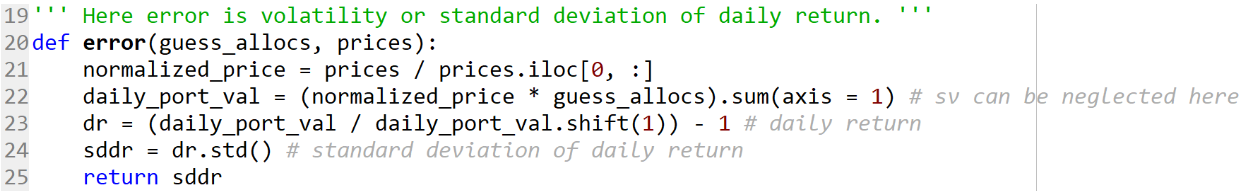
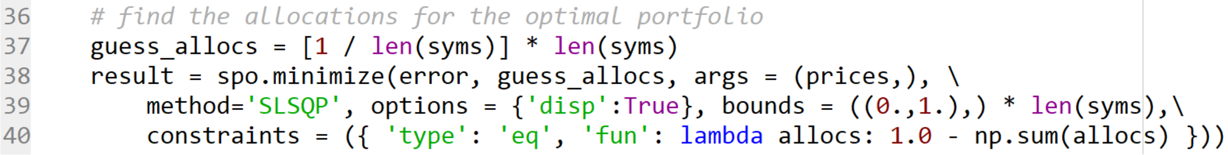
Project 2 Optimize something



```python
''' Here error is volatility or standard deviation of daily return. '''
def error(guess_allocs, prices):
    normalized_price = prices / prices.iloc[0, :]
    daily_port_val = (normalized_price * guess_allocs).sum(axis = 1) # sv can be neglected here
    dr = (daily_port_val / daily_port_val.shift(1)) - 1 # daily return
    sddr = dr.std() # standard deviation of daily return
    return sddr
```

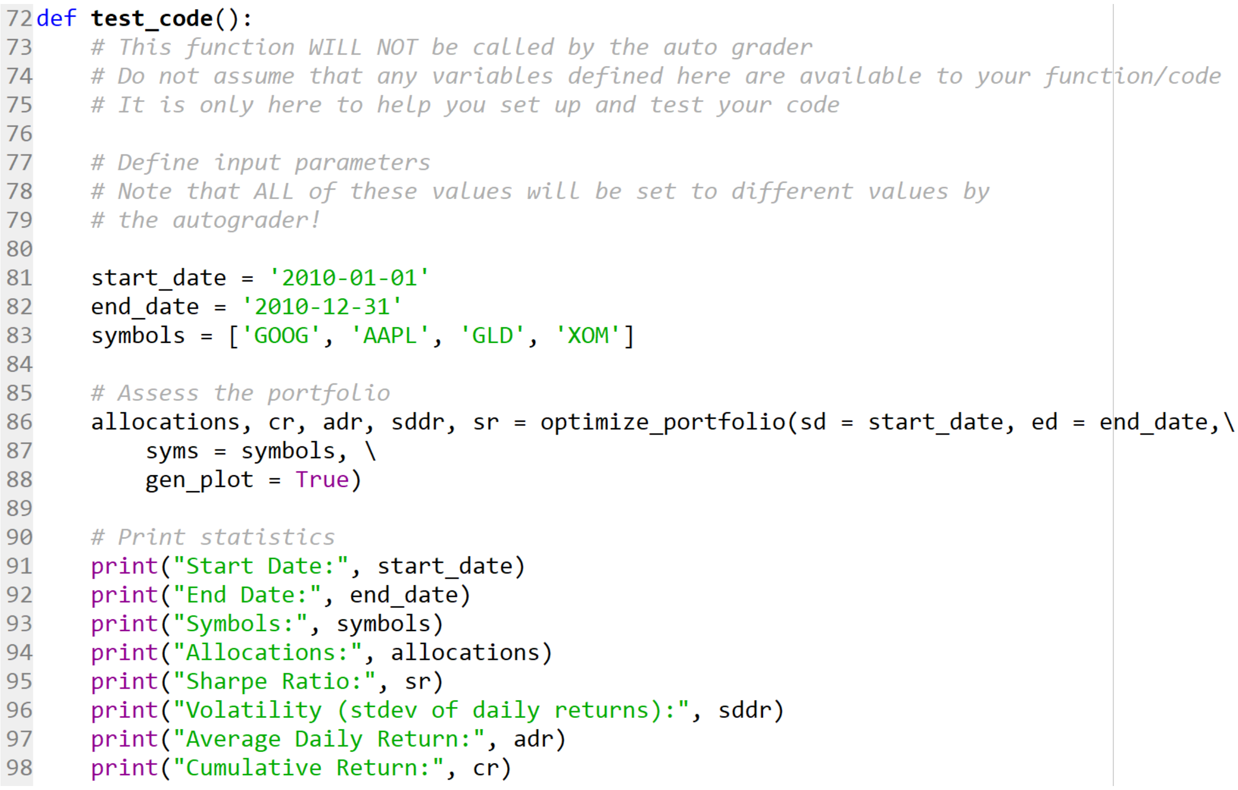


```python
    # find the allocations for the optimal portfolio
    guess_allocs = [1 / len(syms)] * len(syms)
    result = spo.minimize(error, guess_allocs, args = (prices,), \
        method='SLSQP', options = {'disp':True}, bounds = ((0.,1.),) * len(syms),\
        constraints = ({ 'type': 'eq', 'fun': lambda allocs: 1.0 - np.sum(allocs) }))
```

Minimize volatility (std of daily return) to achieve 'optimal'.

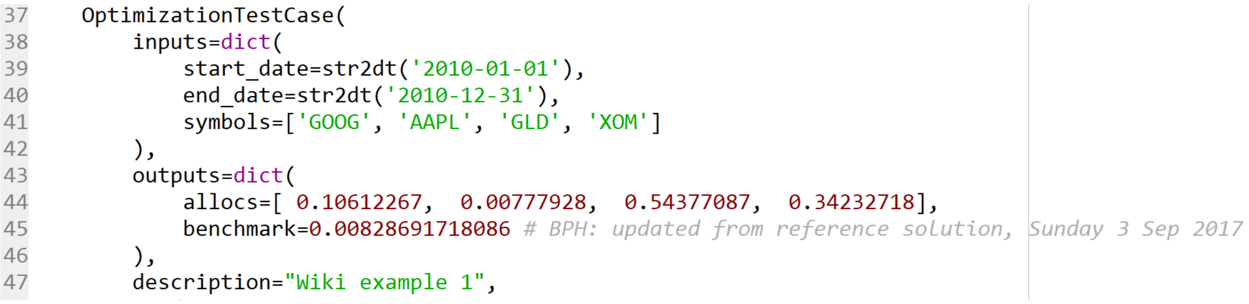
https://stackoverflow.com/questions/35631192/element-wise-constraints-in-scipy-optimize-minimize

Check StackOverflow answer to see how to use constrains to set sum of allocations to 1.

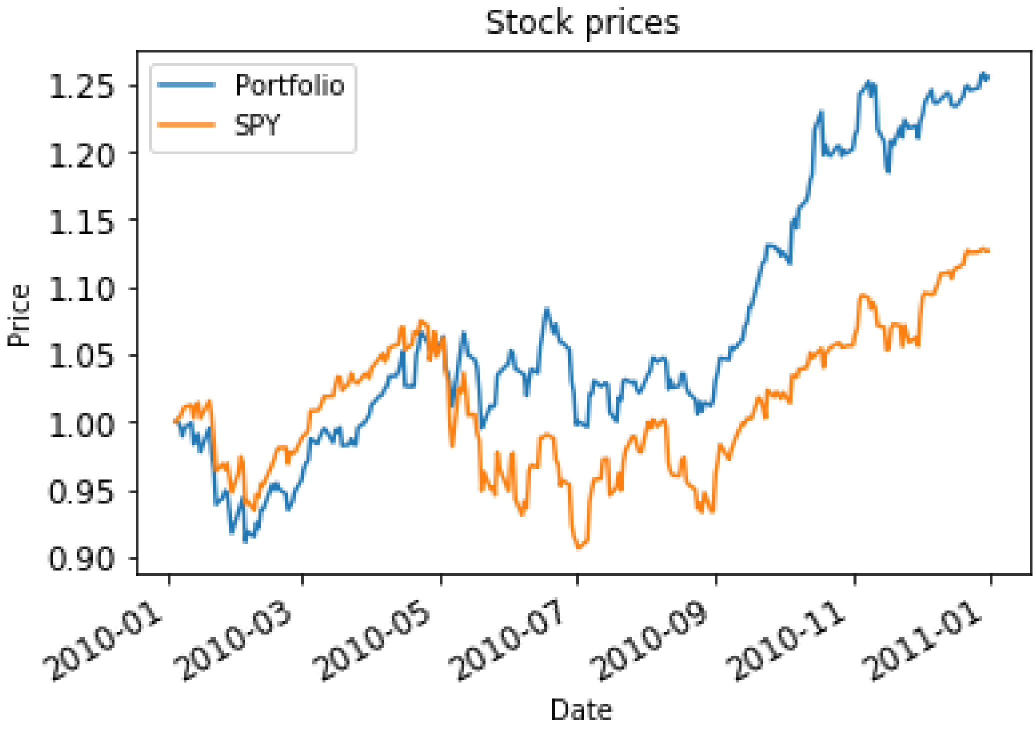


```python
def test_code():
    # This function WILL NOT be called by the auto grader
    # Do not assume that any variables defined here are available to your function/code
    # It is only here to help you set up and test your code

    # Define input parameters
    # Note that ALL of these values will be set to different values by
    # the autograder!

    start_date = '2010-01-01'
    end_date = '2010-12-31'
    symbols = ['GOOG', 'AAPL', 'GLD', 'XOM']

    # Assess the portfolio
    allocations, cr, adr, sddr, sr = optimize_portfolio(sd = start_date, ed = end_date,\
        syms = symbols, \
        gen_plot = True)

    # Print statistics
    print("Start Date:", start_date)
    print("End Date:", end_date)
    print("Symbols:", symbols)
    print("Allocations:", allocations)
    print("Sharpe Ratio:", sr)
    print("Volatility (stdev of daily returns):", sddr)
    print("Average Daily Return:", adr)
    print("Cumulative Return:", cr)
```

Example Wiki Example 1



```
    OptimizationTestCase(
        inputs=dict(
            start_date=str2dt('2010-01-01'),
            end_date=str2dt('2010-12-31'),
            symbols=['GOOG', 'AAPL', 'GLD', 'XOM']
        ),
        outputs=dict(
            allocs=[ 0.10612267,  0.00777928,  0.54377087,  0.34232718],
            benchmark=0.00828691718086 # BPH: updated from reference solution, Sunday 3 Sep 2017
        ),
        description="Wiki example 1",
```



Start Date: 2010-01-01

End Date: 2010-12-31

Symbols: ['GOOG', 'AAPL', 'GLD', 'XOM']

Allocations: [0.10612268 0.00777931 0.54377086 0.34232715]

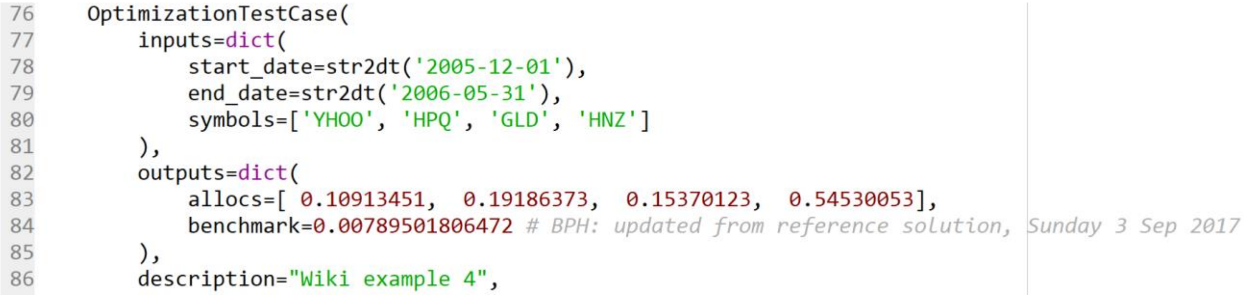
Sharpe Ratio: 1.2787144472012433

Volatility (stdev of daily returns): 0.008286917194624346

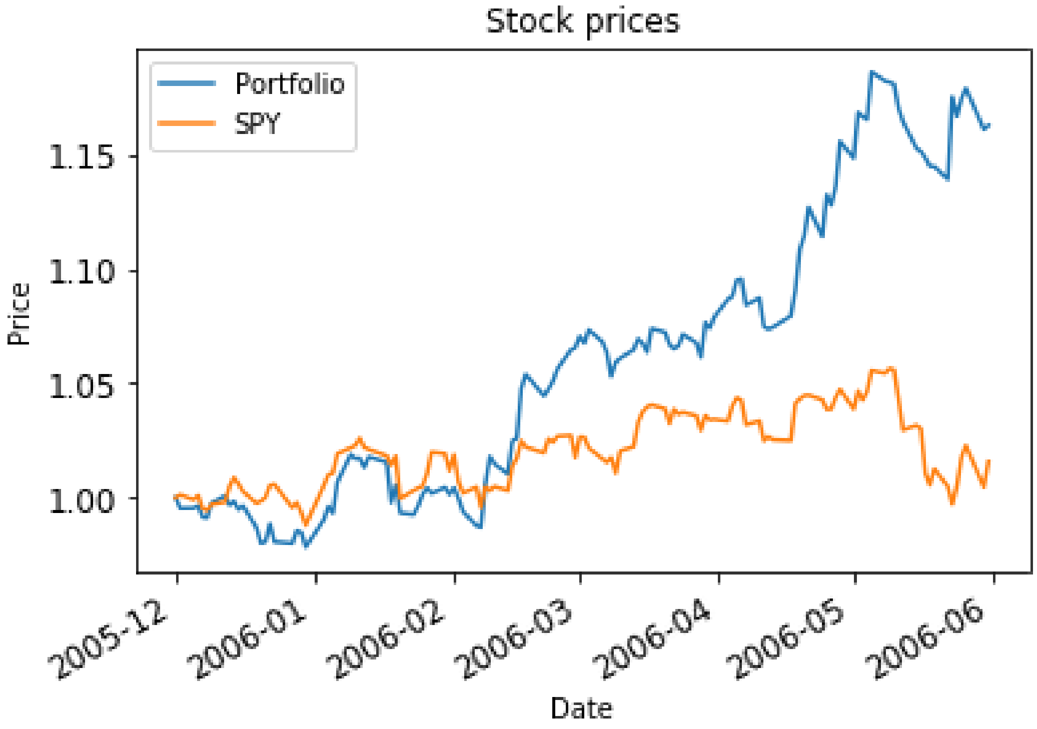
Average Daily Return: 0.0006635848786022384

Cumulative Return: 0.17105107382716445

Example Wiki Example 4



```
    OptimizationTestCase(
        inputs=dict(
            start_date=str2dt('2005-12-01'),
            end_date=str2dt('2006-05-31'),
            symbols=['YHOO', 'HPQ', 'GLD', 'HNZ']
        ),
        outputs=dict(
            allocs=[ 0.10913451,  0.19186373,  0.15370123,  0.54530053],
            benchmark=0.00789501806472 # BPH: updated from reference solution, Sunday 3 Sep 2017
        ),
        description="Wiki example 4",
```



Start Date: 2005-12-01

End Date: 2006-05-31

Symbols: ['YHOO', 'HPQ', 'GLD', 'HNZ']

Allocations: [0.10913452 0.19186371 0.15370125 0.54530052]

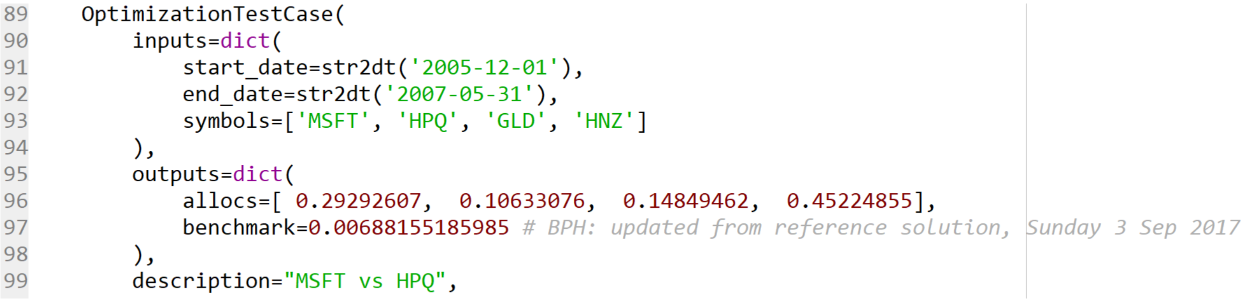
Sharpe Ratio: 2.5438700682811914

Volatility (stdev of daily returns): 0.007895018068826138

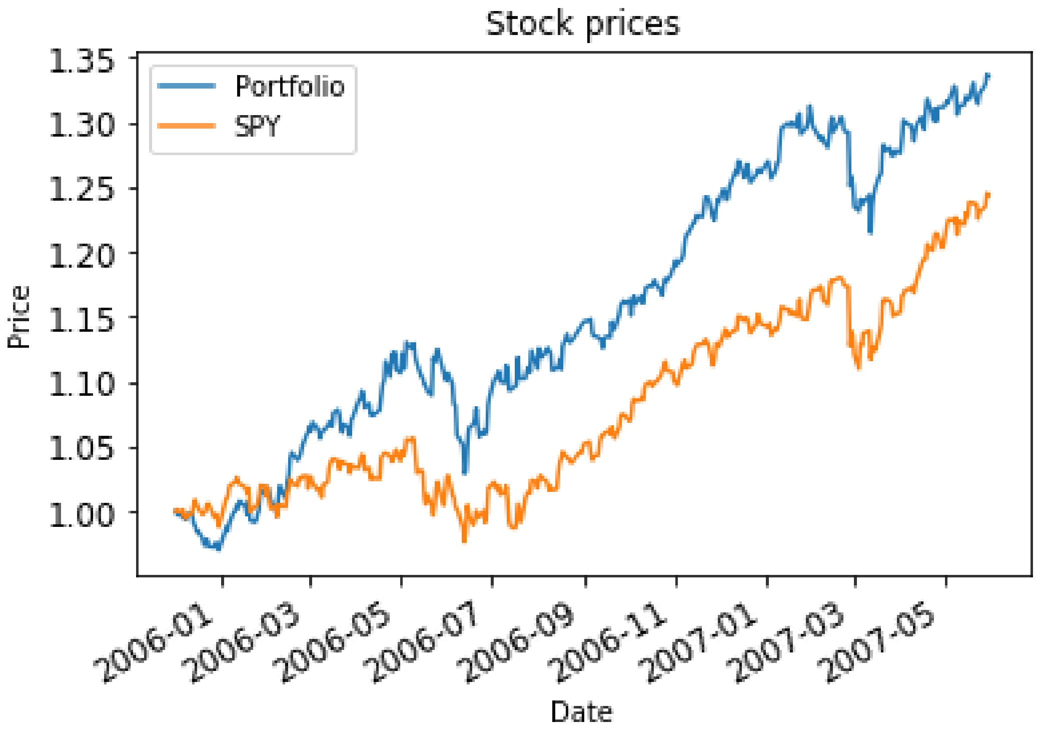
Average Daily Return: 0.0012577026136054606

Cumulative Return: 0.16278942404396135

MSFT vs HPQ



```
    OptimizationTestCase(
        inputs=dict(
            start_date=str2dt('2005-12-01'),
            end_date=str2dt('2007-05-31'),
            symbols=['MSFT', 'HPQ', 'GLD', 'HNZ']
        ),
        outputs=dict(
            allocs=[ 0.29292607,  0.10633076,  0.14849462,  0.45224855],
            benchmark=0.00688155185985 # BPH: updated from reference solution, Sunday 3 Sep 2017
        ),
        description="MSFT vs HPQ",
```



Start Date: 2005-12-01

End Date: 2007-05-31

Symbols: ['MSFT', 'HPQ', 'GLD', 'HNZ']

Allocations: [0.29292602 0.10633095 0.14849463 0.4522484]

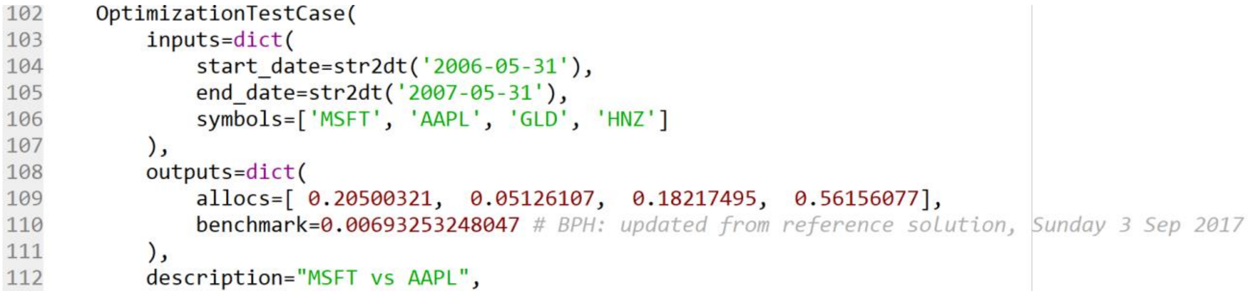
Sharpe Ratio: 1.8503796549242824

Volatility (stdev of daily returns): 0.006881551888743965

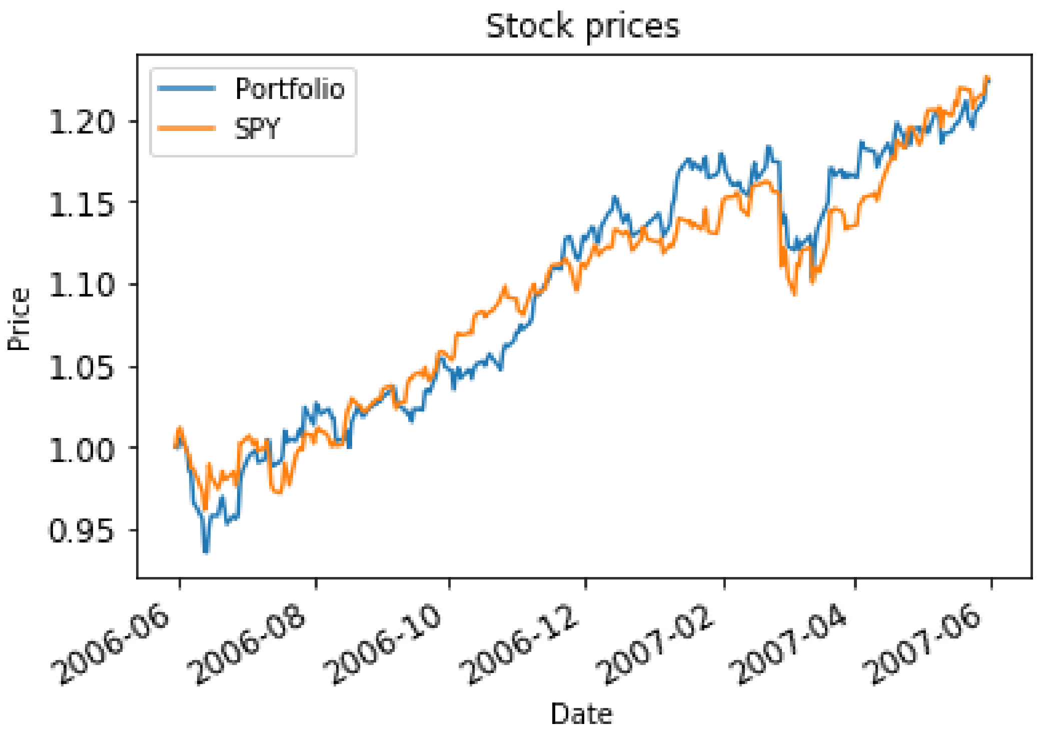
Average Daily Return: 0.0007974016746239188

Cumulative Return: 0.3354664739924902

MSFT vs AAPL



```
    OptimizationTestCase(
        inputs=dict(
            start_date=str2dt('2006-05-31'),
            end_date=str2dt('2007-05-31'),
            symbols=['MSFT', 'AAPL', 'GLD', 'HNZ']
        ),
        outputs=dict(
            allocs=[ 0.20500321,  0.05126107,  0.18217495,  0.56156077],
            benchmark=0.00693253248047 # BPH: updated from reference solution, Sunday 3 Sep 2017
        ),
        description="MSFT vs AAPL",
```



Start Date: 2006-05-31

End Date: 2007-05-31

Symbols: ['MSFT', 'AAPL', 'GLD', 'HNZ']

Allocations: [0.20500314 0.05126108 0.18217498 0.56156079]

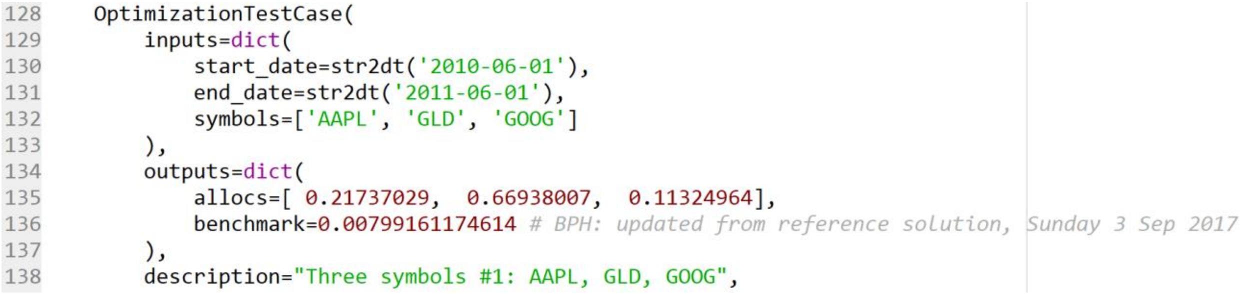
Sharpe Ratio: 1.9036061637809298

Volatility (stdev of daily returns): 0.006932532458168915

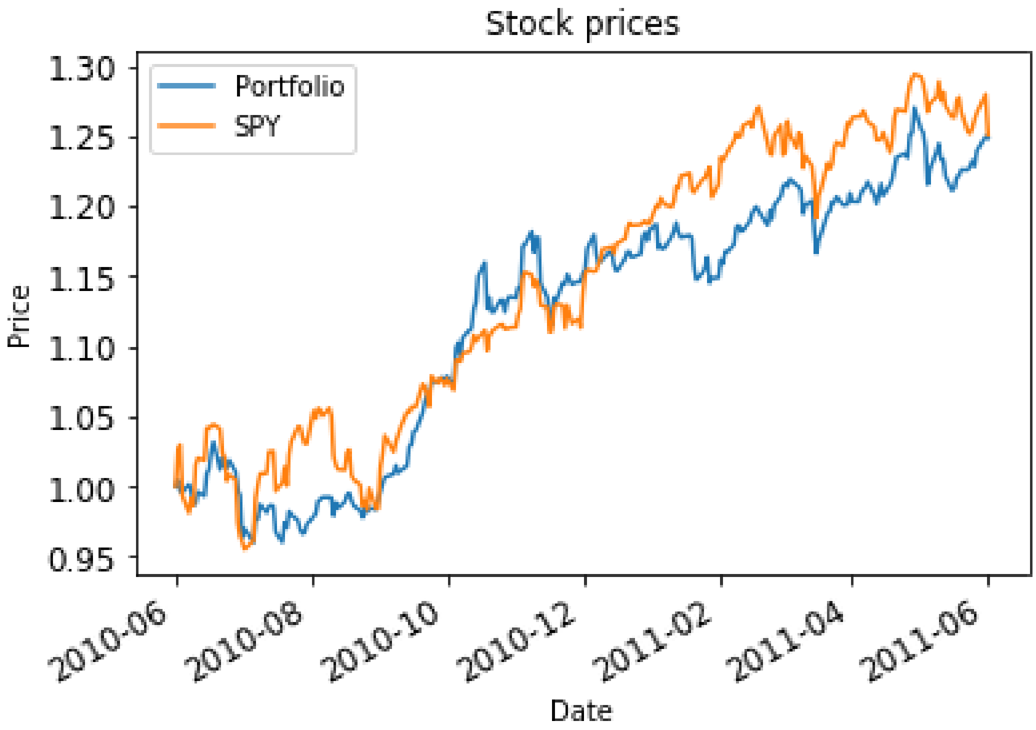
Average Daily Return: 0.000826416393743243

Cumulative Return: 0.22302855884452155

Apple Gld Goog



```
    OptimizationTestCase(
        inputs=dict(
            start_date=str2dt('2010-06-01'),
            end_date=str2dt('2011-06-01'),
            symbols=['AAPL', 'GLD', 'GOOG']
        ),
        outputs=dict(
            allocs=[ 0.21737029,  0.66938007,  0.11324964],
            benchmark=0.00799161174614 # BPH: updated from reference solution, Sunday 3 Sep 2017
        ),
        description="Three symbols #1: AAPL, GLD, GOOG",
```



Start Date: 2010-06-01

End Date: 2011-06-01

Symbols: ['AAPL', 'GLD', 'GOOG']

Allocations: [0.21737034 0.66938008 0.11324958]

Sharpe Ratio: 1.815440096841097

Volatility (stdev of daily returns): 0.00799161179739867

Average Daily Return: 0.0009085445182779028

Cumulative Return: 0.24819364129477273