```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# movoirAMIGAcone



import sampy
coordMessage = "coord.pointAt.sky"
coneSearchEndpoint='http://vizier.u-strasbg.fr/viz-bin/votable/-A?-source=J/A+A/472/121&'

amigaCSmd = {
    'samp.name': 'AMIGA isolation',
    'samp.description.text': 'coord.pointAt.sky client that provides a ConeSearch to the
AMIGA V. Isolation parameters (Verley+, 2007) at VizieR.',
    'movoir.version': '0.1',
    'samp.icon.url': 'http://thesaurus.maths.org/mmkb/media/png/Cone.png'
}

def returnConeSearchUrl(sender_id, ra, dec, sr):
    # We want to send a "table.load.votable" message
    # with the corresponding ConeSearch query URL
    csUrl = coneSearchEndpoint+"RA="+str(ra)+"&DEC="+str(dec)
    csUrl = csUrl + "&SR="+str(sr)
    print "CS URL:", csUrl
    tableLoadMsg = {'samp.mtype': 'table.load.votable', 'samp.params': {'url': csUrl,
'table-id': csUrl, 'name': 'AMIGA CS clickAt response'}}
    print "Mensaje:", tableLoadMsg
    senderMeta = amigaCSclient.getMetadata(sender_id)
    print "Sender_id:", sender_id, "App: ", senderMeta['samp.name']
    amigaCSclient.call(sender_id, "clickAMIGA", tableLoadMsg)



def notificationHandler(private_key, sender_id, mtype, params, extra):
    #pdb.set_trace()
    print ("notifHandler", private_key, sender_id, mtype, params, extra)
    if mtype == coordMessage:
        if params.has_key('ra') and params.has_key('dec'):
            ra = params['ra']
            dec = params['dec']
            sr = 0.5 # Half a degree
            if params.has_key('sr'):
                sr = params['sr']
            print sender_id, ra,dec,sr
            returnConeSearchUrl(sender_id, ra, dec, sr)



def callHandler(private_key, sender_id, msg_id, mtype, params, extra):
    # As this is a call, we should return an empty samp.result
    # after having processed it
    print ("callHandler: ", private_key, sender_id, msg_id, mtype, params, extra)
    if mtype == coordMessage:
        if params.has_key('ra') and params.has_key('dec'):
            ra = params['ra']
            dec = params['dec']
            sr = 0.5 # Half a degree
            if params.has_key('sr'):
                sr=params['sr']
            print sender_id, ra,dec,sr
            returnConeSearchUrl(sender_id, ra, dec, sr)
            replyMsg = {'samp.status': sampy.SAMP_STATUS_OK,
                        'samp.result': {}}
            amigaCSclient.reply(msg_id, replyMsg)
        else:
            print "In the error branch"
            replyMsg = {'samp.status': sampy.SAMP_STATUS_ERROR,
                        'samp.result': {},
                        'samp.error': {'samp.errortxt': 'No url parameter provided.'}}
            amigaCSclient.reply(sender_id, msg_id, replyMsg)



def responseHandler(private_key, sender_id, msg_id, response):
    # Response received, typically an OK response,
    # we should not be receiving none of these
    print ("Receiving response", private_key, sender_id, msg_id, response)



# Start binding behaviours
amigaCSclient = sampy.SAMPIntegratedClient(amigaCSmd)
amigaCSclient.connect()
amigaCSclient.bindReceiveNotification(coordMessage, notificationHandler)
amigaCSclient.bindReceiveCall(coordMessage, callHandler)
amigaCSclient.bindReceiveResponse("clickAMIGA", responseHandler)

#movoirAMIGAcone.amigaCSclient.disconnect()
```