Python API

apply python function to call the optimization and risk model APIs

```
# Input: Postman token
test_token = '5bc0a1a2-f3f9-4f29-8bcb-0eca2cb76a25'
```

```
import requests
import json
import pandas as pd
```

Optimization API

feature argument - json format:

```
{
   "alpha": "QES_LEAP_1_SCORE",
   "template" : "default",
   "portfolio" : "SP500",
   "startDate" : "2018-09-30",
   "endDate" : "2018-12-31",
   "freq" : "1me",
   "notionalValue" : 100000000,
   "baseCurrency" : "USD",
   "riskModel" :
       {
         "universe": "SP500",
         "template" : "default"
       }
}
```

optimization_call:

call the API and derive the optimal portfolio weights of pandas dataframe format

```
def optimization_call(postman_token,
                      alpha = "QES_LEAP_1_SCORE",
                      template = "default",
                      portfolio = "SP500",
                      startDate = "2018-09-30",
                      endDate = "2018-12-31",
                      freq = "1me", notionalValue = 100000000,
                      baseCurrency = "USD",
                      riskModel = '{"universe": "SP500","template" : "default"}',
                      verbose = True, saveFlag = True, savePath = '.'):
    '''
```

```
    call the optimization API and generate the corresponding Postman UUID token,
    fectch and save the optimal weights and summary into pandas dataframe type
    Inputs:
        alpha: signal
        portfolio: universe
        startDate: start trading date
        endDate: end trading date
        freq: trading frequency
        verbose: boolean value of whether display the running status
        saveFlag: boolean value of whether save the file
        savePath: file path for saving data
    Outputs:
        dictionary contains the status of API calling, portfolio optimal weights and summary
    '''
    # get the header & args
    headers = {
    'Authorization': 'Basic aGphaW46aGphaW4xMjM=','Content-Type': 'application/json',
    'Postman-Token': postman_token,
    'cache-control': 'no-cache',
    }
    args_data = {"alpha": alpha,
                 "template" : template,
                 "portfolio" : portfolio,
                 "startDate" : startDate,
                 "endDate" : endDate,
                 "freq" : freq,
                 "notionalValue" : notionalValue,
                 "baseCurrency" : baseCurrency,
                 "riskModel" : riskModel
                }
    # derive the API token
    uid = token_access(args_data, headers, verbose= verbose)

    callFlag = True
    if uid is None:
        # unsuccessful call
        callFlag = False

    # accesss Weight
    df_weight = data_process(uid, headers, saveFlag=saveFlag, savePath=savePath)
    # access Summary
    df_summary = data_process(uid, headers, feature= 'Summary', saveFlag=saveFlag, savePath=savePath)

    return {'status': callFlag, 'weight': df_weight, 'summary':df_summary}
```

- usage showcase

```
# Optimization results dictionary with default inputs
result_dic = optimization_call(test_token)
```

```
# Weight display
result_dic['weight']
```

	DATE	ID	Sedol	Ticker	Company Name	Sector	Industry Group	Country	Currency	IssuerId	WEIGHT

259	2018-09-30	9KRRG25L84	BD0PJ08	COL	Inmobiliaria Colonial	SOCIMI	S.A.	60	6010	ES	50
322	2018-09-30	9NJJO3GPRZ	7025471	EGL	Mota-Engil	SGPS	S.A.	20	2010	PT	50
363	2018-09-30	ZVVV283MPZ	5962934	SEM	Semapa - Sociedade de Investimento e Gestão	SGPS	S.A.	15	1510	PT	50
490	2018-09-30	ZVVVKK0K1Z	5973992	SON	Sonae	SGPS	S.A.	30	3010	PT	50
633	2018-09-30	92VVWDL039	4657736	COR	Corticeira Amorim	S.G.P.S.	S.A.	15	1510	PT	50

```
# Summary
result_dic['summary']
```

	Alpha	Ex-Ante Risk	Ex-Ante Tracking Error	Turnover	Max Turnover
1	0.34771553678194	0.0975385350142646	0.0975385350142646	1.99999962340783	NA
2	0.355137553618081	0.107704547829215	0.107704547829215	1.44999721432476	NA
3	0.288000411658759	0.104215536770768	0.104215536770768	1.49999918674998	NA
4	0.24134166758503	0.0971503585210817	0.0971503585210817	1.49654598552549	NA

Inner Tool Function

- token_access

```
# init a Postman headers and optimization argument for fetch the data
headers = "{'Authorization': 'Basic aGphaW46aGphaW4xMjM=','Content-Type': 'application/json','Postman-Token':
'8732d840-1fcf-4f21-a92e-d4390dd88fc6','cache-control': 'no-cache'}"

data = '{\n    "alpha": "QES_LEAP_1_SCORE",\n    "template": "default",\n    "portfolio": "SP500",\n
"startDate": "2018-08-30",\n    "endDate": "2018-12-31",\n    "freq": "1me",\n    "notionalValue":
100000000,\n    "baseCurrency": "USD",\n    "riskModel": {\n        "universe": "SP500",\n        "template":
"default"\n    }\n}'
```

```
def token_access(args_data, headers, function = 'optimization', verbose = True):
    '''
    access the token for accessing the API results
    Input:
        args_data: feature with each value
        function: {optimization|risk}
        verbose: boolean value to display the argument or not
```

```
    Output:
        unique id
    '''
    # call the
    response = requests.post('http://feed.luoquant.com/{}/'.format(function),
                             headers=headers, data=data)
    if response.status_code == 200:
        return response.text

    if verbose:
        print('Unsuccessful call')
    return None
```

```
# Example : generat
uid =  token_access(data, headers)
print(uid)
```

```
78b871e5-982b-419c-8398-e2d31496fdf6
```

- data_process

```
def data_process(uid, headers, feature = 'Weights', function = 'optimization', saveFlag = True, savePath =
'.'):
    # I. get the data content
    response = requests.get('http://feed.luoquant.com/{0}/{1}/{2}.csv'.format(function, uid, feature),
                            headers=headers)
    data_text = response.text

    # II. generate the corresponding pandas df
    df_raw = pd.DataFrame([[word.replace("\"", "") for word in line.split(',')] for line in
data_text.split('\n')])
    # process the data df
    df = df_raw.rename(columns=df_raw.iloc[0]).iloc[1:]
    # drop the all-none value
    df.dropna(inplace = True)
    # III. save the data into local path with csv format
    if saveFlag:
        # by default save into the current directory
        df.to_csv(savePath + '/{}.csv'.format(feature))

    return df
```

```
# Example : generate and save the Weights data
df_weight = data_process(uid, headers)

df_weight.tail()
```

	DATE	ID	Sedol	Ticker	Company Name	Sector	Industry Group	Country	Currency	WEIGHT
2566	2018-12-31	4MOOJ25EK9	B6ZC3N6	TRIP	TripAdvisor Inc	50	5020	US	160	0
2567	2018-	4M7KNQL719	B6WVMH3	CBRE	CBRE Group	60	6010	US	160	0

	12-31				Inc					
2568	2018-12-31	4M7KY5YDK9	B01R258	WCG	WellCare Health Plans Inc	35	3510	US	160	0
2569	2018-12-31	4MK60M351Z	B3SPXZ3	LYB	LyondellBasell Industries NV	15	1510	US	160	0
2570	2018-12-31	4M3YQMNYKZ	BFRT3W7	ALLE	Allegion Plc	20	2010	US	160	0