Artist 1
Genre: Pop
user1's Playlists:
My Playlist
```