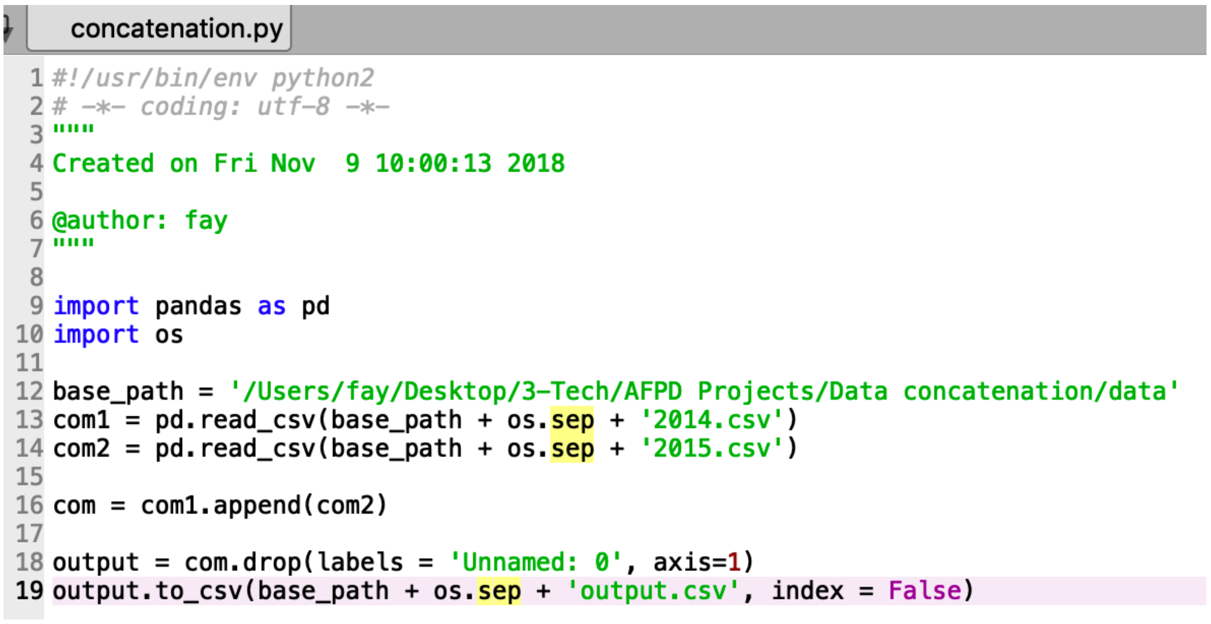
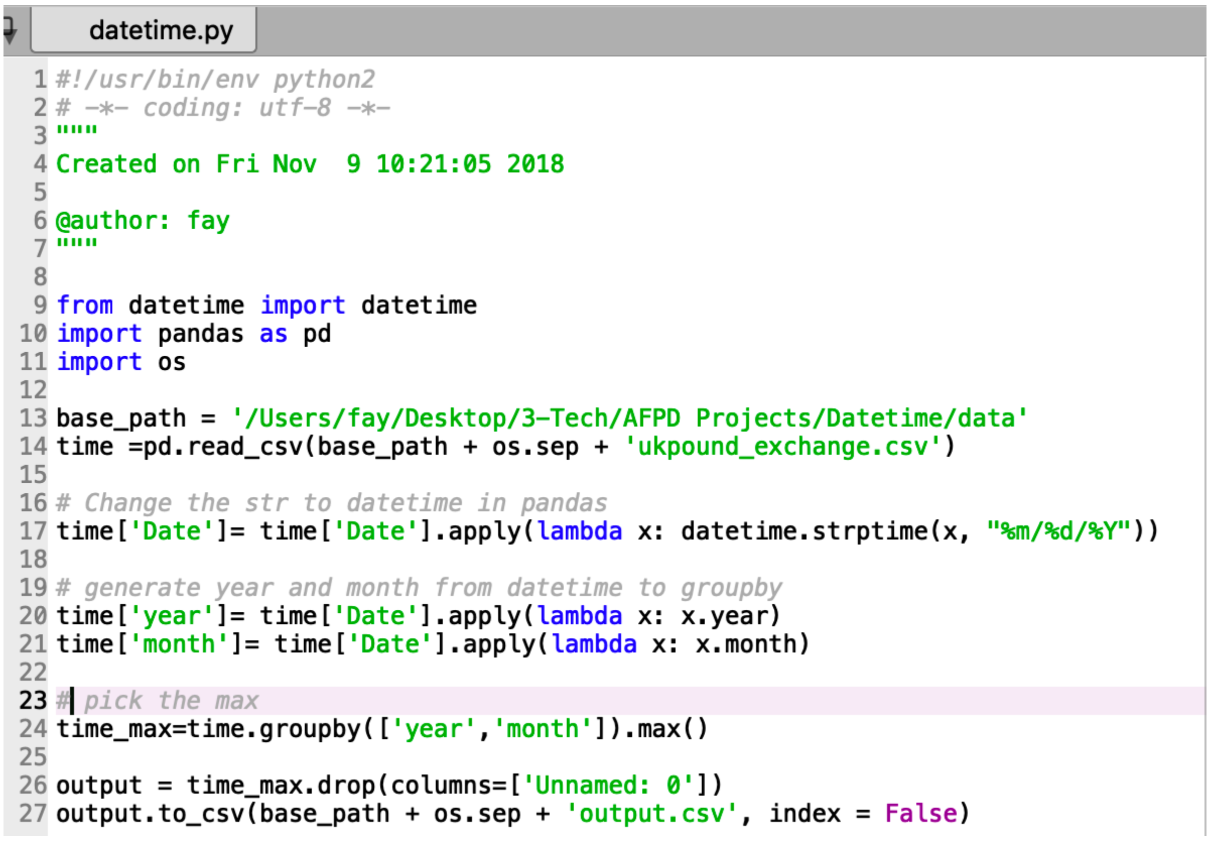
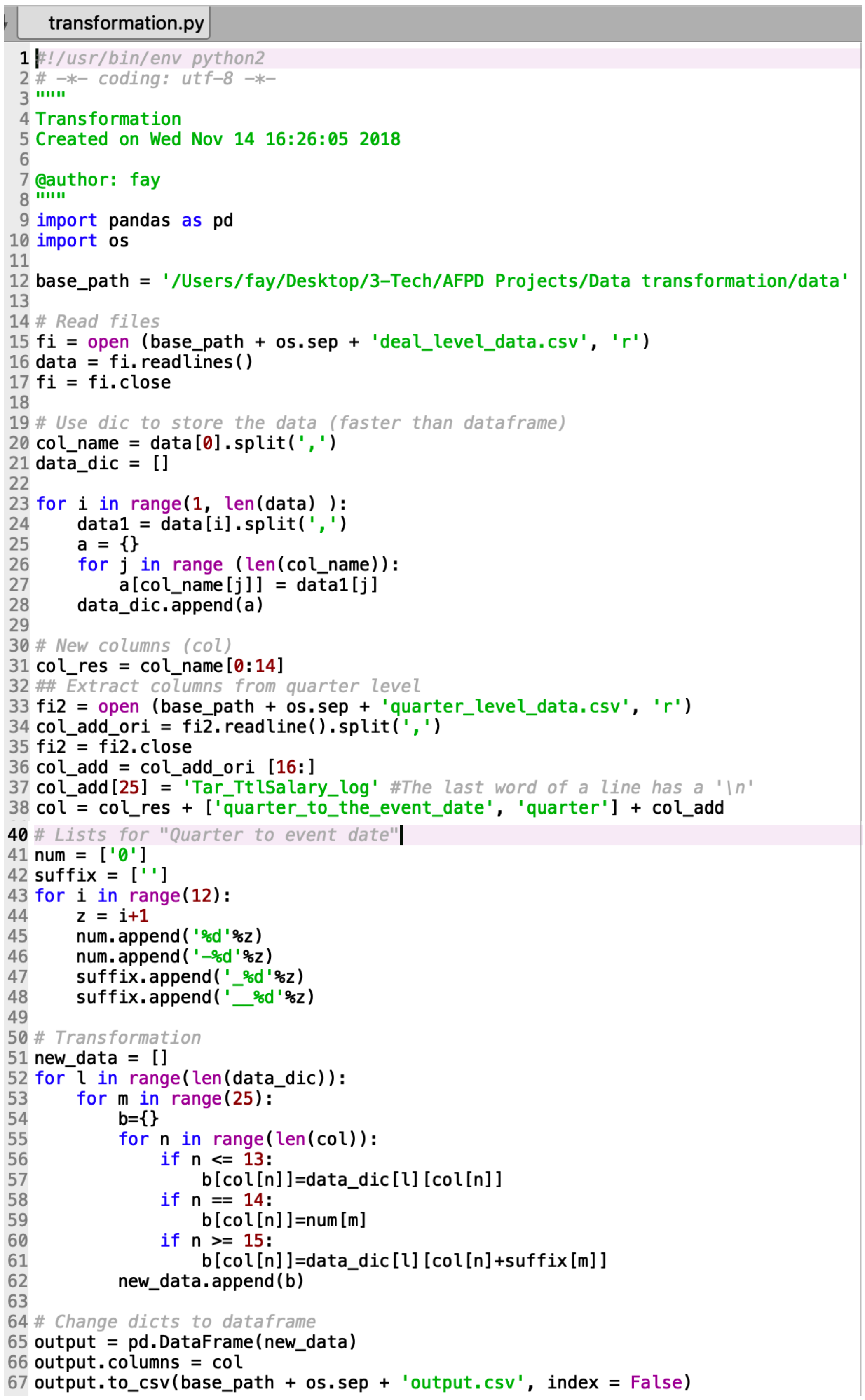


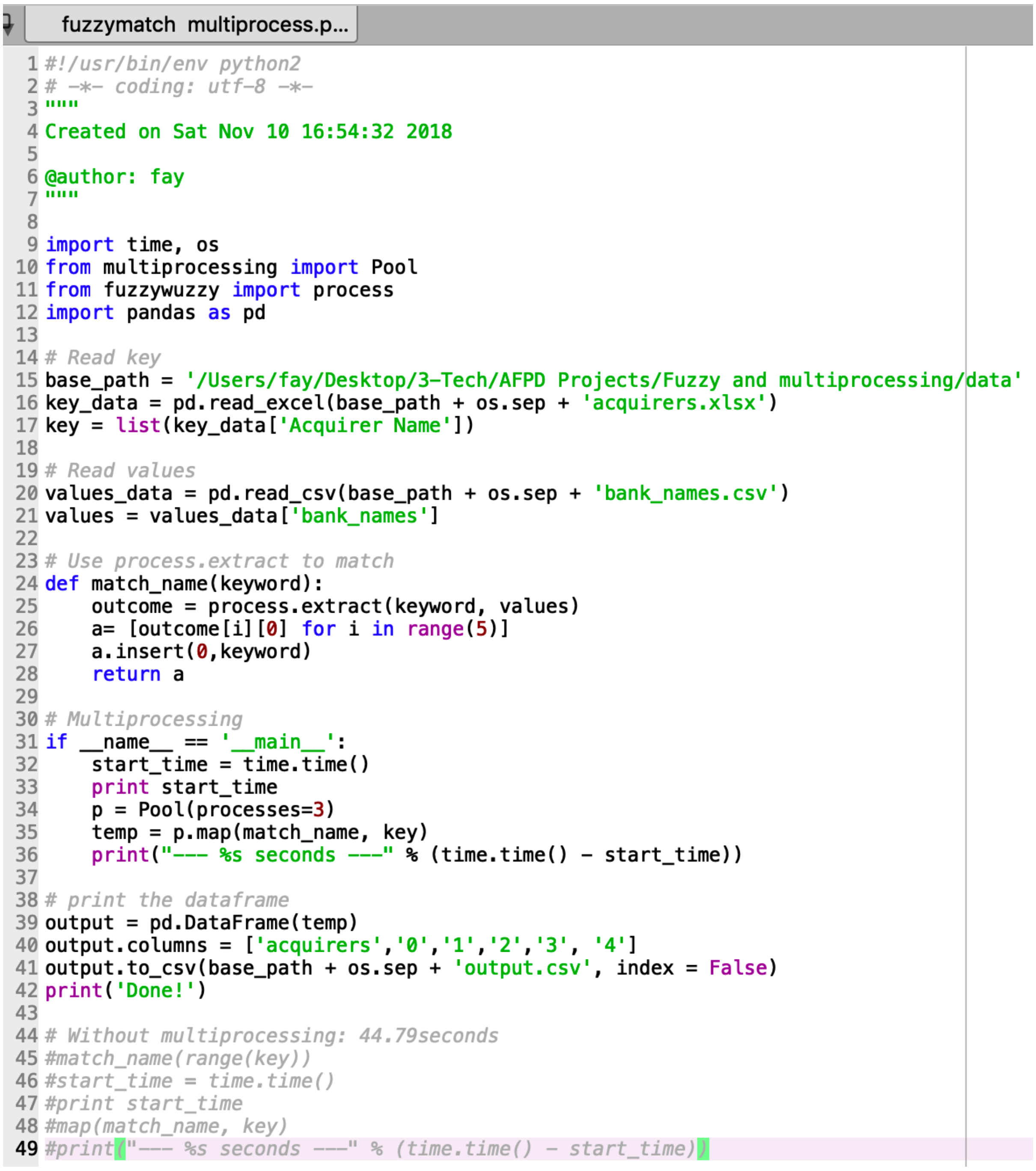
```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Nov  9 10:00:13 2018

@author: fay
"""

import pandas as pd
import os

base_path = '/Users/fay/Desktop/3-Tech/AFPD Projects/Data concatenation/data'
com1 = pd.read_csv(base_path + os.sep + '2014.csv')
com2 = pd.read_csv(base_path + os.sep + '2015.csv')

com = com1.append(com2)

output = com.drop(labels = 'Unnamed: 0', axis=1)
output.to_csv(base_path + os.sep + 'output.csv', index = False)
```



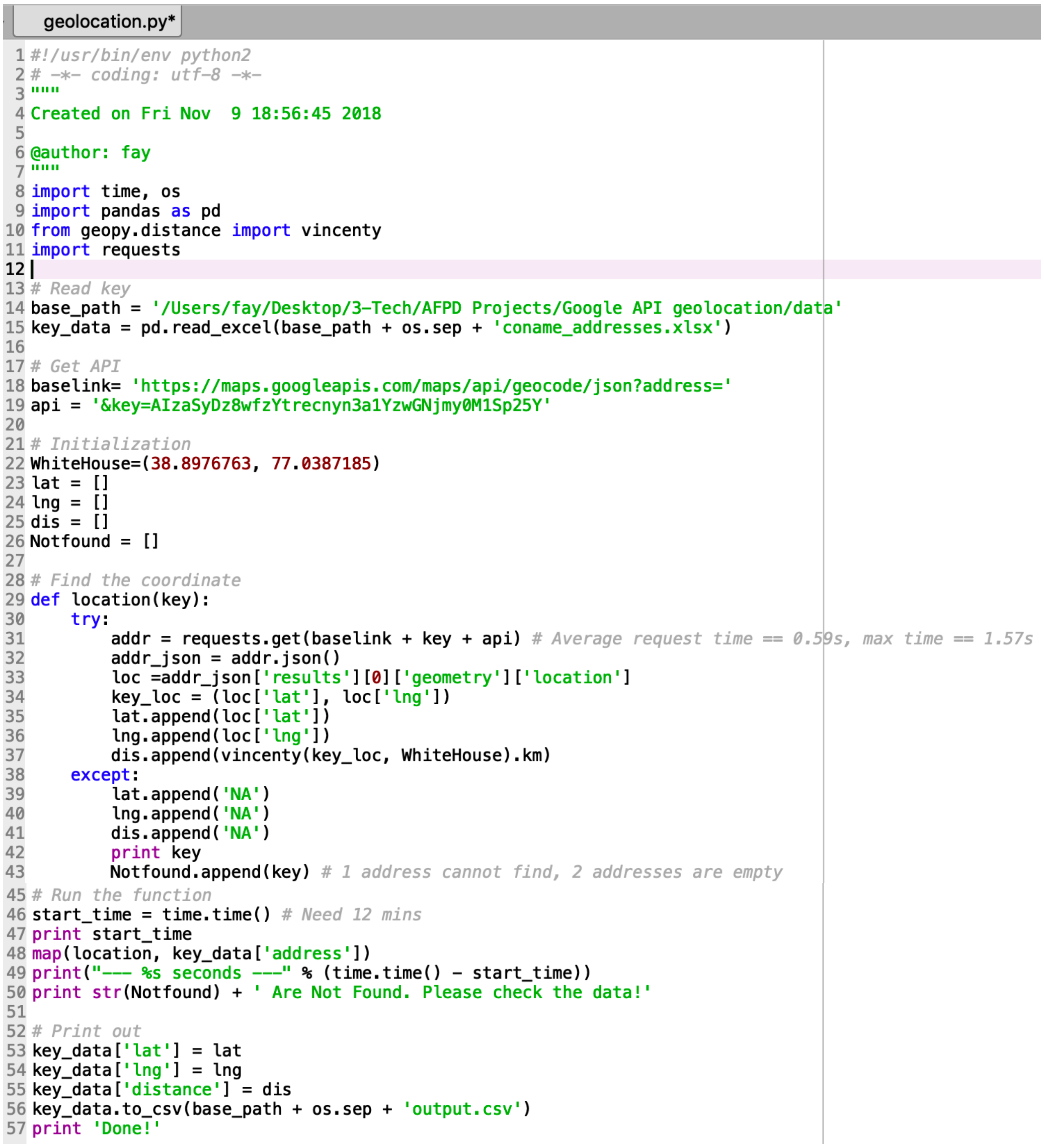
```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Nov  9 10:21:05 2018

@author: fay
"""

from datetime import datetime
import pandas as pd
import os

base_path = '/Users/fay/Desktop/3-Tech/AFPD Projects/Datetime/data'
time =pd.read_csv(base_path + os.sep + 'ukpound_exchange.csv')

# Change the str to datetime in pandas
time['Date']= time['Date'].apply(lambda x: datetime.strptime(x, "%m/%d/%Y"))

# generate year and month from datetime to groupby
time['year']= time['Date'].apply(lambda x: x.year)
time['month']= time['Date'].apply(lambda x: x.month)

# pick the max
time_max=time.groupby(['year','month']).max()

output = time_max.drop(columns=['Unnamed: 0'])
output.to_csv(base_path + os.sep + 'output.csv', index = False)
```



```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Transformation
Created on Wed Nov 14 16:26:05 2018

@author: fay
"""
import pandas as pd
import os

base_path = '/Users/fay/Desktop/3-Tech/AFPD Projects/Data transformation/data'

# Read files
fi = open (base_path + os.sep + 'deal_level_data.csv', 'r')
data = fi.readlines()
fi = fi.close

# Use dic to store the data (faster than dataframe)
col_name = data[0].split(',')
data_dic = []

for i in range(1, len(data) ):
    data1 = data[i].split(',')
    a = {}
    for j in range (len(col_name)):
        a[col_name[j]] = data1[j]
    data_dic.append(a)

# New columns (col)
col_res = col_name[0:14]
## Extract columns from quarter level
fi2 = open (base_path + os.sep + 'quarter_level_data.csv', 'r')
col_add_ori = fi2.readline().split(',')
fi2 = fi2.close
col_add = col_add_ori [16:]
col_add[25] = 'Tar_TtlSalary_log' #The last word of a line has a '\n'
col = col_res + ['quarter_to_the_event_date', 'quarter'] + col_add
# Lists for "Quarter to event date"
num = ['0']
suffix = ['']
for i in range(12):
    z = i+1
    num.append('%d'%z)
    num.append('-%d'%z)
    suffix.append('_%d'%z)
    suffix.append('__%d'%z)

# Transformation
new_data = []
for l in range(len(data_dic)):
    for m in range(25):
        b={}
        for n in range(len(col)):
            if n <= 13:
                b[col[n]]=data_dic[l][col[n]]
            if n == 14:
                b[col[n]]=num[m]
            if n >= 15:
                b[col[n]]=data_dic[l][col[n]+suffix[m]]
        new_data.append(b)

# Change dicts to dataframe
output = pd.DataFrame(new_data)
output.columns = col
output.to_csv(base_path + os.sep + 'output.csv', index = False)
```



```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 10 16:54:32 2018

@author: fay
"""

import time, os
from multiprocessing import Pool
from fuzzywuzzy import process
import pandas as pd

# Read key
base_path = '/Users/fay/Desktop/3-Tech/AFPD Projects/Fuzzy and multiprocessing/data'
key_data = pd.read_excel(base_path + os.sep + 'acquirers.xlsx')
key = list(key_data['Acquirer Name'])

# Read values
values_data = pd.read_csv(base_path + os.sep + 'bank_names.csv')
values = values_data['bank_names']

# Use process.extract to match
def match_name(keyword):
    outcome = process.extract(keyword, values)
    a= [outcome[i][0] for i in range(5)]
    a.insert(0,keyword)
    return a

# Multiprocessing
if __name__ == '__main__':
    start_time = time.time()
    print start_time
    p = Pool(processes=3)
    temp = p.map(match_name, key)
    print("--- %s seconds ---" % (time.time() - start_time))

# print the dataframe
output = pd.DataFrame(temp)
output.columns = ['acquirers','0','1','2','3', '4']
output.to_csv(base_path + os.sep + 'output.csv', index = False)
print('Done!')

# Without multiprocessing: 44.79seconds
#match_name(range(key))
#start_time = time.time()
#print start_time
#map(match_name, key)
#print("--- %s seconds ---" % (time.time() - start_time))
```



geolocation.py*

```python
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Fri Nov  9 18:56:45 2018

@author: fay
"""
import time, os
import pandas as pd
from geopy.distance import vincenty
import requests

# Read key
base_path = '/Users/fay/Desktop/3-Tech/AFPD Projects/Google API geolocation/data'
key_data = pd.read_excel(base_path + os.sep + 'coname_addresses.xlsx')

# Get API
baselink= 'https://maps.googleapis.com/maps/api/geocode/json?address='
api = '&key=AIzaSyDz8wfzYtrecnyn3a1YzwGNjmy0M1Sp25Y'

# Initialization
WhiteHouse=(38.8976763, 77.0387185)
lat = []
lng = []
dis = []
Notfound = []

# Find the coordinate
def location(key):
    try:
        addr = requests.get(baselink + key + api) # Average request time == 0.59s, max time == 1.57s
        addr_json = addr.json()
        loc =addr_json['results'][0]['geometry']['location']
        key_loc = (loc['lat'], loc['lng'])
        lat.append(loc['lat'])
        lng.append(loc['lng'])
        dis.append(vincenty(key_loc, WhiteHouse).km)
    except:
        lat.append('NA')
        lng.append('NA')
        dis.append('NA')
        print key
        Notfound.append(key) # 1 address cannot find, 2 addresses are empty
# Run the function
start_time = time.time() # Need 12 mins
print start_time
map(location, key_data['address'])
print("--- %s seconds ---" % (time.time() - start_time))
print str(Notfound) + ' Are Not Found. Please check the data!'

# Print out
key_data['lat'] = lat
key_data['lng'] = lng
key_data['distance'] = dis
key_data.to_csv(base_path + os.sep + 'output.csv')
print 'Done!'
```