```
# -*- coding: utf-8 -*-
"""
Created on Tue July 6 17:12:44 2021

@author: Lindsay Kelly
Purpose: City of Vancouver Street Sweeping Data Assembly

"""
import json
import pytz
import pandas as pd
import numpy as np
import datetime as dt
import os
import sys
file_dir = os.path.dirname(__file__)
sys.path.append(file_dir)
import build_config as c
import geotab_testing as gps
import assemble_data as data
import map1 as m
import test_connections as t



pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)

cwd=os.getcwd()

utc = pytz.UTC
pst = pytz.timezone('America/Los_Angeles')

datetime_str_format = lambda d: pd.to_datetime(d).__format__('%Y-%m-%d %H:%M:%S')
datetime_dt_format = lambda d: pst.localize(dt.datetime.strptime(d, '%Y-%m-%d'))
datetime_dt_str = lambda d: dt.datetime.strftime(d,'%Y-%m-%d')

file_dir=lambda f, x: os.path.join(f,x)

spacer_start = '------------------------\n'
spacer_end = '\n------------------------'


def make_config(time_period,custom={}):
    """
    Parameters
    ----------
    time_period : (required) uses the time period variable to build the appropriate
calendar for the search parameters
    custom : (optional - default {}) builds any custom elements required within the
```

```
program, including filenames and variables

    Returns: config.json
    -------
    """
    inputs=c.config(time_period,output_file=True,custom_items=custom)
    file=inputs.build_config()
    return file


def user_input(geotab_permissions):
    if geotab_permissions==True:
        user_input_use_files=input("Do you want to use stored data files? (Y/N)
")[0].upper()
        if user_input_use_files=='Y':
            use_file=True
            print(spacer_start,'building project from stored files',spacer_end)
        else:
            use_file=False
            print(spacer_start,'building project from API connections',spacer_end)
    else:
        use_file=True
        print(spacer_start,'You do not have the username / password file for the
Geotab connecetion \nbuilding project from stored files',spacer_end)

    user_input_descriptive=input("Do you want to to see descriptive progress and
data previews? (Y/N) ")[0].upper()

    if user_input_descriptive=='Y':
        verbose_output=True
        print(spacer_start,'building project with descriptive outputs',spacer_end)
    else:
        verbose_output=False
        print(spacer_start,'building project with typical progress
descriptions',spacer_end)
    return use_file, verbose_output


def system_check():

    geotab = t.check_geotab()

    return geotab


### ------------------------------------------------------------------------
###

if __name__=='__main__':
```

```
    time_period="WEEKLY" ##"MONTHLY" "WEEKLY" "DAILY"

    custom={"filenames": {
        "all_trips": "all_trips",
        "file_type": ".xlsx",
        "onsite_trips": "onsite_trips"
        },
    "vehicle details":{"b466":"D1506",
                       "b4BD":"D1507",
                       "b464":"D1508",
                       "b60":"D1554",
                       "bBC":"D2408",
                       "bD":"E2461",
                       "bE":"E2462",
                       "b3A":"E2464",
                       "b3B":"E2465",
                       "b5C6":"F2466"
                       },
    'use file':False,
    'verbose':False,
    "folder_location": str(os.path.join(cwd,"data")),
    'trip_summary_json': "trip_summary.json",
    'trip_gps_json': "trip_gps.json",
    "exception_gps_json": "exception_gps.json",
    "device_status_json": "device_status.json",
    "cov_bikelanes_json": "cov_bikelanes.json",
    "cov_arterials_json": 'cov_arterials.json',
    "json_orient":"records",
    "street_shape_file": "COV_Streets.shp",
    "trip_shape_file": "street_sweeping_trips.shp",

"COV_open_data_api_key":"9f5582ff69d05faa9e1baba68656553fa2db03dd5bd32f3a3558c6b4",
    "time_period": time_period,
    "email_details":{'filter_name':"STREET SWEEPING"},

    "gps_rules":{1:{'rule':"street sweeper engaged",'id':'aC8HtvKrVzUq7CkhZ9P1VvA',
'description':'auxiliary vehicle equipment engaged - aux 1 (left broom), aux 2
(right broom), aux 4 (water)'},
                 2:{'rule':"street cleaning - seatbelt low
speed",'id':'aIRmwe_3EI06AzjPlg0Mm-Q','descripion': ' driver seatbelt unbuckled
while the vehicle is travelling under 30 km/hr for more than 250 meters'},
                 3:{'rule':'street sweeper - high speed
exception','id':'aXeLaht6Mb0OdmEhrDVl6YA','description':'auxiliary vehicle equipment
engaged while the vehicle is travelling at speeds above the acceptable limits (12
km/hr) for more than 20 seconds'}
                 },
    "cov_street_gis_data_url":
'https://opendata.vancouver.ca/api/records/1.0/search/?dataset=public-streets&q=&row
s=10000&facet=streetuse&refine.streetuse=Arterial',
    "cov_bikelane_gis_data_url":
```

```
"https://opendata.vancouver.ca/api/records/1.0/search/?dataset=bikeways&q=&rows=1000
0&facet=bike_route_name&facet=bikeway_type&facet=status&facet=aaa_network&facet=snow
_removal&facet=year_of_construction"
    }


    ########################################
    ## system check and build configuration
    geotab_permissions = system_check()
    use_file, verbose_output = user_input(geotab_permissions)

    custom['verbose'] = verbose_output
    custom['use file'] = use_file

    config_file = make_config(time_period,custom) ### builds configuration file from
common/config based on the reporting month
    params = json.load(open(config_file, 'r'))

    ############################################
    ### generate the dataset

    gps_data=gps.gps_data() ### generates the geotab connection string for all
geotab queries


    # 1) get the device list of gps data to be evaluated
    device_list=list(params["vehicle details"])
    device_status_df = data.get_device_info_data(params, device_list,
gps_conn=gps_data)

    if params['verbose']==True: print("current vehicle device gps
status\n",device_status_df.head())
    ## TODO: filter the device status to query only the devices in use recently

    # TODO conditional for the data source to be used - as GEOTAB API requires
username, password ad data permissions for vehicles
    #### if user permissions are not available the stored datafiles will be used for
the reporting

    # 2) datasets to be used

    ## summary trip details for the vehicle list provided
    trip_summary_df = data.get_trip_summary_data(params, device_list,
gps_conn=gps_data)
    ## gps data logs for each trip contained within the trips_summary df
    trip_df = data.get_trip_gps_data(params, trip_summary_df, gps_conn=gps_data)

    ## gps data for equipment sensor telemetry
    exception_df = data.get_exception_gps_data(params, device_list=[],
gps_conn=None)
```

```
    ## bikelane gis data
    bikelane_df = data.get_cov_bikelane_data(params)
    ## cov street gis data
    arterial_streets_df = data.get_cov_arterial_streets_data(params)

    # 3) assemble datasets
    ######################################################
    ## assemble trip summary with line segments built from the gps points
    print(spacer_end)
    print("geo_df : indivudual trip gps points combined to geo pandas linestring -
in order to display trips on a map")
    geo_df,trip_summary_df = gps.generate_trip_geom(trip_summary_df,trip_df, params)

    ######################################################
    ## assemble trip summary map based on line segments built from the gps points


    print(spacer_end)
    print("map of last week's street sweeping\nNOTE: this map is generated using
plotly - use your mouse cursor to zoom and pan within the map window")

    m.plot_sweeper_map(geo_df)
    # m.daily_summary_map(geo_df)
    print("All data access requests have completed")
```