election.py

```python
import matplotlib
import numpy as np
import pandas as pd
import re
import folium
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
from icecream import ic
from matplotlib import pyplot as plt, font_manager
from context.domains import Reader, File
import platform
import matplotlib.pyplot as plt


class Solution(Reader):
    def __init__(self):
        self.file = File()
        self.file.context = './data/'

    def hook(self):
        def print_menu():
            print('0. Exit')
            print('1. preprocess')
            print('2. draw_korea')
            print('3. draw_korea_geo')
            return input('      \n')

        while 1:
            menu = print_menu()
            if menu == '0':
                break
            if menu == '1':
                self.preprocess()
            if menu == '2':
                self.draw_korea()
            if menu == '3':
                self.draw_korea_geo()
            elif menu == '0':
                break

    def preprocess(self):
        file = self.file
        file.fname = 'election_result'
        election_result = self.csv(file)
        # ic(election_result.head())
        election_result = self.change_char_sido(election_result)
        election_result = self.calc_percent_vote(election_result)
        file.fname = 'draw_korea'
        draw_korea = self.csv(file)
        # draw_korea.head()
        self.create_final_data(draw_korea, election_result)

    def compare_percent_vote(self, final_elect_data):
        #
        final_elect_data['moon_vs_hong'] = final_elect_data['rate_moon'] - final_elect_data['rate_hong']
        #
        final_elect_data['moon_vs_ahn'] = final_elect_data['rate_moon'] - final_elect_data['rate_ahn']
        #
        final_elect_data['ahn_vs_hong'] = final_elect_data['rate_ahn'] - final_elect_data['rate_hong']
        ic(final_elect_data.head())
        '''
         Unnamed: 0  Unnamed: 0_x         ... moon_vs_hong  moon_vs_ahn  ahn_vs_hong
         0           0             0         ...    19.681961    19.693661    -0.011700
         1           1             1         ...    19.505866    17.730411     1.775455
         2           2             2         ...    15.423503    17.530053    -2.106549
         3           3             3         ...    22.699643    20.185554     2.514089
         4           4             4         ...    24.640253    21.950590     2.689664
        '''
        #
        #                         ,
        final_elect_data.sort_values(['moon_vs_hong'], ascending=[False]).head(10)
        '''
        Unnamed: 0  Unnamed: 0_x         ... moon_vs_hong  moon_vs_ahn  ahn_vs_hong
        181         181           182         ...    65.069909    45.282749    19.787160
        165         165           166         ...    63.958791    45.331283    18.627508
        164         164           165         ...    63.724549    45.320941    18.403608
        63          63            63         ...    62.644384    37.757293    24.887091
        171         171           172         ...    62.349928    43.476869    18.873059
        '''
        #   ,
        final_elect_data.sort_values(['moon_vs_hong'], ascending=[True]).head(10)
        '''
        Unnamed: 0  Unnamed: 0_x        ... moon_vs_hong  moon_vs_ahn  ahn_vs_hong
        218         218           219        ...  -53.327282     1.770012   -55.097294
        219         219           220        ...  -48.672566     1.579712   -50.252278
        222         222          223       ...  -47.954067     2.124402   -50.078469
        215         215          216       ...  -42.391498     5.152706   -47.544204
        212         212          213       ...  -42.342174     2.596190   -44.938364
        '''
        final_elect_data.to_csv('./save/final_elect_data.csv', index=False)
        return final_elect_data

    def create_final_data(self, draw_korea, election_result):
        # draw_korea  ID  election_result  ID
        #             ,                          .
        #           , set()         .
        set(draw_korea['ID'].unique()) - set(election_result['ID'].unique())
        set(election_result['ID'].unique()) - set(draw_korea['ID'].unique())

        # '  '    ,                        .
        # ic(election_result[election_result['ID'] == '  '])
        '''
         Unnamed: 0                pop    moon     hong     ahn  ID
         125    125            18692.0  5664.0   6511.0  3964.0
         233    233            34603.0  9848.0  16797.0  4104.0
        '''
        election_result.loc[125, 'ID'] = '  (  )'
        election_result.loc[233, 'ID'] = '  (  )'
        # ic(election_result[election_result['  '] == '   '])
        '''
        Unnamed: 0                pop    moon     hong     ahn      ID
         125   125            18692.0  5664.0   6511.0  3964.0    (  )
         233   233            34603.0  9848.0  16797.0  4104.0    (  )
        '''
        #            '  '      ,             '  '        .
        # ic(election_result[election_result['    '] == '    '])
        '''
        Unnamed: 0                      pop      moon     hong      ahn       ID
        228    228                 119281.0   35592.0  54488.0  14686.0
        229    229                 136757.0   45014.0  56340.0  17744.0
        '''
        election_result.loc[228, 'ID'] = '      '
        election_result.loc[229, 'ID'] = '      '
        # ic(election_result[election_result['    '] == '    '])
        '''
        Unnamed: 0                      pop      moon     hong      ahn      ID
        228    228                 119281.0   35592.0  54488.0  14686.0
        229    229                 136757.0   45014.0  56340.0  17744.0
        '''

        # draw_korea                     , election_result     .
        #                                          ,           '3'            .
        #  ,      'rate_moon', 'rate_hong', 'rate_ahn'  '3'             .

        # ic(election_result[election_result['  '] == '   '])
        '''
        Unnamed: 0                pop      moon      hong       ahn  ID
        85      85           543777.0  239697.0  100544.0  128297.0
        '''

        # '   '      '3(  ,   ,   )'
        ahn_tmp = election_result.loc[85, 'ahn'] / 3
        hong_tmp = election_result.loc[85, 'hong'] / 3
        moon_tmp = election_result.loc[85, 'moon'] / 3
        pop_tmp = election_result.loc[85, 'pop'] / 3

        #  ,      '3'
        rate_moon_tmp = election_result.loc[85, 'rate_moon']
        rate_hong_tmp = election_result.loc[85, 'rate_hong']
        rate_ahn_tmp = election_result.loc[85, 'rate_ahn']

        election_result.loc[250] = [250, '   ', '   ',
                                    pop_tmp, moon_tmp, hong_tmp, ahn_tmp, '      ',
                                    rate_moon_tmp, rate_hong_tmp, rate_ahn_tmp]
        election_result.loc[251] = [251, '   ', '   ',
                                    pop_tmp, moon_tmp, hong_tmp, ahn_tmp, '      ',
                                    rate_moon_tmp, rate_hong_tmp, rate_ahn_tmp]
        election_result.loc[252] = [252, '   ', '   ',
                                    pop_tmp, moon_tmp, hong_tmp, ahn_tmp, '      ',
                                    rate_moon_tmp, rate_hong_tmp, rate_ahn_tmp]

        #          '[85]   '      .
        election_result.drop([85], inplace=True)

        #       draw_korea  ID  election_result  ID                .
        set(draw_korea['ID'].unique()) - set(election_result['ID'].unique())
        set(election_result['ID'].unique()) - set(draw_korea['ID'].unique())

        # election_result  draw_korea  merge()                      .
        final_elect_data = pd.merge(election_result, draw_korea, how='left', on=['ID'])
        # ic(final_elect_data.head())
        final_elect_data = self.compare_percent_vote(final_elect_data)
    return final_elect_data

# ' ',' '                        ,'  ',' '                    ' ',' '         .
def cut_char_sigu(self, name):
    return name if len(name) == 2 else name[:-1]

def change_char_sido(self, election_result):
    # '    '
    sido_candi = election_result['    ']
    sido_candi = [name[:2] if name[:2] in ['  ',' ',' ',' ',' ',' ',' ']
                  else '' for name in sido_candi]
    '''
   Unnamed: 0                  pop      moon     hong      ahn
                 0          0              102566.0   42512.0  22325.0  22313.0
                 1          1               82852.0   34062.0  17901.0  19372.0
                 2          2              148157.0   58081.0  35230.0  32109.0
                 3          3              203175.0   86686.0  40566.0  45674.0
                 4          4              240030.0  105512.0  46368.0  52824.0
    '''

    #         ' '      '     ',' '     '           .
    sigun_candi = [''] * len(election_result)
    for n in election_result.index:
        each = election_result['  '][n]
        if each[:2] in ['  ',' ',' ',' ',' ',' ',
                        ' ',' ',' ',' ',' ',' ',' ']:
            sigun_candi[n] = re.split(' ', each)[0] + ' ' + \
                             self.cut_char_sigu(re.split(' ', each)[1])
        else:
            sigun_candi[n] = self.cut_char_sigu(each)
    # ic(sigun_candi)

    # sido_candi                                            .
    #                   ,                       .
    #   ,'  '                   .
    ID_candi = [sido_candi[n] + ' ' + sigun_candi[n] for n in range(0, len(sigun_candi))]
    ID_candi = [name[1:] if name[0] == ' ' else name for name in ID_candi]
    ID_candi = [name[:2] if name[:2] == '  ' else name for name in ID_candi]
    election_result['ID'] = ID_candi
    # ic(election_result.head(10))
    return election_result

def calc_percent_vote(self, election_result):
    #     =     /
    #    ,    ,
    election_result[['rate_moon', 'rate_hong', 'rate_ahn']] = election_result[['moon', 'hong', 'ahn']] \
        .div(election_result['pop'], axis=0)
    election_result[['rate_moon', 'rate_hong', 'rate_ahn']] *= 100
    # ic(election_result.head())
    '''
     Unnamed: 0                  pop  ...     ID  rate_moon  rate_hong   rate_ahn
    0           0              102566.0  ...        41.448433  21.766472  21.754773
    1           1               82852.0  ...        41.111862  21.605996  23.381451
    2           2              148157.0  ...        39.202333  23.778829  21.672280
    3           3              203175.0  ...        42.665682  19.966039  22.480128
    4           4              240030.0  ...        43.957839  19.317585  22.007249
    '''

    #
    election_result.sort_values(['rate_moon'], ascending=[False]).head(10)
    '''
        Unnamed: 0                 ...  rate_moon  rate_hong   rate_ahn
    182         182                 ...  67.563695   2.493786  22.280946
    166         166                 ...  66.716865   2.758074  21.385582
    165         165                 ...  66.687114   2.962565  21.366173
    175         175                 ...  66.633497   4.459233  20.853287
    184         184                 ...  65.927955   4.253818  20.833333
    '''
    #
    election_result.sort_values(['rate_hong'], ascending=[False]).head(10)
    '''
     Unnamed: 0                pop  ...  ID  rate_moon  rate_hong   rate_ahn
    219         219            17627.0  ...      12.770182  66.097464  11.000170
    220         220            37855.0  ...      14.172500  62.845067  12.592788
    223         223            26125.0  ...      14.491866  62.445933  12.367464
    247         247            33021.0  ...      21.631689  59.655976   9.318313
    216         216            22396.0  ...      16.761922  59.153420  11.609216
    '''
    #
    election_result.sort_values(['rate_ahn'], ascending=[False]).head(10)
    '''
     Unnamed: 0                  pop  ...     ID  rate_moon  rate_hong   rate_ahn
   196         196              21189.0  ...        49.044315   2.411629  41.790552
   201         201              28950.0  ...        49.637306   2.462867  41.450777
   193         193              25175.0  ...        49.557100   2.991063  40.325720
   195         195              48351.0  ...        53.568696   2.394987  37.552481
```

```
       197         197              36402.0  ...         52.192187   2.266359  37.388056
        '''
        return election_result

    def visualize_percent_vote(self, target_data, blocked_map, cmap_name):
        BORDER_LINES = [
            [(5, 1), (5, 2), (7, 2), (7, 3), (11, 3), (11, 0)],  #
            [(5, 4), (5, 5), (2, 5), (2, 7), (4, 7), (4, 9), (7, 9),(7, 7), (9, 7), (9, 5), (10, 5), (10, 4), (5, 4)],  #
            [(1, 7), (1, 8), (3, 8), (3, 10), (10, 10), (10, 7),(12, 7), (12, 6), (11, 6), (11, 5), (12, 5), (12, 4),(11, 4), (11, 3)],  #
            [(8, 10), (8, 11), (6, 11), (6, 12)],  #
            [(12, 5), (13, 5), (13, 4), (14, 4), (14, 5), (15, 5),(15, 4), (16, 4), (16, 2)],  #
            [(16, 4), (17, 4), (17, 5), (16, 5), (16, 6), (19, 6),(19, 5), (20, 5), (20, 4), (21, 4), (21, 3), (19, 3), (19, 1)],  #
            [(13, 5), (13, 6), (16, 6)],  #
            [(13, 5), (14, 5)],  #
            [(21, 2), (21, 3), (22, 3), (22, 4), (24, 4), (24, 2), (21, 2)],  #
            [(20, 5), (21, 5), (21, 6), (23, 6)],  #
            [(10, 8), (12, 8), (12, 9), (14, 9), (14, 8), (16, 8), (16, 6)],  #
            [(14, 9), (14, 11), (14, 12), (13, 12), (13, 13)],  #
            [(15, 8), (17, 8), (17, 10), (16, 10), (16, 11), (14, 11)],  #
            [(17, 9), (18, 9), (18, 8), (19, 8), (19, 9), (20, 9), (20, 10), (21, 10)],  #
            [(16, 11), (16, 13)],  #
            [(27, 5), (27, 6), (25, 6)]]

        gamma = 0.75
        whitelabelmin = 20.
        datalabel = target_data
        tmp_max = max([np.abs(min(blocked_map[target_data])), np.abs(max(blocked_map[target_data]))])
        vmin, vmax = -tmp_max, tmp_max

        mapdata = blocked_map.pivot_table(index='y', columns='x', values=target_data)
        masked_mapdata = np.ma.masked_where(np.isnan(mapdata), mapdata)

        plt.figure(figsize=(9, 11))
        plt.pcolor(masked_mapdata, vmin=vmin, vmax=vmax, cmap=cmap_name,
                   edgecolor='#aaaaaa', linewidth=0.5)

        #
        for idx, row in blocked_map.iterrows():
            #                                    .
            # (  ,   )
            if len(row['ID'].split()) == 2:
                dispname = '{}\n{}'.format(row['ID'].split()[0], row['ID'].split()[1])
            elif row['ID'][:2] == '  ':
                dispname = '  '
            else:
                dispname = row['ID']

            #     ,             3                      .
            if len(dispname.splitlines()[-1]) >= 3:
                fontsize, linespacing = 10.0, 1.1
            else:
                fontsize, linespacing = 11, 1.

            annocolor = 'white' if np.abs(row[target_data]) > whitelabelmin else 'black'
            plt.annotate(dispname, (row['x'] + 0.5, row['y'] + 0.5), weight='bold',
                         fontsize=fontsize, ha='center', va='center', color=annocolor,
                         linespacing=linespacing)

        #          .
        for path in BORDER_LINES:
            ys, xs = zip(*path)
            plt.plot(xs, ys, c='black', lw=2)

        plt.gca().invert_yaxis()

        plt.axis('off')

        cb = plt.colorbar(shrink=0.1, aspect=10)
        cb.set_label(datalabel)

        plt.tight_layout()
        plt.show()

    def draw_korea(self):
        #
        path = "c:\Windows\Fonts\gulim.ttc"
        font_name = font_manager.FontProperties(fname=path).get_name()
        matplotlib.rc('font', family=font_name)

        file = self.file
        file.fname = 'final_elect_data'
        self.file.context = './save/'
        final_elect_data = self.csv(file)
        # "       vs       "
        self.visualize_percent_vote('moon_vs_hong', final_elect_data, 'RdBu')
        # "       vs       "
        self.visualize_percent_vote('moon_vs_ahn', final_elect_data, 'RdBu')
```

```python
        # "       vs       "
        self.visualize_percent_vote('ahn_vs_hong', final_elect_data, 'RdBu')

    def draw_korea_geo(self):
        file = self.file
        file.fname = 'final_elect_data'
        self.file.context = './save/'
        final_elect_data = self.csv(file)
        self.file.context = './data/'
        file.fname = 'skorea_municipalities_geo_simple'
        geo_path = self.mpa_json(file)

        # Folium
        # 'ID'  index
        pop_folium = final_elect_data.set_index('ID')
        # '    ', '  '
        del pop_folium['    ']
        del pop_folium['  ']
        pop_folium.head()
        map = folium.Map(location=[36.2002, 127.054], zoom_start=6)
        # "       vs       "
        map.choropleth(geo_data=geo_path,
                       data=pop_folium['moon_vs_hong'],
                       columns=[pop_folium.index, pop_folium['moon_vs_hong']],
                       fill_color='PuBu',  # 'PuRd', 'YlGnBu'
                       key_on='feature.id')
        map.save('./save/moon_vs_hong_map.html')
        # "       vs       "
        map.choropleth(geo_data=geo_path,
                       data=pop_folium['moon_vs_ahn'],
                       columns=[pop_folium.index, pop_folium['moon_vs_ahn']],
                       fill_color='PuBu',  # 'PuRd', 'YlGnBu'
                       key_on='feature.id')
        map.save('./save/moon_vs_ahn_map.html')
        # "       vs       "
        map.choropleth(geo_data=geo_path,
                       data=pop_folium['ahn_vs_hong'],
                       columns=[pop_folium.index, pop_folium['ahn_vs_hong']],
                       fill_color='PuBu',  # 'PuRd', 'YlGnBu'
                       key_on='feature.id')
        map.save('./save/ahn_vs_hong_map.html')


if __name__ == '__main__':
    Solution().hook()
```