pip install the following software modules:

astropy==6.0.0

geopy==2.4.1

numpy==1.26.4

pytest==8.0.0

requests==2.25.1

xmltodict==0.13.0

Then import them into Python:

```
import json
import math
import requests
import time
import xmltodict
from flask import Flask, request
from geopy.geocoders import Nominatim
from astropy import coordinates
from astropy import units
from astropy.time import Time
```

The following code loads the NASA data into a Python Dictionary:

```
data= {}


global data
data.clear()
url =
'https://nasa-public-data.s3.amazonaws.com/iss-coords/current/ISS_OEM
/ISS.OEM_J2K_EPH.xml'

response = requests.get(url)
data = xmltodict.parse(response.text)

print (data)
```

The following code takes the state vector and returns a dictionary of Longitude/Latitude/Altitude

```
def compute_location_astropy(sv:dict) -> dict:
    """
    Send this function a state vector (including epoch, x, y, z),
return location dict
    """
    x = float(sv['X']['#text'])
    y = float(sv['Y']['#text'])
    z = float(sv['Z']['#text'])
    # assumes epoch is in format '2024-067T08:28:00.000Z'
    this_epoch=time.strftime('%Y-%m-%d %H:%m:%S',
time.strptime(sv['EPOCH'][:-5], '%Y-%jT%H:%M:%S'))

    cartrep = coordinates.CartesianRepresentation([x, y, z],
unit=units.km)
    gcrs = coordinates.GCRS(cartrep, obstime=this_epoch)
    itrs = gcrs.transform_to(coordinates.ITRS(obstime=this_epoch))
    loc = coordinates.EarthLocation(*itrs.cartesian.xyz)

    lat=loc.lat.value
    lon=loc.lon.value
    alt=loc.height.value

    geoloc = geocoder.reverse((lat, lon), zoom=15, language='en')
    if geoloc != None:
        return {'location':{'latitude':lat, 'longitude':lon,
'altitude':{ 'value':alt, 'units':'km'},
'geo':geoloc.raw['address']}}
    else:
        return {'location':{'latitude':lat, 'longitude':lon,
'altitude':{ 'value':alt, 'units':'km'}, 'geo':'no data - perhaps
over ocean'}}
```

This bit of code, computes speed from the velocity vector:

```
def compute_speed(sv:dict) -> dict:
    """
    Send this function a state vector (including x_dot, y_dot,
z_dot), return speed dict
    """
    speed = math.sqrt( float(sv['X_DOT']['#text'])**2 +
                       float(sv['Y_DOT']['#text'])**2 +
```

```
                        float(sv['Z_DOT']['#text'])**2 )
    return {'speed':{'value':speed, 'units':'km/s'}}
```

This code prints a list of epochs() (getting rid of the comments)

```
def print_list_of_epochs() -> str:
    """
    Return a list of all Epochs in the data set
    """
    if not bool(data): # data is empty
        return json.dumps(data, indent=2)

    offset = request.args.get('offset', '0')
    limit = request.args.get('limit', '0')

    if not offset.isnumeric():
        return 'Error: offset must be an integer\n'
    if not limit.isnumeric():
        return 'Error: limit must be an integer\n'

    offset = int(offset)
    limit = int(limit)

    epoch_list = []
    counter = 0
    for sv in
data['ndm']['oem']['body']['segment']['data']['stateVector']:
        if offset > counter:
            counter += 1
            continue
        epoch_list.append(sv['EPOCH'])
        if len(epoch_list) == limit:
            return json.dumps(epoch_list, indent=2)
    return json.dumps(epoch_list, indent=2)
```