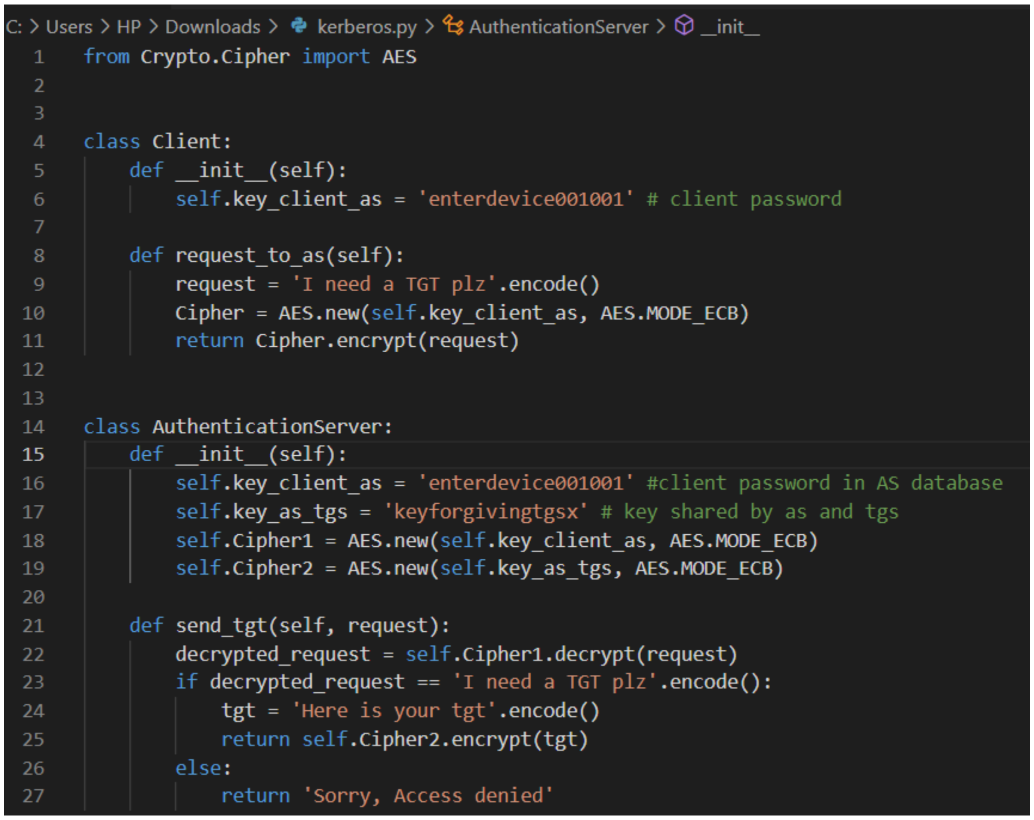
Q. Simulate Kerberos system in a dummy application.

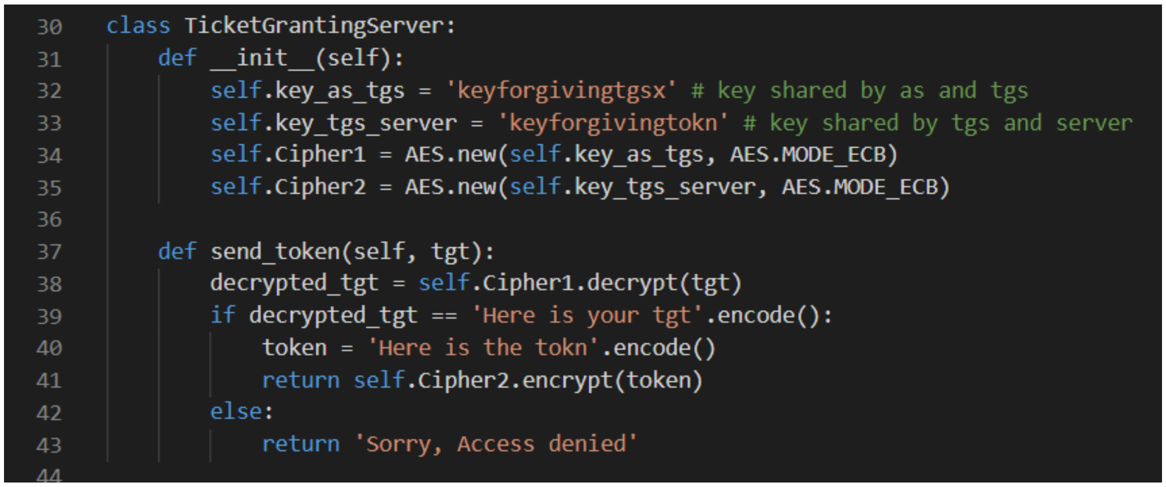
Kerberos System

Kerberos is a computer network security protocol that authenticates service requests across an untrusted network, such as the internet, between two or more verified hosts. It authenticates client-server applications and verifies users' identities using secret-key cryptography and a trusted third party. Kerberos uses cryptographic tickets to avoid transmitting plain text passwords. Kerberos shares a secret key with the KDC. This secret key is known only to the KDC and the service principal on each IBM Streams resource. The service principal for IBM Streams is the authentication and authorization service. The authentication and authorization service on each resource must be registered with the KDC. The Kerberos administrator generates a keytab file that the authentication and authorization service uses to authenticate to the KDC.

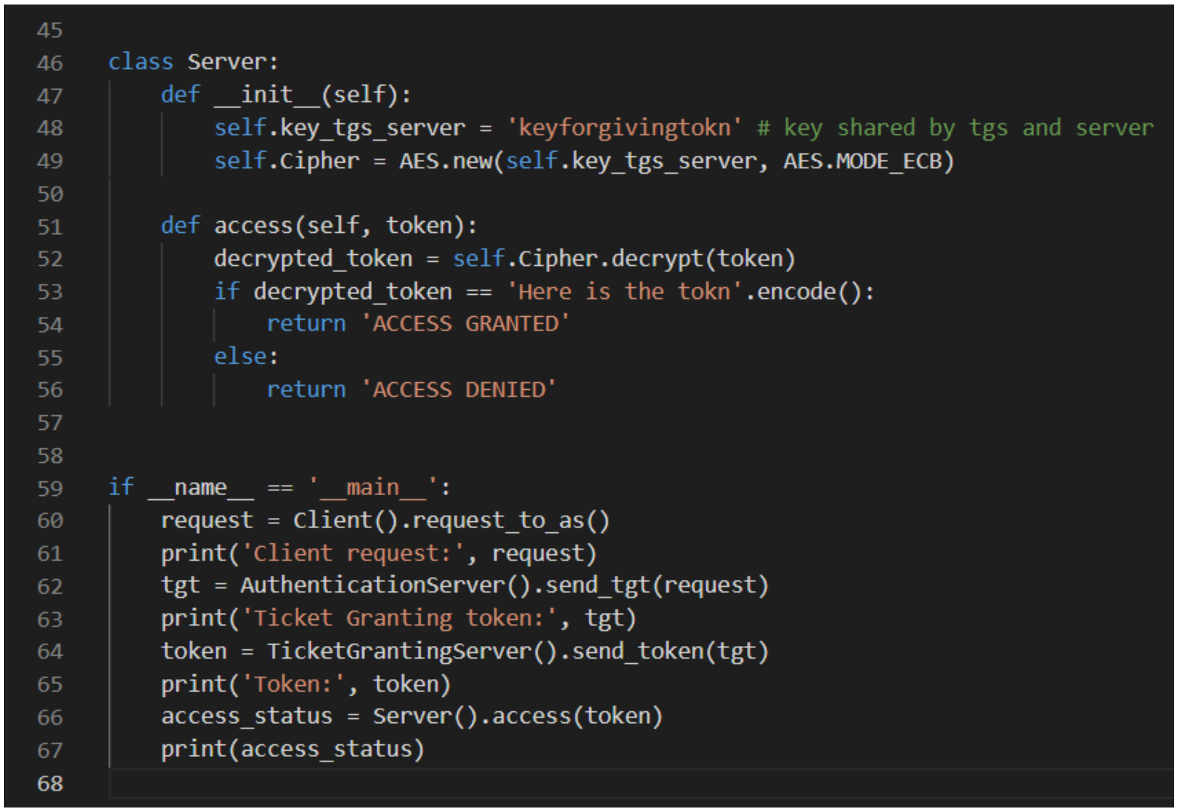
Code


C: > Users > HP > Downloads > kerberos.py > AuthenticationServer > __init__

```python
from Crypto.Cipher import AES


class Client:
    def __init__(self):
        self.key_client_as = 'enterdevice001001' # client password

    def request_to_as(self):
        request = 'I need a TGT plz'.encode()
        Cipher = AES.new(self.key_client_as, AES.MODE_ECB)
        return Cipher.encrypt(request)


class AuthenticationServer:
    def __init__(self):
        self.key_client_as = 'enterdevice001001' #client password in AS database
        self.key_as_tgs = 'keyforgivingtgsx' # key shared by as and tgs
        self.Cipher1 = AES.new(self.key_client_as, AES.MODE_ECB)
        self.Cipher2 = AES.new(self.key_as_tgs, AES.MODE_ECB)

    def send_tgt(self, request):
        decrypted_request = self.Cipher1.decrypt(request)
        if decrypted_request == 'I need a TGT plz'.encode():
            tgt = 'Here is your tgt'.encode()
            return self.Cipher2.encrypt(tgt)
        else:
            return 'Sorry, Access denied'
```

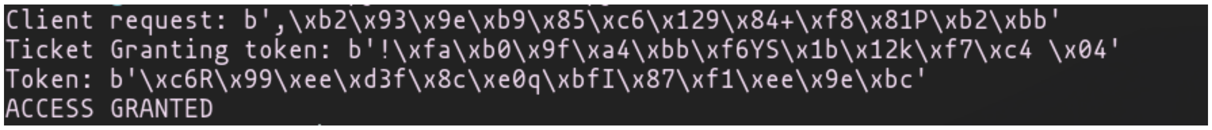


```python
class TicketGrantingServer:
    def __init__(self):
        self.key_as_tgs = 'keyforgivingtgsx' # key shared by as and tgs
        self.key_tgs_server = 'keyforgivingtokn' # key shared by tgs and server
        self.Cipher1 = AES.new(self.key_as_tgs, AES.MODE_ECB)
        self.Cipher2 = AES.new(self.key_tgs_server, AES.MODE_ECB)

    def send_token(self, tgt):
        decrypted_tgt = self.Cipher1.decrypt(tgt)
        if decrypted_tgt == 'Here is your tgt'.encode():
            token = 'Here is the tokn'.encode()
            return self.Cipher2.encrypt(token)
        else:
            return 'Sorry, Access denied'
```



```python

class Server:
    def __init__(self):
        self.key_tgs_server = 'keyforgivingtokn' # key shared by tgs and server
        self.Cipher = AES.new(self.key_tgs_server, AES.MODE_ECB)

    def access(self, token):
        decrypted_token = self.Cipher.decrypt(token)
        if decrypted_token == 'Here is the tokn'.encode():
            return 'ACCESS GRANTED'
        else:
            return 'ACCESS DENIED'


if __name__ == '__main__':
    request = Client().request_to_as()
    print('Client request:', request)
    tgt = AuthenticationServer().send_tgt(request)
    print('Ticket Granting token:', tgt)
    token = TicketGrantingServer().send_token(tgt)
    print('Token:', token)
    access_status = Server().access(token)
    print(access_status)

```

Output:



```
Client request: b',\xb2\x93\x9e\xb9\x85\xc6\x129\x84+\xf8\x81P\xb2\xbb'
Ticket Granting token: b'!\xfa\xb0\x9f\xa4\xbb\xf6YS\x1b\x12k\xf7\xc4 \x04'
Token: b'\xc6R\x99\xee\xd3f\x8c\xe0q\xbfI\x87\xf1\xee\x9e\xbc'
ACCESS GRANTED
```

Description

During login, the Authentication Server (AS) authenticates the user. The Ticket Granting Server (TGS) is responsible for issuing identification tickets. The server performs the role of a service provider. It is AS's job to authenticate each user at login time. On AS, each user has their own personal password. TGS's job is to verify that a user is who he or she claims to be to the network's servers. To showcase this, tickets are utilized (which enable admittance onto a server in the same manner that a ticket lets you park a car or enter a music event). Every AS user has his or her own unique password. TGS's job is to make sure a user is who he or she says he or she is to the network's servers. The mechanism of tickets (which allow admission into a server in the same manner as a ticket allows parking or access into a concert) is used to demonstrate this.