

“건물주 위에 배당주”

파이썬 프로젝트

데이터로

투자하라

파이썬 데이터 분석을 통한

미국 배당성장주 투자

실전 가이드

목차

1. 주식투자기본 – 쌀사비파
 - 1.1 배당 성장 투자는 경기 침체에 강하다!
2. 파이선 주가 데이터 분석
 - 2.1 윈도우 개발 환경 구축
 - 2.2 배당귀족주 리스트 다운로드
 - 2.3 야후 파이낸스 모듈 설치
 - 2.4 데이터 수집
 - 2.5 데이터 처리
 - 2.6 데이터 분석
3. 데이터 분석 응용 – 한 방에 계산하자
4. 포트폴리오 운영 팁
 - 4.1 타임머신을 타고 과거에 투자했다면?
 - 4.2 세금을 아끼자
5. 마치며 – 지금 시작하자

1. 주식 투자 기본 - 쌀사비파

“좋은 회사의 주식을 싸게 사서 비싸게 팔아라~”

누구나 아는 주식 투자의 기본 원리입니다. 줄여서 ‘쌀사비파’라고도 합니다. 그러나 실전에서는 대부분 실패합니다. 왜 그럴까요?

첫째, 좋은 회사를 고르기가 어렵기 때문입니다. 더군다나 직장인이 퇴근 후에 매크로를 공부하고 유망한 산업에서 미래에 1등할 기업을 찾기란 짚더미에서 바늘 찾기입니다.

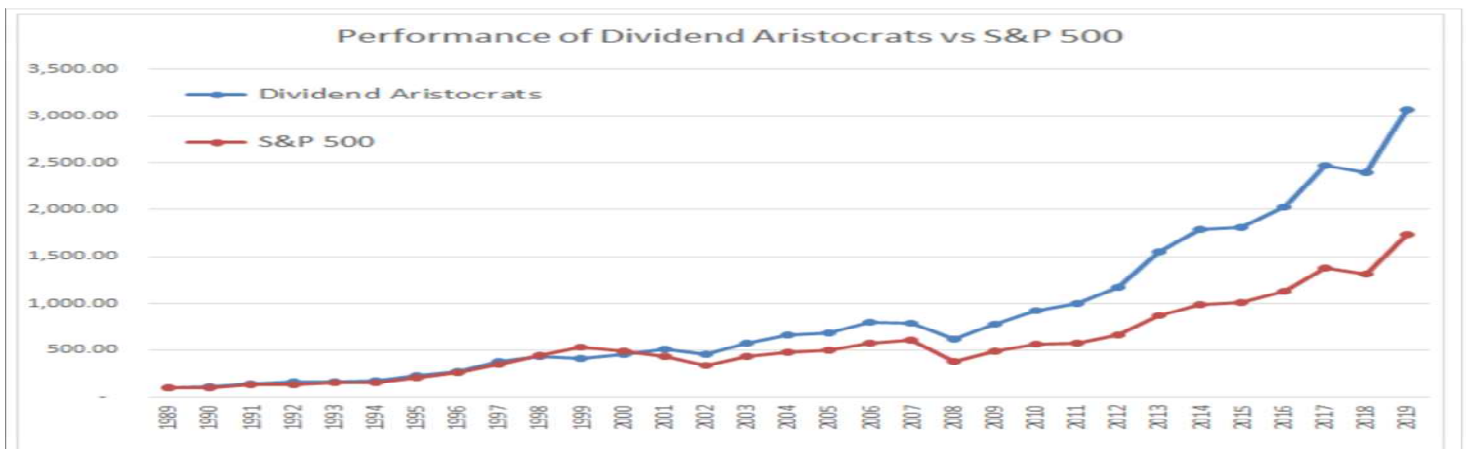
둘째, 싼 가격이 얼마이고 비싼 가격이 얼마인지 모르기 때문입니다. 설사 유망한 기업을 찾았다고 해도 현재 가격이 싼 가격인지 비싼 가격인지 평가할 방법도 없습니다.

하지만 실망하지 마세요. 우리같은 평범한 직장인도 좋은 회사를 싸게 사서 비싸게 팔 수 있는 방법이 있습니다.

바로 “배당 가치 투자법”입니다.

주식 시장의 원조라고 할 수 있는 미국에서는 이미 배당 가치 투자가 지수 투자보다 수익률이 높다는 것은 이미 증명되었습니다.

파란 선은 Dividend Aristocrats, 한국말로 번역하면 배당 귀족주이고 빨간 선은 미국의 대표 지수인 S&P 500 수익률입니다.



1. 주식 투자 기본 - 쌀사비파

배당귀족주란 S&P 500에 속한 기업으로 25년 이상 배당금을 인상해야 합니다. 그리고 시가총액 30억 달러 이상이고 하루 평균 거래량이 최소 500만 달러가 되어야 합니다.

과거 10년 기준으로 S&P 500 연평균 복리 수익률은 약 13%, 배당귀족주는 14%입니다.

원금 1억 원을 투자해서 10년 동안 14% 복리 수익을 올리면 최종 평가금액이 얼마인가요?

$$100,000,000 \times (1 + 0.14)^{10} = 370,722,131$$

3억 7천만 원입니다.

만일 5% 예금 상품에 10년간 묻어 두면 얼마가 될까요?

$$100,000,000 \times (1 + 0.05)^{10} = 162,889,462$$

1억 6천만 원입니다.

물론 과거 수익률이 미래 수익률을 100% 보장하지는 않습니다. 그렇게 될 확률이 높다는 겁니다.

여기서 핵심은 이런 겁니다.

“배당은 투자 수익과 밀접한 상관관계가 있다”

1. 주식 투자 기본 - 쌀사비파

이제 배당 가치 투자 전략에 대해서 설명해 보겠습니다.

첫째, 좋은 회사를 선정합니다. 전문 용어로 블루칩 주식이라고 합니다.

- 배당금 인상 - 지난 12년간 최소 5배 증가
- S&P 품질 순위 - "A" 등급 이상
- 유동성 - 최소 500만 주 발행
- 유동성 - 최소 80개 기관 투자자
- 연속 배당 - 최소 25년 연속
- 수입 - 지난 12년 중 7년 동안 개선

둘째, 배당수익률을 기준으로 매수 시점과 매매 시점을 찾는다.

- 배당수익률이 역사적 고점(주가 저점)일 때 매수한다.
- 배당수익률이 역사적 저점(주가 고점)일 때 매도한다.

배당수익률은 어떻게 계산하나요?

- 배당수익률 = 배당금 / 주가

배당수익률은 주가와 반비례 관계입니다. 따라서 배당수익률이 높을 때 사면 싸게 살 수 있고, 배당수익률이 낮을 때 팔면 비싸게 파는 겁니다.

1.1 배당 성장 투자는 경기 침체에 강하다!

배당주 중 성장률이 높은 종목에 투자하는 전략을 '배당 성장 투자'라고 하는데 다음과 같은 특징이 있습니다.

- 경기 침체기에도 큰 영향을 받지 않는다.
- 채권과 비슷한 변동성으로 주식의 수익률을 제공한다. (낮은 변동성 대비 높은 수익률 제공)
- 물가상승률과 시장 수익률을 이긴다.
- 장기 투자의 성과의 대부분은 배당으로 발생한다.

배당 성장 주식 투자는 포트폴리오를 구성해서 운영합니다.

- 종목은 10~20개를 권장한다.
- 산업 섹터별로 다양하게 분산한다.
- 배당률보다 배당 성장률에 집중한다.

섹터별로 분산하는 이유는 산업별로 경기 사이클이 다르기 때문에 포트폴리오 변동성을 줄여 줍니다. 그리고 변동성 낮아야 장기 수익률이 향상됩니다.

아래 그림은 제가 투자하는 배당 성장 포트폴리오 일부입니다.

<input type="checkbox"/>	KMB	김벌리클라크	129.1400	2.45%	134.8777
<input type="checkbox"/>	LMT	록히드 마틴	1,082.9232	33.00%	364.5808
<input type="checkbox"/>	LQD	ISHARES IBOXX \$ INVE	-1,092.3200	-21.86%	135.0338
<input type="checkbox"/>	MCD	맥도날드	770.0610	16.88%	227.9910
<input type="checkbox"/>	MRK	머크	1,712.9852	46.77%	76.3009
<input type="checkbox"/>	MSFT	마이크로소프트	-473.3800	-9.52%	261.5263
<input type="checkbox"/>	NEE	넥스테라 에너지	534.9400	11.89%	74.9732
<input type="checkbox"/>	O	리얼티 인컴	-125.8900	-2.77%	65.8436
<input type="checkbox"/>	OGN	오가논	109.6900	5,484.50%	0.5000
<input type="checkbox"/>	OHI	오메가 헬스케어 인베	-1,289.4700	-21.12%	35.6961

요즘같이 금리 인상과 인플레이션으로 시장이 폭락하는 상황에도 방산, 제약, 식음료 등 평가수익이 발생하는 업종이 있습니다.

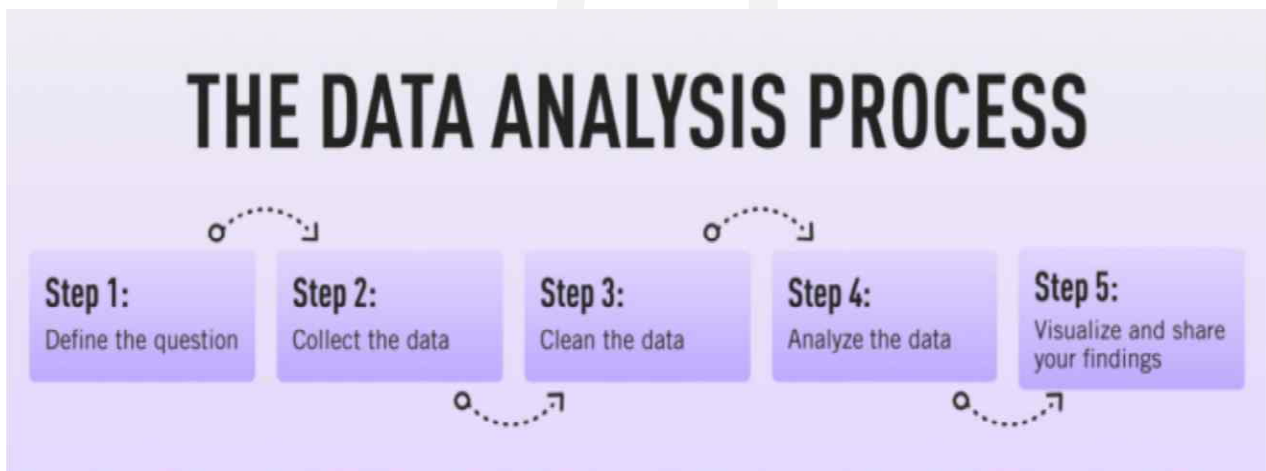
2. 파이썬 주가 데이터 분석 개요

“구슬이 서말이라도 꿰어야 보배다!”

배당 가치 투자 전략이 좋은 건 알겠는데 실제로 어떻게 적용할 수 있을까요?

아무리 좋은 전략이라도 써 먹을 수 없다면 말짱 도루묵입니다. 하지만 걱정하지 마세요. 파이썬 데이터 분석으로 배당 가치 투자 전략을 만들면 됩니다.

데이터 분석은 “문제 정의 -> 데이터 수집 -> 데이터 분석 -> 결과 정리”의 과정을 거쳐서 최종적으로 의사 결정에 사용됩니다.



데이터 분석을 배당 가치 투자 전략에 적용하면 다음과 같이 됩니다.

첫째, 좋은 회사를 선정한다.

- 미국 배당 귀족주 리스트를 확인한다.
- 옵션. 투자 대상을 확장하고 싶다면 ‘Dividend Achievers’를 사용한다.

둘째, 배당수익률을 기준으로 매수 시점과 매매 시점을 찾는다.

- 야후 파이낸스 API를 이용해서 주식 데이터를 불러 온다.
- 과거 배당 이력을 계산하고 역사적 고점과 저점을 찾는다.
- 현재 배당률을 기준으로 매수 점수를 산출한다.
- 매수 점수를 기준으로 투자 의견을 결정한다.

2. 파이썬 주가 데이터 분석 개요

참고. 미국 배당주는 배당 인상 기간에 따라 3가지 등급으로 분류합니다.

- 배당킹: 50년 배당 인상
- 배당귀족: 25년 이상 배당 인상
- 배당 성취자(Dividend Achievers): 10년 이상 배당 인상

코딩
장인

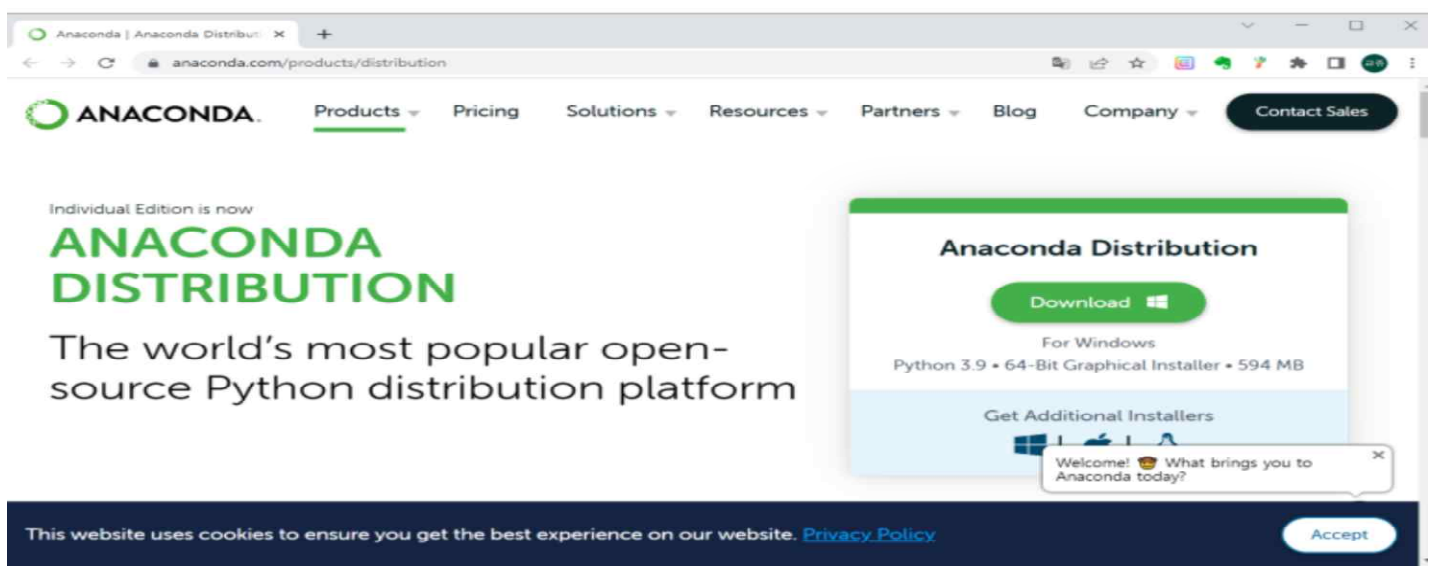
2.1 개발 환경 구축

먼저 윈도우 기반 개발 환경을 설정하겠습니다.

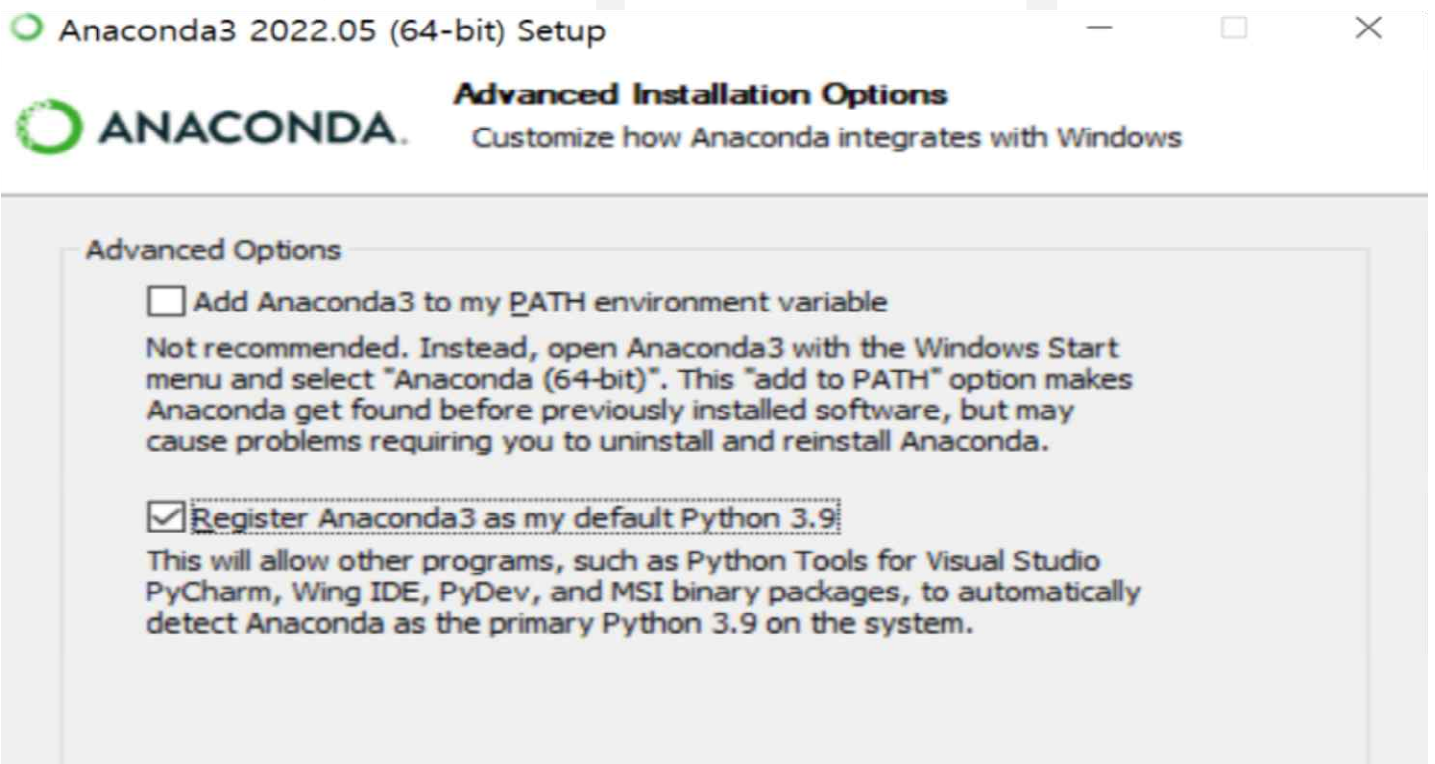
파이썬 데이터 분석에는 주피터 노트북이 사용됩니다. 반응형 UI에 그림 그리기도 쉬워서 '데이터 분석, 시각화'에 좋습니다.

먼저 아나콘다를 다운로드합니다.

<https://www.anaconda.com/>

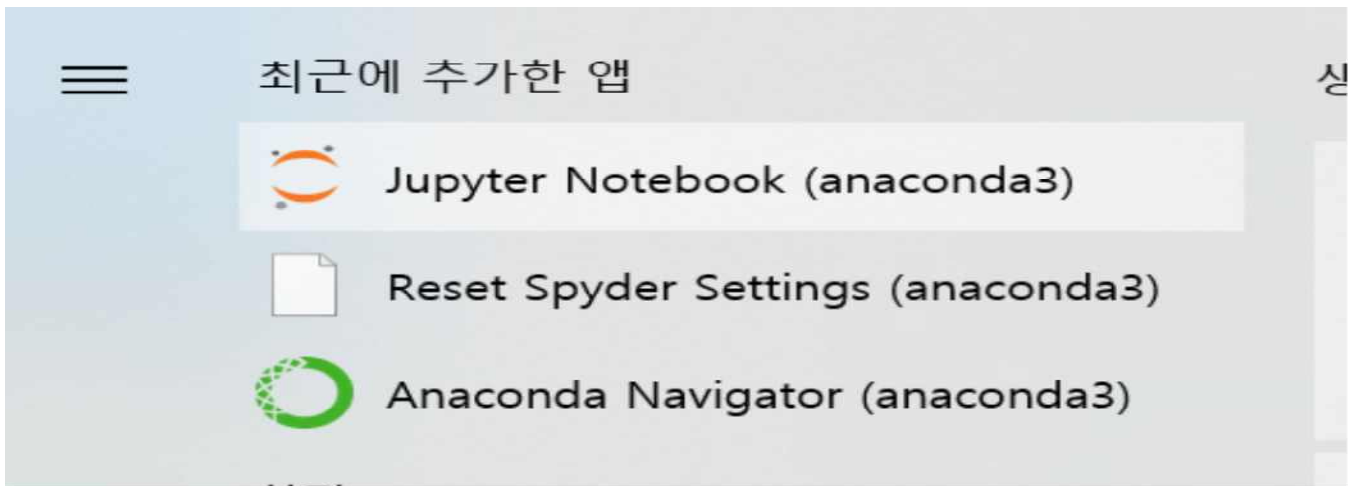


Setup 파일을 실행하고 옵션이 나오면 "Register Anaconda3 ~"만 체크합니다.

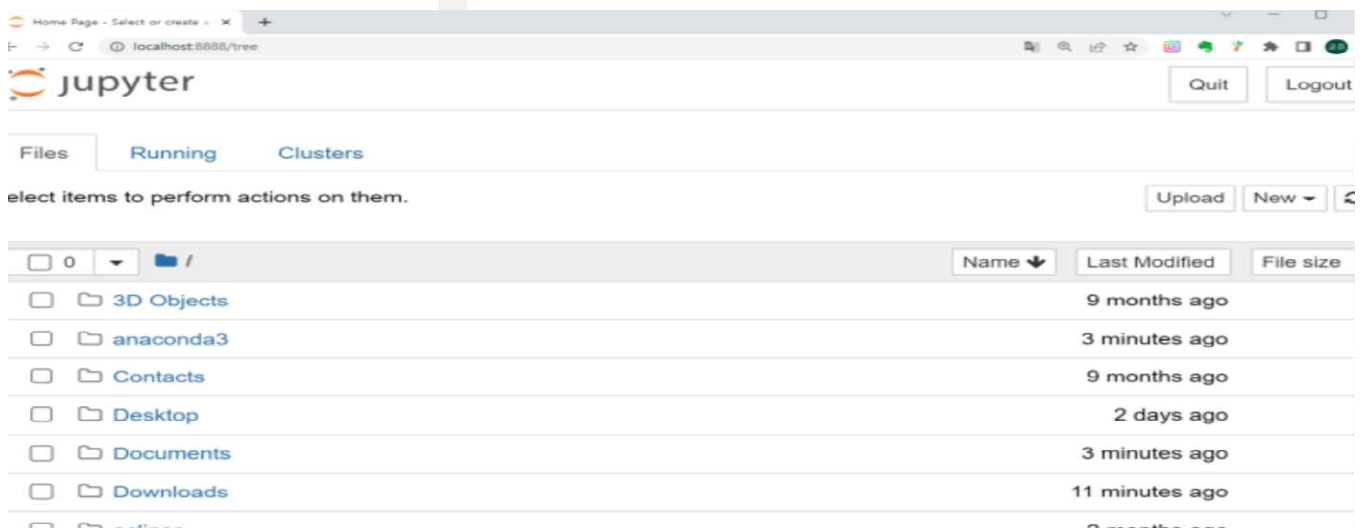


2.1 개발 환경 구축

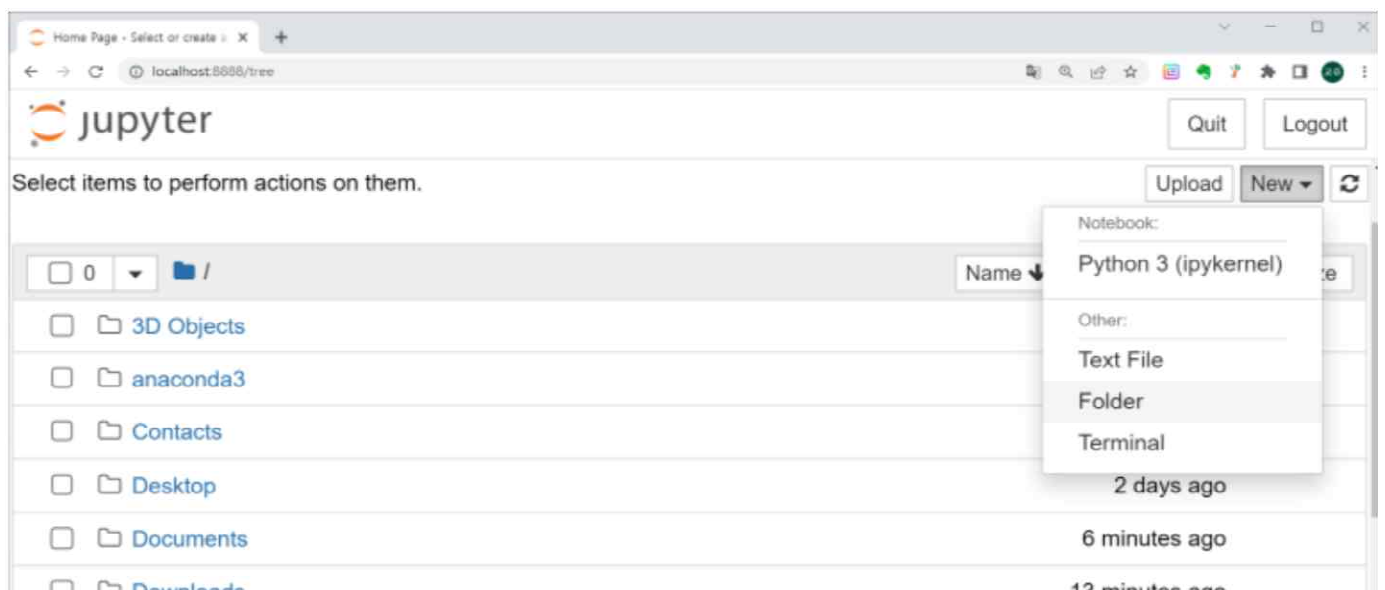
설치가 끝나면 윈도우 시작 메뉴에 주피터 노트북이 있습니다.



주피터 노트북을 실행합니다.

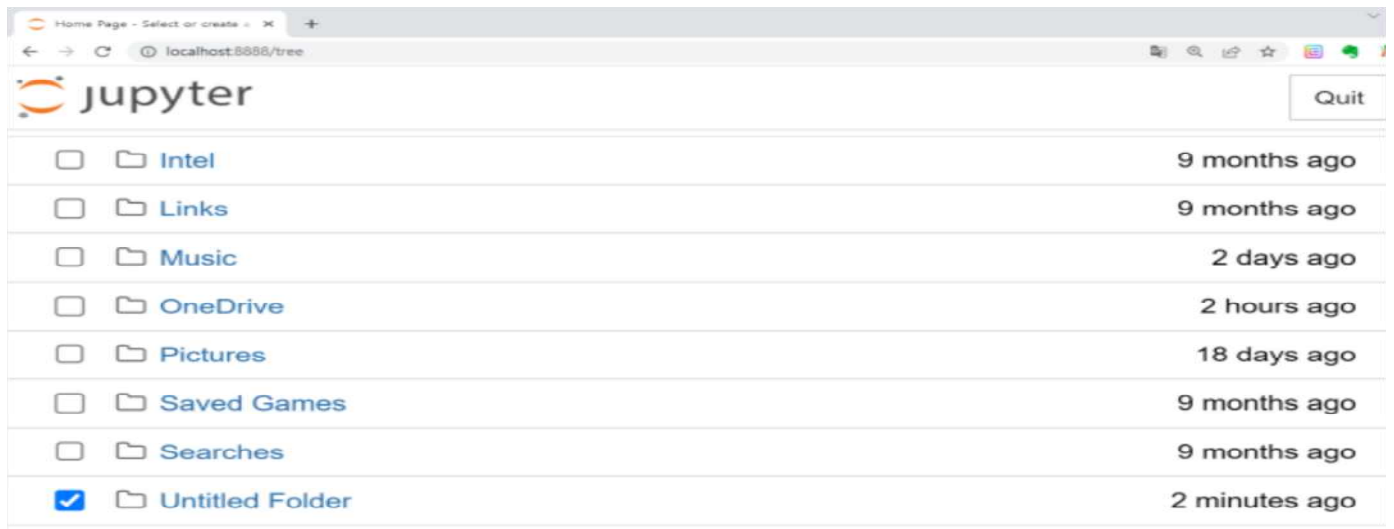


우측 상단에 있는 New Folder를 클릭합니다.



2.1 개발 환경 구축

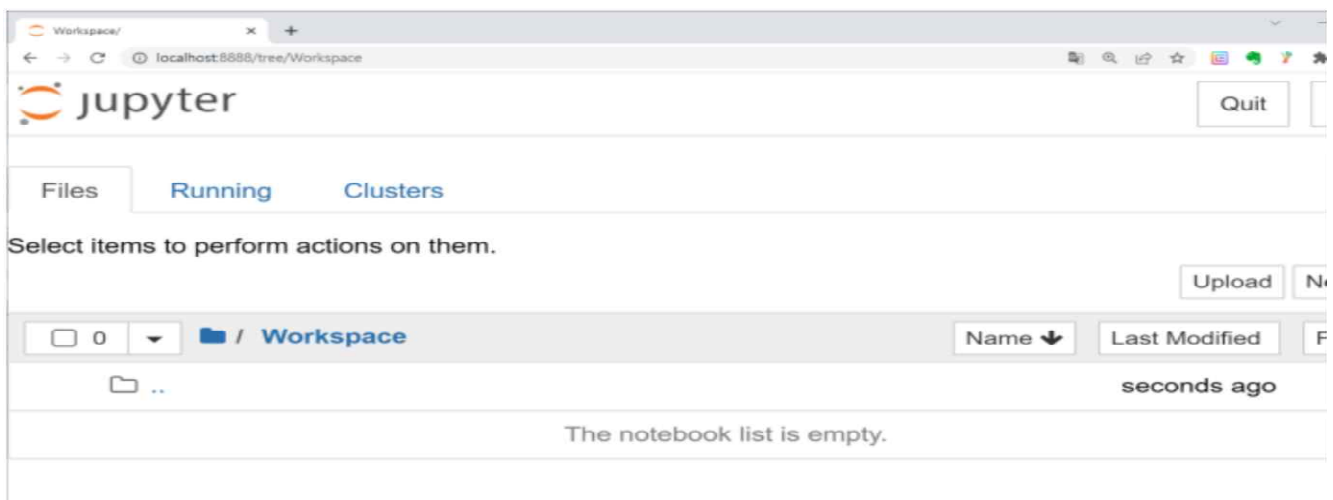
그럼 "Untitled Folder"가 생기는데요, 체크합니다.



좌측 상단에 있는 Rename 메뉴를 클릭하고 'Workspace'로 변경합니다.



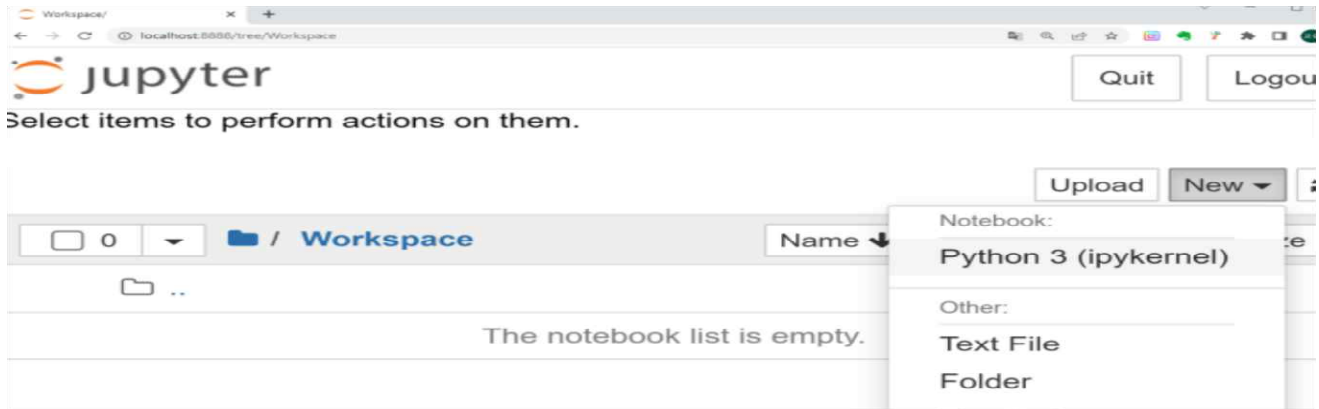
Workspace 폴더를 클릭해서 들어갑니다.



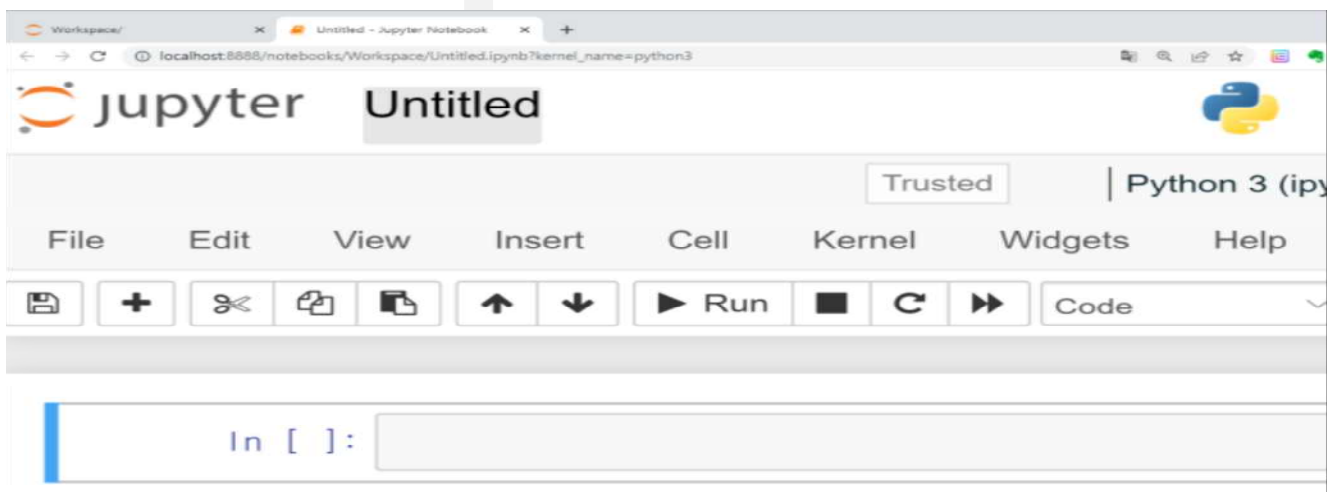
2.1 개발 환경 구축

이제 파이썬 커널을 만들겠습니다.

다시 우측 상단에서 'New -> Python3'를 클릭합니다.

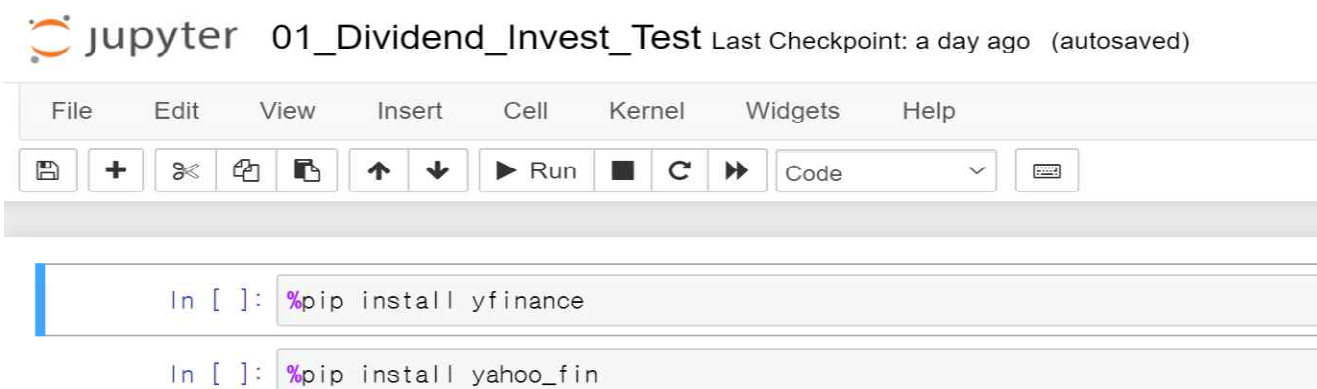


파이썬 커널이 만들어졌습니다.



“Untitled”를 클릭하고 파일 이름을 바꿔 줍니다.

“01_Dividend_Invest_Test”



이제 개발 환경 설정이 끝났습니다.

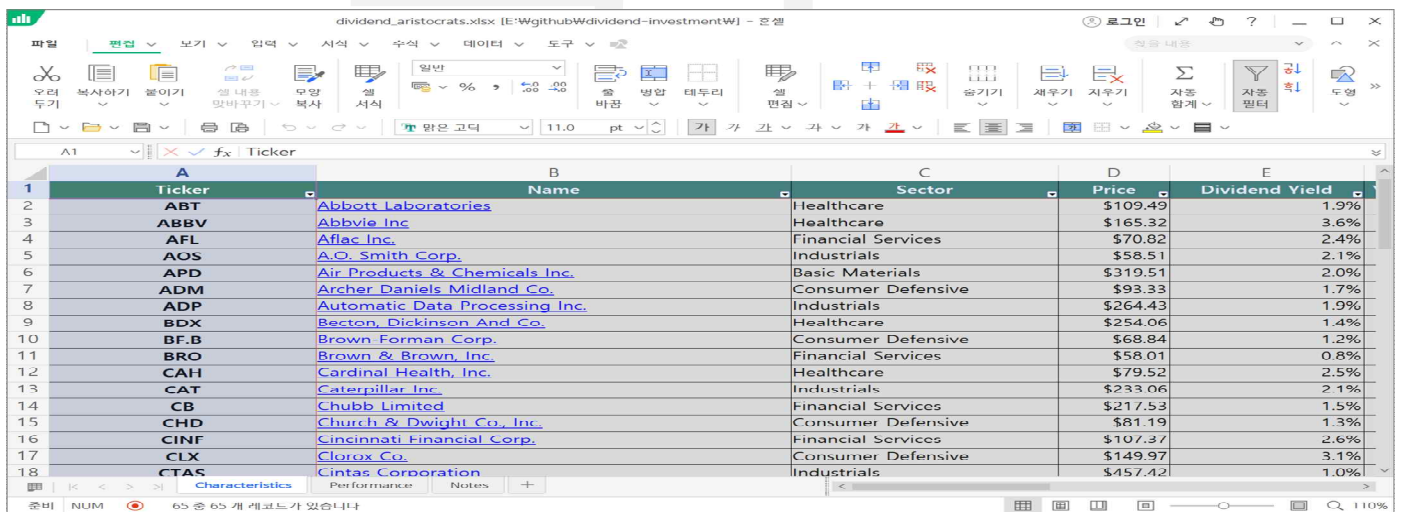
2.2 배당귀족주 리스트 다운로드

다음은 미국 배당 귀족주 리스트를 가져 오겠습니다.

미국 배당귀족주 전체 리스트는 아래 페이지에서 다운로드 가능합니다. 이메일 주소만 입력하면 엑셀 파일을 무료로 받을 수 있습니다.

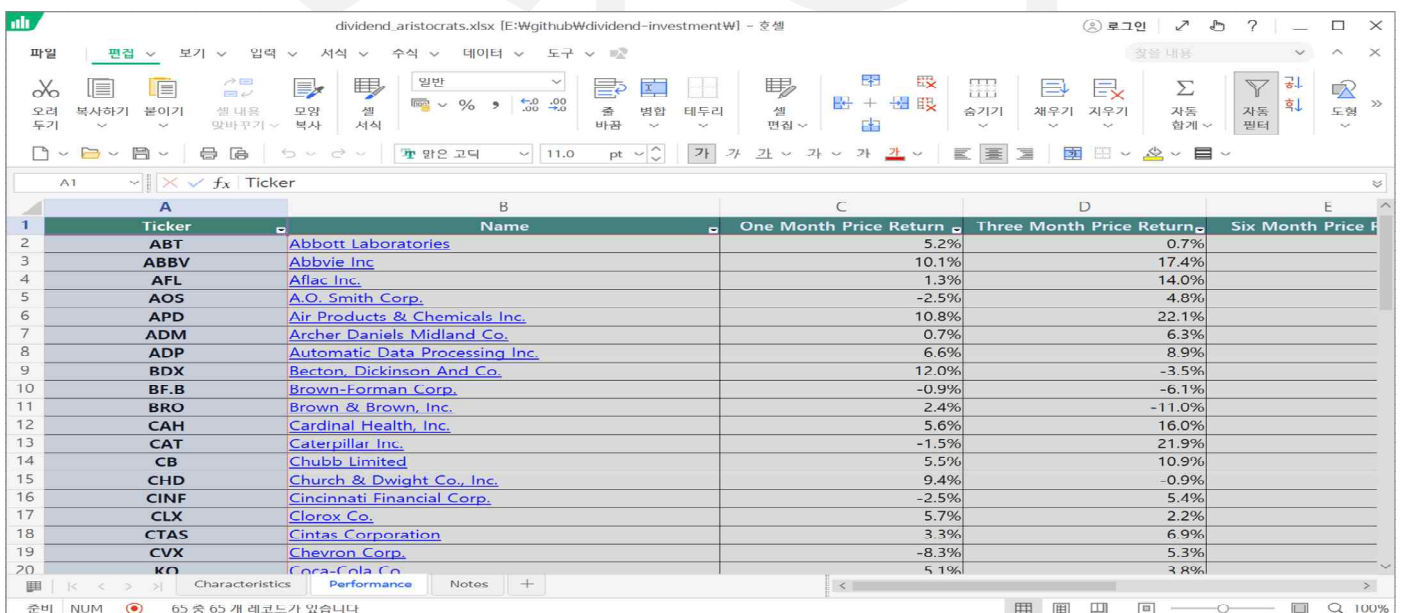
<https://www.suredividend.com/dividend-aristocrats-list/>

현재 배당귀족주는 총 65개입니다.



	A	B	C	D	E
	Ticker	Name	Sector	Price	Dividend Yield
1	ABT	Abbott Laboratories	Healthcare	\$109.49	1.9%
2	ABBV	Abbvie Inc.	Healthcare	\$165.32	3.6%
3	AFL	Aflac Inc.	Financial Services	\$70.82	2.4%
4	AOS	A.O. Smith Corp.	Industrials	\$58.51	2.1%
5	APD	Air Products & Chemicals Inc.	Basic Materials	\$319.51	2.0%
6	ADM	Archer Daniels Midland Co.	Consumer Defensive	\$93.33	1.7%
7	ADP	Automatic Data Processing Inc.	Industrials	\$264.43	1.9%
8	BDX	Becton, Dickinson And Co.	Healthcare	\$254.06	1.4%
9	BF.B	Brown Forman Corp.	Consumer Defensive	\$68.84	1.2%
10	BRO	Brown & Brown, Inc.	Financial Services	\$58.01	0.8%
11	CAH	Cardinal Health, Inc.	Healthcare	\$79.52	2.5%
12	CAT	Caterpillar Inc.	Industrials	\$233.06	2.1%
13	CB	Chubb Limited	Financial Services	\$217.53	1.5%
14	CHD	Church & Dwight Co., Inc.	Consumer Defensive	\$81.19	1.3%
15	CINF	Cincinnati Financial Corp.	Financial Services	\$107.37	2.6%
16	CLX	Clorox Co.	Consumer Defensive	\$149.97	3.1%
17	CTAS	Cintas Corporation	Industrials	\$457.42	1.0%

티커, 회사명, 섹터, 가격 등 기본 정보가 제공됩니다.



	A	B	C	D	E
	Ticker	Name	One Month Price Return	Three Month Price Return	Six Month Price Return
1	ABT	Abbott Laboratories	5.2%	0.7%	
2	ABBV	Abbvie Inc.	10.1%	17.4%	
3	AFL	Aflac Inc.	1.3%	14.0%	
4	AOS	A.O. Smith Corp.	-2.5%	4.8%	
5	APD	Air Products & Chemicals Inc.	10.8%	22.1%	
6	ADM	Archer Daniels Midland Co.	0.7%	6.3%	
7	ADP	Automatic Data Processing Inc.	6.6%	8.9%	
8	BDX	Becton, Dickinson And Co.	12.0%	-3.5%	
9	BF.B	Brown-Forman Corp.	-0.9%	-6.1%	
10	BRO	Brown & Brown, Inc.	2.4%	-11.0%	
11	CAH	Cardinal Health, Inc.	5.6%	16.0%	
12	CAT	Caterpillar Inc.	-1.5%	21.9%	
13	CB	Chubb Limited	5.5%	10.9%	
14	CHD	Church & Dwight Co., Inc.	9.4%	-0.9%	
15	CINF	Cincinnati Financial Corp.	-2.5%	5.4%	
16	CLX	Clorox Co.	5.7%	2.2%	
17	CTAS	Cintas Corporation	3.3%	6.9%	
18	CVX	Chevron Corp.	-8.3%	5.3%	
19	KO	Coca-Cola Co.	5.1%	3.8%	

2번째 시트에는 퍼포먼스 정보도 제공됩니다.

2.3 야후 파이낸스 모듈 설치

이제 파이썬으로 야후 파이낸스 API를 사용하기 위해서 모듈을 설치합니다. 필요한 모듈은 2가지입니다.

- `yfinance`
- `yahoo_fin`

주피터 노트북에 아래와 같이 입력하고 “shift + enter”를 클릭합니다.

`%pip install yfinance`

```
In [1]: %pip install yfinance
```

```
Requirement already satisfied: yfinance in c:\Users\shaws\Anaconda3\lib\site-packages (0.1.90)  
Requirement already satisfied: lxml>=4.9.1 in c:\Users\shaws\Anaconda3\lib\site-packages (from yfinance) (4.9.2)  
Requirement already satisfied: appdirs>=1.4.4 in c:\Users\shaws\Anaconda3\lib\site-packages (from yfinance) (1.4.4)  
Requirement already satisfied: pandas>=1.3.0 in c:\Users\shaws\Anaconda3\lib\site-packages (from yfinance) (1.4.2)
```

다시 아래와 같이 입력하고 “shift + enter”를 클릭합니다.

`%pip install yahoo_fin`

```
In [65]: %pip install yahoo_fin
```

```
Requirement already satisfied: yahoo_fin in c:\Users\shaws\Anaconda3\lib\site-packages (0.8.9.1)  
Requirement already satisfied: pandas in c:\Users\shaws\Anaconda3\lib\site-packages (from yahoo_fin) (1.4.2)  
Requirement already satisfied: requests-html in c:\Users\shaws\Anaconda3\lib\site-packages (from yahoo_fin) (0.10.0)  
Requirement already satisfied: feedparser in c:\Users\shaws\Anaconda3\lib\site-packages (from yahoo_fin) (6.0.10)  
Requirement already satisfied: requests in c:\Users\shaws\Anaconda3\lib\site-packages (from yahoo_fin) (2.27.1)
```


2.4 데이터 수집

이제 야후 파이낸스 모듈 설치가 끝났습니다.

쥬피터 노트북에 데이터 분석에 필요한 모듈을 import 하겠습니다.

```
import yfinance as yf
from yahoo_fin import stock_info
from datetime import datetime
from dateutil.relativedelta import relativedelta
```

```
In [4]: import yfinance as yf
        from yahoo_fin import stock_info
```

```
In [5]: from datetime import datetime
        from dateutil.relativedelta import relativedelta
```

배당 귀족주 중에서 월배당으로 유명한 리얼티 인컴을 분석해 보겠습니다.

57	IBM	International Business Machines Corp.
58	NEE	NextEra Energy Inc
59	WST	West Pharmaceutical Services, Inc.
60	AMCR	Amcor Plc
61	ATO	Atmos Energy Corp.
62	O	Realty Income Corp.
63	ESS	Essex Property Trust, Inc.
64	ALB	Albemarle Corp.
65	EXPD	Expeditors International Of Washington, Inc.
66	XOM	Exxon Mobil Corp.
67		

TICKER라는 변수를 만들고 'O'를 입력합니다.

```
TICKER = 'O'
```

DIV_PERIODS라는 변수를 만들고 7을 입력합니다. 과거 7년간 배당 정보를 분석하겠습니다.

```
DIV_PERIODS = 7
```

```
In [6]: TICKER = 'O'
        DIV_PERIODS = 7
```

2.4 데이터 수집

이제 야후 파이낸스로 과거 배당금 정보를 가져 옵니다.

stock_info 모듈의 get_dividends 함수를 사용하면 됩니다.

아래 페이지에 API 설명이 있습니다.

http://theautomatic.net/yahoo_fin-documentation/#get_dividends

get_dividends(ticker, start_date = None, end_date = None, index_as_date = True)

Downloads historical dividend data of a stock into a pandas data frame.

Possible parameters

ticker

Stock ticker (e.g. 'MSFT', 'AMZN', etc.). Case insensitive.
This is the only required argument.

start_date

The date the dividend history should begin.

end_date

The date the dividend history should end.

ticker는 주식에 해당하는 코드입니다. 리얼티인컴은 “O”입니다.

start_date와 end_date은 배당 정보를 가져올 기간의 시작과 끝입니다.

2.4 데이터 수집

`end_date`는 오늘 날짜를 입력하면 됩니다.

`datetime` 모듈의 `now()`를 이용하면 현재 년월일시를 가져 옵니다.

`datetime.now()`

```
In [1]: from datetime import datetime
        from dateutil.relativedelta import relativedelta
```

```
In [2]: datetime.now()
```

```
Out[2]: datetime.datetime(2022, 12, 25, 13, 44, 29, 630494)
```

년월일만 있으면 됩니다. `strftime()`함수를 사용하면 됩니다.

`datetime.now().strftime("%Y-%m-%d")`

```
In [3]: datetime.now().strftime("%Y-%m-%d")
```

```
Out[3]: '2022-12-25'
```

오늘 날짜를 `end_date` 변수에 저장하겠습니다.

`end_date = datetime.now().strftime("%Y-%m-%d")`

`print("end date: " , end_date)`

```
In [10]: end_date = datetime.now().strftime("%Y-%m-%d")
         print("end date: " , end_date)
```

```
end date: 2022-12-25
```

2.4 데이터 수집

이제 start_date을 찾아야 합니다. relativedelta를 사용하면 됩니다.

```
datetime.now() - relativedelta(years=7)
```

```
In [5]: datetime.now() - relativedelta(years=7)
```

```
Out[5]: datetime.datetime(2015, 12, 25, 13, 52, 28, 890800)
```

결과를 start_date 변수에 저장하겠습니다.

```
start_date = (datetime.now()
               - relativedelta(years=DIV_PERIODS)).strftime("%Y-%m-%d")
print("start date: ", start_date)
```

```
In [11]: start_date = (datetime.now() - relativedelta(years=DIV_PERIODS)).strftime("%Y-%m-%d")
         print("start date: ", start_date)
```

```
start date: 2015-12-25
```

stock_info모듈의 get_dividends 함수에 전달할 파라미터가 준비되었습니다.

```
dividends = stock_info.get_dividends(TICKER, start_date=start_date,
                                     end_date=end_date)
```

리얼티 인컴의 과거 7년 배당 정보를 데이터프레임으로 가져 왔습니다.

```
In [12]: dividends = stock_info.get_dividends(TICKER, start_date=start_date, end_date=end_date)
```

```
In [13]: dividends
```

```
Out[13]:
```

	dividend	ticker
2015-12-30	0.185078	O
2016-01-28	0.192829	O
2016-02-26	0.192829	O
2016-03-30	0.192829	O

2.4 데이터 수집

ticker는 이미 알고 있으니 제거하겠습니다. drop 함수를 사용하면 됩니다.

```
dividends.drop(['ticker'], axis=1, inplace=True)
```

```
In [18]: dividends.drop(['ticker'], axis=1, inplace=True)
```

```
In [19]: dividends
```

Out[19]:

dividend	
2015-12-30	0.185078
2016-01-28	0.192829
2016-02-26	0.192829

배당금 정보를 차트로 그려서 확인해 보겠습니다.

차트를 그리려면 matplotlib.pyplot 모듈이 필요합니다.

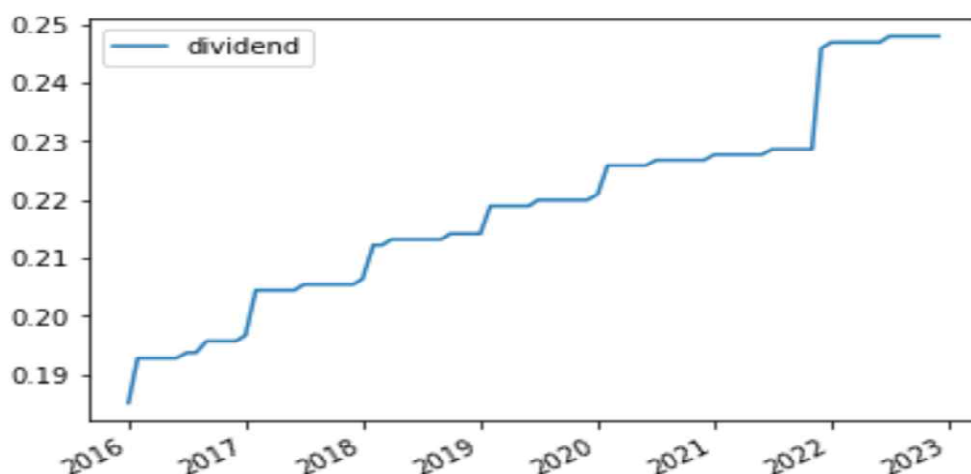
```
import matplotlib.pyplot as plt
```

그리고 plot() 함수를 사용하면 됩니다.

```
dividends.plot()
```

```
plt.show()
```

```
In [30]: dividends.plot()  
plt.show()
```



2.4 데이터 수집

배당금이 꾸준히 증가했다는 사실을 눈으로 쉽게 확인할 수 있습니다.

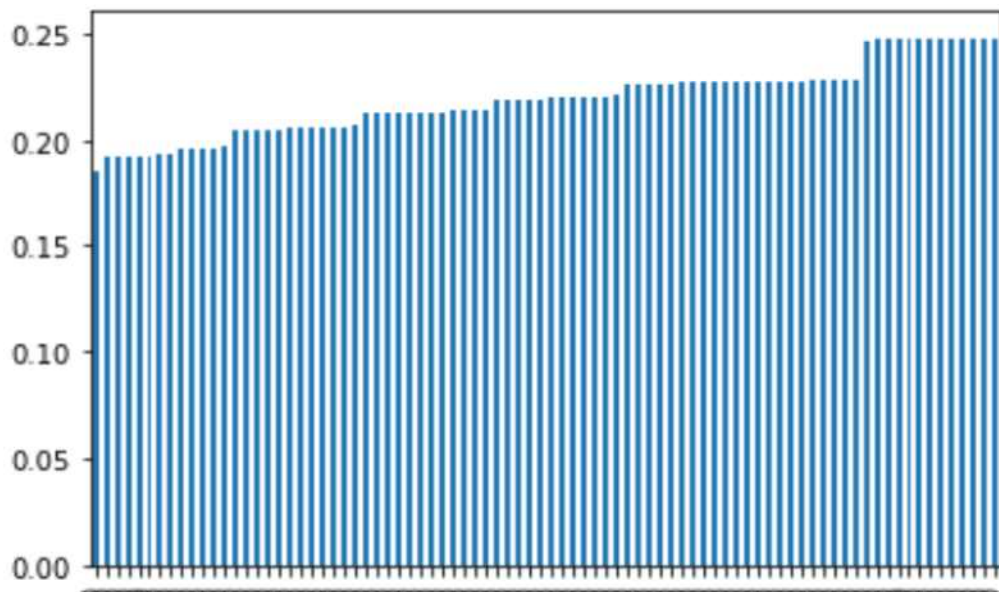
이렇게 데이터를 시각화하면 직관적으로 이해할 수 있습니다.

차트 종류도 쉽게 바꿀 수 있습니다. 바 차트로 그려 보겠습니다.

```
dividends.plot(kind='bar')
```

```
plt.show()
```

```
In [64]: dividends.plot(kind='bar')  
plt.show()
```



2.4 데이터 수집

배당률을 구하기 위해서는 배당일에 해당하는 주가 정보도 필요합니다.

주가 정보는 yfinance 모듈의 download 함수를 사용하면 됩니다. 방법은 get_dividends와 동일합니다.

```
stock_data = yf.download(TICKER, start_date, end_date)
```

```
In [14]: # Get stock price data
stock_data = yf.download(TICKER, start_date, end_date)

[*****100%*****] 1 of 1 completed
```

```
In [15]: stock_data
```

Out[15]:

	Open	High	Low	Close	Adj Close	Volume
Date						
2015-12-28	49.670544	50.155041	49.496124	50.145348	37.113575	1129627
2015-12-29	50.310078	50.629845	50.125969	50.562016	37.421963	1566473
2015-12-30	50.465115	50.784885	50.242249	50.310078	37.372280	1331693

배당률을 계산하려면 배당일에 해당하는 데이터만 있으면 됩니다. 그리고 종가를 사용합니다. 열 이름은 'Close'입니다.

loc 함수를 사용하면 원하는 데이터를 가져올 수 있습니다.

```
df_close_price = stock_data.loc[dividends.index]['Close']
```

```
In [31]: df_close_price = stock_data.loc[dividends.index]['Close']
```

```
In [32]: df_close_price
```

```
Out[32]: 2015-12-30    50.310078
          2016-01-28    53.168606
          2016-02-26    56.812016
          2016-03-30    60.281010
          2016-04-28    57.625969
```

2.4 데이터 수집

dividends 데이터프레임에 'close'라는 열 이름으로 추가하겠습니다.

```
dividends['close'] = df_close_price
```

```
In [33]: dividends['close'] = df_close_price
```

```
In [35]: dividends
```

```
Out[35]:
```

	dividend	close
2015-12-30	0.185078	50.310078
2016-01-28	0.192829	53.168606
2016-02-26	0.192829	56.812016

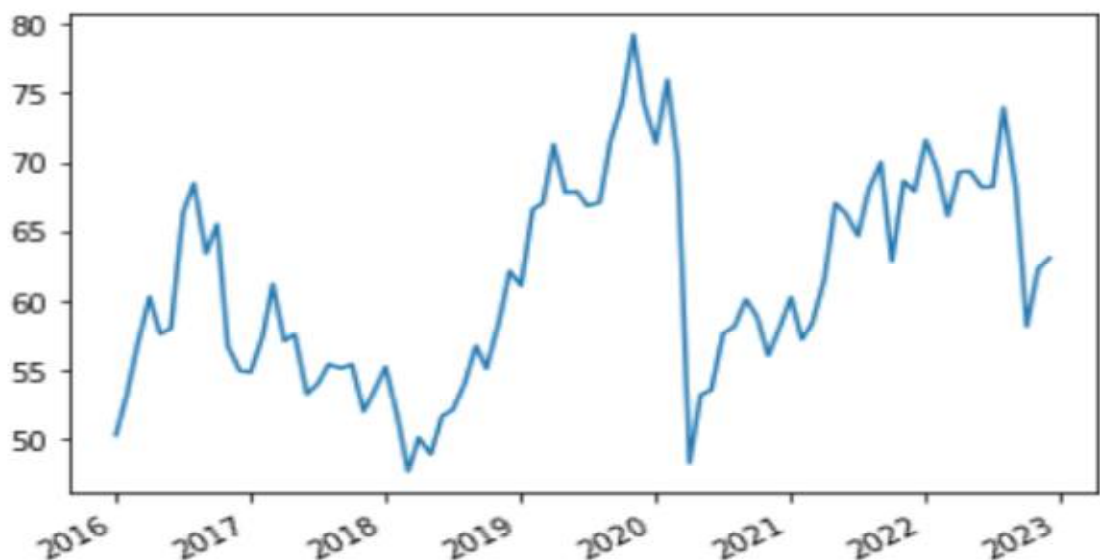
데이터가 정상적으로 추가되었습니다.

주가 정보를 그래프로 그려서 확인해 보겠습니다.

```
stock_data['Close'].plot()
```

```
plt.show()
```

```
In [7]: stock_data['Close'].plot()  
plt.show()
```



2.5 데이터 처리

배당금과 주가 정보가 있으니 7년 동안의 배당률 이력을 계산해 볼 수 있습니다.

$$\text{배당률} = (\text{1년 배당금}) / (\text{주가}) * 100$$

그 전에 1년에 배당을 몇 번 주는지 계산해야 합니다. 리얼티인컴은 월배당이니까 배당주기가 12가 나오면 정상입니다.

$$\text{div_freq} = \text{round}(\text{len}(\text{dividends.index}) / \text{DIV_PERIODS})$$

```
In [37]: div_freq = round(len(dividends.index)/DIV_PERIODS)
```

```
In [38]: div_freq
```

```
Out[38]: 12
```

이제 배당률을 계산해 보겠습니다. dividend 열의 데이터를 close 열의 데이터로 나누면 됩니다.

$$\text{div_yield} = \text{dividends['dividend']} * \text{div_freq} / \text{dividends['close']} * 100$$

```
In [39]: div_yield = dividends['dividend'] * div_freq / dividends['close'] * 100
```

```
In [40]: div_yield
```

```
Out[40]: 2015-12-30    4.414495
         2016-01-28    4.352095
         2016-02-26    4.072991
         2016-03-30    3.838602
         2016-04-28    4.015160
```

2.5 데이터 처리

배당률을 소수점 2자리에서 자르겠습니다. round함수를 사용합니다.

```
div_yield = round(div_yield, 2)
```

```
In [41]: div_yield = round(div_yield, 2)
div_yield
```

```
Out[41]: 2015-12-30    4.41
         2016-01-28    4.35
         2016-02-26    4.07
         2016-03-30    3.84
```

배당률 정보도 dividends 데이터프레임에 추가하겠습니다. 열 이름은 'div yield'로 하겠습니다.

```
dividends['div yield'] = div_yield
```

```
In [42]: dividends['div yield'] = div_yield
```

```
In [43]: dividends
```

```
Out[43]:
```

	dividend	close	div yield
2015-12-30	0.185078	50.310078	4.41
2016-01-28	0.192829	53.168606	4.35
2016-02-26	0.192829	56.812016	4.07

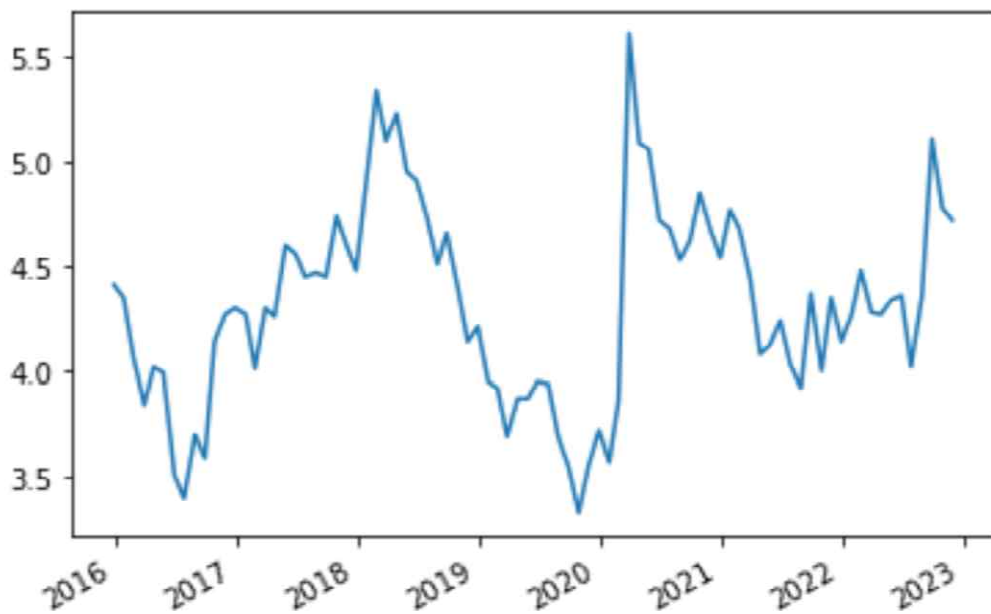
정상적으로 입력되었습니다.

2.5 데이터 처리

배당률도 차트로 그려 보겠습니다.

```
dividends['div yield'].plot()  
plt.show()
```

```
In [45]: dividends['div yield'].plot()  
plt.show()
```



최종적으로 필요한 데이터는 역사적인 배당률의 고점과 저점을 확인하면 됩니다.
min, max 함수를 사용합니다.

```
div_min = min(dividends['div yield'])  
div_max = max(dividends['div yield'])
```

```
In [99]: div_min = min(dividends['div yield'])  
div_max = max(dividends['div yield'])  
  
In [100]: div_min, div_max  
Out [100]: (3.33, 5.61)
```

최저 배당률은 3.33, 최고 배당률은 5.61입니다.

2.5 데이터 처리

그리고 현재 배당률을 확인합니다. stock_data의 제일 마지막 데이터를 가져오면 됩니다.

```
current_price = stock_data.iloc[-1]['Close']
```

```
In [105]: current_price = stock_data.iloc[-1]['Close']
```

```
In [106]: current_price
```

```
Out[106]: 64.66000366210938
```

그리고 제일 마지막 배당금 액수도 확인합니다.

```
last_dividend = dividends.iloc[-1]['dividend']
```

```
In [107]: last_dividend = dividends.iloc[-1]['dividend']  
last_dividend
```

```
Out[107]: 0.248
```

이제 현재 배당률을 계산할 수 있습니다.

```
current_div_yield = round(last_dividend*div_freq/current_price*100, 2)
```

```
In [115]: current_div_yield = round(last_dividend*div_freq/current_price*100, 2)  
current_div_yield
```

```
Out[115]: 4.6
```

현재 배당률은 4.6%입니다.

2.6 데이터 분석

최고값은 5.61 최저값은 3.33 현재 배당률은 4.6입니다. 대략 중간 정도에 있는 것 같은데 퍼센티지로 환산해서 매수 점수라고 이름지어 보겠습니다.

함수를 만들어 보았습니다.

```
def calculate_buy_score(cur_div, min_div, max_div):  
    return round((cur_div - min_div) / (max_div - min_div) * 100)
```

실행해 볼까요?

```
buy_score = calculate_buy_score(current_div_yield, div_min, div_max)  
print(f'buy score = {buy_score}')
```

```
In [61]: buy_score = calculate_buy_score(current_div_yield, div_min, div_max)  
         print(f'buy score = {buy_score}')
```

buy score = 57

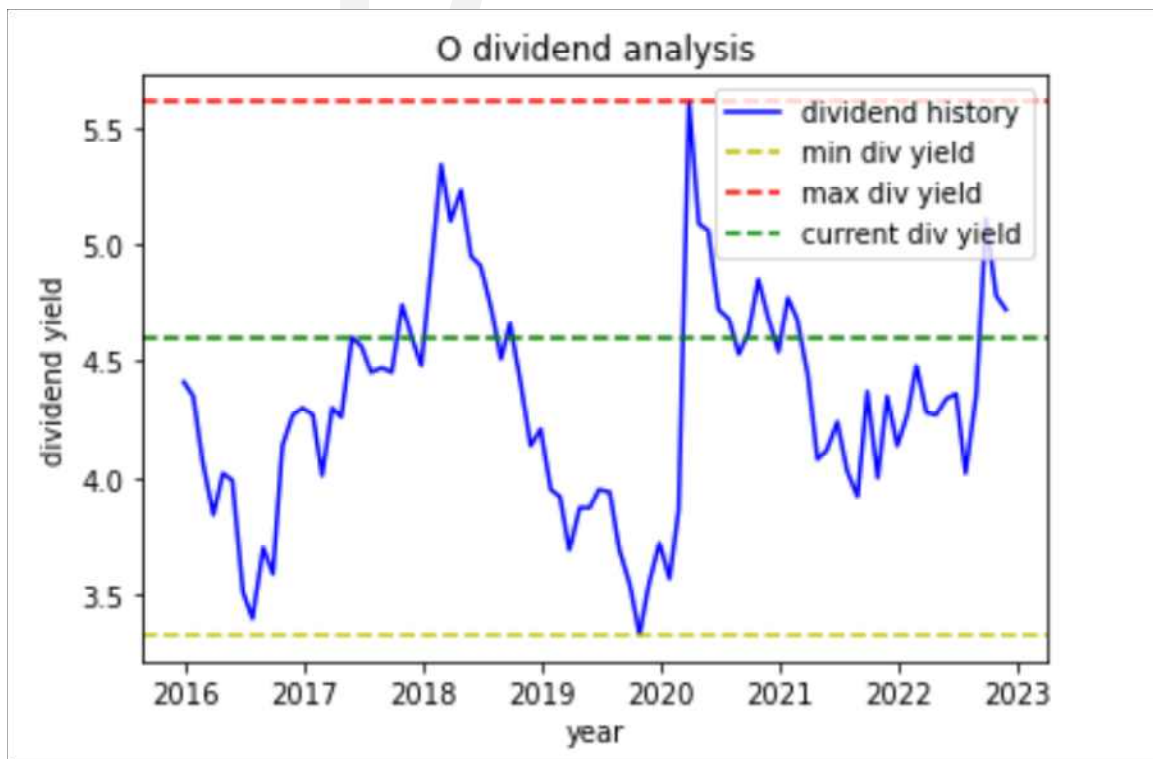
매수 점수는 57점입니다. 리얼티 인컴은 현재 매수하기에 좋은 가격은 아닙니다. 따라서 매수 의견은 '중립'이라고 결정하겠습니다. 시가 배당률을 지켜 보다가 5.61에 가까워지면 매수에 들어가고 배당을 받다가 3.33 근처로 떨어지면 매도를 하면 됩니다.

매수, 매도도 가능하면 일정한 간격으로 3~4회에 걸쳐 진행합니다.

2.6 데이터 분석

배당률 최고, 최저, 현재 값을 차트로 그려 보겠습니다.

```
plt.plot(dividends['div yield'], color='b', label = 'dividend history')
plt.axhline(y=div_min, color='y', linestyle='--', label='min div yield')
plt.axhline(y=div_max, color='r', linestyle='--', label='max div yield')
plt.axhline(y=current_div_yield, color='g', linestyle='--', label='current div
yield')
plt.title(f'{TICKER} dividend analysis')
plt.xlabel('year')
plt.ylabel('dividend yield')
plt.legend()
plt.show()
```



이렇게 시각화를 하면 직관적으로 매수, 매도 결정을 할 수 있습니다.

2.6 데이터 분석 – 신대륙 발견

연습 삼아서 다른 주식도 확인해 보겠습니다.

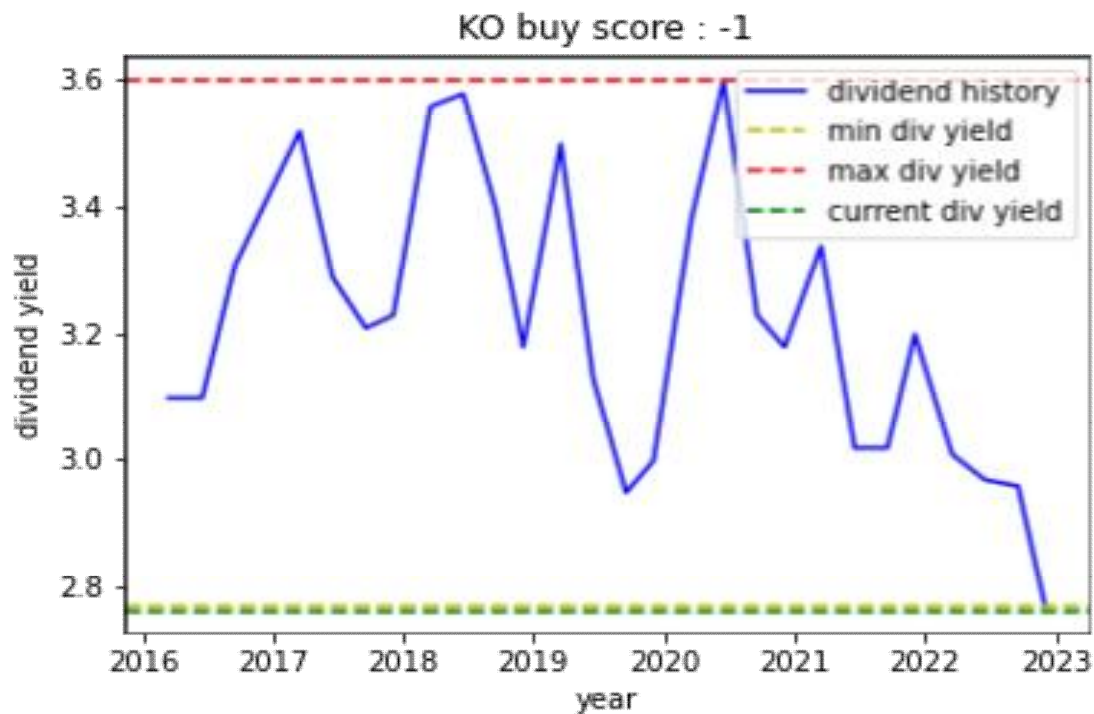
매수 점수가 92점인 주식을 발견했습니다. 티커는 'AOS', 처음 보는 회사입니다.
매수 후보에 선정합니다.



이렇게 퀀트 방식으로 분석하다 보면 처음 보는 주식들이 발견됩니다. 일종의 신대륙 발견 같은 겁니다. 구글을 이용해서 해당 기업을 공부해야 합니다.

2.6 데이터 분석

매수 점수가 -1인 주식도 있습니다. 티커는 “KO” 워런 버핏이 사랑하는 코카콜라입니다.



절대 사면 안 되겠죠. 아무리 훌륭한 회사도 비싸게 사면 의미없습니다. 혹시 보유 중이고 수익구간이라면 매도를 고려해야 할 시점입니다.

3. 데이터 분석 응용

이제 배당귀족주 리스트 전체에 대해서 매수 점수를 일괄적으로 계산해 보겠습니다.

코딩을 하는 이유 중 하나죠. 동일한 작업을 반복할 때 반복문으로 쉽고 빠르게 해결할 수 있습니다.

2장에서 사용된 코드를 정리해서 함수로 만듭니다.

함수 이름은 'getBuyScore'로 하겠습니다. ticker를 입력하면 cur_div, cur_div_yield, buy_score 를 리턴합니다.

```
def getBuyScore(ticker, periods=7):
    try:
        print(f'ticker = {ticker}')
        stock_basic_info = yf.Ticker(ticker).info
        end_date = datetime.now().strftime("%Y-%m-%d")
        start_date = (datetime.now() -
relative delta(years=periods)).strftime("%Y-%m-%d")
        dividends = stock_info.get_dividends(ticker, start_date=start_date,
end_date=end_date)
        result = dividends.drop('ticker', axis=1)
        stock_data = yf.download(ticker, start_date, end_date)
        df_close_price = stock_data.loc[dividends.index]['Close']
        result['close'] = df_close_price
        div_freq = round(len(result.index)/periods)
        result['div yield'] = round(result['dividend'] * div_freq /
result['close'] * 100, 2)
        div_min = min(result['div yield'])
        div_max = max(result['div yield'])
```

3. 데이터 분석 응용

```
buy_price = result['dividend'][-1] * div_freq * 100 / div_min
    sell_price = result['dividend'][-1] * div_freq * 100 / div_max
    current_datetime = datetime.strptime(end_date, "%Y-%m-%d")
current_price = stock_data.iloc[-1]['Close']
    last_dividend = result.iloc[-1]['dividend']
    current_dividend = last_dividend * div_freq
    current_div_yield = round(current_dividend/current_price*100, 2)
    buy_score = calculate_buy_score(current_div_yield, div_min,
div_max)
    return cur_div, cur_div_yield, buy_score
except Exception as e:
    raise(e)
```

장인

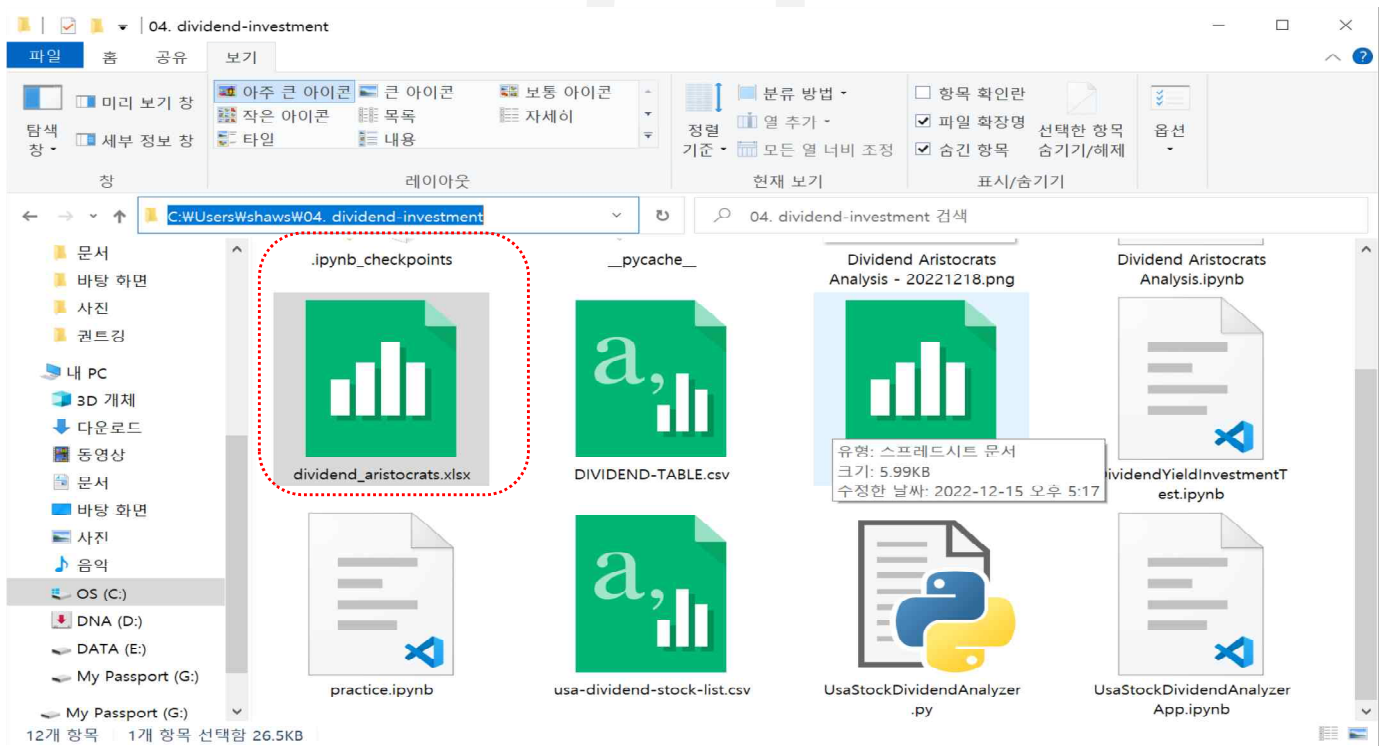
3. 데이터 분석 응용

다운받은 배당귀족주 엑셀 파일을 주피터 노트북을 실행하는 폴더로 복사합니다.
작업 중인 폴더 위치는 아래와 같이 확인할 수 있습니다.

```
import os
os.getcwd()
```

```
In [1]: import os
        os.getcwd()

Out[1]: 'C:\\Users\\User\\workspace'
```



그리고 주피터 노트북에서 엑셀 파일을 읽을 수 있도록 openpyxl 모듈을 설치합니다.

```
%pip install openpyxl
```

```
In [1]: %pip install openpyxl
```

```
Requirement already satisfied: openpyxl in c:\\Users\\Wshaws\\
Requirement already satisfied: et-xmlfile in c:\\Users\\Wshaws\\
0)
Note: you may need to restart the kernel to use updated p
```

3. 데이터 분석 응용

배당귀족주 엑셀 파일을 데이터프레임으로 읽어 옵니다.

```
import pandas as pd
stock_df= pd.read_excel('dividend_aristocrats.xlsx')
stock_df.head(3)
```

```
In [18]: stock_df= pd.read_excel('dividend_aristocrats.xlsx')
stock_df.head(3)
```

Out[18]:

	Ticker	Name	Sector	Price	Dividend Yield	Years of Dividend Increases	1-Year Dividend Growth	5-Year Dividend Growth (Annualized)	Divide Per S (1
0	ABT	0	Healthcare	109.49	0.018632	51	0.044444	0.109144	1.867
1	ABBV	0	Healthcare	165.32	0.035809	51	0.084615	0.147076	5.550

정상적으로 로딩이 되었습니다. 우리가 필요한 데이터는 'Ticker' 열입니다.

```
tickers = stock_df['Ticker']
```

```
In [20]: tickers = stock_df['Ticker']
tickers
```

```
Out[20]: 0    ABT
1    ABBV
2    AFL
3    AOS
4    ABB
```

3. 데이터 분석 응용

이제 for문을 이용해서 매 수 점수를 계산하고 리스트에 저장하겠습니다.

```
start = time.time()
new_tickers = []
cur_divs = []
cur_div_yields = []
buy_scores = []
for ticker in tickers:
    try:
        cur_div, cur_div_yield, buy_score = getBuyScore(ticker)
        new_tickers.append(ticker)
        cur_divs.append(cur_div)
        cur_div_yields.append(cur_div_yield)
        buy_scores.append(buy_score)
    except Exception as e:
        print(e)
end = time.time()
print(f'{end - start} seconds')
```

```
[*****100%*****] 1 of 1 completed
ticker = ALB
[*****100%*****] 1 of 1 completed
ticker = EXPD
[*****100%*****] 1 of 1 completed
ticker = XOM
[*****100%*****] 1 of 1 completed
262.31689047813416 seconds
```

데이터를 다운로드하고 계산하는데 262초 걸렸습니다.

3. 데이터 분석 응용

결과를 데이터프레임으로 만듭니다.

```
result = pd.DataFrame({'ticker': new_tickers,  
                        'annual dividend': cur_divs,  
                        'dividend yield': cur_div_yields,  
                        'buy score': buy_scores})
```

```
In [33]: result = pd.DataFrame({'ticker': new_tickers,  
                                'annual dividend': cur_divs,  
                                'dividend yield': cur_div_yields,  
                                'buy score': buy_scores})
```

```
In [34]: result
```

Out[34]:

	ticker	annual dividend	dividend yield	buy score
0	ABT	1.880	1.73	31
1	ABBV	5.640	3.46	20
2	AFL	1.600	2.21	16
3	AOS	1.200	2.08	90

buy score 순으로 내림차순 정렬해 보겠습니다.

```
result_sorted = result.sort_values('buy score', ascending=False)
```

```
In [41]: result_sorted = result.sort_values('buy score', ascending=False)  
result_sorted
```

Out[41]:

	ticker	annual dividend	dividend yield	buy score
46	SWK	3.20	4.30	115
37	MMM	5.96	4.96	110
50	VFC	2.04	7.51	105
31	LEG	1.76	5.42	101
60	ESS	8.80	4.17	101
...
16	CVX	5.68	3.16	2
63	XOM	3.64	3.30	1

3. 데이터 분석 응용

의미없는 인덱스를 없애고 ticker로 바꿉니다.index에 티커 칼럼의 정보를 입력하고 티커 칼럼은 drop 함수로 삭제합니다.

```
result_sorted.index = result_sorted['ticker']  
result_sorted.drop('ticker', axis=1, inplace=True)
```

```
In [43]: result_sorted.index = result_sorted['ticker']  
result_sorted.drop('ticker', axis=1, inplace=True)
```

```
In [44]: result_sorted
```

Out[44]:

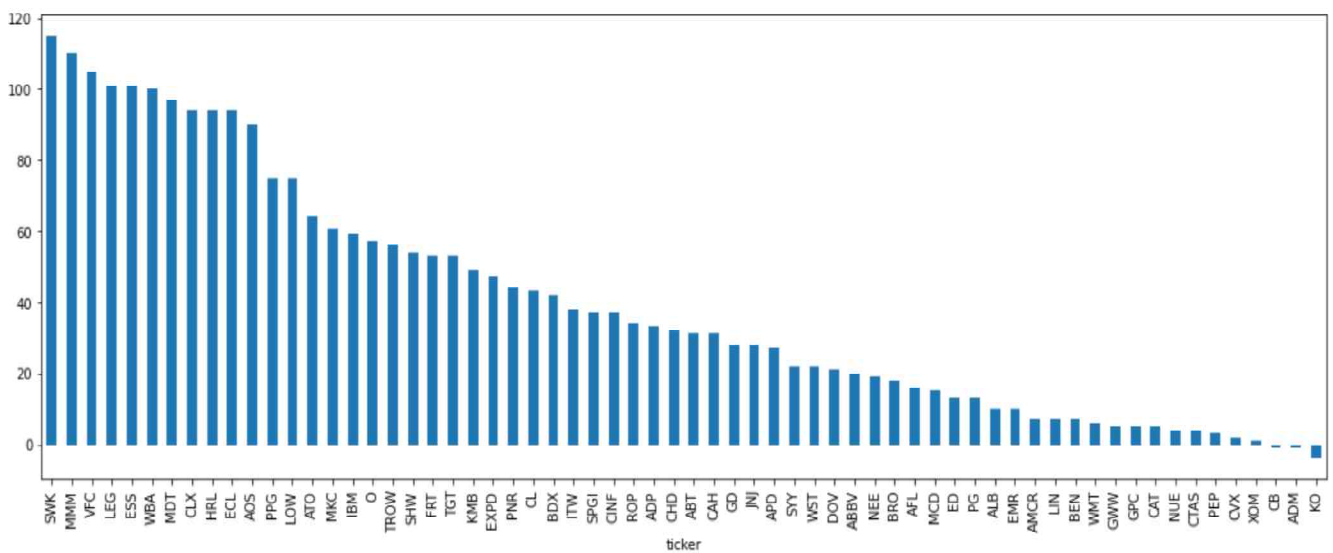
	annual dividend	dividend yield	buy score
ticker			
SWK	3.20	4.30	115
MMM	5.96	4.96	110
VFC	2.04	7.51	105
LEG	1.76	5.42	101
ESS	8.80	4.17	101

3. 데이터 분석 응용

이제 그래프로 그려서 확인해 보겠습니다.

```
plt.figure(figsize=(18,6))  
result_sorted['buy score'].plot(kind='bar')  
plt.show()
```

```
: plt.figure(figsize=(18,6))  
result_sorted['buy score'].plot(kind='bar')  
plt.show()
```



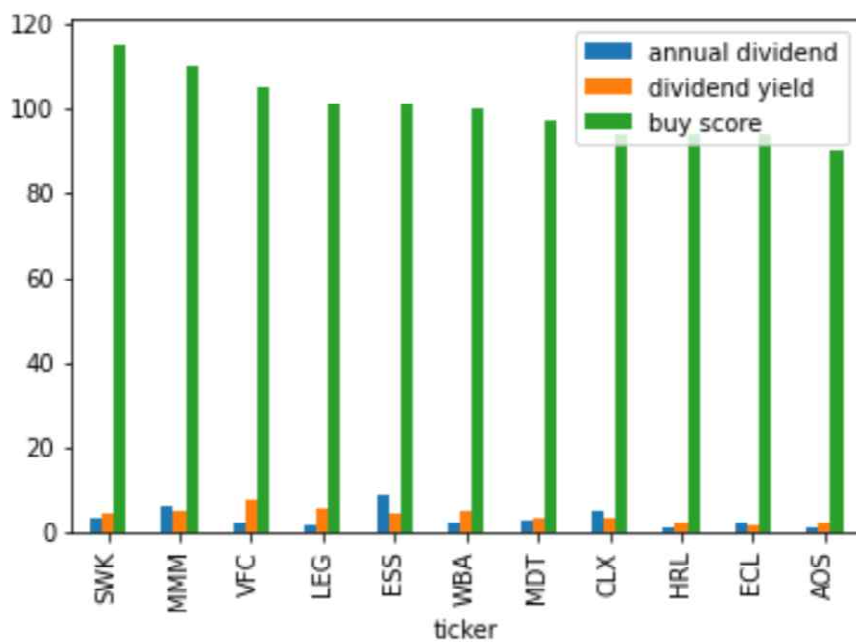
매수 점수가 높은 주식 순으로 시각화를 했습니다.

3. 데이터 분석 응용

주식 개수가 너무 많으니까 매수 점수가 80점 이상인 것만 다시 그려 보겠습니다.

```
Result_sorted[result_sorted['buy score'] > 80].plot(kind='bar')  
Plt.show()
```

```
In [52]: result_sorted[result_sorted['buy score'] > 80].plot(kind='bar')  
plt.show()
```



3. 데이터 분석 응용

이제 가격이 싼 주식은 뽑았습니다. 그러나 꼭 해야 할 일이 남았습니다. 회사가 얼마나 돈을 잘 버는지 그리고 번 돈에서 배당을 얼마나 주는지 확인합니다. 즉, ROE(주가 수익 비율), Payout Ratio(배당 성향)을 확인합니다.

- ROE는 최소 10%이상은 되어야 합니다.
- Payout Ratio는 20~60%가 적당합니다.

2가지 지표만 봐도 이 회사가 앞으로 배당금을 계속 올려 줄 수 있을지 확인할 수 있습니다.

StockBasicInfo 클래스에서 getRoe, getPayoutRatio를 이용하면 됩니다.

```
class StockBasicInfo:
    def __init__(self, ticker):
        self.ticker = ticker
        self.stock_stats = stock_info.get_stats(ticker)

    def getRoe(self):
        stock_stats = self.stock_stats
        roe = stock_stats['Value'][stock_stats['Attribute'] == 'Return on Equity (ttm)'].values[0]
        return roe

    def getPayoutRatio(self):
        stock_stats = self.stock_stats
        payout_ratio = stock_stats['Value'][stock_stats['Attribute'] == 'Payout Ratio 4'].values[0]
        return payout_ratio
```


3. 데이터 분석 응용 - 대박 발견

for 문을 이용해서 전체 주식의 roe와 payoutratio를 읽어 옵니다.

```
roes = []
payoutratios = []
for ticker in result.index:
    #print(ticker)
    stockBasicInfo = StockBasicInfo(ticker)
    roes.append(stockBasicInfo.getRoe())
    payoutratios.append(stockBasicInfo.getPayoutRatio())
```

그리고 result에 추가하겠습니다.

```
result['roe'] = roes
result['payout ratio'] = payoutratios
```

이제 result 데이터프레임을 출력해 보겠습니다.

```
result.head(3)
```

```
In [23]: result.head(3)
```

```
Out[23]:
```

	annual dividend	dividend yield	buy score	roe	payout ratio
ticker					
SWK	3.20	4.30	115	5.73%	85.68%
MMM	5.96	4.96	110	45.93%	51.92%
VFC	2.04	7.51	105	12.68%	185.19%

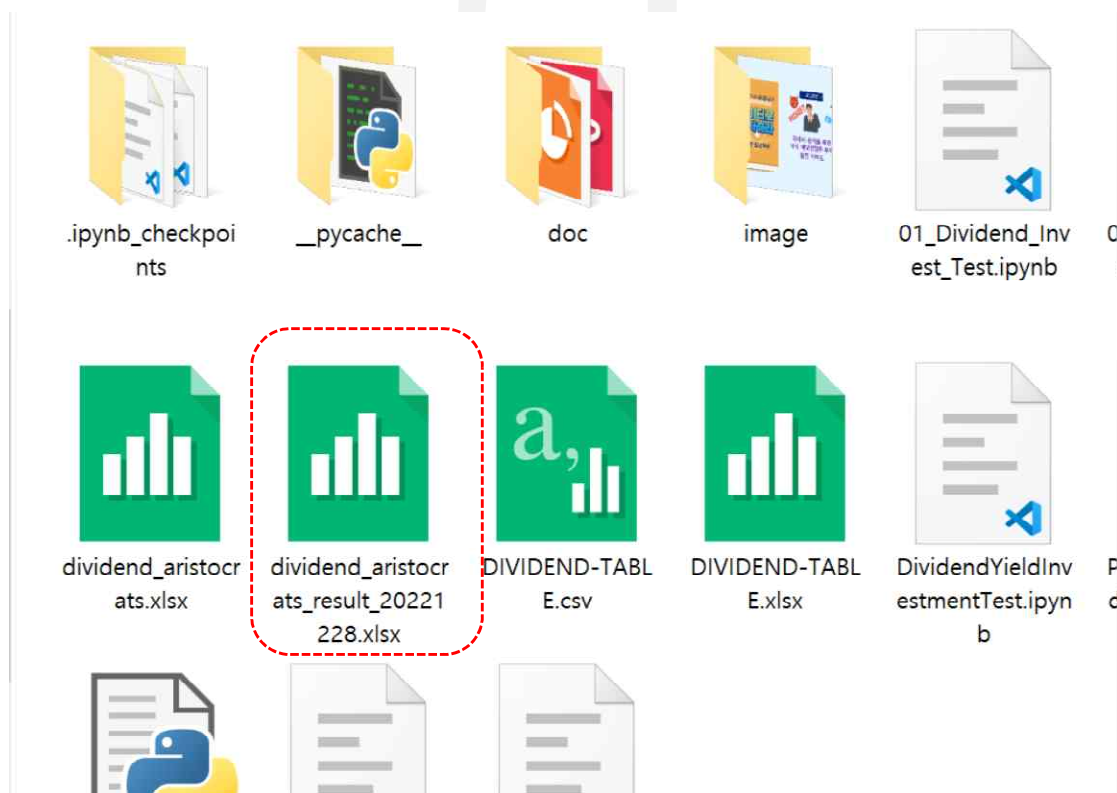
MMM은 포스트잇을 만드는 3M입니다. roe가 45에 배당성향이 51.9%! 대박을 발견했습니다.

3. 데이터 분석 응용 - 결과를 엑셀로 저장

이제 분석한 결과를 엑셀 파일로 저장하겠습니다.

```
from datetime import date
today = date.today().strftime("%Y%m%d")
result_file_name = STOCK_FILE_NAME.split(".")[0] + "_result_" + today +
".xlsx"
result.to_excel(result_file_name)
```

파일로 저장되었는지 확인해 보겠습니다.



파일이 정상적으로 저장되었습니다.

3. 데이터 분석 응용 - 결과를 엑셀로 저장

엑셀 파일을 열어 보겠습니다.

	A	B	C	D	E	F	G	H	I	J
1	ticker	annual dividend	dividend yield	buy score	roe	ayout ratio				
2	SWK	3.2	4.3	115	5.73%	85.68%				
3	MMM	5.96	4.96	110	45.93%	51.92%				
4	VFC	2.04	7.51	105	12.68%	185.19%				
5	LEG	1.76	5.42	101	23.10%	64.91%				
6	ESS	8.8	4.17	101	6.34%	157.43%				
7	WBA	1.92	5.01	100	14.90%	38.17%				
8	MDT	2.72	3.5	97	8.33%	81.37%				
9	CLX	4.72	3.27	94	79.58%	142.51%				
10	HRL	1.04	2.27	94	13.78%	57.14%				
11	ECL	2.12	1.46	94	16.29%	52.04%				
12	AOS	1.2	2.08	90	27.19%	35.67%				
13	PPG	2.48	1.96	75	17.33%	53.71%				
14	LOW	4.2	2.08	75		35.71%				
15	ATO	2.96	2.56	64	8.94%	48.57%				
16	MKC	1.48	1.74	61	15.40%	56.64%				
17	IBM	6.6	4.63	59	6.48%	435.76%				

데이터도 모두 입력되어 있습니다.

이 중에서 투자할 종목을 골라 잡으면 됩니다.

장인

4.1 타임머신을 타고 과거에 투자했다면?

미국 주식은 무료로 백테스트할 수 있습니다.

백테스트란 과거 수익률을 확인하는 방법입니다. 타임머신을 타고 옛날에 이렇게 투자했다면 오늘 얼마나 벌었을까 알아 볼 수 있습니다.

포트폴리오를 구성했다면 과거 데이터로 수익률을 꼭 예상해 보세요.

포트폴리오 비주얼라이저를 이용하면 됩니다. 그리고 무료입니다.

<https://www.portfoliovisualizer.com/>

Start Year 1985
End Year 2021
Include YTD No
Initial Amount \$ 100000 .00
Cashflows None
Rebalancing Rebalance annually
Reinvest Dividends Yes

Portfolio Assets에 포트폴리오의 Ticker와 비중을 입력합니다. 보통 동일 비중으로 합니다. 10종목이라면 비중을 10%씩 입력하면 됩니다.

Asset 1	JNJ	Q	10	%		%
Asset 2	T	Q	10	%		%
Asset 3	AFL	Q	10	%		%
Asset 4	MRK	Q	10	%		%
Asset 5	CSCO	Q	10	%		%
Asset 6						

4.1 타임머신을 타고 과거에 투자했다면?

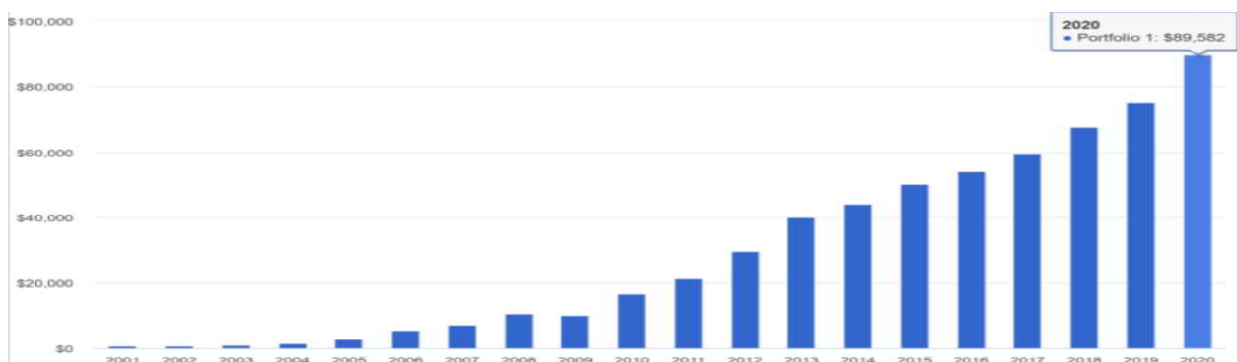
실행하면 1~2초 안에 투자 결과를 확인할 수 있습니다.



연평균 복리수익률(CAGR)도 중요하지만 Max Drawdown도 눈여겨 보세요. 보유 기간중 저만큼 평가 손실이 발생할 수 있다는 겁니다.

이렇게 과거 데이터로 미래의 모습을 그려 보면서 자기 암시를 하면 요즘 같은 대세 하락장에도 장기 투자에 필요한 심신의 안정을 찾을 수 있습니다.

그리고 배당 투자의 가장 큰 장점은 주가가 하락해도 배당금은 계속 올라간다는 겁니다.



자세한 사용 방법은 아래 페이지를 참고하시기 바랍니다.

<https://blog.naver.com/shawgibal/222916920688>

4.2 세금을 아끼자!

미국 주식은 양도세가 있습니다.

250만 원을 초과하는 금액에 대하여 22% 양도소득세가 발생합니다. 그래서 1년에 한 번씩 반드시 최소 250만 원은 수익을 확정하세요. 그래야 절세할 수 있습니다.

저도 2022년 12월이 끝나기 전에 일부 주식을 익절했습니다.

1 [3142] 해외주식 양도소득세(T) - 해외주식 양도소득세 가계산(전계좌)

전계좌가계산

계좌별가계산

고객IDshawsk

ID 비밀번호*****

년도2022년

조회

다음

양도소득총합계2,794,046

기본공제적용금액0

양도소득산출세액614,689

거래소명	주식종목명	종목코드	양도주식수	주당양도가액	양도가액	주당취득가액	취득가액	제비용	비용차감후손익	주식종목코드
미국	록히드 마틴	LMT	4	652,898	2,611,593	425,219	1,700,876	3,011	907,706	US5398301094
미국	맥도날드	MCD	1	353,058	353,058	240,654	240,654	426	111,978	US5801351017
미국	머크	MRK	16	139,100	2,225,602	86,245	1,379,924	2,531	843,147	US589333V1055
미국	화이자	PFE	4	68,801	275,204	40,125	160,501	350	114,353	US7170811035
미국	VANGUARD SHORT-TER	VGSH	120	74,339	8,920,790	67,433	8,091,977	11,951	816,862	US92206C1027
합 계					14,386,247		11,573,932	18,269	2,794,046	

양도소득 : 2,794,046

이렇게 되면 250만을 초과하는 약 294,000원은 양도소득세 대상이 됩니다.

4.2 세금을 아끼자!

그래서 마이너스 구간에 있는 종목을 일부 팔아서 양도차익을 250만원 이내로 줄였습니다. 그리고 계속 보유할 생각이라 곧바로 다시 매수했습니다.

1 [3142] 해외주식 양도소득세(T) - 해외주식 양도소득세 가계산(전계좌)											
전계좌가계산		계좌별가계산									
고객ID	shawsk	ID 비밀번호	*****	년도	2022년						
양도소득출합계		2,259,357		기본공제적용금액		0		양도소득산출세액		497,05	
거래소명	주식종목명	종목코드	양도주식수	주당양도가액	양도가액	주당취득가액	취득가액	제비용	비용차감후손익	주식종목코드	
미국	인텔	INTC	20	38,166	763,333	64,829	1,296,596	1,426	-534,689	US4581401001	
미국	록히드 마틴	LMT	4	652,898	2,611,593	425,219	1,700,876	3,011	907,706	US5398301094	
미국	맥도날드	MCD	1	353,058	353,058	240,654	240,654	426	111,978	US5801351017	
미국	대크	MRK	16	139,100	2,225,602	86,245	1,379,924	2,531	843,147	US58933V1055	
미국	화이자	PFE	4	68,801	275,204	40,125	160,501	350	114,353	US7170811035	
미국	VANGUARD SHORT-TEF	VGSH	120	74,339	8,920,790	67,433	8,091,977	11,951	816,862	US92206C1027	
합 계					15,149,580		12,870,528	19,695	2,259,357		

양도소득 : 2,259,357

미국 주식 양도소득세 절세 관련한 세부 내용은 아래 글을 확인하시면 됩니다.

<https://blog.naver.com/shawgibal/222949463731>

5. 지금 시작하자!

그리고 가장 중요한 투자 원칙은 “지금 바로 시작해야 한다~”는 겁니다.

배당 성장 투자의 핵심은 복리수익이고, 복리의 마법은 시간이 지날수록 강력해집니다.

“복리의 법칙은 세계 8대 불가사의 중의 하나다.”

– 알버트 아인슈타인 –

마지막으로 퀴즈를 풀어 보겠습니다.

만일, 1억을 투자해서 복리 수익률 10%를 올릴 수 있다면 원금이 2배가 되는데 걸리는 시간은 얼마일까요?

10년일까요? 당연히 아닙니다.

약 7.2년이면 투자금이 2배가 되어 총 자산이 2억이 됩니다.

14.4년 후에는 4억이 되고 21.6년 후에는 8억이 되고 28.8년 후에는 16억이 됩니다.

여러분이 30살에 1억을 모아서 10%의 수익률을 꾸준히 올릴 수 있다면 60살에 총 자산은 대략 16억원이 됩니다.

복리의 마법, 72의 법칙

$$\frac{72}{\text{연간 이자율}} = \text{자금이 2배 되는 예상시간}$$

이것이 그 유명한 72의 법칙입니다.

여러분 모두 자기만의 투자 스타일을 만들어서 성투하시길 기원합니다.

이 자료는 대한민국 저작권법의 보호를 받습니다. 작성된 모든 내용의 권리는 작성자에게 있으며, 작성자의 동의 없는 사용이 금지됩니다.

본 자료의 일부 혹은 전체 내용을 무단으로 복제/배포하거나 2차적 저작물로 재편집하는 경우, 5년 이하의 징역 또는 5천만원 이하의 벌금과 민사상 손해배상을 청구합니다.

※ 저작권법 제 30조(사적이용을 위한 복제)

공표된 저작물을 영리를 목적으로 하지 아니하고, 개인적으로 이용하거나 가정 및 이에 준하는 한정된 범위 안에서 이용하는 경우에는 그 이용자는 이를 복제할 수 있다. 다만, 공중의 사용에 제공하기 위하여 설치된 복사기기에 의한 복제는 그러지 아니하다.

※ 저작권법 제 136조(벌칙)의 ① 다음 각 호의 어느 하나에 해당하는 자는 5년 이하의 징역 또는 5천만원 이하의 벌금에 처하거나 이를 병과할 수 있다.

지적재산권 및 이 법에 따라 보호되는 재산적 권리(제93조에 따른 권리는 제외한다)를 복제, 공연, 공중송신, 전시, 배포, 대여, 2차적 저작물 작성의 방법으로 침해한 자

※ 민법 제750조(불법행위의 내용)

고의 또는 과실로 인한 위법행위로 타인에게 손해를 가한 자는 그 손해를 배상할 책임이 있다.