STK INTEGRATING PYTHON

Autor: Celina Bossa

ÍNDICE

OBJETOS

ESCENARIO

Descripción

Proveé acceso a los métodos y propiedades respectivas del escenario de STK

Métodos

```
def change_time(InicialTime:typing.Any, FinalTime:typing.Any,
                StepTime:typing.Any)->None
def change_units(Variable:str, Unit:str)->None
```

Propiedades

```
ScName: str
InicialTm: str
FinalTm: str
StepTm: float
```

ESTACIÓN TERRENA

Descripción

Proveé acceso a los métodos y propiedades respectivas de la estación terrena.

Métodos

```
def new_GdSta(Name:str,Lat:float,Lon:float,Alt:float)->None
def setUseTerrainTrue(FacilityName:str)->None
```

Propiedades

```
CBA_GdStaName: str
CBA_GdStaLat: float
CBA_GdStaLon: float
CBA_GdStaAlt: float
POLAR_GdStaName: str
POLAR_GdStaLat: float
POLAR_GdStaLon: float
POLAR_GdStaAlt: float
```

RECEPTOR

Descripción

Proveé acceso a los métodos y propiedades respectivas del receptor de las estaciones terrenas

Métodos

```
def setRecDemodulation(FacilityName:str, ReceptorName:str,
                       Demodulation:str)->None
def setRecGainOverT(FacilityName:str,ReceptorName:str,GT:float)->None
```

Propiedades

```
CBA_RecName: str
RecType: str
POLAR_RecName: str
DemOptions: list[4] = [str,str,str,str,str]
Dem: DemOptions[x]
```

SATÉLITE

Descripción

Proveé acceso a los métodos y propiedades respectivas del satélite

Métodos

```
def new_Satellite(SatelliteName:str,StepTime:float)->None
def boolean_AutoUpdateEnabled(SatelliteName:str,state:boolean)->None
def getAttAvailableRefAcex(SatelliteName:str)-> list
def setAttReferenceAxes(SatelliteName:str,referece:str)->None
def setYPR(SatelliteName:str,Yaw:float,Pitch:float,Roll:float)->None
def setSaMass(SatelliteName:str,Mass:float)->None
```

Propiedades

```
SaName: str
```

ANTENA

Descripción

Proveé acceso a los métodos y propiedades respectivas de la antena del satélite.

Métodos

```
def setDiameterAnt(SatelliteName:str, AntennaName:str,
                   Diemater:float) ->None
def setFrecuencyAnt(SatelliteName:str, AntennaName:str,
                    Frecuency:float)->None
def setAzimuthElevation (SatelliteName:str ,AntennaName:str, Azimuth:float,
                         Elevation:float) -> None
```

Propiedades

```
AntName: str
ElvOptions: list[4] = [float,float,float,float,float]
Elv: ElvOptions[x]
```

TRANSMISOR

Descripción

Proveé acceso a los métodos y propiedades respectivas del transmisor del satélite.

Métodos

```
def new_Transmitter(SatelliteName:str,TransmitterName:str)->None
def setTraDemodulation(SatelliteName:str,TransmitterName:str,
                       Demodulation:str)->None
def setTraFrecuency(SatelliteName:str,TransmitterName:str,
                    Frecuency:float)->None
def setTraPower(SatelliteName:str,TransmitterName:str, Power:float)->None
def setTraDataRate(SatelliteName:str,TransmitterName:str,
                   Data_Rate:float)->None
```

Propiedades

```
TraName: str
DemOptions: list[4] = [str,str,str,str,str]
Dem: DemOptions[x]
DataRateOptions: list[4] = [float,float,float,float,float]
DataRate: DataRateOptions[x]
```

FUNCIONES

```
def change_time(InicialTime:typing.Any, FinalTime:typing.Any,
                StepTime:typing.Any):
  Sc_STKObj          = root.CurrentScenario
  Sc_ScObj           = Sc_STKObj.QueryInterface(STKObjects.IAgScenario)
  Sc_ScObj.SetTimePeriod(InicialTime,FinalTime)
  Sc_ScObj.Animation.AnimStepValue = StepTime
  root.Rewind();


def change_units(Variable:str,Unit:str):
    root.UnitPreferences.Item(Variable).SetCurrentUnit(Unit)

def new_GdSta(Name:str,Lat:float,Lon:float,Alt:float):
  GdSta_STKObj       = root.CurrentScenario.Children.New(8, Name)
  GdSta_FaObj        = GdSta_STKObj.QueryInterface(STKObjects.IAgFacility)
  root.UnitPreferences.Item('LatitudeUnit').SetCurrentUnit('deg')
  root.UnitPreferences.Item('LongitudeUnit').SetCurrentUnit('deg')
  GdSta_FaObj.UseTerrain  = False
  GdSta_FaObj.Position.AssignGeodetic(Lat, Lon, Alt)


def setUseTerrainTrue(FacilityName:str):
  GdSta_STKObj       = root.CurrentScenario.Children.Item(FacilityName)
  GdSta_FaObj        = GdSta_STKObj.QueryInterface(STKObjects.IAgFacility)
  GdSta_FaObj.UseTerrain = 'True'

def setRecDemodulation(FacilityName:str, ReceptorName:str,
                       Demodulation:str):
  GdSta_STKObj       = root.CurrentScenario.Children.Item(FacilityName)
  Rec_STKObj         = GdSta_STKObj.Children.Item(ReceptorName)
  Rec_RecObj         = Rec_STKObj.QueryInterface(STKObjects.IAgReceiver)
  RecModel_ModObj    = Rec_RecObj.Model
  RecModel_SModObj   = RecModel_ModObj.QueryInterface(STKObjects.IAgReceiverModelSimple)
  RecModel_SModObj.AutoSelectDemodulator = False
  RecModel_SModObj.SetDemodulator(Demodulation)
  print("Make sure you have change the Transmitter's Demodulation too")


def setRecGainOverT(FacilityName:str,ReceptorName:str,GT:float):
  GdSta_STKObj     = root.CurrentScenario.Children.Item(FacilityName)
  Rec_STKObj       = GdSta_STKObj.Children.Item(ReceptorName)
```

```
  Rec_RecObj       = Rec_STKObj.QueryInterface(STKObjects.IAgReceiver)
  RecModel_ModObj  = Rec_RecObj.Model
  RecModel_SModObj = RecModel_ModObj.QueryInterface(STKObjects.IAgReceiverModelSimple)
  RecModel_SModObj.GOverT  = GT #dB/K


def new_Satellite(SatelliteName:str,StepTime:float):
  Sa_STKObj    = root.CurrentScenario.Children.New(18, SatelliteName)  #
eSatellite
  Sa_SaObj     = Sa_STKObj.QueryInterface(STKObjects.IAgSatellite)
  Sa_SaObj.SetPropagatorType(STKObjects.ePropagatorSGP4)
  Prop_PropObj = Sa_SaObj.Propagator
   Prop_SGP4Obj = Prop_PropObj.QueryInterface(STKObjects.IAgVePropagatorSGP4)
   Prop_SGP4Obj.EphemerisInterval.SetImplicitInterval(root.CurrentScenario.Vgt.Ev
   entIntervals.Item("AnalysisInterval"))  # Link to scenario period
  Prop_SGP4Obj.Step  = StepTime
  Prop_SGP4Obj.AutoUpdateEnabled = False
  Prop_SGP4Obj.Propagate()
  print("You can upload the TLE")


def boolean_AutoUpdateEnabled(SatelliteName:str,state:boolean):
  Sa_STKObj    = root.CurrentScenario.Children.Item(SatelliteName)
  Sa_SaObj     = Sa_STKObj.QueryInterface(STKObjects.IAgSatellite)
  Prop_PropObj = Sa_SaObj.Propagator
  Prop_SGP4Obj = Prop_PropObj.QueryInterface(STKObjects.IAgVePropagatorSGP4)
  Prop_SGP4Obj.AutoUpdateEnabled = state
  Prop_SGP4Obj.Propagate()


def getAttAvailableRefAcex(SatelliteName:str):
  Sa_STKObj  = root.CurrentScenario.Children.Item(SatelliteName)
  Sa_SaObj   = Sa_STKObj.QueryInterface(STKObjects.IAgSatellite)
  Att_AttObj = Sa_SaObj.Attitude
   Att_OrbitAttStdObj=Att_AttObj.QueryInterface(STKObjects.IAgVeOrbitAttitudeStandard)
  Att_BasicObj = Att_OrbitAttStdObj.Basic
  Att_ProfObj  = Att_BasicObj.Profile
   Att_FIAObj   = Att_ProfObj.QueryInterface(STKObjects.IAgVeProfileFixedInAxes)
  return (Att_FIAObj.AvailableReferenceAxes)


def setAttReferenceAxes(SatelliteName:str,referece:str):
  Sa_STKObj      = root.CurrentScenario.Children.Item(SatelliteName)
  Sa_SaObj       = Sa_STKObj.QueryInterface(STKObjects.IAgSatellite)
```

```
  Att_AttObj      = Sa_SaObj.Attitude
  Att_OrbitAttStdObj=Att_AttObj.QueryInterface(STKObjects.IAgVeOrbitAttitudeStandard)
  Att_BasicObj    = Att_OrbitAttStdObj.Basic
  Att_ProfObj     = Att_BasicObj.Profile
  Att_FIAObj      = Att_ProfObj.QueryInterface(STKObjects.IAgVeProfileFixedInAxes)
  Att_FIAObj.ReferenceAxes = referece


def setYPR(SatelliteName:str,Yaw:float,Pitch:float,Roll:float):
  Sa_STKObj           = root.CurrentScenario.Children.Item(SatelliteName)
  Sa_SaObj            = Sa_STKObj.QueryInterface(STKObjects.IAgSatellite)
  Att_AttObj          = Sa_SaObj.Attitude
  Att_OrbitAttStdObj=Att_AttObj.QueryInterface(STKObjects.IAgVeOrbitAttitudeStandard)
  Att_BasicObj        = Att_OrbitAttStdObj.Basic
  Att_ProfObj     = Att_BasicObj.Profile
  Att_FIAObj      = Att_ProfObj.QueryInterface(STKObjects.IAgVeProfileFixedInAxes)
  Att_OrintObj    = Att_FIAObj.Orientation
  Att_OrintObj.AssignYPRAngles(4,Yaw,Pitch,Roll) #YPR sequence


def setSaMass(SatelliteName:str,Mass:float):
  Sa_STKObj             = root.CurrentScenario.Children.Item(SatelliteName)
  Sa_SaObj              = Sa_STKObj.QueryInterface(STKObjects.IAgSatellite)
  SaMass                = Sa_SaObj.MassProperties
  SaMass.Mass           = Mass

def setDiameterAnt(SatelliteName:str, AntennaName:str,
                   Diemater:float):
  Sa_STKObj             = root.CurrentScenario.Children.Item(SatelliteName)
  Ant_STKObj            = Sa_STKObj.Children.Item(AntennaName)
  Ant_AntObj            = Ant_STKObj.QueryInterface(STKObjects.IAgAntenna)
  Ant_AntModObj         = Ant_AntObj.Model
  Ant_AntSABObj=
Ant_AntModObj.QueryInterface(STKObjects.IAgAntennaModelApertureCircularBessel)
  Ant_AntSABObj.Diameter = Diemater #m

def setFrecuencyAnt(SatelliteName:str, AntennaName:str,
                    Frecuency:float):
  Sa_STKObj             = root.CurrentScenario.Children.Item(SatelliteName)
  Ant_STKObj            = Sa_STKObj.Children.Item(AntennaName)
  Ant_AntObj            = Ant_STKObj.QueryInterface(STKObjects.IAgAntenna)
  Ant_AntModObj     = Ant_AntObj.Model
  Ant_AntModObj.DesignFrequency = Frecuency #GHz
```

```
def setAzimuthElevation(SatelliteName:str ,AntennaName:str, Azimuth:float,
                        Elevation:float):
  Sa_STKObj             = root.CurrentScenario.Children.Item(SatelliteName)
  Ant_STKObj            = Sa_STKObj.Children.Item(AntennaName)
  Ant_AntObj            = Ant_STKObj.QueryInterface(STKObjects.IAgAntenna)
  Ant_OrintObj          = Ant_AntObj.Orientation
  Ant_OrintObj.AssignAzEl(Azimuth, Elevation, 1)  # 1 represents Rotate
About Boresight 'Value 0° = 1.27222e-14 °''


def new_Transmitter(SatelliteName:str,TransmitterName:str):
  Sa_STKObj           = root.CurrentScenario.Children.Item(SatelliteName)
  Sa_STKObj.Children.New(24, TransmitterName)

def setTraDemodulation(SatelliteName:str,TransmitterName:str,
                       Demodulation:str):
  Sa_STKObj         = root.CurrentScenario.Children.Item(SatelliteName)
  Tra_STKObj        = Sa_STKObj.Children.Item(TransmitterName)
  Tra_TraObj        = Tra_STKObj.QueryInterface(STKObjects.IAgTransmitter)
  TxModel_ModObj    = Tra_TraObj.Model
  TxModel_CmxModObj  =
TxModel_ModObj.QueryInterface(STKObjects.IAgTransmitterModelComplex)
  TxModel_CmxModObj.SetModulator(Demodulation)
  if Demodulation == DemOptions[0]:
        DataRate = DataRateOptions[0]
  elif Demodulation == DemOptions[1]:
        DataRate = DataRateOptions[1]
  elif Demodulation ==  DemOptions[2]:
        DataRate = DataRateOptions[2]
  elif Demodulation ==  DemOptions[3]:
        DataRate = DataRateOptions[3]
  elif Demodulation ==  DemOptions[4]:
        DataRate = DataRateOptions[4]
  CBAtxModel_CmxModObj.DataRate = DataRate  # Mb/sec
  print("Make sure you have change the Receiver's Demodulation too")

def setTraFrecuency(SatelliteName:str,TransmitterName:str,
                      Frecuency:float):
  Sa_STKObj       = root.CurrentScenario.Children.Item(SatelliteName)
  Tra_STKObj      = Sa_STKObj.Children.Item(TransmitterName)
  Tra_TraObj      = Tra_STKObj.QueryInterface(STKObjects.IAgTransmitter)
```

```
  TxModel_ModObj         = Tra_TraObj.Model
  TxModel_CmxModObj   =
TxModel_ModObj.QueryInterface(STKObjects.IAgTransmitterModelComplex)
  TxModel_CmxModObj.Frequency = Frecuency # GHz


def setTraPower(SatelliteName:str,TransmitterName:str, Power:float):
  Sa_STKObj          = root.CurrentScenario.Children.Item(SatelliteName)
  Tra_STKObj         = Sa_STKObj.Children.Item(TransmitterName)
  Tra_TraObj         = Tra_STKObj.QueryInterface(STKObjects.IAgTransmitter)
  TxModel_ModObj     = Tra_TraObj.Model
  TxModel_CmxModObj   =
TxModel_ModObj.QueryInterface(STKObjects.IAgTransmitterModelComplex)
  TxModel_CmxModObj.Power = Power # dBW

def setTraDataRate(SatelliteName:str,TransmitterName:str,
                   Data_Rate:float):
  Sa_STKObj          = root.CurrentScenario.Children.Item(SatelliteName)
  Tra_STKObj         = Sa_STKObj.Children.Item(TransmitterName)
  Tra_TraObj         = Tra_STKObj.QueryInterface(STKObjects.IAgTransmitter)
  TxModel_ModObj     = Tra_TraObj.Model
  TxModel_CmxModObj   =
TxModel_ModObj.QueryInterface(STKObjects.IAgTransmitterModelComplex)
  TxModel_CmxModObj.DataRate = Data_Rate # Mb/sec
  print("Make sure that you select the demodulation you want")
```

DEFINICIÓN DE PROPIEDADES

ScName

No necesita seguir un formato en particular. Por ejemplo: `ScName = 'Paper'`

InicialTm

La variable deberá ser de tipo str y temporalmente deberá ser anterior a la fecha definida en FinalTm. Por ejemplo:
```
InicialTm =  '18 May 2022 09:21:00.000'
InicialTm =  'Today'
InicialTm =  'Tomorrow'
```

FinalTm

La variable deberá ser de tipo str y temporalmente deberá ser posterior a la fecha definida en InicialTm. Por ejemplo:
```
FinalTm =  '3 Jun 2022 23:32:30.830'
FinalTm =  'Today'
FinalTm =  '+3 days'
FinalTm =  '+5 hours'
FinalTm =  '+1 week'
```

StepTm

La variable deberá ser de tipo float y definirá cada cuanto tiempo se tomará un dato de la simulación. Por ejemplo:
```
StepTm  = 8
StepTm  = 3.56
```

CBA/POLAR_GdStaName

No necesita seguir un formato en particular. Por ejemplo:
```
CBA_GdStaName   = 'CordBS'
POLAR_GdStaName = 'polarBS'
```

CBA/POLAR_GdStaLat

No necesita seguir un formato en particular. Por ejemplo:
```
CBA_GdStaLat   = -31.4343
POLAR_GdStaLat = -90
```

CBA/POLAR_GdStaLon

No necesita seguir un formato en particular. Por ejemplo:
```
CBA_GdStaLon   = -64.2672
POLAR_GdStaLon = -90
```

CBA/POLAR_GdStaAlt

No necesita seguir un formato en particular. Por ejemplo:
```
CBA_GdStaAlt   = 0
POLAR_GdStaAlt = 0
```

CBA/POLAR_RecName

No necesita seguir un formato en particular. Por ejemplo: `CBA_RecName = 'Receiver2'`
`POLAR_RecName = 'Receiver3'`

RecType

La variable acepta como parámetros las siguientes opciones: `'Cable Receiver Model'`, `'Complex Receiver Model'`, `'Laser Receiver Model'`, `'Medium Receiver Model'`, `'Multibeam Receiver Model'`, `'Script Plugin Laser Receiver Model'`, `'Script Plugin RF Receiver Model'` y `'Simple Receiver Model'`. Por el momento solo se encuentra habilitada la configuración por código del tipo `'Simple Receiver Model'`, para otras opciones se deberá realizar la configuración manual. Por ejemplo: `RecType = 'Simple Receiver Model'`

DemOptions

Debe definirse como una lista de str de longitud 4 que contenga los 5 tipos de modulación que desea simular. Acepta como parámetros de demodulaciones las siguientes opciones: `'16PSK'`, `'8PSK'`, `'BOC'`, `'BPSK'`, `'BPSK-BCH-127-64'`, `'BPSK-BCH-255-123'`, `'BPSK-BCH-511-259'`, `'BPSK-BCH-63-30'`, `'BPSK-Conv-2-1-6'`, `'BPSK-Conv-2-1-8'`, `'BPSK-Conv-3-1-6'`, `'BPSK-Conv-3-2-3'`, `'BPSK-Conv-3-2-8'`, `'BPSK-Conv-4-3-6'`, `'BPSK-Conv-4-3-6'`, `'BPSK-Conv-4-3-8'`, `'DPSK'`,`'External'`, `'FSK'`, `'MSK'`, `'Narrowband Uniform'`, `'NFSK'`, `'NFSK-BCH-127-92'`, `'NFSK-BCH-255-192'`, `'NFSK-BCH-511-385'`, `'NFSK-BCH-63-45'`, `'OQPSK'`, `'Pulsed Signal'`, `'QAM1024'`, `'QAM128'`, `'QAM16'`, `'QAM256'`, `'QAM32'`, `'QAM62'`, `'QPSK'`, `'Script'`, `'Wideband Gaussian'` y `'Wideband Uniform'`.

Por ejemplo: `DemOptions= ['QPSK','8PSK','16PSK','QAM16','QAM32']`

Dem

Se define como una de las 5 demodulaciones de DemOptions. Por ejemplo: `Dem = DemOptions[0]`

SaName

No necesita seguir un formato en particular. Por ejemplo: `SaName = 'Saocom-1-B'`

AntName

No necesita seguir un formato en particular. Por ejemplo: `AntName = 'SAOCOMantenna'`

ElvOptions

Debe definirse como una lista de floats de longitud 4 que contenga los 5 posicionamientos de antena (Elevation) que desea simular. Acepta como parámetros valores entre -1.57079633 rad a 1.57079633 rad, ó, -90° a 90°. Por ejemplo: `ElvOptions = [-65,-32.5,0,32.5,65]`

`Elv`

Se define como una de las 5 demodulaciones de ElvOptions. Por ejemplo: `Elv = ElvOptions[0]`

`TraName`

No necesita seguir un formato en particular. Por ejemplo: TraName = 'Transmitter2'

`DataRateOptions`

Debe definirse como una lista de floats de longitud 4 que contenga los 5 valores de Data Rate. El orden en el que se define el Data Rate debe estar en concordancia con el realizado en DemOptions para obtener el ancho de banda deseado. Por ejemplo: DataRateOptions = [2,3,4,4,5]

`DataRate`

Se define como una de los 5 Data Rate de DataRateOptions

INTERACCIÓN CON STK

MÉTODOS

```
def report()->None
def single_report(Demodulation:str, Angle:float)->None
def getAccessTimeData(ReferenceObject:IAgStkObject,
                      ObjectToAccess:IAgStkObject,
                      element:'Stop Time'ó 'Start Time')->list
def getTmIntervals(ReferenceObject:IAgStkObject,
                   ObjectToAccess:IAgStkObject, StartTime:typing.Any,
                   StopTime:typing.Any, StepTime:typing.Any)->list
def getTmData(ReferenceObject:IAgStkObject, ObjectToAccess:IAgStkObject,
              ItemName:str, GroupName:str,StartTime:typing.Any,
              StopTime:typing.Any, StepTime:typing.Any, element:str)->list
def getTmRealData(ReferenceObject:IAgStkObject,
                  ObjectToAccess:IAgStkObject, ItemName:str, GroupName:str,
                  StartTime:typing.Any, StopTime:typing.Any,
                  StepTime:typing.Any, elements:list=[str])->list
def getStaticData(ReferenceObject:IAgStkObject,ItemName:str,GroupName:str,
                   element:str)->list
def Step(ReferenceObject:IAgStkObject, ObjectToAccess:IAgStkObject,
         ItemNameList:list=[str], GroupNameList:list=[str],
         StartTime:typing.Any, StopTime:typing.Any,StepTime:typing.Any,
         elementsList:list=[list=[str]],SatelliteName:str,
         TransmitterName:str,FacilityName:str, ReceptorName:str,
    AntennaName:str,Demodulation:str, Angle:float, Azimuth:float)->list
```

report

Genera los 25 reportes correspondientes a las 5 modulaciones y los 5 posicionamientos de antena definidos. **Debe ejecutarse una vez configurado todo el escenario.**

FUNCIÓN

```
def report():
    for modulation in range(len(DemOptions)):
        for angle in range(len(ElvOptions)):
            single_report(DemOptions[modulation],ElvOptions[angle])
    print('Done')
```

single_report

Genera un reporte correspondiente a la modulación y posicionamiento de antena indicados en los parámetros de la función. **Debe ejecutarse una vez configurado todo el escenario.**

FUNCIÓN

```
def single_report(Demodulation:srt, Angle:float):
    Dem = Demodulation
    Elv = Angle
    if Demodulation == DemOptions[0]:
        DataRate = DataRateOptions[0]
    elif Demodulation == DemOptions[1]:
        DataRate = DataRateOptions[1]
    elif Demodulation ==  DemOptions[2]:
        DataRate = DataRateOptions[2]
    elif Demodulation ==  DemOptions[3]:
        DataRate = DataRateOptions[3]
    elif Demodulation ==  DemOptions[4]:
        DataRate = DataRateOptions[4]
    CBArecModel_SModObj.SetDemodulator(Dem)
    POLARrecModel_SModObj.SetDemodulator(Dem)
    CBAtxModel_CmxModObj.DataRate = DataRate  # Mb/sec
    SAOCOMant_OrintObj.AssignAzEl(0, Elv, 1)
    CBAtxModel_CmxModObj.SetModulator(Dem)
    access = CBArec_STKObj.GetAccessToObject(CBAtra_STKObj)
    access.ComputeAccess()
    AccessData        = access.DataProviders.Item('Access Data')
    AccessData_ProvG  =
AccessData.QueryInterface(STKObjects.IAgDataPrvInterval)
    AccessData_results        =
AccessData_ProvG.Exec(scenario_ScObj.StartTime, scenario_ScObj.StopTime)
    accessStartTime = AccessData_results.DataSets.GetDataSetByName('Start
Time').GetValues()
    accessStopTime  = AccessData_results.DataSets.GetDataSetByName('Stop
Time').GetValues()
    #print(accessStartTime, accessStopTime)

     ######################################
    ##    Task 7
    ##    1. Retrive and view the altitud of the satellite during an access
interval.

    ##Data provider de AER Data -> Default -> Azimuth - Elevation - Range
    AERdata                 = access.DataProviders.Item('AER Data')
```

```
    AERdata_GroupObj        =
AERdata.QueryInterface(STKObjects.IAgDataProviderGroup)
    AERdata_DataObj         = AERdata_GroupObj.Group
    AERdata_Default         = AERdata_DataObj.Item('Default')
    AERdata_TimeVar         =
AERdata_Default.QueryInterface(STKObjects.IAgDataPrvTimeVar)
    AERdata_elements        = ['Access Number', 'Time', 'Azimuth',
'Elevation', 'Range']
    accessTime              = []
    accessAccessNumber      = []
    accessAzimuth           = []
    accessElevation         = []
    accessRange             = []
    for i in range(len(accessStartTime)):
        AERdata_results         =
AERdata_TimeVar.ExecElements(accessStartTime[i],accessStopTime[i],StepTm,AE
Rdata_elements)
        Time =
list(AERdata_results.DataSets.GetDataSetByName('Time').GetValues())
        AccessNumber =
list(AERdata_results.DataSets.GetDataSetByName('Access
Number').GetValues())
        Azimuth =
list(AERdata_results.DataSets.GetDataSetByName('Azimuth').GetValues())
        Elevation =
list(AERdata_results.DataSets.GetDataSetByName('Elevation').GetValues())
        Range =
list(AERdata_results.DataSets.GetDataSetByName('Range').GetValues())
        for j in range (len(AccessNumber)):
            accessTime.append(Time[j])
            accessAccessNumber.append(AccessNumber[j])
            accessAzimuth.append(round(Azimuth[j],3))
            accessElevation.append(round(Elevation[j],3))
            accessRange.append(round(Range[j],6))

    ##Data provider de To Position Velocity -> ICRF -> x - y - z - xVel -
yVel - zVel - RelSpeed
    ToPositionVel           = access.DataProviders.Item('To Position
Velocity')
    ToPositionVel_GroupObj  =
ToPositionVel.QueryInterface(STKObjects.IAgDataProviderGroup)
    ToPositionVel_DataObj   = ToPositionVel_GroupObj.Group
    ToPositionVel_ICRF      = ToPositionVel_DataObj.Item('ICRF')
```

```
    ToPositionVel_TimeVar   =
ToPositionVel_ICRF.QueryInterface(STKObjects.IAgDataPrvTimeVar)
    ToPositionVel_elements  = ['x', 'y', 'z', 'xVel', 'yVel', 'zVel',
'RelSpeed']
    accessX                 = []
    accessY                 = []
    accessZ                 = []
    accessXVel              = []
    accessYVel              = []
    accessZVel              = []
    accessRelSpeed          = []
    for i in range(len(accessStartTime)):
        ToPositionVel_results   =
ToPositionVel_TimeVar.ExecElements(accessStartTime[i],accessStopTime[i],Ste
pTm,ToPositionVel_elements)
        X =
list(ToPositionVel_results.DataSets.GetDataSetByName('x').GetValues())
        Y =
list(ToPositionVel_results.DataSets.GetDataSetByName('y').GetValues())
        Z =
list(ToPositionVel_results.DataSets.GetDataSetByName('z').GetValues())
        XVel =
list(ToPositionVel_results.DataSets.GetDataSetByName('xVel').GetValues())
        YVel =
list(ToPositionVel_results.DataSets.GetDataSetByName('yVel').GetValues())
        ZVel =
list(ToPositionVel_results.DataSets.GetDataSetByName('zVel').GetValues())
        RelSpeed =
(ToPositionVel_results.DataSets.GetDataSetByName('RelSpeed').GetValues())
        for j in range(len(X)):
            accessX.append(round(X[j],6))
            accessY.append(round(Y[j],6))
            accessZ.append(round(Z[j],6))
            accessXVel.append(round(XVel[j],6))
            accessYVel.append(round(YVel[j],6))
            accessZVel.append(round(ZVel[j],6))
            accessRelSpeed.append(round(RelSpeed[j],6))

    ##Data provider de Link Information -> Prop Loss - EIRP - Rcvd.
Frequency - Freq. Doppler Shift -
    #                                     - Bandwidth Overlap - Rcvd.
Iso. Power - Flux Density -
```

```
    #                                           - g/T - C/No - Bandwidth - C/N
- Spectral Flux Density -
    #                                           - Eb/No - BER
    LinkInfo                = access.DataProviders.Item('Link Information')
    LinkInfo_TimeVar        =
LinkInfo.QueryInterface(STKObjects.IAgDataPrvTimeVar)
    LinkInfo_elements       = ['Prop Loss', 'EIRP', 'Rcvd. Frequency',
'Freq. Doppler Shift', 'Bandwidth Overlap','Rcvd. Iso. Power', 'Flux
Density', 'g/T', 'C/No', 'Bandwidth', 'C/N', 'Spectral Flux Density',
'Eb/No','BER']
    accessPropLoss          = []
    accessEIRP              = []
    accessRcvdFrequency     = []
    accessFreqDopplerShift  = []
    accessBandwidthOverlap  = []
    accessRcvdIsoPower      = []
    accessFluxDensity       = []
    accessgT                = []
    accessCNo               = []
    accessBandwidth         = []
    accessCN                = []
    accessSpectralFluxDensity = []
    accessEbNo              = []
    accessBER               = []
    for i in range(len(accessStartTime)):
        LinkInfo_results        =
LinkInfo_TimeVar.ExecElements(accessStartTime[i],accessStopTime[i],StepTm,L
inkInfo_elements)
        PropLoss = list(LinkInfo_results.DataSets.GetDataSetByName('Prop
Loss').GetValues())
        EIRP =
list(LinkInfo_results.DataSets.GetDataSetByName('EIRP').GetValues())
        RcvdFrequency =
list(LinkInfo_results.DataSets.GetDataSetByName('Rcvd.
Frequency').GetValues())
        FreqDopplerShift =
list(LinkInfo_results.DataSets.GetDataSetByName('Freq. Doppler
Shift').GetValues())
        BandwidthOverlap =
list(LinkInfo_results.DataSets.GetDataSetByName('Bandwidth
Overlap').GetValues())
```

```
        RcvdIsoPower =
list(LinkInfo_results.DataSets.GetDataSetByName('Rcvd. Iso.
Power').GetValues())
        FluxDensity = list(LinkInfo_results.DataSets.GetDataSetByName('Flux
Density').GetValues())
        gT =
list(LinkInfo_results.DataSets.GetDataSetByName('g/T').GetValues())
        CNo =
list(LinkInfo_results.DataSets.GetDataSetByName('C/No').GetValues())
        Bandwidth =
list(LinkInfo_results.DataSets.GetDataSetByName('Bandwidth').GetValues())
        CN =
list(LinkInfo_results.DataSets.GetDataSetByName('C/N').GetValues())
        SpectralFluxDensity =
list(LinkInfo_results.DataSets.GetDataSetByName('Spectral Flux
Density').GetValues())
        EbNo =
list(LinkInfo_results.DataSets.GetDataSetByName('Eb/No').GetValues())
        BER =
list(LinkInfo_results.DataSets.GetDataSetByName('BER').GetValues())
        for j in range (len(BER)):
            accessPropLoss.append(round(PropLoss[j],4))
            accessEIRP.append(round(EIRP[j],3))
            accessRcvdFrequency.append(round(RcvdFrequency[j],3))
            accessFreqDopplerShift.append(round(FreqDopplerShift[j],3))
            accessBandwidthOverlap.append(round(BandwidthOverlap[j],4))
            accessRcvdIsoPower.append(round(RcvdIsoPower[j],3))
            accessFluxDensity.append(round(FluxDensity[j],6))
            accessgT.append(round(gT[j],6))
            accessCNo.append(round(CNo[j],6))
            accessBandwidth.append(round(Bandwidth[j],3))
            accessCN.append(round(CN[j],4))

accessSpectralFluxDensity.append(round(SpectralFluxDensity[j],6))
            accessEbNo.append(round(EbNo[j],4))
            accessBER.append(round(BER[j],6))

    accessModulation        = []
    accessAntAngle          = []
    for i in range(len(accessTime)):
        accessModulation.append(Dem)
        accessAntAngle.append(str(Elv))
```

```
    import pandas as pd
    tabla = {
                "Access Number": accessAccessNumber,
                'Time (UTCG)': accessTime,
                  'Modulation': accessModulation,
                  'Angulo Antenna' : accessAntAngle,
                'Azimuth (deg)': accessAzimuth,
                  'Elevation (deg)': accessElevation,
                'Range (km)': accessRange,
                  'x (km)': accessX,
                  'y (km)': accessY,
                  'z (km)': accessZ,
                  'xVel (km/sec)': accessXVel,
                  'yVel (km/sec)': accessYVel,
                  'zVel (km/sec)': accessZVel,
                  'RelSpeed (km/sec)': accessRelSpeed,
                  'Prop Loss (dB)': accessPropLoss,
                  'EIRP (dBW)': accessEIRP,
                  'Rcvd. Frequency (GHz)': accessRcvdFrequency,
                  'Freq. Doppler Shift (GHz)': accessFreqDopplerShift,
                  'Bandwidth Overlap (dB)': accessBandwidthOverlap,
                  'Rcvd. Iso. Power (dBW)': accessRcvdIsoPower,
                  'Flux Density (dBW/m^2)': accessFluxDensity,
                  'g/T (dB/K)': accessgT,
                  'C/No (dB*MHz)': accessCNo,
                  'Bandwidth (MHz)': accessBandwidth,
                  'C/N (dB)': accessCN,
                  'Spectral Flux Density (dBW*m^-2*Hz^-1)':
accessSpectralFluxDensity,
                  'Eb/No (dB)': accessEbNo,
                  'BER': accessBER,
    }

    reporte = pd.DataFrame(tabla)
    reporte.to_csv("Reporte_"+Dem+"_"+str(Elv)+".csv")
    reporte.to_excel("Reporte_"+Dem+"_"+str(Elv)+".xlsx")
```

getAccessTimeData

Devuelve una lista con los valores iniciales o finales (dependiendo si el argumento de la variable 'elemento' es `'Start Time'` o `'Stop Time'`) que se produjo el acceso.

Por ejemplo, si obtenemos:

```
accessStartTime = ['18 May 2022 10:42:01.759586608', '19 May 2022
02:12:50.532451501', '19 May 2022 03:53:07.820825650', ...]
accessStopTime = ['18 May 2022 10:51:39.810447213', '19 May 2022
02:23:27.053174090', '19 May 2022 04:05:51.273799883, ...]
```

Significa que el primer acceso se dio entre 18 May 2022 10:42:01.76 y el 18 May 2022 10:51:39.81, el segundo acceso de dió entre el 19 May 2022 02:12:50.53 y el 19 May 2022 02:23:27.05 y así sucesivamente.

Debe ejecutarse una vez configurado todo el escenario.

FUNCIÓN

```
getAccessTimeData(ReferenceObject:IAgStkObject,
                        ObjectToAccess:IAgStkObject, element:str):
#  element = ['Start Time','Stop Time']
  access = CBArec_STKObj.GetAccessToObject(CBAtra_STKObj)
  access.ComputeAccess()
  AccessData            = access.DataProviders.Item('Access Data')
  AccessData_ProvG      =
AccessData.QueryInterface(STKObjects.IAgDataPrvInterval)
  AccessData_results    = AccessData_ProvG.Exec(scenario_ScObj.StartTime,
scenario_ScObj.StopTime)
  accessTime            =
AccessData_results.DataSets.GetDataSetByName(element).GetValues()
  return accessTime
```

getTmIntervals

Devuelve una lista con instantes de tiempo distanciados por `StepTime` dentro del intervalo de acceso ([`StartTime:StopTime`]) para el cual se tomará una muestra de las variables del simulador. Los valores de `StartTime` y `StopTime` son los encontrados en la función getAccessTimeData.

FUNCIÓN

```
def getTmIntervals(ReferenceObject:IAgStkObject,
                   ObjectToAccess:IAgStkObject, StartTime:typing.Any,
                   StopTime:typing.Any, StepTime:typing.Any):
  Time = getTmData(ReferenceObject,ObjectToAccess,'AER
Data','Default',StartTime,StopTime,StepTime,'Time')
  return Time
```

getTmData

Devuelve una lista con la información de una variable dependiente del tiempo entre el intervalo [StartTime:StopTime].
Si se desea obtener la información para un step de la simulación primero deberá correr la función getAccessTimeData, luego deberá correr la función getTmIntervals y utilizar los instantes de tiempo de tiempo encontrados en la segunda función para obtener los datos siguiendo la siguiente lógica:

```
accessStartTime = getAccessTimeData(CBArec_STKObj,CBAtra_STKObj,'Start Time')
accessStopTime = getAccessTimeData(CBArec_STKObj,CBAtra_STKObj,'Stop Time')
Tiempo = getTmIntervals (CBArec_STKObj, CBAtra_STKObj, accessStartTime[0],
accessStopTime[0], StepTm)
Datos = getTmData(ReferenceObject:IAgStkObject, ObjectToAccess:IAgStkObject,
                  ItemName:str, GroupName:str,StartTime=Tiempo[i],
                  StopTime=Tiempo[i+1], StepTime:typing.Any, element:str)
```

Para no tener que cambiar manualmente la variable `i` se recomienda crear una variable global e ir incrementandola en 1 en cada step.

Para determinar el nombre del Item, del Grupo y del Elemento se debe seguir manualmente los siguientes pasos en el STK.

1. Click derecho en el objeto que se quiere realizar el acceso
2. Click izquierdo en “Access…”
3. En la parte inferior derecha clickeamos “Report & Graph Manager…”
4. Click derecho en “My Styles”
5. Click izquierdo en “New” -> “Report”
6. Enter

Se abrirá una ventana con “Data Providers”. Los ‘Item’ son los nombres que figuran de mayor jerarquía, los ‘Groups’ son las carpetas internas que tienen los ‘Item’ (si no tiene ‘Groups’ debe colocar la palabra None) y los ‘elements’ son las variables que figuran dentro de la carpeta del Group.

FUNCIÓN

```
def getTmData(ReferenceObject:IAgStkObject, ObjectToAccess:IAgStkObject,
                  ItemName:str, GroupName:str,StartTime:typing.Any,
                  StopTime:typing.Any, StepTime:typing.Any, element:str):
  access = ReferenceObject.GetAccessToObject(ObjectToAccess)
  access.ComputeAccess()
  Item              = access.DataProviders.Item(ItemName)
  if GroupName != None:
    GroupObj     = Item.QueryInterface(STKObjects.IAgDataProviderGroup)
    DataObj      = GroupObj.Group
    Item   = DataObj.Item(GroupName)
  TimeVar           = Item.QueryInterface(STKObjects.IAgDataPrvTimeVar)
  Elements          = [element]
  Results    = TimeVar.ExecElements(StartTime,StopTime,StepTime,Elements)
  Data       = list(Results.DataSets.GetDataSetByName(element).GetValues())
  return Data
```

getTmRealData

Devuelve una lista con la información de las variables dependientes del tiempo y pertenecientes al mismo Item y Grupo entre un intervalo [StartTime:StopTime].

Ver `getTmData` para instrucciones de uso.

FUNCIÓN

```
def getTmRealDatagetTmRealData(ReferenceObject:IAgStkObject,
                               ObjectToAccess:IAgStkObject, ItemName:str,
                               GroupName:str,StartTime:typing.Any,
                               StopTime:typing.Any,StepTime:typing.Any,
                               elements:list=[str]):
  DataList = []
  for i in range(len(elements)):
    element = elements[i]
    Data = getTmData(ReferenceObject,
ObjectToAccess,ItemName,GroupName,StartTime,StopTime,StepTime,element)
    TmREalData = round(Data[0],3)
    DataList.append(TmREalData)
  return DataList
```

getStaticData

Devuelve una lista con la información de una variable independiente del tiempo.

Ver el final de `getTmData` para instrucciones de cómo definir ItemName, GroupName y element.

FUNCIÓN

```
def getStaticData(ReferenceObject:IAgStkObject,ItemName:str,GroupName:str,
                  element:str):
  Item                  = ReferenceObject.DataProviders.Item(ItemName)
  if GroupName != None:
    GroupObj    = Item.QueryInterface(STKObjects.IAgDataProviderGroup)
    DataObj     = GroupObj.Group
    Item        = DataObj.Item(GroupName)
  DataPrvFixed          = Item.QueryInterface(STKObjects.IAgDataPrvFixed)
  Elements              = [element]
  Results               = DataPrvFixed.ExecElements(Elements)
       Data                                                              =
list(Results.DataSets.GetDataSetByName(element).GetValues())
  return Data
```

Step

Devuelve una lista o grupo de listas con la información de una, o más, variables dependientes del tiempo entre el intervalo [StartTime:StopTime].

Ver `getTmData` para instrucciones de uso.

Para la definición de las variables ItemNameList, GroupNameList y elementsList se debe seguir la siguiente lógica: el nombre del Item en la primera posición de ItemNameList debe estar en concordancia con la primera posición del nombre del Group GroupNameList y con la primera lista posición de la primer posición de elementsList.

Si se quiere obtener datos de un mismo Item pero de diferentes Grupos la definición de ItemNameList, GroupNameList y elementsList deberá ser, por ejemplo, como la siguiente:

```
ItemNameList = ['AER Data','AER Data']
GroupNameList = ['Default','BodyFixed']
elementsList = [['Access Number','Azimuth'],['Elevation','Range']]
```

Entonces del Item `'AER Data'`, Grupo `'Default'` leera los datos de `['Access Number','Azimuth']`. Y del Item `'AER Data'`, Grupo `'BodyFixed'` tomará los datos de `['Elevation','Range']`.

Otro ejemplo de otro caso sería:

```
ItemNameList = ['AER Data','To Position Velocity']
GroupNameList = ['Default','ICRF']
elementsList = [['Access Number','Azimuth','Elevation','Range'],['x', 'y', 'z', 'xVel', 'yVel', 'zVel', 'RelSpeed']]
```

Donde del Item `'AER Data'`, Grupo `'Default'` extraerá los datos `['Access Number','Azimuth','Elevation','Range']` y del Item `'To Position Velocity'`, Grupo `'ICRF'` extraerá los datos `['x', 'y', 'z', 'xVel', 'yVel', 'zVel', 'RelSpeed']`

FUNCIÓN

```
def Step(ReferenceObject:IAgStkObject, ObjectToAccess:IAgStkObject,
          ItemNameList:list=[str], GroupNameList:list=[str],
          StartTime:typing.Any, StopTime:typing.Any,StepTime:typing.Any,
          elementsList:list=[list=[str]],SatelliteName:str,
          TransmitterName:str,FacilityName:str, ReceptorName:str,
          AntennaName:str,Demodulation:str, Angle:float, Azimuth:float):
  setAzimuthElevation(SatelliteName,AntennaName,Azimuth,Angle) #Cambio
angulo
  setTraDemodulation(SatelliteName,TransmitterName, Demodulation)
  setRecDemodulation(FacilityName,ReceptorName, Demodulation)
```

```
  Data = []
  for i in range(len(ItemNameList)):
    ItemName = ItemNameList[i]
    GroupName = GroupNameList[i]
    elements = elementsList[i]
    DataList = getTmRealData(ReferenceObject,
ObjectToAccess,ItemName,GroupName,StartTime,StopTime,StepTime,elements)
    for j in range(len(DataList)):
      Data.append(DataList[j])
  return Data
```

ABREVIACIONES

Para simplificar y facilitar la detección de las variables se siguió la siguiente tabla para asignar a las variables. Algunas de las abreviaciones han sido tomadas de la documentación de *STK Programming Help*

GdSta	Ground Station
Rec	Receiver
Tra	Transmitter
Sa	Satellite
Sc	Scenario
Tm	Time
Ant	Antenna
El or Elev	Elevation
Lon	Longitude
Lat	Latitude
Alt	Altitude
Att	Attitude
Fa	Facility
Prop	Propagator

MODO DE USO

Correr por consola el siguiente código

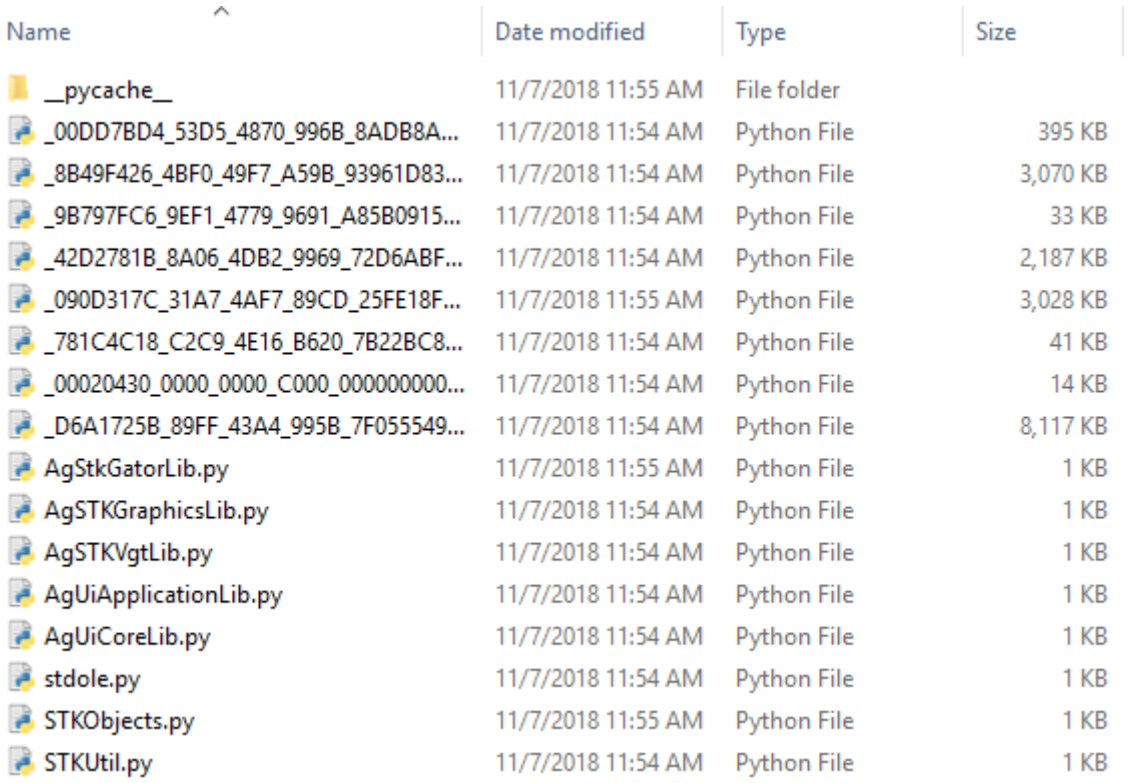
```
### Set the variables

#   Scenrio Properties
```

```
ScName                  = 'Paper'

######
##    Task 1
##    1. Set up your phyton workspace
from comtypes.client import CreateObject

##    2. Get reference to running STK instance
uiApplication = CreateObject('STK11.Application')
uiApplication.Visible = True
uiApplication.UsarControl=True

##    3. Get our IAgStkObjectRoot interface
root = uiApplication.Personality2

from comtypes.gen import STKObjects

#####################################
##    Task 2
##    1. Create a new scenario

root.NewScenario(ScName)
scenario_STKObj          = root.CurrentScenario
```

Si se muestra el error *"ImportError: cannot import name 'STKUtil' from 'comtypes.gen'"* probablemente haya un problema con los archivos de Python localizados en la carpeta comtypes gen. Ejecute el siguiente código para determinar la carpeta de comtypes gen:

```
import comtypes.client
print(comtypes.client.gen_dir, '\n')
```

Copie la ruta del archivo en la barra de direcciones del Explorador de archivos. Verá varios archivos; estos son los archivos que permiten que Python se comunique con STK (como se muestra a continuación).



Name	Date modified	Type	Size
__pycache__	11/7/2018 11:55 AM	File folder	
_00DD7BD4_53D5_4870_996B_8ADB8A...	11/7/2018 11:54 AM	Python File	395 KB
_8B49F426_4BF0_49F7_A59B_93961D83...	11/7/2018 11:54 AM	Python File	3,070 KB
_9B797FC6_9EF1_4779_9691_A85B0915...	11/7/2018 11:54 AM	Python File	33 KB
_42D2781B_8A06_4DB2_9969_72D6ABF...	11/7/2018 11:54 AM	Python File	2,187 KB
_090D317C_31A7_4AF7_89CD_25FE18F...	11/7/2018 11:55 AM	Python File	3,028 KB
_781C4C18_C2C9_4E16_B620_7B22BC8...	11/7/2018 11:54 AM	Python File	41 KB
_00020430_0000_0000_C000_000000000...	11/7/2018 11:54 AM	Python File	14 KB
_D6A1725B_89FF_43A4_995B_7F055549...	11/7/2018 11:54 AM	Python File	8,117 KB
AgStkGatorLib.py	11/7/2018 11:55 AM	Python File	1 KB
AgSTKGraphicsLib.py	11/7/2018 11:54 AM	Python File	1 KB
AgSTKVgtLib.py	11/7/2018 11:54 AM	Python File	1 KB
AgUiApplicationLib.py	11/7/2018 11:54 AM	Python File	1 KB
AgUiCoreLib.py	11/7/2018 11:54 AM	Python File	1 KB
stdole.py	11/7/2018 11:54 AM	Python File	1 KB
STKObjects.py	11/7/2018 11:55 AM	Python File	1 KB
STKUtil.py	11/7/2018 11:54 AM	Python File	1 KB

Si no puede conectarse a STK, es posible que algunos de los archivos, aquellos con un nombre de archivo que contiene un GUID (que parece una secuencia aleatoria de números y letras), no se hayan generado correctamente y probablemente estén vacíos. Puede verificar si este es el problema abriendo los archivos en un editor de texto y revisando el contenido. Vuelva a generar los archivos que estén vacíos.

Nota: en general, estos archivos siempre tendrán que regenerarse entre versiones principales (p. ej., STK 10, STK 11, etc.) de STK.

1. Para comenzar a regenerar cualquier archivo vacío, elimine todo el contenido de esta carpeta.
2. Luego, vuelve a ejecutar:

```
import comtypes
from comtypes.client import CreateObject
app = CreateObject('STK11.Application')  #  Possible error
app.Visible = True
root = app.Personality2 # Possible error
```

3. Cuando se ejecutan "app=CreateObject('STK11.Application')" y "root=uiApplication.Personality2", la biblioteca comtypes crea automáticamente la carpeta gen que contiene los contenedores de Python para las bibliotecas STK.
4. Debería ver varias salidas que dicen "# Generando comtypes.gen. xxxxx", y se completará el contenido de la carpeta gen.
5. Si está intentando acceder a la biblioteca de Astrogator, específicamente, también deberá ejecutar la siguiente línea:

```
comtypes.client.GetModule((comtypes.GUID("{090D317C-31A7-4AF7-89CD-25FE18
F4017C}") ,1,0))
```

6. Una vez que los archivos wrapper se han generado correctamente, se pueden importar directamente:

```
from comtypes.gen import STKObjects
from comtypes.gen import STKUtil
```

Si aún no puede conectarse a STK, verifique dos veces los permisos de la carpeta gen y asegúrese de que Propiedades -> Seguridad esté en Control total. Si intentar conectarse aún no funciona, puede descargar la carpeta zip adjunta para STK 11, extraer el contenido y pegarlo en su carpeta comtypes gen[1]

Una vez que pueda correr esa parte del código sin errores corra el código completo e interaccione con el programa con las funciones definidas previamente.

[1] Fuente: AGI
https://agiweb.secure.force.com/faqs/articles/Keyword/Having-Trouble-Connecting-to-STK-with-Python