Rolling

주식을 하신 분들은 이동평균선에 대하여 많이 들어보셨을 것이라고 생각합니다. rolling 은 이동평균선을 간단하게 만들어줄 수 있는 메소드입니다. 예제를 보시면 금방 이해가 되 실 것이라고 생각합니다. 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

일봉의 종가에 대하여 5 일 이동평균선을 만들어 '5 day moving average' 라는 이름의 컬럼에 담았습니다. rolling(5) 은 5 개 row 로 만들어진 창(window) 을 한 단계씩 진행하라는 뜻이고, mean() 을 한 이유는 각 창의 평균값을 구하라는 뜻입니다. 처음 4개의 row 에는 5 일의 창이 만들어지지 않으므로 'NaN'(값 없음) 이 되고 처음으로 시작하는 '5 day moving average' 값은 2021년 1월 8일부터 시작하게 됩니다. 2021년 1월 8일의 5일 이동평균선 값 84,160 은 1월 4일 ~ 1월 8일까지 5일 종가들의 평균값입니다.

```
stock_data['5 day moving average'] = stock_data['Close'].rolling(5).mean()
stock_data.head(6).style.set_table_attributes('style="font-size: 12px"')
```

Date	Open	High	Low	Close	Volume	Change	5 day moving average
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	nan
2021-01-06 00:00:00	83300	84500	82100	82200	42089013	-0.020262	nan
2021-01-07 00:00:00	82800	84200	82700	82900	32644642	0.008516	nan
2021-01-08 00:00:00	83300	90000	83000	88800	59013307	0.071170	84160.000000
2021-01-11 00:00:00	90000	96800	89500	91000	90306177	0.024775	85760.000000

같은 방식으로 20일 이동평균선도 만들어 보겠습니다. 그리고 골든크로스(5일 이동평균선이 20일 이동평균선을 뚫고 올라가는) 지점이 어디 인지도 알아보겠습니다. 5일 이동평균선과 동일하게 20일 이동평균선은 20번째 열부터 존재합니다.

```
stock_data['20 day moving average'] = stock_data['Close'].rolling(20).mean()
stock_data.head(21).style.set_table_attributes('style="font-size: 12px"')
```

Date	Open	High	Low	Close	Volume	Change	5 day moving average	20 day moving average
2021-01-04 00:00:00	81000	84400	80200	83000	38655276	0.024691	nan	nan
2021-01-05 00:00:00	81600	83900	81600	83900	35335669	0.010843	nan	nan
2021-01-06 00:00:00	83300	84500	82100	82200	42089013	-0.020262	nan	nan
2021-01-07 00:00:00	82800	84200	82700	82900	32644642	0.008516	nan	nan
2021-01-08 00:00:00	83300	90000	83000	88800	59013307	0.071170	84160.000000	nan
2021-01-11 00:00:00	90000	96800	89500	91000	90306177	0.024775	85760.000000	nan
2021-01-12 00:00:00	90300	91400	87800	90600	48682416	-0.004396	87100.000000	nan
2021-01-13 00:00:00	89800	91200	89100	89700	36068848	-0.009934	88600.000000	nan
2021-01-14 00:00:00	88700	90000	88700	89700	26393970	0.000000	89960.000000	nan
2021-01-15 00:00:00	89800	91800	88000	88000	33431809	-0.018952	89800.000000	nan
2021-01-18 00:00:00	86600	87300	84100	85000	43227951	-0.034091	88600.000000	nan
2021-01-19 00:00:00	84500	88000	83600	87000	39895044	0.023529	87880.000000	nan
2021-01-20 00:00:00	89000	89000	86500	87200	25211127	0.002299	87380.000000	nan
2021-01-21 00:00:00	87500	88600	86500	88100	25318011	0.010321	87060.000000	nan
2021-01-22 00:00:00	89000	89700	86800	86800	30861661	-0.014756	86820.000000	nan
2021-01-25 00:00:00	87000	89900	86300	89400	27258534	0.029954	87700.000000	nan
2021-01-26 00:00:00	88800	89200	86500	86700	33178936	-0.030201	87640.000000	nan
2021-01-27 00:00:00	86600	87700	85600	85600	26423070	-0.012687	87320.000000	nan
2021-01-28 00:00:00	83200	85600	83200	83700	31859808	-0.022196	86440.000000	nan
2021-01-29 00:00:00	84500	85000	82000	82000	39615978	-0.020311	85480.000000	86565.000000
2021-02-01 00:00:00	81700	83400	81000	83000	28046832	0.012195	84200.000000	86565.000000

우선 NaN 으로 표시가 된 값이 없는 모든 열을 제거하고 싶습니다. dropna 라는 메소드도 활용할 것인데요. dropna 를 하면 NaN 가 있는 모든 열을 제거합니다. 제거한 후 자기 자신을 덮어쓰라고 명령하는 것은 inplace=True 라는 인수인데요. 새로운 DataFrame 을 만들지 않고 dropna(inplace=True) 하여 값이 없는 모든 열을 제거한 후, 자기 자신을 덮어쓰도록 하겠습니다.

```
stock_data.dropna(inplace=True) # NaN 이 있는 모든 row 제거
stock_data.head().style.set_table_attributes('style="font-size: 12px"')
```

Date	Open	High	Low	Close	Volume	Change	5 day moving average	20 day moving average
2021-01-29 00:00:00	84500	85000	82000	82000	39615978	-0.020311	85480.000000	86565.000000
2021-02-01 00:00:00	81700	83400	81000	83000	28046832	0.012195	84200.000000	86565.000000
2021-02-02 00:00:00	84100	86400	83700	84400	26302077	0.016867	83740.000000	86590.000000
2021-02-03 00:00:00	84800	85400	83400	84600	22112205	0.002370	83540.000000	86710.000000
2021-02-04 00:00:00	83500	83800	82100	82500	24171688	-0.024823	83300.000000	86690.000000

이제 5일 이동평균선이 20일 이동평균선보다 작았다가 커지는 지점을 찾으면 됩니다. DataFrame의 필터링에 대하여는 아직 다루지 않았습니다. 설명을 드리면, df(DataFrame) 에서 원하는 row를 가져오고 싶을 때는 df[조건] 처럼 대괄호 안에 조건을 넣어 주면 됩니다. 아래에서 stock_data['cross_flag'==1] 은 stock_data 에서 True 인 열과 False 인 열을 구분하는 역할을 합니다. stock_data['cross_flag'].shift(1)==0 은 전 날의 cross_flag 값이 0 인 경우를 찾는 것인데요. 결국 전날은 cross_flag 값이 0, 당일은 cross_flag 값이 1 날을 찾는 조건이 됩니다. 최종 결과를 보시면 2021년은 3월 3일에 최초 골든크로스가 일어났습니다.

```
stock_data['cross_flag'] = (stock_data['5 day moving average'] > stock_data['20 day moving average']).astype(int) # True/False 결과 값을 1/0 으로 바꿔줌
s = stock_data[(stock_data['cross_flag'].shift(1)==0) & (stock_data['cross_flag']==1)] # 조건 - 전날에는 5일 이평선이 20일 이평선보다 작거나 같아는데, 당일은 5일 이평선이 20일 이평선 보다 커짐
s.style.set_table_attributes('style="font-size: 12px"')
```

Date	Open	High	Low	Close	Volume	Change	5 day moving average	20 day moving average	cross_flag
2021-03-03 00:00:00	83500	84000	82800	84000	19882132	0.004785	83480.000000	83195.000000	1
2021-03-18 00:00:00	82800	83800	82600	82900	18585244	0.007290	82520.000000	82485.000000	1
2021-04-02 00:00:00	84000	85200	83900	84800	22997538	0.022919	82580.000000	82060.000000	1
2021-06-03 00:00:00	81300	83000	81100	82800	29546007	0.024752	80960.000000	80490.000000	1
2021-06-29 00:00:00	81900	82100	80800	81000	15744317	-0.010989	81160.000000	81150.000000	1
2021-08-04 00:00:00	82200	83100	81800	82900	25642368	0.018428	80220.000000	79590.000000	1
2021-09-03 00:00:00	76400	76700	76000	76600	12096419	0.007895	76140.000000	76060.000000	1
2021-11-03 00:00:00	71700	71700	70100	70400	12770428	-0.015385	70460.000000	70355.000000	1
2021-11-15 00:00:00	71700	71900	70900	71400	12420710	0.011331	70520.000000	70460.000000	1