Rebalancing Portfolios

In [1]:
```
import pandas as pd
import numpy  as np

from tiingo import TiingoClient
tiingo = TiingoClient({'api_key':'XXXX'})

import matplotlib.pyplot as plt                          # Basic plot library.
plt.style.use('ggplot')                                  # Make plots look nice.
```

Get S&P500 sector ETFs for technology and consumer staples:

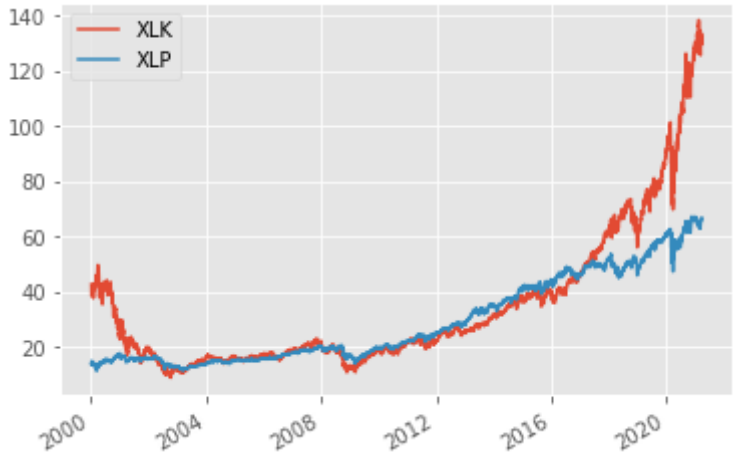
In [2]:
```
PRICE       = tiingo.get_dataframe(['XLK', 'XLP'],'2000-01-01', metric_name='adj
PRICE.index = pd.to_datetime(PRICE.index).tz_convert(None)
PRICE[-3:]
```

Out[2]:

	XLK	XLP
2021-03-18	129.721302	65.797993
2021-03-19	129.332108	66.026769
2021-03-22	131.870000	66.720000

In [3]:
```
PRICE.plot()
```

Out[3]: <AxesSubplot:>



Calculate returns:

In [4]:
```
RET = PRICE.pct_change()
RET
```

Out[4]:

	XLK	XLP
2000-01-03	NaN	NaN
2000-01-04	-0.050685	-0.028132

	XLK	XLP
2000-01-05	0.000570	0.014925
2000-01-06	-0.048044	0.021390
2000-01-07	0.017355	0.063700
...	...	...
2021-03-16	0.007530	0.001204
2021-03-17	-0.000822	-0.001503
2021-03-18	-0.027749	-0.004065
2021-03-19	-0.003000	0.003477
2021-03-22	0.019623	0.010499

5338 rows × 2 columns

Suppose we form an equal-weight portfolio:

In [5]:
```
weights = pd.Series({'XLK':1/2, 'XLP':1/2})
weights
```

Out[5]:
```
XLK    0.5
XLP    0.5
dtype: float64
```

Calculate portfolio return:

In [6]:
```
RET * weights
```

Out[6]:

	XLK	XLP
2000-01-03	NaN	NaN
2000-01-04	-0.025343	-0.014066
2000-01-05	0.000285	0.007463
2000-01-06	-0.024022	0.010695
2000-01-07	0.008677	0.031850
...	...	...
2021-03-16	0.003765	0.000602
2021-03-17	-0.000411	-0.000752
2021-03-18	-0.013874	-0.002033
2021-03-19	-0.001500	0.001738
2021-03-22	0.009812	0.005250

5338 rows × 2 columns

In [7]:
```
r_equal_weight = (RET * weights).sum('columns')
```

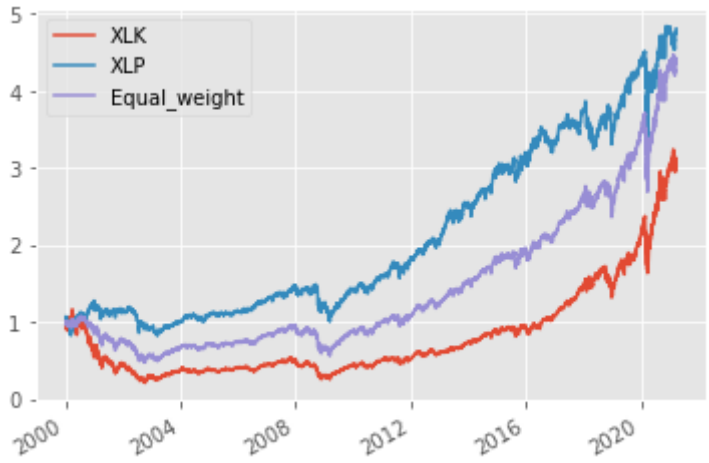
```
r_equal_weight
```

Out[7]:
```
2000-01-03    0.000000
2000-01-04   -0.039409
2000-01-05    0.007748
2000-01-06   -0.013327
2000-01-07    0.040527
                ...
2021-03-16    0.004367
2021-03-17   -0.001163
2021-03-18   -0.015907
2021-03-19    0.000238
2021-03-22    0.015061
Length: 5338, dtype: float64
```

Merge the return tables and the portfolio return and plot them together:

In [8]:
```
RET.join(r_equal_weight.rename('Equal_weight'))  .add(1).cumprod().plot()
```

Out[8]:
```
<AxesSubplot:>
```



Calculate compound returns for each asset:

In [9]:
```
cum_ret = RET[1:].add(1).cumprod()
cum_ret
```

Out[9]:

	XLK	XLP
2000-01-04	0.949315	0.971868
2000-01-05	0.949856	0.986374
2000-01-06	0.904221	1.007473
2000-01-07	0.919913	1.071648
2000-01-10	0.954906	1.047473
...	...	...
2021-03-16	3.128862	4.766872
2021-03-17	3.126290	4.759705
2021-03-18	3.039539	4.740357

	XLK	XLP
2021-03-19	3.030420	4.756839
2021-03-22	3.089886	4.806782

5337 rows × 2 columns

Weight the compound returns:

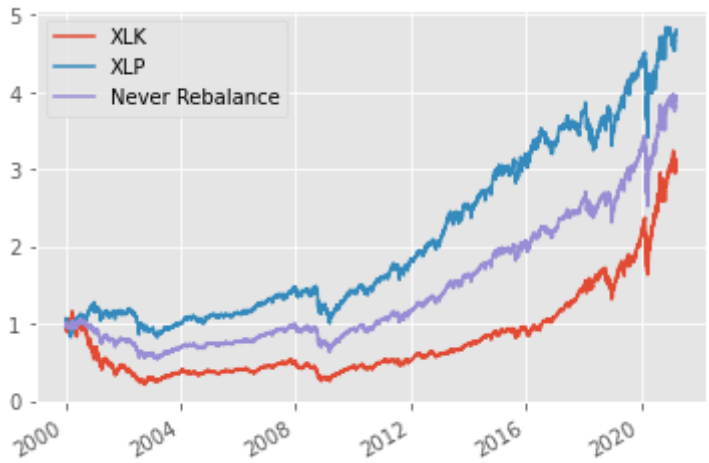
In [10]:
```
r_never_rebalance = (cum_ret * weights).sum('columns')
r_never_rebalance
```

Out[10]:
```
2000-01-04    0.960591
2000-01-05    0.968115
2000-01-06    0.955847
2000-01-07    0.995781
2000-01-10    1.001189
                ...
2021-03-16    3.947867
2021-03-17    3.942998
2021-03-18    3.889948
2021-03-19    3.893630
2021-03-22    3.948334
Length: 5337, dtype: float64
```

Plot the compound returns and the weighted compound return:

In [11]:
```
cum_ret.join(r_never_rebalance.rename('Never Rebalance')) .plot()
```

Out[11]:
```
<AxesSubplot:>
```



Rebalance Dates

We form the initial portfolio on the first date:

In [12]:
```
PRICE[:1].index
```

Out[12]:
```
DatetimeIndex(['2000-01-03'], dtype='datetime64[ns]', freq=None)
```

Now suppose we rebalance at the end of each month.

Get the last trading date of each month:

In [13]:
```
PRICE.groupby([PRICE.index.year, PRICE.index.month]).tail(1)
```

Out[13]:

	XLK	XLP
2000-01-31	38.921303	14.173248
2000-02-29	43.016662	12.519804
2000-03-31	46.619346	12.988841
2000-04-28	42.339234	13.674397
2000-05-31	37.935954	14.653763
...	...	...
2020-11-30	122.939063	65.995006
2020-12-31	129.751240	67.091075
2021-01-29	128.663493	63.748955
2021-02-26	130.419855	62.963158
2021-03-22	131.870000	66.720000

255 rows × 2 columns

In [14]:
```
PRICE.groupby([PRICE.index.year, PRICE.index.month]).tail(1).index
```

Out[14]:
```
DatetimeIndex(['2000-01-31', '2000-02-29', '2000-03-31', '2000-04-28',
               '2000-05-31', '2000-06-30', '2000-07-31', '2000-08-31',
               '2000-09-29', '2000-10-31',
               ...
               '2020-06-30', '2020-07-31', '2020-08-31', '2020-09-30',
               '2020-10-30', '2020-11-30', '2020-12-31', '2021-01-29',
               '2021-02-26', '2021-03-22'],
              dtype='datetime64[ns]', length=255, freq=None)
```

Now combine the first portfolio formation date with all the rebalance dates:

In [15]:
```
PRICE[:1].index.union(PRICE.groupby([PRICE.index.year, PRICE.index.month]).tail(
```

Out[15]:
```
DatetimeIndex(['2000-01-03', '2000-01-31', '2000-02-29', '2000-03-31',
               '2000-04-28', '2000-05-31', '2000-06-30', '2000-07-31',
               '2000-08-31', '2000-09-29',
               ...
               '2020-06-30', '2020-07-31', '2020-08-31', '2020-09-30',
               '2020-10-30', '2020-11-30', '2020-12-31', '2021-01-29',
               '2021-02-26', '2021-03-22'],
              dtype='datetime64[ns]', length=256, freq=None)
```

Same for quarterly rebalance dates:

In [16]:
```
PRICE[:1].index.union(PRICE.groupby([PRICE.index.year, PRICE.index.quarter]).tai
```

Out[16]:
```
DatetimeIndex(['2000-01-03', '2000-03-31', '2000-06-30', '2000-09-29',
               '2000-12-29', '2001-03-30', '2001-06-29', '2001-09-28',
```

```
               '2001-12-31', '2002-03-28', '2002-06-28', '2002-09-30',
               '2002-12-31', '2003-03-31', '2003-06-30', '2003-09-30',
               '2003-12-31', '2004-03-31', '2004-06-30', '2004-09-30',
               '2004-12-31', '2005-03-31', '2005-06-30', '2005-09-30',
               '2005-12-30', '2006-03-31', '2006-06-30', '2006-09-29',
               '2006-12-29', '2007-03-30', '2007-06-29', '2007-09-28',
               '2007-12-31', '2008-03-31', '2008-06-30', '2008-09-30',
               '2008-12-31', '2009-03-31', '2009-06-30', '2009-09-30',
               '2009-12-31', '2010-03-31', '2010-06-30', '2010-09-30',
               '2010-12-31', '2011-03-31', '2011-06-30', '2011-09-30',
               '2011-12-30', '2012-03-30', '2012-06-29', '2012-09-28',
               '2012-12-31', '2013-03-28', '2013-06-28', '2013-09-30',
               '2013-12-31', '2014-03-31', '2014-06-30', '2014-09-30',
               '2014-12-31', '2015-03-31', '2015-06-30', '2015-09-30',
               '2015-12-31', '2016-03-31', '2016-06-30', '2016-09-30',
               '2016-12-30', '2017-03-31', '2017-06-30', '2017-09-29',
               '2017-12-29', '2018-03-29', '2018-06-29', '2018-09-28',
               '2018-12-31', '2019-03-29', '2019-06-28', '2019-09-30',
               '2019-12-31', '2020-03-31', '2020-06-30', '2020-09-30',
               '2020-12-31', '2021-03-22'],
              dtype='datetime64[ns]', freq=None)
```

Define a function to get rebalance dates:

In [17]:

```
def get_rebalance_dates(frequency):
    return PRICE[:1].index.union(PRICE.groupby([PRICE.index.year, PRICE.index.qu

get_rebalance_dates('month')
```

Out[17]:

```
DatetimeIndex(['2000-01-03', '2000-03-31', '2000-06-30', '2000-09-29',
               '2000-12-29', '2001-03-30', '2001-06-29', '2001-09-28',
               '2001-12-31', '2002-03-28', '2002-06-28', '2002-09-30',
               '2002-12-31', '2003-03-31', '2003-06-30', '2003-09-30',
               '2003-12-31', '2004-03-31', '2004-06-30', '2004-09-30',
               '2004-12-31', '2005-03-31', '2005-06-30', '2005-09-30',
               '2005-12-30', '2006-03-31', '2006-06-30', '2006-09-29',
               '2006-12-29', '2007-03-30', '2007-06-29', '2007-09-28',
               '2007-12-31', '2008-03-31', '2008-06-30', '2008-09-30',
               '2008-12-31', '2009-03-31', '2009-06-30', '2009-09-30',
               '2009-12-31', '2010-03-31', '2010-06-30', '2010-09-30',
               '2010-12-31', '2011-03-31', '2011-06-30', '2011-09-30',
               '2011-12-30', '2012-03-30', '2012-06-29', '2012-09-28',
               '2012-12-31', '2013-03-28', '2013-06-28', '2013-09-30',
               '2013-12-31', '2014-03-31', '2014-06-30', '2014-09-30',
               '2014-12-31', '2015-03-31', '2015-06-30', '2015-09-30',
               '2015-12-31', '2016-03-31', '2016-06-30', '2016-09-30',
               '2016-12-30', '2017-03-31', '2017-06-30', '2017-09-29',
               '2017-12-29', '2018-03-29', '2018-06-29', '2018-09-28',
               '2018-12-31', '2019-03-29', '2019-06-28', '2019-09-30',
               '2019-12-31', '2020-03-31', '2020-06-30', '2020-09-30',
               '2020-12-31', '2021-03-22'],
              dtype='datetime64[ns]', freq=None)
```

But this function always returns the same frequency (quarters).

How can we change "PRICE.index.quarter" to depend on the input frequency of the function?

Try:

In [18]:

```
frequency = 'month'

PRICE.index.frequency
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-18-897d86383884> in <module>
      1 frequency = 'month'
      2
----> 3 PRICE.index.frequency

AttributeError: 'DatetimeIndex' object has no attribute 'frequency'
```

So this does not work. If we want to use the variable frequency as an attribute we have to write:

In [19]:
```
getattr(PRICE.index, frequency)  # Same as Price.index.month if frequency = 'mon
```

Out[19]:
```
Int64Index([1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            ...
            3, 3, 3, 3, 3, 3, 3, 3, 3, 3],
           dtype='int64', length=5338)
```

And put this into our function:

In [20]:
```
def get_rebalance_dates(frequency):
    group = getattr(PRICE.index, frequency)
    return PRICE[:1].index.union(PRICE.groupby([PRICE.index.year, group]).tail(1
```

And now we can use the function like this:

In [21]:
```
get_rebalance_dates('year')
```

Out[21]:
```
DatetimeIndex(['2000-01-03', '2000-12-29', '2001-12-31', '2002-12-31',
               '2003-12-31', '2004-12-31', '2005-12-30', '2006-12-29',
               '2007-12-31', '2008-12-31', '2009-12-31', '2010-12-31',
               '2011-12-30', '2012-12-31', '2013-12-31', '2014-12-31',
               '2015-12-31', '2016-12-30', '2017-12-29', '2018-12-31',
               '2019-12-31', '2020-12-31', '2021-03-22'],
              dtype='datetime64[ns]', freq=None)
```

In [22]:
```
get_rebalance_dates('month')
```

Out[22]:
```
DatetimeIndex(['2000-01-03', '2000-01-31', '2000-02-29', '2000-03-31',
               '2000-04-28', '2000-05-31', '2000-06-30', '2000-07-31',
               '2000-08-31', '2000-09-29',
               ...
               '2020-06-30', '2020-07-31', '2020-08-31', '2020-09-30',
               '2020-10-30', '2020-11-30', '2020-12-31', '2021-01-29',
               '2021-02-26', '2021-03-22'],
              dtype='datetime64[ns]', length=256, freq=None)
```