import os

import sys

import random

import asyncio

import time

from datetime import datetime

from os import execl

from telethon.network.connection.tcpabridged import ConnectionTcpAbridged

from telethon.sessions import StringSession

from telethon import TelegramClient, events

from telethon.tl.functions.account import UpdateProfileRequest

from Config import GROUP\_USERNAME, STRING, SUDO, BIO\_MESSAGE, API\_ID, API\_ID2, API\_ID3, API\_ID4, API\_ID5, API\_ID6, API\_ID7, API\_ID8, API\_ID9, API\_ID10, API\_ID11, API\_ID12, API\_ID13, API\_ID14, API\_ID15, API\_ID16, API\_ID17, API\_ID18, API\_ID19, API\_ID20, API\_ID21, API\_ID22, API\_ID23, API\_ID24, API\_ID25, API\_HASH, API\_HASH2, API\_HASH3, API\_HASH4, API\_HASH5, API\_HASH6, API\_HASH7, API\_HASH8, API\_HASH9, API\_HASH10, API\_HASH11, API\_HASH12, API\_HASH13, API\_HASH14, API\_HASH15, API\_HASH16, API\_HASH17, API\_HASH18, API\_HASH19, API\_HASH20, API\_HASH21, API\_HASH22, API\_HASH23, API\_HASH24, API\_HASH25, STRING2, STRING3, STRING4 ,STRING5, STRING6, STRING7, STRING8 ,STRING9, STRING10, STRING11, STRING12 , STRING13 , STRING14 , STRING15 ,STRING16 , STRING17 , STRING18 , STRING19 , STRING20 , STRING21 , STRING22 , STRING23 , STRING24 , STRING25

import asyncio

import telethon.utils

from telethon.tl import functions

from telethon.tl.functions.channels import LeaveChannelRequest

from telethon.tl.functions.channels import JoinChannelRequest

from telethon.errors import (

ChannelInvalidError,

ChannelPrivateError,

ChannelPublicGroupNaError,

)

from telethon.tl.functions.channels import GetFullChannelRequest, InviteToChannelRequest

grp = GROUP\_USERNAME

if "@" in grp:

grp = grp.replace("@", "")

sup = API\_ID

aa = API\_ID2 or sup

ba = API\_ID3 or sup

ca = API\_ID4 or sup

da = API\_ID5 or sup

ea = API\_ID6 or sup

fa = API\_ID7 or sup

ga = API\_ID8 or sup

ha = API\_ID9 or sup

ia = API\_ID10 or sup

ja = API\_ID11 or sup

ka = API\_ID12 or sup

la = API\_ID13 or sup

ma = API\_ID14 or sup

na = API\_ID15 or sup

oa = API\_ID16 or sup

pa = API\_ID17 or sup

qa = API\_ID18 or sup

ra = API\_ID19 or sup

sa = API\_ID20 or sup

ta = API\_ID21 or sup

ua = API\_ID22 or sup

va = API\_ID23 or sup

wa = API\_ID24 or sup

xa = API\_ID25 or sup

sap = API\_HASH

ab = API\_HASH2 or sap

bb = API\_HASH3 or sap

cb = API\_HASH4 or sap

db = API\_HASH5 or sap

eb = API\_HASH6 or sap

fb = API\_HASH7 or sap

gb = API\_HASH8 or sap

hb = API\_HASH9 or sap

ib = API\_HASH10 or sap

jb = API\_HASH11 or sap

kb = API\_HASH12 or sap

lb = API\_HASH13 or sap

mb = API\_HASH14 or sap

nb = API\_HASH15 or sap

ob = API\_HASH16 or sap

pb = API\_HASH17 or sap

qb = API\_HASH18 or sap

rb = API\_HASH19 or sap

sb = API\_HASH20 or sap

tb = API\_HASH21 or sap

ub = API\_HASH22 or sap

vb = API\_HASH23 or sap

wb = API\_HASH24 or sap

xb = API\_HASH25 or sap

smex = STRING

smexx = STRING2

smexxx = STRING3

smexxxx = STRING4

smexxxxx = STRING5

sixth = STRING6

seven = STRING7

eight = STRING8

ninth = STRING9

tenth = STRING10

eleve = STRING11

twelv = STRING12

thirt = STRING13

forte = STRING14

fifth = STRING15

sieee = STRING16

seeee = STRING17

eieee = STRING18

nieee = STRING19

gandu = STRING20

ekish = STRING21

baish = STRING22

teish = STRING23

tfour = STRING24

tfive = STRING25

idk = ""

ydk = ""

wdk = ""

sdk = ""

hdk = ""

adk = ""

bdk = ""

cdk = ""

edk = ""

ddk = ""

vkk = ""

kkk = ""

lkk = ""

mkk = ""

sid = ""

shy = ""

aan = ""

ake = ""

eel = ""

khu = ""

shi = ""

yaa = ""

dav = ""

raj = ""

put = ""

que = {}

SMEX\_USERS = []

for x in SUDO:

SMEX\_USERS.append(x)

async def start\_yukki():

global idk

global ydk

global wdk

global sdk

global hdk

global adk

global bdk

global cdk

global ddk

global edk

global vkk

global kkk

global lkk

global mkk

global sid

global shy

global aan

global ake

global eel

global khu

global shi

global yaa

global dav

global raj

global put

if smex:

session\_name = StringSession(str(smex))

print("String 1 Found")

idk = TelegramClient(

session=session\_name,

api\_id=sup,

api\_hash=sap,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#idk = TelegramClient(session\_name, sup, sap)

try:

print("Booting Up The Client 1")

await idk.start()

botme = await idk.get\_me()

await idk(functions.channels.JoinChannelRequest(channel="@Mm\_Userbot"))

await idk(functions.channels.JoinChannelRequest(channel="@mm\_ub\_updates"))

await idk(functions.channels.JoinChannelRequest(channel="@professor\_agora"))

await idk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

idk = "smex"

print(e)

pass

else:

print("Session 1 not Found")

session\_name = "startup"

idk = TelegramClient(session\_name, sup, sap)

try:

await idk.start()

except Exception as e:

pass

if smexx:

session\_name = StringSession(str(smexx))

print("String 2 Found")

ydk = TelegramClient(

session=session\_name,

api\_id=aa,

api\_hash=ab,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

try:

print("Booting Up The Client 2")

await ydk.start()

await ydk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await ydk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

await ydk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot\_bot\_BOT"))

await ydk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await ydk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 2 not Found")

pass

session\_name = "startup"

ydk = TelegramClient(session\_name, aa, ab)

try:

await ydk.start()

except Exception as e:

pass

if smexxx:

session\_name = StringSession(str(smexxx))

print("String 3 Found")

wdk = TelegramClient(

session=session\_name,

api\_id=ba,

api\_hash=bb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#wdk = TelegramClient(StringSession(session\_name), ba, bb)

try:

print("Booting Up The Client 3")

await wdk.start()

await wdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await wdk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

await wdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot\_bot\_BOT"))

await wdk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await wdk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 3 not Found")

pass

session\_name = "startup"

wdk = TelegramClient(

session=session\_name,

api\_id=ba,

api\_hash=bb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

# wdk = TelegramClient(session\_name, ba, bb)

try:

await wdk.start()

except Exception as e:

pass

if smexxxx:

session\_name = StringSession(str(smexxxx))

print("String 4 Found")

hdk = TelegramClient(

session=session\_name,

api\_id=ca,

api\_hash=cb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#hdk = TelegramClient(StringSession(session\_name), ca, cb)

try:

print("Booting Up The Client 4")

await hdk.start()

await hdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await hdk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

await hdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await hdk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await hdk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 4 not Found")

pass

session\_name = "startup"

hdk = TelegramClient(

session=session\_name,

api\_id=ca,

api\_hash=cb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#hdk = TelegramClient(session\_name, ca, cb)

try:

await hdk.start()

except Exception as e:

pass

if smexxxxx:

session\_name = StringSession(str(smexxxxx))

print("String 5 Found")

sdk = TelegramClient(

session=session\_name,

api\_id=da,

api\_hash=db,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#sdk = TelegramClient(StringSession(session\_name), da, db)

try:

print("Booting Up The Client 5")

await sdk.start()

await sdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await sdk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

await sdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await sdk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await sdk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 5 not Found")

pass

session\_name = "startup"

sdk = TelegramClient(

session=session\_name,

api\_id=da,

api\_hash=db,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#sdk = TelegramClient(session\_name, da, db)

try:

await sdk.start()

except Exception as e:

pass

if sixth:

session\_name = StringSession(str(sixth))

print("String 6 Found")

adk = TelegramClient(

session=session\_name,

api\_id=ea,

api\_hash=eb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#adk = TelegramClient(StringSession(session\_name), ea, eb)

try:

print("Booting Up The Client 6")

await adk.start()

await adk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await adk(functions.channels.JoinChannelRequest(channel="@Offcial\_LegendBot"))

await adk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await adk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await adk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 6 not Found")

pass

session\_name = "startup"

adk = TelegramClient(

session=session\_name,

api\_id=ea,

api\_hash=eb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#adk = TelegramClient(session\_name, ea, eb)

try:

await adk.start()

except Exception as e:

pass

if seven:

session\_name = StringSession(str(seven))

print("String 7 Found")

bdk = TelegramClient(

session=session\_name,

api\_id=fa,

api\_hash=fb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#bdk = TelegramClient(StringSession(session\_name), fa, fb)

try:

print("Booting Up The Client 7")

await bdk.start()

await bdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await bdk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

await bdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await bdk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await bdk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 7 not Found")

pass

session\_name = "startup"

bdk = TelegramClient(

session=session\_name,

api\_id=fa,

api\_hash=fb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#bdk = TelegramClient(session\_name, fa, fb)

try:

await bdk.start()

except Exception as e:

pass

if eight:

session\_name = StringSession(str(eight))

print("String 8 Found")

cdk = TelegramClient(

session=session\_name,

api\_id=ga,

api\_hash=gb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#cdk = TelegramClient(StringSession(session\_name), ga, gb)

try:

print("Booting Up The Client 8")

await cdk.start()

await cdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await cdk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

await cdk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await cdk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await cdk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 8 not Found")

pass

session\_name = "startup"

cdk = TelegramClient(session\_name, ga, gb)

try:

await cdk.start()

except Exception as e:

pass

if ninth:

session\_name = StringSession(str(ninth))

print("String 9 Found")

ddk = TelegramClient(

session=session\_name,

api\_id=ha,

api\_hash=hb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#ddk = TelegramClient(StringSession(session\_name), ha, hb)

try:

print("Booting Up The Client 9")

await ddk.start()

await ddk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await ddk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

await ddk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await ddk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await ddk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 9 not Found")

pass

session\_name = "startup"

ddk = TelegramClient(session\_name, ha, hb)

try:

await ddk.start()

except Exception as e:

pass

if tenth:

session\_name = StringSession(str(tenth))

print("String 10 Found")

edk = TelegramClient(

session=session\_name,

api\_id=ia,

api\_hash=ib,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

# edk = TelegramClient(StringSession(session\_name), ia, ib)

try:

print("Booting Up The Client 10")

await edk.start()

await edk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await edk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

await edk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await edk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await edk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 10 not Found")

pass

session\_name = "startup"

edk = TelegramClient(session\_name, ia, ib)

try:

await edk.start()

except Exception as e:

pass

if eleve:

session\_name = StringSession(str(eleve))

print("String 11 Found")

vkk = TelegramClient(

session=session\_name,

api\_id=ja,

api\_hash=jb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#vkk = TelegramClient(StringSession(session\_name), ja, jb)

try:

print("Booting Up The Client 11")

await vkk.start()

await vkk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await vkk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

await vkk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await vkk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

botme = await vkk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 11 not Found")

pass

session\_name = "startup"

vkk = TelegramClient(session\_name, ja, jb)

try:

await vkk.start()

except Exception as e:

pass

if twelv:

session\_name = StringSession(str(twelv))

kkk = TelegramClient(

session=session\_name,

api\_id=ka,

api\_hash=kb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#kkk = TelegramClient(StringSession(session\_name), ka, kb)

try:

print("Booting Up The Client 12")

await kkk.start()

await kkk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await kkk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await kkk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await kkk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botme = await kkk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 12 not Found")

pass

session\_name = "startup"

kkk = TelegramClient(session\_name, ka, kb)

try:

await kkk.start()

except Exception as e:

pass

if thirt:

session\_name = StringSession(str(thirt))

print("String 13 Found")

lkk = TelegramClient(

session=session\_name,

api\_id=la,

api\_hash=lb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#lkk = TelegramClient(StringSession(session\_name), la, lb)

try:

print("Booting Up The Client 13")

await lkk.start()

await lkk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await lkk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await lkk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await lkk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botme = await lkk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 13 not Found")

pass

session\_name = "startup"

lkk = TelegramClient(session\_name, la, lb)

try:

await lkk.start()

except Exception as e:

pass

if forte:

session\_name = StringSession(str(forte))

print("String 14 Found")

mkk = TelegramClient(

session=session\_name,

api\_id=ma,

api\_hash=mb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#mkk = TelegramClient(StringSession(session\_name), ma, mb)

try:

print("Booting Up The Client 14")

await mkk.start()

await mkk(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await mkk(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await mkk(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await mkk(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botme = await mkk.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 14 not Found")

pass

session\_name = "startup"

mkk = TelegramClient(session\_name, ma, mb)

try:

await mkk.start()

except Exception as e:

pass

if fifth:

session\_name = StringSession(str(fifth))

print("String 15 Found")

sid = TelegramClient(

session=session\_name,

api\_id=na,

api\_hash=nb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#sid = TelegramClient(StringSession(session\_name), na, nb)

try:

print("Booting Up The Client 15")

await sid.start()

await sid(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await sid(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await sid(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await sid(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botme = await sid.get\_me()

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 15 not Found")

pass

session\_name = "startup"

sid = TelegramClient(session\_name, na, nb)

try:

await sid.start()

except Exception as e:

pass

if sieee:

session\_name = StringSession(str(sieee))

print("String 16 Found")

shy = TelegramClient(

session=session\_name,

api\_id=oa,

api\_hash=ob,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#shy = TelegramClient(StringSession(session\_name), oa, ob)

try:

print("Booting Up The Client 16")

await shy.start()

botme = await shy.get\_me()

await shy(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await shy(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await shy(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await shy(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 16 not Found")

session\_name = "startup"

shy = TelegramClient(session\_name, oa, ob)

try:

await shy.start()

except Exception as e:

pass

if seeee:

session\_name = StringSession(str(seeee))

print("String 17 Found")

aan = TelegramClient(

session=session\_name,

api\_id=pa,

api\_hash=pb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#aan = TelegramClient(StringSession(session\_name), pa, pb)

try:

print("Booting Up The Client 17")

await aam.start()

botme = await aan.get\_me()

await aan(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await aan(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await aan(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await aan(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 17 not Found")

session\_name = "startup"

aan = TelegramClient(session\_name, pa, pb)

try:

await aan.start()

except Exception as e:

pass

if eieee:

session\_name = StringSession(str(eieee))

print("String 18 Found")

ake = TelegramClient(

session=session\_name,

api\_id=qa,

api\_hash=qb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#ake = TelegramClient(StringSession(session\_name), qa, qb)

try:

print("Booting Up The Client 18")

await ake.start()

botme = await ake.get\_me()

await ake(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await ake(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await ake(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await ake(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 18 not Found")

session\_name = "startup"

ake = TelegramClient(session\_name, qa, qb)

try:

await ake.start()

except Exception as e:

pass

if nieee:

session\_name = StringSession(str(nieee))

print("String 19 Found")

eel = TelegramClient(

session=session\_name,

api\_id=ra,

api\_hash=rb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#eel = TelegramClient(StringSession(session\_name), ra, rb)

try:

print("Booting Up The Client 19")

await eel.start()

botme = await eel.get\_me()

await eel(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await eel(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await eel(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await eel(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 19 not Found")

session\_name = "startup"

eel = TelegramClient(session\_name, ra, rb)

try:

await idk.start()

except Exception as e:

pass

if gandu:

session\_name = StringSession(str(gandu))

print("String 20 Found")

khu = TelegramClient(

session=session\_name,

api\_id=sa,

api\_hash=sb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#khu = TelegramClient(StringSession(session\_name), sa, sb)

try:

print("Booting Up The Client 20")

await khu.start()

botme = await khu.get\_me()

await khu(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await khu(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await khu(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await khu(functions.channels.JoinChannelRequest(channel="@Official\_\_LegendBot"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 20 not Found")

session\_name = "startup"

khu = TelegramClient(session\_name, sa, sb)

try:

await khu.start()

except Exception as e:

pass

if ekish:

session\_name = StringSession(str(ekish))

print("String 21 Found")

shi = TelegramClient(

session=session\_name,

api\_id=ta,

api\_hash=tb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#shi = TelegramClient(StringSession(session\_name), ta, tb)

try:

print("Booting Up The Client 21")

await shi.start()

botme = await shi.get\_me()

await shi(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await shi(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await shi(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await shi(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 21 not Found")

session\_name = "startup"

shi = TelegramClient(session\_name, ta, tb)

try:

await shi.start()

except Exception as e:

pass

if baish:

session\_name = StringSession(str(baish))

print("String 22 Found")

yaa = TelegramClient(

session=session\_name,

api\_id=ua,

api\_hash=ub,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#yaa = TelegramClient(StringSession(session\_name), ua, ub)

try:

print("Booting Up The Client 22")

await yaa.start()

botme = await yaa.get\_me()

await yaa(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await yaa(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await yaa(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await yaa(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 22 not Found")

session\_name = "startup"

yaa = TelegramClient(session\_name, ua, ub)

try:

await yaa.start()

except Exception as e:

pass

if teish:

session\_name = StringSession(str(teish))

print("String 23 Found")

dav = TelegramClient(

session=session\_name,

api\_id=va,

api\_hash=vb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#dav = TelegramClient(StringSession(session\_name), va, vb)

try:

print("Booting Up The Client 23")

await dav.start()

botme = await dav.get\_me()

await dav(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await dav(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await dav(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await dav(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 23 not Found")

session\_name = "startup"

dav = TelegramClient(session\_name, va, vb)

try:

await dav.start()

except Exception as e:

pass

if tfour:

session\_name = StringSession(str(tfour))

print("String 24 Found")

raj = TelegramClient(

session=session\_name,

api\_id=wa,

api\_hash=wb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#raj = TelegramClient(StringSession(session\_name), wa, wb)

try:

print("Booting Up The Client 24")

await raj.start()

botme = await raj.get\_me()

await raj(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await raj(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await raj(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await raj(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 24 not Found")

session\_name = "startup"

raj = TelegramClient(session\_name, wa, wb)

try:

await raj.start()

except Exception as e:

pass

if tfive:

session\_name = StringSession(str(tfive))

print("String 25 Found")

put = TelegramClient(

session=session\_name,

api\_id=xa,

api\_hash=xb,

connection=ConnectionTcpAbridged,

auto\_reconnect=True,

connection\_retries=None,

)

#put = TelegramClient(StringSession(session\_name), xa, xb)

try:

print("Booting Up The Client 25")

await put.start()

botme = await put.get\_me()

await put(functions.channels.JoinChannelRequest(channel="@Mm\_userbot"))

await put(functions.channels.JoinChannelRequest(channel=f"@{grp}"))

await put(functions.channels.JoinChannelRequest(channel="@Mm\_userbotSpam"))

await put(functions.channels.JoinChannelRequest(channel="@MM\_UB\_UPDATES"))

botid = telethon.utils.get\_peer\_id(botme)

SMEX\_USERS.append(botid)

except Exception as e:

print(e)

pass

else:

print("Session 25 not Found")

session\_name = "startup"

put = TelegramClient(session\_name, xa, xb)

try:

await put.start()

except Exception as e:

pass

loop = asyncio.get\_event\_loop()

loop.run\_until\_complete(start\_yukki())

async def gifspam(e, smex):

try:

await e.client(

functions.messages.SaveGifRequest(

id=types.InputDocument(

id=sandy.media.document.id,

access\_hash=smex.media.document.access\_hash,

file\_reference=smex.media.document.file\_reference,

),

unsave=True,

)

)

except Exception as e:

pass

def user\_full\_name(user):

names = [user.first\_name, user.last\_name]

names = [i for i in list(names) if i]

full\_name = " ".join(names)

return full\_name

async def get\_chatinfo(event):

yukki = ("".join(event.text.split(maxsplit=1)[1:])).split(" ", 1)

chat = yukki[0]

chat\_info = None

if chat:

try:

chat = int(chat)

except ValueError:

pass

if not chat:

if event.reply\_to\_msg\_id:

replied\_msg = await event.get\_reply\_message()

if replied\_msg.fwd\_from and replied\_msg.fwd\_from.channel\_id is not None:

chat = replied\_msg.fwd\_from.channel\_id

else:

chat = event.chat\_id

try:

chat\_info = await event.client(GetFullChatRequest(chat))

except:

try:

chat\_info = await event.client(GetFullChannelRequest(chat))

except ChannelInvalidError:

await event.edit("`Invalid channel/group`")

return None

except ChannelPrivateError:

await event.edit(

"`This is a private channel/group or I am banned from there`"

)

return None

except ChannelPublicGroupNaError:

await event.reply("`Channel or supergroup doesn't exist`")

return None

except (TypeError, ValueError):

await event.reply("`Invalid channel/group`")

return None

return chat\_info

@idk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@ydk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@wdk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@hdk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@sdk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@adk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@bdk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@cdk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@edk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@ddk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@vkk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@kkk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@lkk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@mkk.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@sid.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@shy.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@aan.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@ake.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@eel.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@khu.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@shi.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@yaa.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@dav.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@raj.on(events.NewMessage(incoming=True, pattern=r"\.join"))

@put.on(events.NewMessage(incoming=True, pattern=r"\.join"))

async def \_(e):

usage = "𝗠𝗼𝗱𝘂𝗹𝗲 𝗡𝗮𝗺𝗲 = 𝗝𝗼𝗶𝗻\n\nCommand:\n\n.join <Public Channel or Group Link/Username>"

if e.sender\_id in SMEX\_USERS:

yukki = ("".join(e.text.split(maxsplit=1)[1:])).split(" ", 1)

if len(e.text) > 6:

bc = yukki[0]

text = "Joining..."

event = await e.reply(text, parse\_mode=None, link\_preview=None )

try:

await e.client(functions.channels.JoinChannelRequest(channel=bc))

await event.edit("𝐉𝐎𝐢𝐍 𝐇𝐎𝐆𝐘𝐀 𝐕𝐀𝐈 AB BATA KISKO MARU PAHLE🔥")

except Exception as e:

await event.edit(str(e))

else:

await e.reply(usage, parse\_mode=None, link\_preview=None )

@idk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@ydk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@wdk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@hdk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@sdk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@adk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@bdk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@cdk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@edk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@ddk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@vkk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@kkk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@lkk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@mkk.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@sid.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@shy.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@aan.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@ake.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@eel.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@khu.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@shi.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@yaa.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@dav.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@raj.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

@put.on(events.NewMessage(incoming=True, pattern=r"\.pjoin"))

async def \_(e):

usage = "𝗠𝗼𝗱𝘂𝗹𝗲 𝗡𝗮𝗺𝗲 = 𝗣𝗿𝗶𝘃𝗮𝘁𝗲 𝗝𝗼𝗶𝗻\n\nCommand:\n\n.pjoin <Private Channel or Group's access hash>\n\nExample :\nLink = https://t.me/joinchat/HGYs1wvsPUplMmM1\n\n.pjoin HGYs1wvsPUplMmM1"

if e.sender\_id in SMEX\_USERS:

yukki = ("".join(e.text.split(maxsplit=1)[1:])).split(" ", 1)

if len(e.text) > 7:

bc = yukki[0]

text = "Joining...."

event = await e.reply(text, parse\_mode=None, link\_preview=None )

try:

await e.client(ImportChatInviteRequest(bc))

await event.edit("𝐉𝐎𝐢𝐍 𝐇𝐎𝐆𝐘𝐀 𝐕𝐀𝐈 𝐀𝐁 𝐁𝐓𝐀 𝐊𝐈𝐒𝐊𝐈 𝐌𝐀𝐑𝐔😏🔥")

except Exception as e:

await event.edit(str(e))

else:

await e.reply(usage, parse\_mode=None, link\_preview=None )

@idk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@ydk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@wdk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@hdk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@sdk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@adk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@bdk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@cdk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@edk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@ddk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@vkk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@kkk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@lkk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@mkk.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@sid.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@shy.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@aan.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@ake.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@eel.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@khu.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@shi.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@yaa.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@dav.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@raj.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

@put.on(events.NewMessage(incoming=True, pattern=r"\.pleave"))

async def \_(e):

usage = "𝗠𝗼𝗱𝘂𝗹𝗲 𝗡𝗮𝗺𝗲 = 𝗟𝗲𝗮𝘃𝗲\n\nCommand:\n\n.leave <Channel or Chat ID>"

if e.sender\_id in SMEX\_USERS:

yukki = ("".leave(e.text.split(maxsplit=1)[1:])).split(" ", 1)

if len(e.text) == 7:

bc = yukki[0]

bc = int(bc)

text = "BOT Leaving....."

event = await e.reply(text, parse\_mode=None, link\_preview=None )

try:

await event.client(LeaveChannelRequest(bc))

await event.edit("Succesfully Left")

except Exception as e:

await event.edit(str(e))

else:

await e.reply(usage, parse\_mode=None, link\_preview=None )

@idk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@ydk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@wdk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@hdk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@sdk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@adk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@bdk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@cdk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@edk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@ddk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@vkk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@kkk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@lkk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@mkk.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@sid.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@shy.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@aan.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@ake.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@eel.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@khu.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@shi.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@yaa.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@dav.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@raj.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

@put.on(events.NewMessage(incoming=True, pattern=r"\.ping"))

async def ping(e):

if e.sender\_id in SMEX\_USERS:

start = datetime.now()

text = "Pong!"

event = await e.reply(text, parse\_mode=None, link\_preview=None )

end = datetime.now()

ms = (end-start).microseconds / 1000

await event.edit(f"PING PONG!\n`{ms}` 𝗺𝘀")

@idk.on(events.NewMessage(incoming=True, pattern=r"\.invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await idk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@ydk.on(events.NewMessage(incoming=True, pattern=r"\\*invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await ydk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@wdk.on(events.NewMessage(incoming=True, pattern=r"\?invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await wdk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@hdk.on(events.NewMessage(incoming=True, pattern=r"\+invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await hdk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@sdk.on(events.NewMessage(incoming=True, pattern=r"\-invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await sdk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@adk.on(events.NewMessage(incoming=True, pattern=r"\×invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await adk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@bdk.on(events.NewMessage(incoming=True, pattern=r"\÷invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await bdk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@cdk.on(events.NewMessage(incoming=True, pattern=r"\=invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await cdk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@edk.on(events.NewMessage(incoming=True, pattern=r"\.invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await edk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@ddk.on(events.NewMessage(incoming=True, pattern=r"\\*invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await ddk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@vkk.on(events.NewMessage(incoming=True, pattern=r"\?invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await vkk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@kkk.on(events.NewMessage(incoming=True, pattern=r"\+invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await kkk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@lkk.on(events.NewMessage(incoming=True, pattern=r"\-invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await lkk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@mkk.on(events.NewMessage(incoming=True, pattern=r"\×invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await mkk(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@sid.on(events.NewMessage(incoming=True, pattern=r"\÷invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await sid(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@shy.on(events.NewMessage(incoming=True, pattern=r"\=invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await shy(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@aan.on(events.NewMessage(incoming=True, pattern=r"\.invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await aan(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@ake.on(events.NewMessage(incoming=True, pattern=r"\\*invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await ake(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@eel.on(events.NewMessage(incoming=True, pattern=r"\?invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await eel(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@khu.on(events.NewMessage(incoming=True, pattern=r"\+invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await khu(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@shi.on(events.NewMessage(incoming=True, pattern=r"\-invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await shi(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@yaa.on(events.NewMessage(incoming=True, pattern=r"\×invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await yaa(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@dav.on(events.NewMessage(incoming=True, pattern=r"\÷invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await dav(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@raj.on(events.NewMessage(incoming=True, pattern=r"\=invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

legend = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*⚜️[Ͳєямιиαℓ Տτατυѕ](https://t.me/Mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(legend.full\_chat.id):

try:

await raj(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@put.on(events.NewMessage(incoming=True, pattern=r"\.invitesall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

agora = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit(

"\*\*💰[Ͳєямιиαℓ Տτατυѕ]🤣(https://t.me/mm\_userbot)\*\*\n\n`🔸Inviting Users.......`"

)

async for user in event.client.iter\_participants(agora.full\_chat.id):

try:

await put(InviteToChannelRequest(channel=chat, users=[user.id]))

s = s + 1

await agora.edit(

f"🤟\*\*Inviting Users👇 \*\*\n\n\*\*⚜Invited :\*\* `{s}` users \n\*\*🔰Failed to Invite :\*\* `{f}` users.\n\n\*\*×Error :\*\* `{error}`"

)

except Exception as e:

error = str(e)

f = f + 1

return await agora.edit(

f"[τєямנиαℓ ƒιиιѕнє∂](https://t.me/Mm\_userbot) \n\n🔸 Sυϲϲєѕѕƒυℓℓγ ιиνιτє∂ `{s}` ρєορℓє \n⚠️ ƒαιℓє∂ το ιиνιτє `{f}` ρєορℓє"

)

@idk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@ydk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@wdk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@hdk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@sdk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@adk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@bdk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@cdk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@edk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@ddk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@vkk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@kkk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@lkk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@mkk.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@sid.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@shy.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@aan.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@ake.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@eel.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@khu.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@shi.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@yaa.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@dav.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@raj.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

@put.on(events.NewMessage(incoming=True, pattern=r"\.inviteall"))

async def get\_users(event):

if event.sender\_id in SMEX\_USERS:

sender = await event.get\_sender()

me = await event.client.get\_me()

if not sender.id == me.id:

text = "Processing...."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

else:

text = "processing.."

agora = await event.reply(text, parse\_mode=None, link\_preview=None )

aura = await get\_chatinfo(event)

chat = await event.get\_chat()

if event.is\_private:

return await agora.edit("`Sorry, Cant add users here`")

s = 0

f = 0

error = "None"

await agora.edit("\*\*TerminalStatus\*\*\n\n`Collecting Users.......`")

async for user in event.client.iter\_participants(aura.full\_chat.id):

try:

if error.startswith("Too"):

return await agora.edit(

f"\*\*Terminal Finished With Error\*\*\n(`May Got Limit Error from telethon Please try agin Later`)\n\*\*Error\*\* : \n`{error}`\n\n• Invited `{s}` people \n• Failed to Invite `{f}` people"

)

await event.client(

functions.channels.InviteToChannelRequest(channel=chat, users=[user.id])

)

s = s + 1

await agora.edit(

f"\*\*Terminal Running...\*\*\n\n• Invited `{s}` people \n• Failed to Invite `{f}` people\n\n\*\*× LastError:\*\* `{error}`"

)

except Exception as k:

error = str(k)

f = f + 1

await agora.edit(

f"\*\*Terminal Finished\*\* \n\n• Successfully Invited `{s}` people \n• failed to invite `{f}` people"

)

await asyncio.sleep(300)

await agora.delete()

@idk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@ydk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@wdk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@hdk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@sdk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@adk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@bdk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@cdk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@edk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@ddk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@vkk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@kkk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@lkk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@mkk.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@sid.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@shy.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@aan.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@ake.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@eel.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@khu.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@shi.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@yaa.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@dav.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@raj.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

@put.on(events.NewMessage(incoming=True, pattern=r"\.restart"))

async def restart(e):

if e.sender\_id in SMEX\_USERS:

text = "𝙍𝙚𝙨𝙩𝙖𝙧𝙩𝙚𝙙\n\nPlease wait till it reboots..."

await e.reply(text, parse\_mode=None, link\_preview=None )

try:

await idk.disconnect()

except Exception as e:

pass

try:

await ydk.disconnect()

except Exception as e:

pass

try:

await wdk.disconnect()

except Exception as e:

pass

try:

await hdk.disconnect()

except Exception as e:

pass

try:

await sdk.disconnect()

except Exception as e:

pass

try:

await adk.disconnect()

except Exception as e:

pass

try:

await bdk.disconnect()

except Exception as e:

pass

try:

await cdk.disconnect()

except Exception as e:

pass

try:

await ddk.disconnect()

except Exception as e:

pass

try:

await edk.disconnect()

except Exception as e:

pass

os.execl(sys.executable, sys.executable, \*sys.argv)

quit()

text = """

CONGRATS🥳🥳🥳 &THANKS TO PROFESSOR AGORA """

print(text)

print("")

print("🙏🔥🔥 BOT STARTED SUCCESFULLY.🔥🔥🙏")

if len(sys.argv) not in (1, 3, 4):

try:

idk.disconnect()

except Exception as e:

pass

try:

ydk.disconnect()

except Exception as e:

pass

try:

wdk.disconnect()

except Exception as e:

pass

try:

hdk.disconnect()

except Exception as e:

pass

try:

sdk.disconnect()

except Exception as e:

pass

try:

adk.disconnect()

except Exception as e:

pass

try:

bdk.disconnect()

except Exception as e:

pass

try:

cdk.disconnect()

except Exception as e:

pass

try:

edk.disconnect()

except Exception as e:

pass

try:

ddk.disconnect()

except Exception as e:

pass

try:

vkk.disconnect()

except Exception as e:

pass

try:

kkk.disconnect()

except Exception as e:

pass

try:

lkk.disconnect()

except Exception as e:

pass

try:

mkk.disconnect()

except Exception as e:

pass

try:

sid.disconnect()

except Exception as e:

pass

try:

shy.disconnect()

except Exception as e:

pass

try:

aan.disconnect()

except Exception as e:

pass

try:

ake.disconnect()

except Exception as e:

pass

try:

eel.disconnect()

except Exception as e:

pass

try:

khu.disconnect()

except Exception as e:

pass

try:

shi.disconnect()

except Exception as e:

pass

try:

yaa.disconnect()

except Exception as e:

pass

try:

dav.disconnect()

except Exception as e:

pass

try:

raj.disconnect()

except Exception as e:

pass

try:

put.disconnect()

except Exception as e:

pass

else:

try:

idk.run\_until\_disconnected()

except Exception as e:

pass

try:

ydk.run\_until\_disconnected()

except Exception as e:

pass

try:

wdk.run\_until\_disconnected()

except Exception as e:

pass

try:

hdk.run\_until\_disconnected()

except Exception as e:

pass

try:

sdk.run\_until\_disconnected()

except Exception as e:

pass

try:

adk.run\_until\_disconnected()

except Exception as e:

pass

try:

bdk.run\_until\_disconnected()

except Exception as e:

pass

try:

cdk.run\_until\_disconnected()

except Exception as e:

pass

try:

edk.run\_until\_disconnected()

except Exception as e:

pass

try:

ddk.run\_until\_disconnected()

except Exception as e:

pass

try:

vkk.run\_until\_disconnected()

except Exception as e:

pass

try:

kkk.run\_until\_disconnected()

except Exception as e:

pass

try:

lkk.run\_until\_disconnected()

except Exception as e:

pass

try:

mkk.run\_until\_disconnected()

except Exception as e:

pass

try:

sid.run\_until\_disconnected()

except Exception as e:

pass

try:

shy.run\_until\_disconnected()

except Exception as e:

pass

try:

aan.run\_until\_disconnected()

except Exception as e:

pass

try:

ake.run\_until\_disconnected()

except Exception as e:

pass

try:

eel.run\_until\_disconnected()

except Exception as e:

pass

try:

khu.run\_until\_disconnected()

except Exception as e:

pass

try:

shi.run\_until\_disconnected()

except Exception as e:

pass

try:

yaa.run\_until\_disconnected()

except Exception as e:

pass

try:

dav.run\_until\_disconnected()

except Exception as e:

pass

try:

raj.run\_until\_disconnected()

except Exception as e:

pass

try:

put.run\_until\_disconnected()

except Exception as e:

pass