In [6]:
```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
```

Read 'SP500_data' csv file.

In [7]:
```
1 sp500 = pd.read_csv('../Data/SP500_data.csv')
2 sp500
```

Out[7]:

	year	dividendYield	Peratio	ShillerPEratio	10yearTyield	SP500level	annual_return
0	1950	0.0744	7.47	11.90	0.0257	21.21	NaN
1	1951	0.0602	9.95	12.53	0.0268	24.19	0.140500
2	1952	0.0541	10.86	13.01	0.0283	26.18	0.082265
3	1953	0.0584	10.10	12.00	0.0248	25.46	-0.027502
4	1954	0.0440	12.58	15.99	0.0261	35.60	0.398272
...	...	...	...	...	...	...	...
65	2015	0.0211	22.18	24.21	0.0209	1918.60	-0.054029
66	2016	0.0203	23.59	28.06	0.0243	2275.12	0.185823
67	2017	0.0185	24.97	33.31	0.0258	2789.80	0.226221
68	2018	0.0209	19.60	28.38	0.0271	2607.39	-0.065385
69	2019	0.0181	24.47	31.31	0.0183	3265.38	0.252356

In pandas, all the mathematical operators will apply the same operation to each element in the Series. It would usually require a ‘for’ loop to iterate each of the items in the sequence. But pandas is built on top of the NumPy library that can operate on entire sequences of data without the writing for loops.

In [8]:
```
sp500['indicator'] = sp500.dividendYield > 0.03
sp500['indicator'] = sp500['indicator'].shift()
sp500['port_return'] = 0.6 * sp500['annual_return'] + 0.4 * sp500['10yea
sp500
```

Out[8]:

	year	dividendYield	Peratio	ShillerPEratio	10yearTyield	SP500level	annual_return	indica
0	1950	0.0744	7.47	11.90	0.0257	21.21	NaN	N
1	1951	0.0602	9.95	12.53	0.0268	24.19	0.140500	T
2	1952	0.0541	10.86	13.01	0.0283	26.18	0.082265	T
3	1953	0.0584	10.10	12.00	0.0248	25.46	-0.027502	T
4	1954	0.0440	12.58	15.99	0.0261	35.60	0.398272	T
...	...	...	...	...	...	...	...	
65	2015	0.0211	22.18	24.21	0.0209	1918.60	-0.054029	Fa
66	2016	0.0203	23.59	28.06	0.0243	2275.12	0.185823	Fa
67	2017	0.0185	24.97	33.31	0.0258	2789.80	0.226221	Fa
68	2018	0.0209	19.60	28.38	0.0271	2607.39	-0.065385	Fa
69	2019	0.0181	24.47	31.31	0.0183	3265.38	0.252356	Fa

70 rows × 9 columns

The ‘where()’ method from numpy will process elements depending on the condition provided. In the below code, we gave the boolean column ‘indicator’ as the condition. If it is true, it will return the result of the second argument, and if not, it will return the result of the third argument.

In [9]:
```
sp500['strat_return'] =  np.where(sp500['indicator'],
                                  0.8 * sp500['annual_return'] + 0.2 * sp500
                                  0.6 * sp500['annual_return'] + 0.4 * sp500

sp500
```

Out[9]:

	year	dividendYield	Peratio	ShillerPEratio	10yearTyield	SP500level	annual_return	indica
0	1950	0.0744	7.47	11.90	0.0257	21.21	NaN	N
1	1951	0.0602	9.95	12.53	0.0268	24.19	0.140500	T
2	1952	0.0541	10.86	13.01	0.0283	26.18	0.082265	T
3	1953	0.0584	10.10	12.00	0.0248	25.46	-0.027502	T
4	1954	0.0440	12.58	15.99	0.0261	35.60	0.398272	T
...	...	...	...	...	...	...	...	
65	2015	0.0211	22.18	24.21	0.0209	1918.60	-0.054029	Fa
66	2016	0.0203	23.59	28.06	0.0243	2275.12	0.185823	Fa
67	2017	0.0185	24.97	33.31	0.0258	2789.80	0.226221	Fa
68	2018	0.0209	19.60	28.38	0.0271	2607.39	-0.065385	Fa
69	2019	0.0181	24.47	31.31	0.0183	3265.38	0.252356	Fa

70 rows × 10 columns

In [10]:
```
sp500[['port_return','strat_return']].describe()
```

Out[10]:

	port_return	strat_return
count	69.000000	69.000000
mean	0.074909	0.079782
std	0.093835	0.109669
min	-0.213242	-0.213242
25%	0.006805	-0.004167
50%	0.086759	0.087278
75%	0.147563	0.150035
max	0.249403	0.323837

The `cumprod()` method will return cumulative product over a DataFrame.

In [12]:
```
sp500['port_cr'] = (sp500['port_return'] + 1).cumprod() - 1
sp500['strat_cr'] = (sp500['strat_return'] + 1).cumprod() - 1
sp500
```

Out[12]:

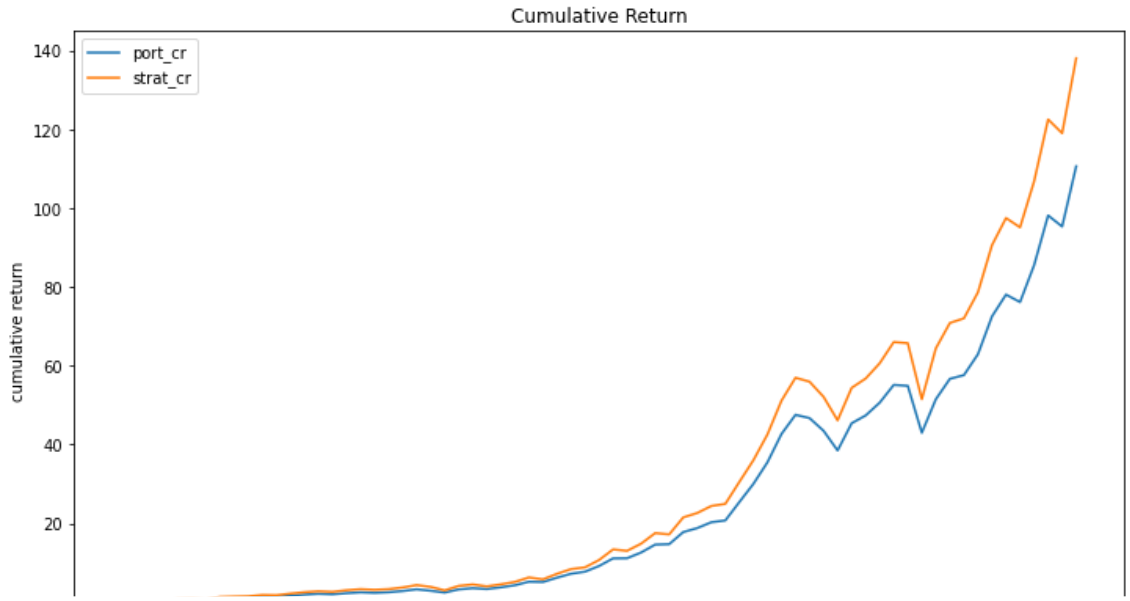
	year	dividendYield	Peratio	ShillerPEratio	10yearTyield	SP500level	annual_return	indica
0	1950	0.0744	7.47	11.90	0.0257	21.21	NaN	N
1	1951	0.0602	9.95	12.53	0.0268	24.19	0.140500	T
2	1952	0.0541	10.86	13.01	0.0283	26.18	0.082265	T
3	1953	0.0584	10.10	12.00	0.0248	25.46	-0.027502	T
4	1954	0.0440	12.58	15.99	0.0261	35.60	0.398272	T
...	...	...	...	...	...	...	...	
65	2015	0.0211	22.18	24.21	0.0209	1918.60	-0.054029	Fa
66	2016	0.0203	23.59	28.06	0.0243	2275.12	0.185823	Fa
67	2017	0.0185	24.97	33.31	0.0258	2789.80	0.226221	Fa
68	2018	0.0209	19.60	28.38	0.0271	2607.39	-0.065385	Fa
69	2019	0.0181	24.47	31.31	0.0183	3265.38	0.252356	Fa

70 rows × 12 columns

We set the `year` as the index so pandas can automatically recognize it and use it as an x axis.

In [13]:
```
1 plot_data = sp500[['year','port_cr','strat_cr']].set_index('year')
2 ax = plot_data.plot(figsize = (12, 7))
3 ax.set_title('Cumulative Return')
4 ax.set_ylabel('cumulative return')
```

Out[13]: Text(0, 0.5, 'cumulative return')



In [ ]:
```
1
```