Shift

Shift 은 이전 row 나 이후 row 에 있는 값을 가져올 수 있는 메소드입니다. 일단 삼성전자 일봉을 가져오겠습니다.

```
import FinanceDataReader as fdr

code = '005930' # 삼성전자
stock_data = fdr.DataReader(code, start='2021-01-03', end='2021-12-31')

stock_data.head().style.set_table_attributes('style="font-size: 12px"')
```

Date	Open	High	Low	Close	Volume	Change
2021-01-04 00:00:00	81000	84400	80200	83000	38655276	0.024691
2021-01-05 00:00:00	81600	83900	81600	83900	35335669	0.010843
2021-01-06 00:00:00	83300	84500	82100	82200	42089013	-0.020262
2021-01-07 00:00:00	82800	84200	82700	82900	32644642	0.008516
2021-01-08 00:00:00	83300	90000	83000	88800	59013307	0.071170

일봉 데이터에서 전날의 종가를 당일로 가져와 보겠습니다. 아래 예제를 보시면 2021년 1월 5일 'Previous Close' 컬럼에 1월 4일 종가가 들어가 있습니다. 1월 4일은 전날이 없어서 NaN (값없음) 처리 되었습니다.

```
stock_data['Previous Close'] = stock_data['Close'].shift(1)
stock_data.head(6).style.set_table_attributes('style="font-size: 12px"')
```

Date	Open	High	Low	Close	Volume	Change	Previous Close
2021-01-04 00:00:00	81000	84400	80200	83000	38655276	0.024691	nan
2021-01-05 00:00:00	81600	83900	81600	83900	35335669	0.010843	83000.000000
2021-01-06 00:00:00	83300	84500	82100	82200	42089013	-0.020262	83900.000000
2021-01-07 00:00:00	82800	84200	82700	82900	32644642	0.008516	82200.000000
2021-01-08 00:00:00	83300	90000	83000	88800	59013307	0.071170	82900.000000
2021-01-11 00:00:00	90000	96800	89500	91000	90306177	0.024775	88800.000000

이제 아주 단순한 전략을 구현해 보겠습니다. 구현해 볼 단순 전략은 '전날 종가보다 오늘 종가가 높으면 내일 시가에 매수하고 내일 종가에 매도' 입니다. 결과가 어떨지 정말 궁금합니다. 이 전략을 구현하면 수익율이 어떻게 될 지 테스트 해보겠습니다. 먼저 전날 종가보다 오늘 종가가 높은 날을 찾아야 합니다. 전날 종가는 이미 만들어서 'Previous Close' 컬럼에 저장해 두었습니다. 오늘 종가와 전날 종가를 비교한 후, True 이면 1 되도록 하겠습니다. 조건 (stock_data['Close'] > stock_data['Previous Close']) 는 True/False 를 반환합니다. 그래서, astype(int) 를 이용해서 정수로 변환합니다.
그 다음 수익율 데이터를 만들어 보겠습니다. 내일의 시가는 stock_data['Open'].shift(-1), 내일의 종가는 stock_data['Close'].shift(-1) 로 가져오면 됩니다. 결과를 컬럼 'return' 에 넣겠습니다.
shift(1) 는 전날의 정보를 shift(-1) 은 다음날의 데이터를 가져옵니다.

```
stock_data['buy'] = (stock_data['Close'] > stock_data['Previous Close']).astype(int)
# 매수 시그널 생성
stock_data['return'] = stock_data['Close'].shift(-1) / stock_data['Open'].shift(-1) #
전략의 수익율
stock_data.head(6).style.set_table_attributes('style="font-size: 12px"')
```

Date	Open	High	Low	Close	Volume	Change	Previous Close	buy	return
2021-01-04 00:00:00	81000	84400	80200	83000	38655276	0.024691	nan	0	1.028186
2021-01-05 00:00:00	81600	83900	81600	83900	35335669	0.010843	83000.000000	1	0.986795
2021-01-06 00:00:00	83300	84500	82100	82200	42089013	-0.020262	83900.000000	0	1.001208
2021-01-07 00:00:00	82800	84200	82700	82900	32644642	0.008516	82200.000000	1	1.066026
2021-01-08 00:00:00	83300	90000	83000	88800	59013307	0.071170	82900.000000	1	1.011111
2021-01-11 00:00:00	90000	96800	89500	91000	90306177	0.024775	88800.000000	1	1.003322

이제 buy 시그널이 1 인 날의 수익율과 0 인 날의 수익율을 groupby 을 이용해서 비교해보겠습니다. 결과가 실망입니다. 좋은 전략이 아닌 것 같습니다. 100 원을 투자했으면 평균 기대수익율이 99.8 원입니다. 여기서 평균 수익율은 buy 가 1 인 날 중 랜덤한 날에 투자했을 때 기대할 수 있는 수익율이 0.998 (0.2% 손실) 이라는 의미입니다. describe 메소드로 수익율의 분포도 확인해 보겠습니다. buy 가 1 인 날(매수)은 0 인 날에 비하여 평균도 낮고, 변동성(std) 이 더 큽니다. 차라리 전날 종가보다 오늘 종가가 높을 때 매수하는 것이 더 좋을 것 같습니다.

```
import pandas as pd
pd.options.display.float_format = '{:,.3f}'.format

stock_data.dropna(inplace=True) # NaN(값 없음) 열 전부 제거
print(stock_data.groupby('buy')['return'].mean()) # 평균 비교
print('\n')
print(stock_data.groupby('buy')['return'].describe()) # 분포 비교
```

```
buy
0   0.999
1   0.998
Name: return, dtype: float64


      count  mean   std   min   25%   50%   75%   max
buy
0   136.000 0.999 0.011 0.970 0.992 1.000 1.005 1.033
1   110.000 0.998 0.013 0.975 0.990 0.997 1.004 1.066
```

위에서 구현한 단순 전략은 손실을 보는 전략입니다. 이번에는 만약 우리가 100 원을 투자했으면 110 영업일 이후에 얼마나 손해를 보는 지 확인 해 보겠습니다. 위 describe 결과에서 buy 가 1 인 날은 110일 입니다. 이번에 쓸 메소드는 prod 입니 다. prod 는 값을 다 곱하라는 뜻입니다. 만약 당일 수익율이 0.9 이고 다음날 1.1 이면, 최종 수익율은 0.99 (=0.9 x 1.1) 가 됩니다. 아래 결과에서와 같이 단순 전략으로 2021년 초에 삼성전자에 100원을 투자하면 110 일 이후인 2021년 연말에는 잔고가 81.1 원이 됩니다. 약 19% 의 손실이 발생했습니다.

```
print(stock_data.groupby('buy')['return'].prod())
```

```
buy
0   0.843
1   0.811
Name: return, dtype: float64
```