

인생은 짧아요, 엑셀 대신 파이썬

파이썬으로 도전 하는 업무 자동화와 RPA(로봇 프로세스 자동화)

이승준, Sean-June Lee

코딩이랑 무관합니다만,



코무합니다.

혹시 엑셀에서 짹수 번째 행을 전부 삭제하는 기능이 있나요?
일일이 클릭해서 삭제하기엔 열이 많습니당~

	A	B	C	D	E	F	G	H
1	data							
2	data							
3	data							
4	data							
5	data							
6	data							
7	data							
8	data							
9	data							
10	data							
11	data							



엑셀에서 코딩 가능하다는 소릴 들었는데 그쪽을 찾아보심

이..?



컬럼하나 추가해서 첫줄에 0 다음줄에 1 넣고 두개를 선택 드래그해거 패턴으로 끝까지 복사해주고 해당열을 필터 걸어서 1인부분만 삭제하면 됩니다.. 이런 일이 지속적이고 반복적으로 일어날때 그때 파이썬이든 브이비든 매크로든 프로그래밍을 하는거지요..—;;



이런거 하라고 만든게 엑셀매크로(VBA) 일껍니다.



그냥.. 셀 삽입하시고, 0 1 쪽 넣으시고 MOD(셀, 2) 수식넣고 데이터 필터해서 0이든 1이든 필터한 후 다 지우시면 될꺼같은데요?

코딩이랑 유관합니다만,

```
import pandas as pd  
  
df = pd.read_excel('data.xlsx')  
df[::2].to_excel('even.xlsx')
```

홀수 행은?

```
df[1::2].to_excel('odd.xlsx')
```

import 포함해도
딱 3줄

게다가,
엑셀이 없어도

거의 매일 반복되는 단순 반복 작업



모 증권사의 김대리가 매일 아침에 하는 작업 (약 40분 소요)

- 브라우저 띄우기
- 한국거래소(KRX) 페이지 열기
- 엑셀 파일로 다운로드하기
- 파일 이름 변경 (혹은 원하는 위치에 복사)
- 엑셀 컬럼 합치기, 엑셀 파일 생성하기
- 서식 보고서 만들기
- 엑셀을 첨부하여 이메일로 보내기

人生苦短, 使用 파이썬+엑셀



OpenPyXL

XlsxWriter

(쓰기)

xlsx 지원

xls 지원

xlrd, xlwt

엑셀 설치 필요 없음

(윈도우, MacOS, Linux)



xlwings

(윈도우, MacOS)

pywin32

(윈도우)

설치된 엑셀이 필요

파이썬+엑셀 추천



KOKEY 2018

OpenPyXL 

엑셀 파일 읽고 쓰기
(엑셀 설치 없이)



xlwings

엑셀 매크로 자동화
(설치된 엑셀)

XlsxWriter

엑셀 파일을 생성
(서버 사이드)

데이터 준비 - 워크북, 시트 읽기



KOKEY2018

	A	B	C	D
1	date	name	magazine	newspaper
2	2017-01	삼성전자	3.3125	31.18191
3	2017-02	삼성전자	1.21	57.21111
4	2017-03	삼성전자	3.266	57.85807
5	2017-04	삼성전자	2.926	64.80993
6	2017-05	삼성전자	3.585	65.10387
7	2017-06	삼성전자	3.828	74.98342
8	2017-07	삼성전자	5.218	69.54232
9	2017-08	삼성전자	4.302	57.77148
10	2017-09	삼성전자	3.917	73.56007
11	2017-10	삼성전자	4.306	79.57186
12	2017-11	삼성전자	3.987	89.35373
13	2017-12	삼성전자	5.132	91.47235
14				

```
import openpyxl  
  
wb = openpyxl.load_workbook('2017년 광고비 - 삼성전자.xlsx')  
sheet = wb.get_sheet_by_name('Sheet1')  
sheet
```

'2017년 광고비 - 삼성전자.xlsx'

셀에 접근

	A	B	C	D
1	date	name	magazine	newspaper
2	2017-01	삼성전자	3.3125	31.18191
3	2017-02	삼성전자	1.21	57.21111
4	2017-03	삼성전자	3.266	57.85807
5	2017-04	삼성전자	2.926	64.80993
6	2017-05	삼성전자	3.585	65.10387
7	2017-06	삼성전자	3.828	74.98342
8	2017-07	삼성전자	5.218	69.54232
9	2017-08	삼성전자	4.302	57.77148
10	2017-09	삼성전자	3.917	73.56007
11	2017-10	삼성전자	4.306	79.57186
12	2017-11	삼성전자	3.987	89.35373
13	2017-12	삼성전자	5.132	91.47235
14				

```
sheet['A2'].value
```

```
'2017-01'
```

```
sheet['B1'].value
```

```
'name'
```

```
sheet['C1'].value = 43700
```

```
sheet.cell(row = 1, column = 3).value
```

43700

범위에 접근

```
multi_cells = sheet['E2':'F14']
multi_cells
```

```
((<Cell 'Sheet1'.E2>,
 <Cell 'Sheet1'.E3>,
 <Cell 'Sheet1'.E4>,
 <Cell 'Sheet1'.E5>,
 <Cell 'Sheet1'.E6>,
 <Cell 'Sheet1'.E7>,
 <Cell 'Sheet1'.E8>,
 <Cell 'Sheet1'.E9>,
 <Cell 'Sheet1'.E10>,
 <Cell 'Sheet1'.E11>,
 <Cell 'Sheet1'.E12>,
 <Cell 'Sheet1'.E13>,
 <Cell 'Sheet1'.E14>,
 # 모든 row 살펴보기
 for row in sheet.rows:
     print([col.value for col in row])
     ['date', 'name', 43700, 'newspaper', 'radio', 'tv', 'total', 'rank']
     ['2017-01', '삼성전자', 3.3125, 31.18191, 2.58051, 75.7927, 112.86761, 1]
     ['2017-02', '삼성전자', 1.21, 57.21111, 2.58306, 60.95579, 121.95995, 1]
     ['2017-03', '삼성전자', 3.266, 57.85807, 2.51379, 65.58804, 129.22589, 2]
     ['2017-04', '삼성전자', 2.926, 64.80993, 2.60663, 119.3088, 189.65136, 1]
     ['2017-05', '삼성전자', 3.585, 65.10387, 2.56629, 99.69942, 170.95458, 1]
     ['2017-06', '삼성전자', 3.828, 74.98342, 2.59602, 58.38074, 139.78818, 1]
     ['2017-07', '삼성전자', 5.218, 69.54232, 2.62309, 69.72003, 147.10345, 1]
     ['2017-08', '삼성전자', 4.302, 57.77148, 3.61393, 111.52323, 177.21063, 1]
     ['2017-09', '삼성전자', 3.917, 73.56007, 2.42469, 116.6872, 196.58896, 1]
     ['2017-10', '삼성전자', 4.306, 79.57186, 2.68068, 198.73793, 285.29646, 1]
     ['2017-11', '삼성전자', 3.987, 89.35373, 2.4953, 223.63268, 319.46871, 1]
     ['2017-12', '삼성전자', 5.132, 91.47235, 2.35521, 39.721, 138.68057, 2]
     [None, None, None, None, None, None, None, None]
```

데이터 병합

	A	B	C	D	E	F	G
1	date	name	magazine	newspaper	radio	tv	total
2	2017-01	삼성전자	3.3125	31.18191	2.58051	75.7927	112.8676
3	2017-02	삼성전자	1.21	57.21111	2.58306	60.95579	121.96
4	2017-03	삼성전자	3.266	57.85807	2.51379	65.58804	129.2259
5	2017-04	삼성전자	2.926	64.80993	2.60663	119.3088	189.6514
6	2017-05	삼성전자	3.585	65.10387	2.56629	99.69942	170.9546
7	2017-06	삼성전자	3.828	74.98342	2.59602	58.38074	139.7882
8	2017-07	삼성전자	5.218	69.54232	2.62309	69.72003	147.1035
9	2017-08	삼성전자	4.302	57.77148	3.61393	111.5232	177.2106
10	2017-09	삼성전자	3.917	73.56007	2.42469	116.6872	196.589
11	2017-10	삼성전자	4.306	79.57186	2.68068	198.7379	285.2965
12	2017-11	삼성전자	3.987	89.35373	2.4953	223.6327	319.4687
13	2017-12	삼성전자	5.132	91.47235	2.35521	39.721	138.6806
14							

'2017년 광고비 - 삼성전자.xlsx'

	A	B	C	D	E	F	G
1	date	name	magazine	newspaper	radio	tv	total
2	2017-01	LG전자	0.645	13.9993	0.90878	43.56334	59.11641
3	2017-02	LG전자	0.86	11.18028	1.07101	29.09002	42.2013
4	2017-03	LG전자	0.756	22.98917	4.58819	113.1466	141.4799
5	2017-04	LG전자	0.61	15.7484	5.58151	148.3207	170.2606
6	2017-05	LG전자	1.754	17.71454	5.29391	111.5183	136.2807
7	2017-06	LG전자	1.704	13.94278	2.92095	97.93962	116.5074
8	2017-07	LG전자	0.899	15.64618	1.82889	86.045	104.4191
9	2017-08	LG전자	1.094	22.81309	0.97381	68.18897	93.06988
10	2017-09	LG전자	1.557	26.79913	3.22223	137.7601	169.3385
11	2017-10	LG전자	1.48	8.97103	3.07307	168.7502	182.2743
12	2017-11	LG전자	1.9629	19.53747	1.75838	156.8406	180.0993
13	2017-12	LG전자	2.46	14.27291	1.24421	227.0468	245.024
14							

'2017년 광고비 - LG전자.xlsx'

두 엑셀 파일을 하나로 합치기

	A	B	C
1	date	삼성전자	LG전자
2	2017-01	112.8676	59.11641
3	2017-02	121.96	42.2013
4	2017-03	129.2259	141.4799
5	2017-04	189.6514	170.2606
6	2017-05	170.9546	136.2807
7	2017-06	139.7882	116.5074
8	2017-07	147.1035	104.4191
9	2017-08	177.2106	93.06988
10	2017-09	196.589	169.3385
11	2017-10	285.2965	182.2743
12	2017-11	319.4687	180.0993
13	2017-12	138.6806	245.024
14			

```
import pandas as pd

df_삼성전자 = pd.read_excel('2017년 광고비 - 삼성전자.xlsx')
df_삼성전자.set_index('date', inplace=True)

df_LG전자 = pd.read_excel('2017년 광고비 - LG전자.xlsx')
df_LG전자.set_index('date', inplace=True)

df_merge = pd.DataFrame()

df_merge['삼성전자'] = df_삼성전자['total']
df_merge['LG전자'] = df_LG전자['total']
```

df_merge

계산 및 계산 결과 저장

A	B	C
1	date	삼성전자
2	2017-01	112.8676
3	2017-02	59.11641
4	2017-03	121.96
5	2017-04	42.2013
6	2017-05	129.2259
7	2017-06	141.4799
8	2017-07	189.6514
9	2017-08	170.2606
10	2017-09	170.9546
11	2017-10	136.2807
12	2017-11	139.7882
13	2017-12	116.5074
14	합계	147.1035
15		104.4191
		177.2106
		196.589
		285.2965
		319.4687
		138.6806
		2128.796
		1640.071

합계 계산

```
sum_삼성전자 = sum([row[0].value for row in sheet['B2':'B13']])
sheet['B14'].value = sum_삼성전자
```

```
sum_LG전자 = sum([row[0].value for row in sheet['C2':'C13']])
sheet['C14'].value = sum_LG전자
```

수식 넣기

	A	B	C
1	date	삼성전자	LG전자
2	2017-01	112.8676	59.11641
3	2017-02	121.96	42.2013
4	2017-03	129.2259	141.4799
5	2017-04	189.6514	170.2606
6	2017-05	170.9546	136.2807
7	2017-06	139.7882	116.5074
8	2017-07	147.1035	104.4191
9	2017-08	177.2106	93.06988
10	2017-09	196.589	169.3385
11	2017-10	285.2965	182.2743
12	2017-11	319.4687	180.0993
13	2017-12	138.6806	245.024
14	합계	2128.796	1640.071
15			

합계 계산

```
sheet[ 'B14' ].value = '=SUM(B2:B13)'  
sheet[ 'C14' ].value = '=SUM(C2:C13)'
```

스타일 지정

	A	B	C
1	date	삼성전자	LG전자
2	2017-01	112.8676	59.11641
3	2017-02	121.96	42.2013
4	2017-03	129.2259	141.4799
5	2017-04	189.6514	170.2606
6	2017-05	170.9546	136.2807
7	2017-06	139.7882	116.5074
8	2017-07	147.1035	104.4191
9	2017-08	177.2106	93.06988
10	2017-09	196.589	169.3385
11	2017-10	285.2965	182.2743
12	2017-11	319.4687	180.0993
13	2017-12	138.6806	245.024
14	합계	2128.796	1640.071
15			

```
from openpyxl.styles import Font, Alignment
from openpyxl.styles import Border, Side, Color, PatternFill

# Font: '맑은 고딕', 크기 15, 굵게
font_15 = Font(name='맑은 고딕', size=15, bold=True)

# Alignment: 가로 세로, 가운데 정렬
align_center = Alignment(horizontal='center', vertical='center')
align_vcenter = Alignment(vertical='center')

# Border: 테두리 지정
border_thin = Border(
    left=Side(style='thin'), right=Side(style='thin'),
    top=Side(style='thin'), bottom=Side(style='thin'))

# PatternFill: 셀 색상 지정
fill_orange = PatternFill(patternType='solid', fgColor=Color('FFC000'))
fill_lightgrey = PatternFill(patternType='solid', fgColor=Color('D3D3D3'))

cell_sum = sheet['A14'] # 합계 제목 셀

cell_sum.font = font_15
cell_sum.alignment = align_center
cell_sum.border = border_thin
cell_sum.fill = fill_orange
```

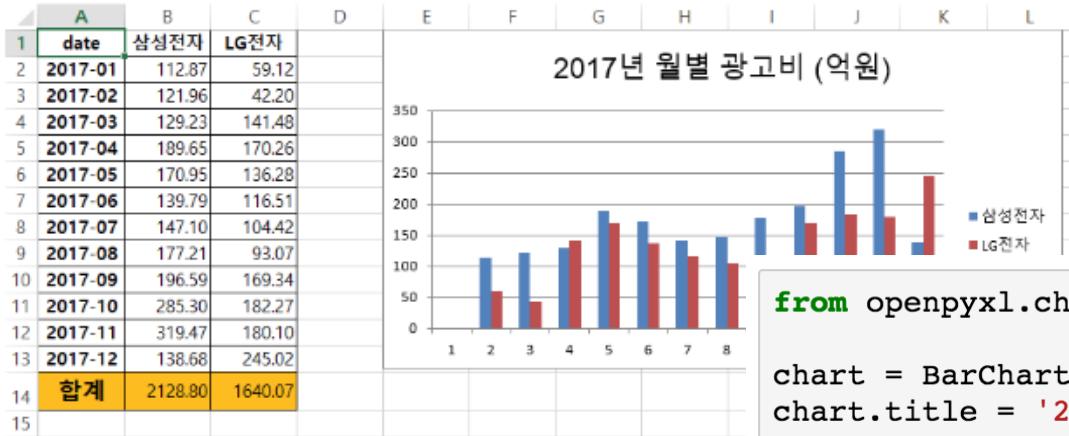
범위에 스타일 지정

	A	B	C
1	date	삼성전자	LG전자
2	2017-01	112.87	59.12
3	2017-02	121.96	42.20
4	2017-03	129.23	141.48
5	2017-04	189.65	170.26
6	2017-05	170.95	136.28
7	2017-06	139.79	116.51
8	2017-07	147.10	104.42
9	2017-08	177.21	93.07
10	2017-09	196.59	169.34
11	2017-10	285.30	182.27
12	2017-11	319.47	180.10
13	2017-12	138.68	245.02
14	합계	2128.80	1640.07
15			

```
for row in sheet['B2:C14']:
    for cell in row:
        cell.border = border_thin
        cell.number_format = '0.00'

for row in sheet['B14:C14']:
    for cell in row:
        cell.alignment = align_vcenter
        cell.fill = fill_orange
```

차트 추가하기



```
from openpyxl.chart import Reference, Series, BarChart

chart = BarChart()
chart.title = '2017년 월별 광고비 (억원)'

vals = Reference(sheet, range_string= 'Sheet1!B1:B13')
series = Series(vals, title="삼성전자")
chart.append(series)

vals = Reference(sheet, range_string= 'Sheet1!C1:C13')
series = Series(vals, title="LG전자")
chart.append(series)

sheet.add_chart(chart, 'E1')
```

엑셀을 쓰는 가장 큰 이유 3가지



1. 필터링과 소트
2. VLOOKUP
3. 피벗 테이블

데이터 준비

A	B	C	D	E	F
1	종목코드	종목명	PER	PBR	주당배당금 배당수익률
2	005930	삼성전자	15.91	1.98	21000 1.05
3	000660	SK하이닉스	8.2	1.67	500 1.02
4	005380	현대차	6.12	0.67	4000 2.74
5	015760	한국전력	2.01	0.4	3100 7.45
6	035420	NAVER	45.41	12.41	1100 0.14
7	005490	POSCO	153.39	0.6	8000 2.83
8	012330	현대모비스	7.62	0.93	3500 1.43
9	028260	삼성물산	5.84	1.4	500 0.41
10	055550	신한지주	9.89	0.75	1200 2.53
11	032830	삼성생명	16.75	0.91	1800 1.67
12	105560	KB금융	11.15	0.66	980 2
13	017670	SK텔레콤	11.77	1.31	10000 4.05
14	051910	LG화학	17	1.52	4500 1.69
15					

```
import pandas as pd
```

```
df_팩터 = pd.read_excel('종목별 팩터 데이터.xlsx', dtype={'종목코드':str})  
df_팩터 = df_팩터.set_index('종목코드')  
df_팩터
```

종목코드	종목명	PER	PBR	주당배당금	배당수익률
005930	삼성전자	15.91	1.98	21000	1.05
000660	SK하이닉스	8.20	1.67	500	1.02
005380	현대차	6.12	0.67	4000	2.74
015760	한국전력	2.01	0.40	3100	7.45
035420	NAVER	45.41	12.41	1100	0.14
005490	POSCO	153.39	0.60	8000	2.83
012330	현대모비스	7.62	0.93	3500	1.43
028260	삼성물산	5.84	1.40	500	0.41
055550	신한지주	9.89	0.75	1200	2.53

필터링



A	B	C	D	E	F	
1	종목코드	종목명	PER	PBR	주당배당금	배당수익률
3	000660	SK하이닉스	8.2	1.67	500	1.02
4	005380	현대차	6.12	0.67	4000	2.74
5	015760	한국전력	2.01	0.4	3100	7.45
8	012330	현대모비스	7.62	0.93	3500	1.43
9	028260	삼성물산	5.84	1.4	500	0.41
10	055550	신한지주	9.89	0.75	1200	2.53

```
df_팩터[df_팩터['PER'] <= 10]
```

종목명	PER	PBR	주당배당금	배당수익률
-----	-----	-----	-------	-------

종목코드

000660	SK하이닉스	8.20	1.67	500	1.02
005380	현대차	6.12	0.67	4000	2.74
015760	한국전력	2.01	0.40	3100	7.45
012330	현대모비스	7.62	0.93	3500	1.43
028260	삼성물산	5.84	1.40	500	0.41
055550	신한지주	9.89	0.75	1200	2.53

정렬

배당수익률을 내림차순으로 정렬

A	B	C	D	E	F	
1	종목코드	종목명	PER	PBR	주당배당	배당수익률
3	015760	한국전력	2.01	0.4	3100	7.45
4	005380	현대차	6.12	0.67	4000	2.74
5	055550	신한지주	9.89	0.75	1200	2.53
8	012330	현대모비스	7.62	0.93	3500	1.43
9	000660	SK하이닉스	8.2	1.67	500	1.02
10	028260	삼성물산	5.84	1.4	500	0.41
15						

```
df = df_팩터[df_팩터['PER'] <= 10].sort_values('배당수익률', ascending=False)
```

종목명	PER	PBR	주당배당금	배당수익률
종목코드				
015760	한국전력	2.01	0.40	3100 7.45
005380	현대차	6.12	0.67	4000 2.74
055550	신한지주	9.89	0.75	1200 2.53
012330	현대모비스	7.62	0.93	3500 1.43
000660	SK하이닉스	8.20	1.67	500 1.02
028260	삼성물산	5.84	1.40	500 0.41

VLOOKUP

=VLOOKUP(A2,H2 :M\$14, 3, FALSE)

C2	=VLOOKUP(\$A2, \$H\$2:\$M\$14, 3, FALSE)											
A	B	C	D	E	F	G	H	I	J	K	L	M
1	종목코드	종목명	PER	PBR	배당수익률		종목코드	종목명	PER	PBR	주당배당금	배당수익률
2	005930	삼성전자	15.91	1.98	1.05		005930	삼성전자	15.91	1.98	21000	1.05
3	000660	SK하이닉스	8.2	1.67	1.02		000660	SK하이닉스	8.2	1.67	500	1.02
4	105560	KB금융	11.15	0.66	2		005380	현대차	6.12	0.67	4000	2.74
5	051910	LG화학	17	1.52	1.69		015760	한국전력	2.01	0.4	3100	7.45
6	090430	아모레퍼시픽	#N/A	#N/A	#N/A		035420	NAVER	45.41	12.41	1100	0.14
7							005490	POSCO	153.39	0.6	8000	2.83
8							012330	현대모비스	7.62	0.93	3500	1.43
9							028260	삼성물산	5.84	1.4	500	0.41
10							055550	신한지주	9.89	0.75	1200	2.53
11							032830	삼성생명	16.75	0.91	1800	1.67
12							105560	KB금융	11.15	0.66	980	2
13							017670	SK텔레콤	11.77	1.31	10000	4.05
14							051910	LG화학	17	1.52	4500	1.69
15												

df_관심종목[['PER', 'PBR', '배당수익률']] = df_팩터[['PER', 'PBR', '배당수익률']]
df_관심종목

엑셀 파일로 저장



컬럼 너비, 테두리, 포맷 지정하여 엑셀로 저장

A	B	C	D	E
1	종목코드	종목명	PER	PBR
2	005930	삼성전자	15.91	1.98
3	000660	SK하이닉스	8.20	1.67
4	105560	KB금융	11.15	0.66
5	051910	LG화학	17.00	1.52
6				1.69

```
import pandas as pd

writer = pd.ExcelWriter("관심팩터병합.xlsx", engine='openpyxl')
df_관심종목.to_excel(writer, sheet_name='Sheet1')

wb = writer.book
sheet = writer.sheets['Sheet1']

# 컬럼 너비 지정
for col in list('BCDE'):
    sheet.column_dimensions[col].width = 14

# 테두리, 숫자포맷 지정
for row in sheet['B2:E5']:
    for cell in row:
        cell.border = border_thin
        cell.number_format = '0.00'

# 헤더 색상 지정
for row in sheet['A1:E1']:
    for cell in row:
        cell.fill = fill_lightgrey

writer.save()
```

피벗 테이블

```
import pandas as pd

df = pd.read_excel('광고비 데이터 3사.xlsx', dtype={'종목코드':str})
table = pd.pivot_table(df, values='total', index='date', columns='name')
table
```

	A	B	C	D	E		
1	date	name	magazine	newspaper	radio	tv	total
2	2018-01	LG전자	0.749	23.40148	0.88007	133.0973	158.12784
3	2018-01	삼성전자	3.057	23.52071	2.61946	80.55491	109.75208
4	2018-01	현대자동차	1.143	27.66826	4.65094	48.63957	82.10177
5	2018-02	삼성전자	2.293	54.42794	2.53148	85.2717	144.52412
6	2018-02	LG전자	0.5815	7.01109	0.85703	113.56652	122.01614
7	2018-02	현대자동차	0.606	23.11229	4.51387	61.49156	89.72372
8	2018-03	LG전자	1.1492	24.50926	0.88181	149.00252	175.54279
9	2018-03	삼성전자	2.538	63.04994	2.57359	98.79365	166.95518
10	2018-03	현대자동차	1.081	17.22717	4.70813	45.72437	68.74067
11							
12		합계 : total	열 레이블				
13		행 레이블	LG전자	삼성전자	현대자동차	총합계	
14		2018-01	158.12784	109.75208	82.10177	349.98169	
15		2018-02	122.01614	144.52412	89.72372	356.26398	
16		2018-03	175.54279	166.95518	68.74067	411.23864	
17		총합계	455.68677	421.23138	240.56616	1117.48431	
18							

name	LG전자	삼성전자	현대자동차
date			
2018-01	158.12784	109.75208	82.10177
2018-02	122.01614	144.52412	89.72372
2018-03	175.54279	166.95518	68.74067

엑셀에서는 약간 까다로운 작업



특정 주기로 데이터 추출 pd.date_range()

- 매주 월요일
- 매월 1일
- 매분기 첫 영업일
- 매월 마지막 영업일

df_01.head()

년/월/일	종가	대비	거래량(주)	거래대금(원)	시가	고가	저가	시가총액(백만)
2017-12-28	2548000	80000	179709	453635894717	2478000	2548000	2475000	328942963
2017-12-27	2468000	58000	214872	526055660822	2448000	2478000	2423000	318615083
2017-12-26	2410000	-75000	320797	787684546000	2488000	2505000	2410000	311127371
2017-12-22	2485000	28000	223993	555690493000	2470000	2498000	2462000	320809758
2017-12-21	2457000	-87000	312486	777078978780	2550000	2553000	2455000	317195000

```
# 2017년, 매주 월요일
inx = pd.date_range('2017-01-01', '2017-12-31', freq='W-MON')

# 2017년, M (매월 말일)
inx = pd.date_range('2017-01-01', '2017-12-31', freq='M')

# 2017년, BQS (매분기 첫 영업일)
inx = pd.date_range('2017-01-01', '2017-12-31', freq='BQS')

# 2017년, BM (매월 마지막 영업일)
inx = pd.date_range('2017-01-01', '2017-12-31', freq='BM')

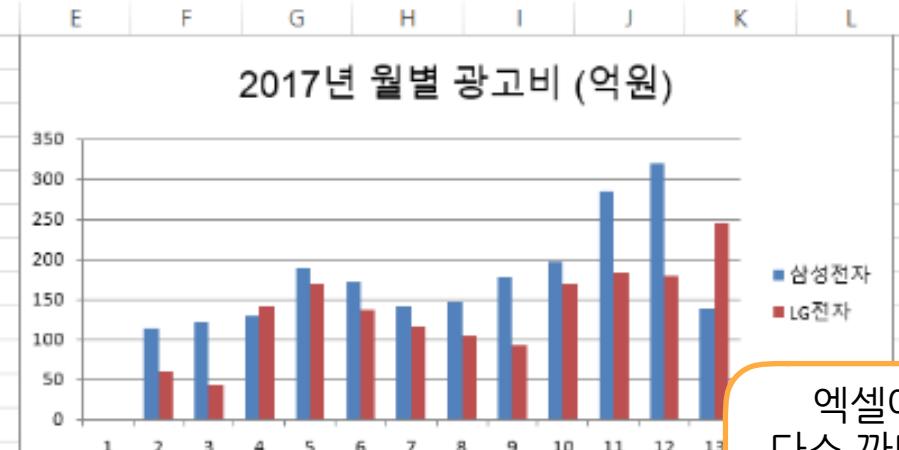
df_x = pd.DataFrame(index=inx)
df_x['종가'] = df_01['종가']
df_x.head(10)
```

종가	
2017-01-31	1973000.0
2017-02-28	1922000.0
2017-03-31	2060000.0
2017-04-28	2231000.0
2017-05-31	2235000.0
2017-06-30	2377000.0
2017-07-31	2410000.0
2017-08-31	2316000.0

파이썬 pandas = 코드로 다루는 엑셀

엑셀 파일
읽기, 쓰기

A	B	C	D	E	F	G	H	I	J	K	L
1	date	삼성전자	LG전자								
2	2017-01	112.87	59.12								
3	2017-02	121.96	42.20								
4	2017-03	129.23	141.48								
5	2017-04	189.65	170.26								
6	2017-05	170.95	136.28								
7	2017-06	139.79	116.51								
8	2017-07	147.10	104.42								
9	2017-08	177.21	93.07								
10	2017-09	196.59	169.34								
11	2017-10	285.30	182.27								
12	2017-11	319.47	180.10								
13	2017-12	138.68	245.02								
14	합계	2128.80	1640.07								
15											



셀 서식

수식
 $=sum(A0:A10)$

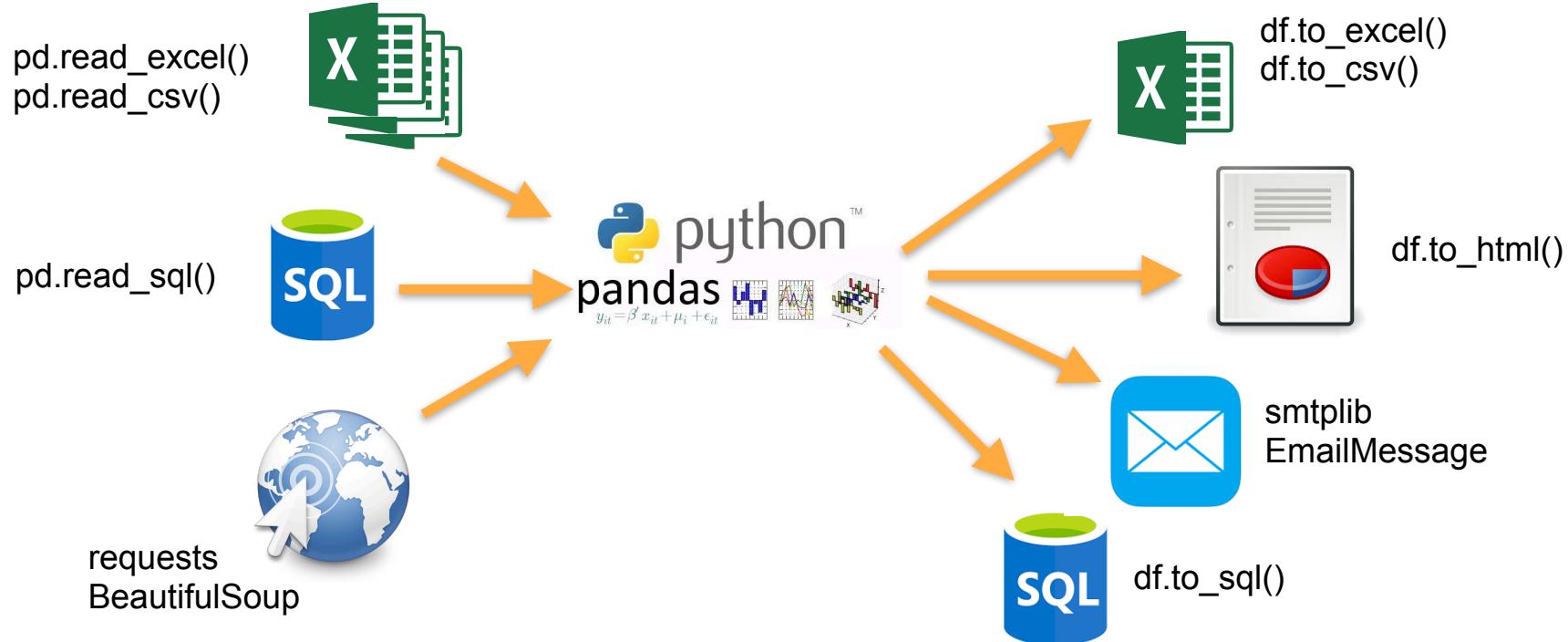
차트 삽입

사실상 대부분의
엑셀 작업 대체 가능

엑셀에서
다소 까다로운
시계열 연산

속도는 둠

파이썬 pandas = 데스크탑 데이터 허브





KRX 상장회사 목록 엑셀 가져오기



1. 브라우저 실행

```
from selenium import webdriver  
driver = webdriver.Chrome('C:/Users/Seung-June/bin/chromedriver.exe')
```

2. 페이지 방문

```
driver.get('http://marketdata.krx.co.kr/mdi#document=040601')
```

3. 다운로드 버튼을 클릭

```
from selenium.webdriver.common.by import By
```

```
button = driver.find_element(By.XPATH, '//button[text()="Excel"]')  
button.click()
```

4. 파일명 바꾸기

```
import os  
os.rename('data.xls', '상장회사.xls')
```

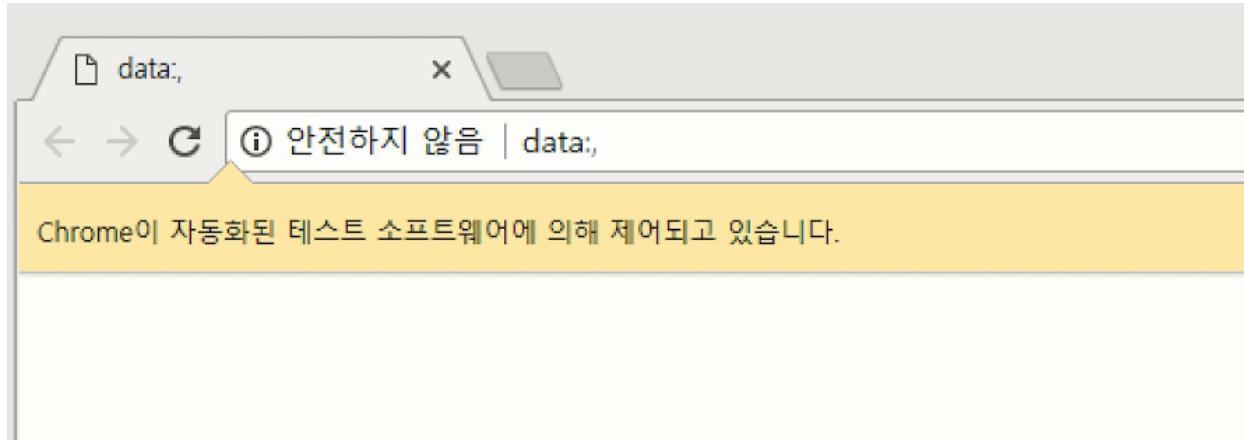
The screenshot illustrates the workflow for extracting KRX listed company data:

- Browser Window:** Shows the KRX market data page with a search form for "상장회사검색". The search parameters are set to "전체" (All) under "시장구분" and "전체" under "총목검색".
- File Download Dialog:** A "다운로드" (Download) dialog is open, showing the file path "Seung-June > 다운로드" and the file name "상장회사목록.xls".
- File Explorer:** A Windows File Explorer window shows the downloaded file "상장회사목록.xls" in the "다운로드" folder.

The extracted data is a table of listed companies:

종목코드	기업명	업종코드
060310	SS	032902
095570	AJ네트웍스	147603
068400	AJ렌티카	147601
006840	AK플러스	116409
054620	APS홀딩스	116409
265520	AP시스템	032902

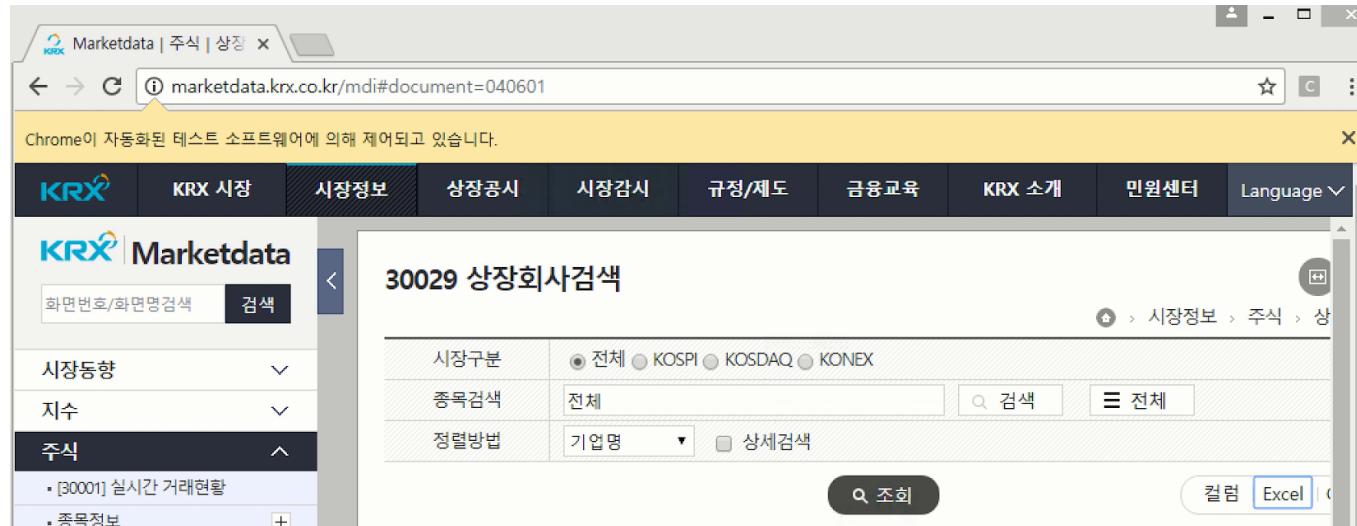
브라우저 자동화 - 1) 브라우저 실행



1. 브라우저 실행

```
from selenium import webdriver
driver = webdriver.Chrome('C:/Users/Seung-June/bin/chromedriver.exe')
```

브라우저 자동화 - 2) 페이지 방문하기



2. 페이지 방문

```
driver.get('http://marketdata.krx.co.kr/mdi#document=040601')
```

브라우저 자동화 - 3) 다운로드 버튼 클릭

The screenshot shows a web application for searching stock market data. On the left, there's a sidebar with dropdown menus for '시장동향', '지수', and '주식'. The '주식' menu is expanded, showing categories like '실시간 거래현황', '종목정보', '투자참고', '순위정보', '기타상품', and '상장현황'. Below these are more collapsed sections for '채권', '파생', '일반상품', 'ETF/ETN/ELW', and '해외여계시작 정보'. A file icon with 'data.xls' is visible at the bottom of the sidebar. The main area has search filters for '시장구분' (All, KOSPI, KOSDAQ, KONEX), '종목검색' (All), and '정렬방법' (기업명). A search button and a download button labeled 'Excel' are present. To the right, a file explorer window titled '다운로드' shows a single item: 'data' (2018-04-22 오전 2:10, Microsoft Excel 9...).

3. 다운로드 버튼을 클릭

```
from selenium.webdriver.common.by import By
```

```
button = driver.find_element(By.XPATH, '//button[text()="Excel"]')
button.click()
```

브라우저 자동화 - 4) 파일명 바꾸기

The screenshot shows a financial trading platform interface. On the left, there's a sidebar with dropdown menus for '시장동향' (Market Trends), '지수' (Indices), '주식' (Stocks), '채권' (Bonds), '파생' (Derivatives), '일반상품' (General Commodities), 'ETF/ETN/ELW' (ETF/ETN/ELW), and '해외연계시작 정보' (Overseas Linkage Start Information). A file named 'data.xls' is selected in the sidebar. The main area has filters for '시장구분' (Market Segment) set to '전체' (All), '종목검색' (Stock Search) set to '전체' (All), and '정렬방법' (Sort Method) set to '기업명' (Company Name). Below these filters is a search bar with a '조회' (Search) button. To the right of the search bar is a '컬럼' (Column) button with 'Excel' selected. A large blue rectangular area covers the top portion of the main content. In the bottom right corner, a file download dialog box is open, titled '다운로드' (Download). It shows a list of files in the 'Seung-June' folder, specifically in the '다운로드' subfolder. One file is selected: '상장회사' (Listed Company), which was downloaded on April 22, 2018, at 2:10 AM, in Microsoft Excel 9 format.

종목코드	기업명	업종코드
060310	3S	032902
095570	AJ네트웍스	147603
068400	AJ렌터카	147601
006840	AK홀딩스	116409
054620	APS홀딩스	116409

4. 파일명 바꾸기

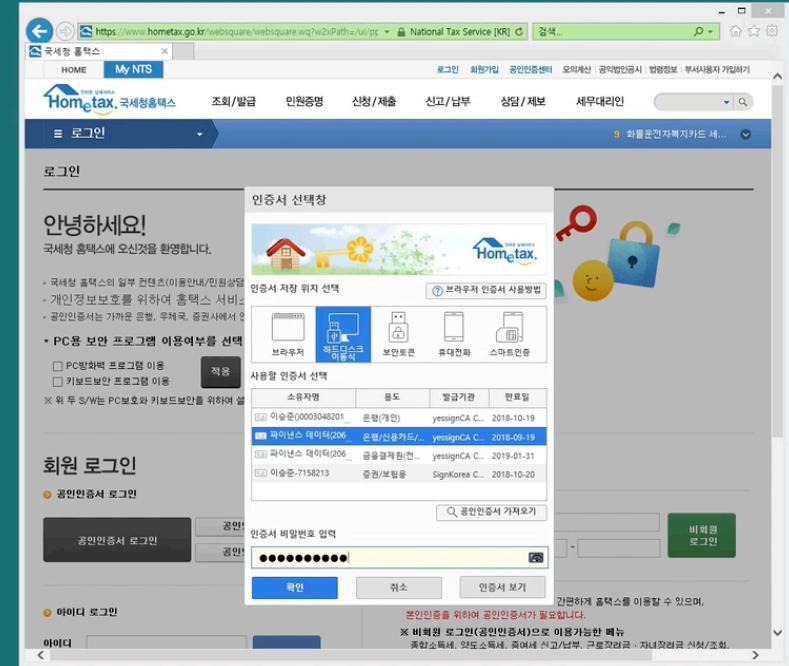
```
import os
```

```
os.rename('data.xls', '상장회사.xls')
```



국세청 홈텍스 로그인(인증서 기반)

PYCON
KOREA 2018



Windows 8.1 Pro K
Build 9600

```
from selenium import webdriver
driver = webdriver.Ie('C:/bin/IEDriverServer.exe')

# 홈텍스로 이동, 상단 로그인
driver.get('https://www.hometax.go.kr/')
driver.find_element_by_id('group1544').click()

# 메인 영역
iframe = driver.find_element_by_id('txppIframe')
driver.switch_to.frame(iframe)

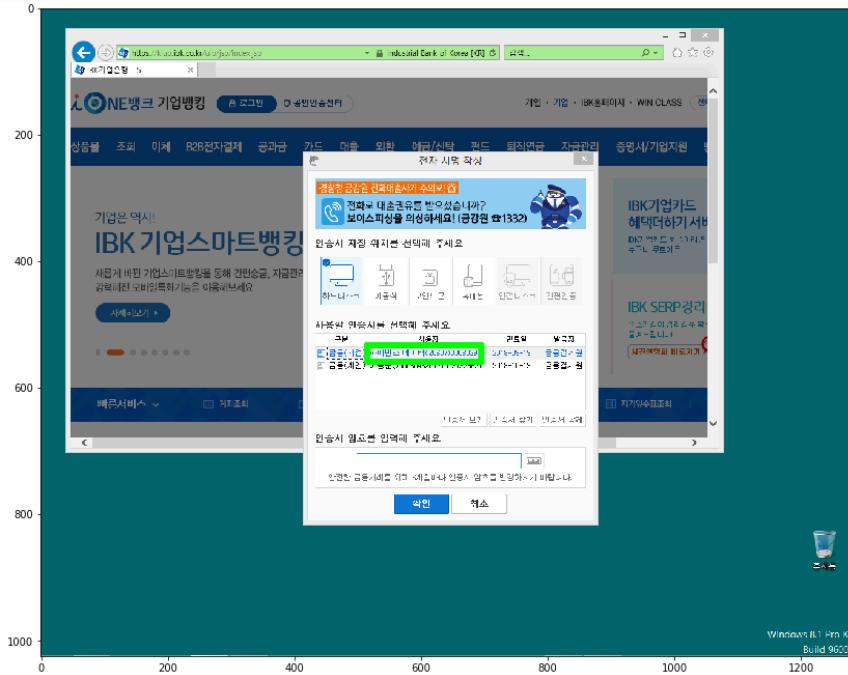
# 로그인 버튼
driver.find_element_by_id('trigger38').click()

# 공인인증서 영역
iframe = driver.find_element_by_id('dscert')
driver.switch_to.frame(iframe)

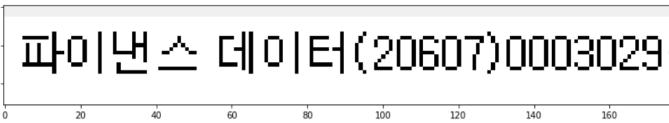
# 공인인증서 선택
driver.find_element_by_xpath('//*[@title="파이낸스 데이터(20607)001"]')

# 인증서 비밀번호 입력
passwd = '*****'
driver.find_element_by_id('input_cert_pw').send_keys(passwd)
driver.find_element_by_id('btn_confirm_iframe').click()
```

OpenCV - matchTemplate()



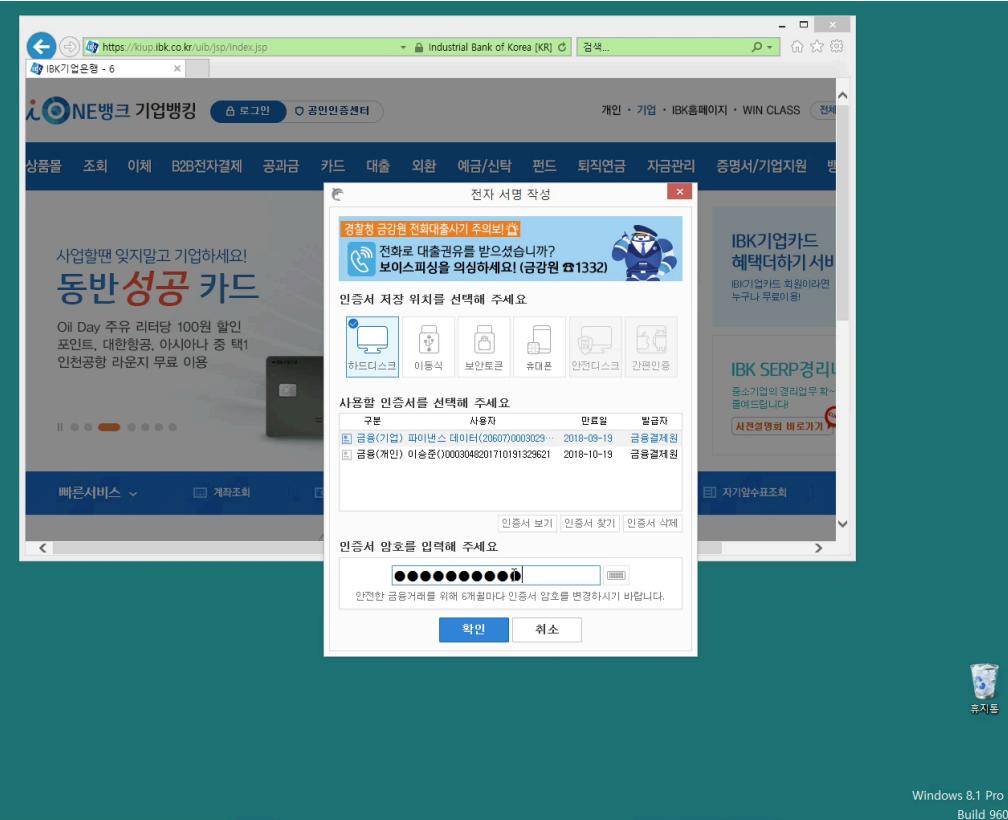
```
capture = pyautogui.screenshot()
```



```
res = cv2.matchTemplate(screen, template, cv2.TM_CCOEFF_NORMED)
min_val,max_val,min_loc, max_loc = cv2.minMaxLoc(res)
top_left = max_loc
h,w = template.shape[:2]
x, y = int(top_left[0] + w/2), int(top_left[1] + h/2)
```



액티브X 컨트롤 제어 자동화



- ActiveX 컴포넌트
- 인증서 로그인
- OpenCV, pyautogui

인증서 암호를 입력해 주세요

안전한 금융거래를 위해 6개월마다 인증서 암호를 변경하시기 바랍니다.

```
template = cv2.imread('img_inputpass.png', cv2.IMREAD_COLOR)
match_pos = match_center_loc(screenshot, template)
pyautogui.moveTo(match_pos)
pyautogui.click()
```

```
passwd = '*****'
pyautogui.typewrite(passwd)
```

네이버 로그인, 쪽지 쓰기

The screenshot shows a Windows desktop environment. On the left is a browser window for 'NAVER 쪽지' (Naver Notes) at the URL [https://note.naver.com/#%7B'sHistoryFunction'%3A" write "%2C'sWriteType'%3A" new "%2C'oParameter'%3A%7B'tai](https://note.naver.com/#%7B'sHistoryFunction'%3A). The browser title bar says '쪽지 쓰기 | 네이버 쪽지'. The page displays a note creation form with fields for '받는 사람' (Recipient), '내게쓰기' (Send to me), and '쪽지쓰기' (Compose note). A message in Korean is visible below the recipient field. On the right is a Python code editor window titled 'Untitled - Python'. The code is a script for automating Naver Note operations:

```
import requests
from selenium import webdriver
from bs4 import BeautifulSoup

# 1. 네이버 로그인
아이디 = '*****'
패스워드 = '*****'

driver = webdriver.Chrome()
driver.get('https://nid.naver.com/nidlogin.login')
driver.find_element_by_name('id').send_keys(아이디)
driver.find_element_by_name('pw').send_keys(패스워드)
driver.find_element_by_xpath('//*[@id="frmNIDLogin"]/fieldset/input').click()

# 2. 쪽지쓰기로 이동
driver.get('https://note.naver.com/')
driver.find_element_by_xpath('//*[@id="menu_write"]/a[1]/span').click()

# 3. 수신자, 메시지에 메시지 쓰고, 전송
to_whom = 'hodoldad'
note = '''안녕하세요 신세 만아오
주인님 산책 공놀이 늘 고맙스이다
'''
driver.find_element_by_id('who').send_keys(to_whom)
driver.find_element_by_id('writeNote').send_keys(note)
driver.find_element_by_class_name('ico_send').click()
```

보고서 생성 - Jinja2

Jinja2 템플릿 엔진을 사용한 보고서 생성

```
html_report_tmpl = """
<!DOCTYPE html>
<html>
<head lang="ko">
    <meta charset="UTF-8">
    <title>{{ title1 }}</title>
</head>
<body>
    <h1>{{ title1 }}</h1>
    {{headline_text}}
    <h2>{{ title2 }}</h2>
    {{ df_html }}
    {{body_text}}
</body>
</html>
"""
```

```
from jinja2 import Template
t = Template(html_report_tmpl)

title1='2018년 환율 전망'
title2 = "2018년 1월 원달러 환율"
headline_text = '다수 전문가들은 올해 환율이 더 낮은 수준에 머물 것으로 내다보고 있다'
body_text = '미국의 기준금리 인상과 감세정책 등 달러화 강세 요인에도 불구하고 .... 때문이란 게 연구원의 분석이다.'

df_html = df.to_html()
html_report = t.render(title1=title1, title2=title2, headline_text=headline_text, body_text=body_text, df_html = df_html)
```

2018년 환율 전망

다수 전문가들은 올해 환율이 더 낮은 수준에 머물 것으로 내다보고 있다

2018년 1월 원달러 환율

DEXKOUS

DATE

2018-01-01	NaN
2018-01-02	1059.39
2018-01-03	1063.11
2018-01-04	1061.50
2018-01-05	1060.07
2018-01-08	1066.26
2018-01-09	1068.80
2018-01-10	1067.55

미국의 기준금리 인상과 감세정책 등 달러화 강세 요인에도 불구하고 때문이란 게 연구원의 분석이다.

이미지와 표 포함된 보고서

```
<div>
  <p>Chart</p>
  
</div>
```

```
df_2018_01 = dpr.DataReader('DEXKOUS', 'fred', '2018-01-01', '2018-01-10')
df_2017_12 = dpr.DataReader('DEXKOUS', 'fred', '2017-01-01', '2017-12-31')
ax = df_2017_12['DEXKOUS'].plot()
fig = ax.get_figure()
fig_data = get_fig_base64(fig)

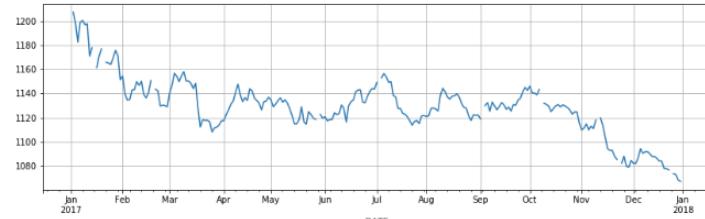
render_data = {
    'title1': '2018년 환율 전망',
    'title2': '2018년 1월 원달러 환율',
    'headline_text': '다수 전문가들은 올해 환율이 더 낮은 수준에 머물 것으로 내다보고 있다',
    'body_text': '미국의 기준금리 인상과 감세정책 등 달러화 강세 요인에도 불구하고 원화 강세를 기대하는 목소리가 있다',
    'df_html': df_2018_01.to_html(),
    'title': "2017년",
    'image_data': fig_data.decode('utf8'),
}

t = Template(html_report_tmpl)
html_report = t.render(render_data)
```

2018년 환율 전망

다수 전문가들은 올해 환율이 더 낮은 수준에 머물 것으로 내다보고 있다

2017년 원달러 환율 추이



지난해 원 · 달러 환율은 줄곧 하락세(원화 강세)를 보였다.

2018년 1월 원달러 환율

DEXKOUS	
DATE	
2018-01-01	NaN
2018-01-02	1059.39
2018-01-03	1063.11
2018-01-04	1061.50
2018-01-05	1060.07
2018-01-08	1066.26
2018-01-09	1068.80
2018-01-10	1067.55

미국의 기준금리 인상과 감세정책 등 달러화 강세 요인에도 불구하고 때문이란 게 연구원의 분석이다.

이메일 자동화 - smtplib, MIMEMultipart



KOKEYDIO

```
gmail_user='alert.messenger.kr@gmail.com'  
gmail_pwd='*****'  
  
def send_gmail(to, subject, html, attach):  
    msg=MIMEMultipart('alternative')  
    msg['From']=gmail_user  
    msg['To'] = to  
    msg['Subject'] = Header(s=subject, charset="utf-8")  
    msg.attach(MIMEText(html, 'html', _charset="utf-8"))  
  
    #첨부파일 발송  
    part=MIMEBase('application','octet-stream')  
    part.set_payload(open(attach, 'rb').read())  
    encoders.encode_base64(part)  
    part.add_header('Content-Disposition', 'attachment; filename="%s"' % os.path.basename(attach))  
    msg.attach(part)  
  
    s = smtplib.SMTP("smtp.gmail.com", 587)  
    s.ehlo()  
    s.starttls()  
    s.ehlo()  
    s.login(gmail_user, gmail_pwd)  
    s.sendmail(gmail_user, to, msg.as_string())  
    s.close()
```

html = "<h2> 첨부 참조하세요</h2> 상세 내용 보내 드립니다"
send_gmail('이승준<plusjune@gmail.com>', '첨부 참조', html, 'myarchive.zip')

메신저 알림 자동화



python-telegram-bot



KOKEY 2018

```
import logging
from telegram.ext import Updater
from telegram.ext import CommandHandler

# Updater 생성
TOKEN = '397331140:*****'
updater = Updater(token=TOKEN)

# Dispatcher 얻기
dispatcher = updater.dispatcher

# 간단한 메시지 핸들러 함수 (단순 에코)
def echo(bot, update):
    bot.send_message(chat_id=update.message.chat_id, text=update.message.text)

# 메시지 핸들러 추가
echo_handler = MessageHandler(Filters.text, echo)
dispatcher.add_handler(echo_handler)

# 실행
updater.start_polling()
```

Robot Framework



<http://robotframework.org/>

- 파이썬 기반
- keyword-driven 테스트 자동화 프레임워크
- 인수 테스팅
- 인수 테스트 기반 개발 (ATDD)

Robot Framework for RPA

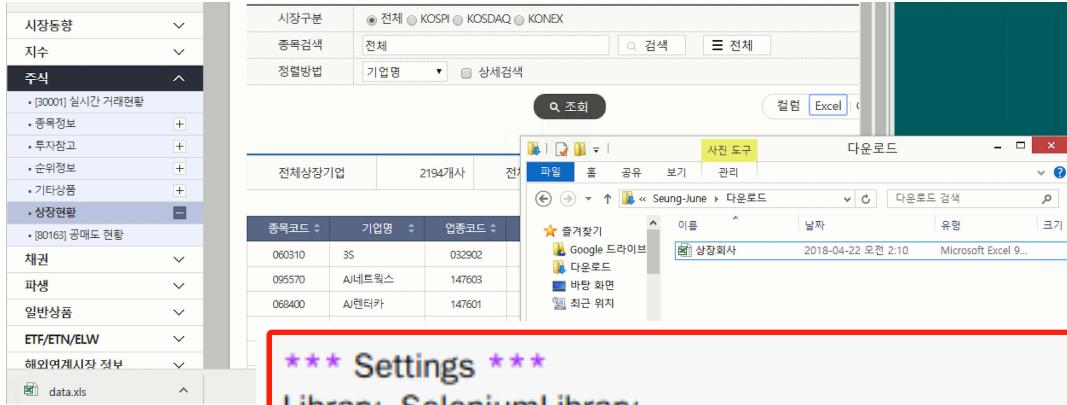


KOKEY TOJO

- SeleniumLibrary (브라우저 자동화)
- ExcelLibrary (엑셀 자동화)
- AutoITLibrary (데스크탑 자동화)
- OperatingSystem (운영체제)
- SSHLibrary (원격 접근, 원격 제어)

STANDARD	EXTERNAL	OTHER
Android library Library for all your Android automation needs. It uses Calabash Android internally.	AnywhereLibrary Library for testing Single-Page Apps (SPA). Uses Selenium Webdriver and Appium internally.	AppiumLibrary Library for Android- and iOS-testing. It uses Appium internally.
Archive library Library for handling zip- and tar-archives.	AutoItLibrary Windows GUI testing library that uses AutoIt freeware tool as a driver.	CncLibrary Library for driving a CNC milling machine.
Database Library (Java) Java-based library for database testing. Usable with Jython. Available also at Maven central .	Database Library (Python) Python based library for database testing. Works with any Python interpreter, including Jython.	Diff Library Library to diff two files together.
Django Library Library for Django , a Python web framework.	Eclipse Library Library for testing Eclipse RCP applications using SWT widgets.	robotframework-faker Library for Faker , a fake test data generator.
FTP library Library for testing and using FTP server with Robot Framework.	HTTP library (livetest) Library for HTTP level testing using livetest tool internally.	HTTP library (Requests) Library for HTTP level testing using Request internally.
HttpRequestLibrary (Java) Library for HTTP level testing using Apache HTTP client. Available also at Maven central .	iOS library Library for all your iOS automation needs. It uses Calabash iOS Server internally.	ImageHorizonLibrary Cross-platform, pure Python library for GUI automation based on image recognition.
JavaFXLibrary Library for testing JavaFX applications, based on TestFX . Has also remote interface support.	MongoDB library Library for interacting with MongoDB using pymongo.	MQTT library Library for testing MQTT brokers and applications.
NcclientLibrary https://github.com/nclient/ncclient	Rambock Generic network protocol test library that offers easy way to specify network packets and inspect the results of sent and received packets.	RemoteSwingLibrary Library for testing and connecting to a java process and using SwingLibrary, especially Java Web Start applications.
RESTinstance Robot Framework test library for HTTP JSON APIs.	SeleniumLibrary Web testing library that uses popular Selenium tool internally.	Selenium2Library Web testing library that uses Selenium 2. Library is deprecated: users should upgrade to SeleniumLibrary described above.

Robot Framework 스크립트



★★★ Settings ★★★

```
Library SeleniumLibrary  
Library OperatingSystem
```

★★★ Test Cases ★★★

TC01

```
Open Browser http://marketdata.krx.co.kr mdi#document=040601 chrome
Click Element xpath://button[text()="Excel"]
Copy File C:/Users/Seung-June/Downloads/data.xls 상장회사.xls
Close Browser
```

자동화를 위한 파이썬 오픈소스들



KOKEYOUNG

영역	파이썬 라이브러리
데이터 크롤링	requests, BeautifulSoup
데이터 전처리	numpy, pandas
브라우저 자동화	Selenium
엑셀 자동화	OpenPyXL, pandas
이메일 자동화	smtplib, EmailMessage

영역	파이썬 라이브러리
보고서 생성 자동화	Jinja2
데스크탑 자동화	pyautogui
자동화 통합 관리	Robot Framework
이미지 캡처, 이미지 프로세싱	OpenCV
OCR (문자 인식, 숫자 인식)	tesseract

작업 스케줄링(주기적 실행): schtasks (윈도우), crontab (리눅스)



로봇 공정 자동화 (RPA) 란 무엇인가

사람이 컴퓨터로 수행하는 정형화되고 반복적인 업무를 소프트웨어로 자동화

BEFORE RPA



AFTER RPA



<https://inatrix.com>

RPA란 무엇인가 또, 무엇이 아닌가?



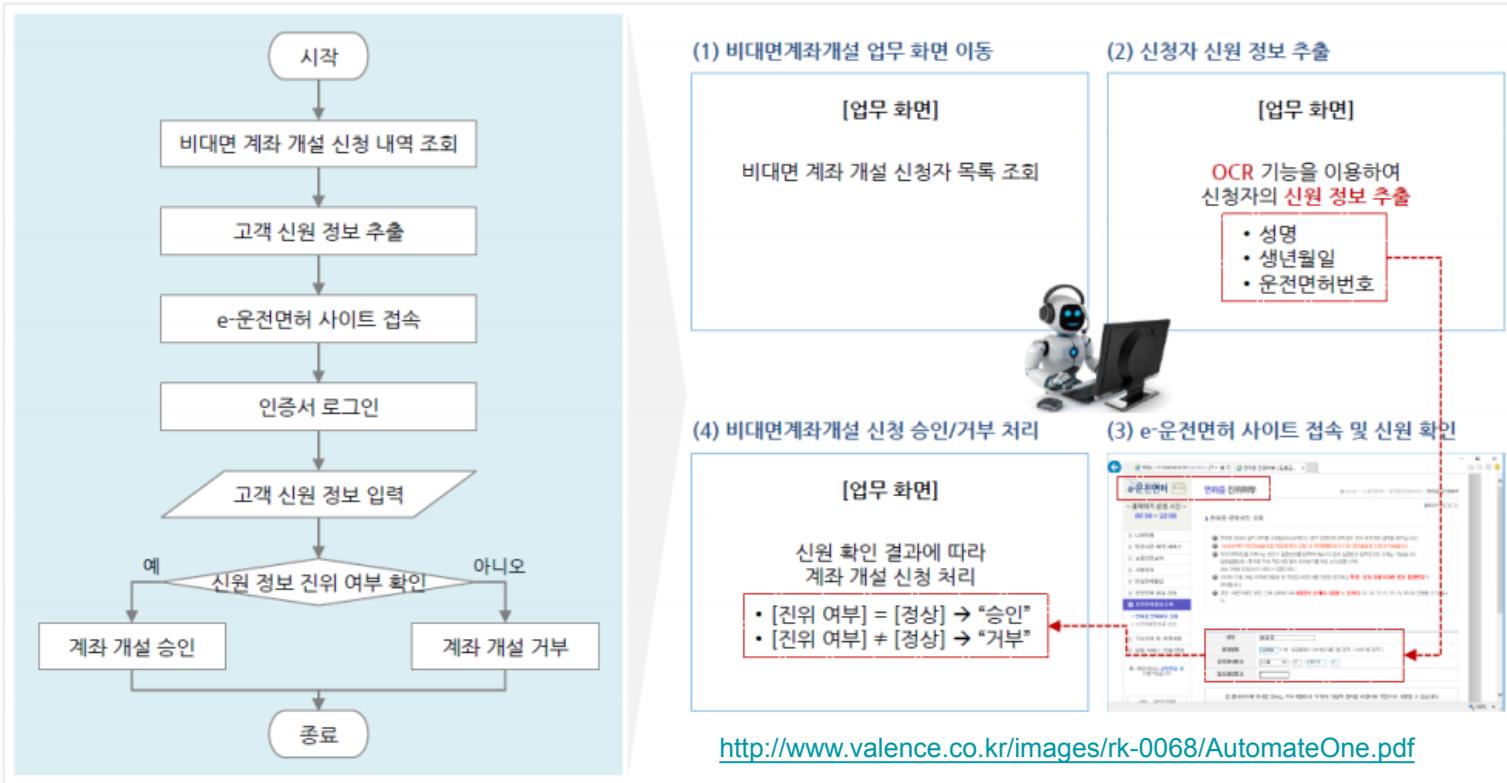
KOKEY 2018

What is	What is NOT
소프트웨어	휴머노이드 로봇
애플리케이션 간의 연계	새로운 IT 시스템 도입
사람이 하던 작업 수행	신규 업무에 적용
비교적 단순 반복 작업	인공지능, 알아서 척척

※ 기존의 시스템과 보안에는 변화를 주지 않음

source: deloitte.com (재정리)

사례 K증권: 비대면 계좌 개설 업무



RPA vs 테스트 자동화 오픈소스

PYCON
KOREA 2018

KOKEY 2018

RPA

- Data Entry, Query
- Transaction
- Business Process



Test Automation

- Functional Testing
- Performance Testing
- Web, UI Testing



RPA vs 파이썬

RPA	파이썬
전사적 자동화	애플리케이션 자동화
상용 솔루션	무료 오픈소스
솔루션 + 개발 + 유지보수 구매	DIY
저작도구 + 전용 스크립트	파이썬 스크립트 코딩

RPA의 고유 기능



KOKEUMDOL

대부분의 자동화 기능은 파이썬 오픈소스로 충분. 하지만, 다음 기능은 RPA에 특화

- 사용자가 하는 동작을 기록하여 재현 (스크립트 생성)
- 자동화 스크립트 관리를 위한 저장소
- 동적인 분기와 반복 워크플로우 저작 기능

“작은 자동화”에 지속적인 관심을 가져야 하는 이유

PYCON
KOREA 2018

KOKEYOUNG

- 거의 모든 산업이 데이터 기반, 데이터 중심으로 이동
- 엑셀의 한계, 개인이 다루어야 하는 데이터 급격 증가
- 변화의 속도: 예측 보다 대응이 중요
- ‘단순 반복 작업을 열심히’ - 거의 도움이 안됨

시스템 구축을
기다릴 수 없다

잘하는 것과 열심히하는 것은 별개

"만들어 주신거 안 돌아가는데 와서 좀 봐주세요"

- 전사적 전략적 업무 시스템의 범위가 아님
- 전사 차원에서 투자대비 효과
- 지속적인 유지보수 비용
- 자동화는 IT부서 만의 고유 영역이 아님!

현업이 코딩을?



KOKEUMDOIO

- 나는 IT가 아닌데? → 스크립트 수준의 코딩이 필요
- 코딩을 해본적이 없는데? → 가장 강력한 코딩 도구 Copy & Paste
- 지금 업무도 많은데 코딩까지? → 단순 반복 업무 소요시간 단축
- 나이도 높은 기술적인 문제는? → 코드 스니펫 공유

LEVEL 01 – 개인

- 엑셀 자동화: OpenPyXL, pandas
- 데이터 크롤링: requests, BeautifulSoup
- 데이터 베이스: sqlite3, hdf5

LEVEL 02 – 개인, 팀/부서 (단위업무)

- 브라우저 자동화: Selenium
- 보고서 생성 자동화: Jinja2
- 이메일 자동화: smtplib, EmailMessage

LEVEL 03 – 팀/부서, 전사 (대외 업무)

- 테스크탑 자동화: pyautogui
- 이미지 캡처, 매칭: OpenCV, 문자 숫자 인식(OCR): tesseract
- 머신러닝, 딥러닝: scikit-learn, Keras (Tensorflow)
- 자동화 통합 관리: Robot Framework (혹은 in-house)

파이썬을 활용한 자동화 도구 3가지



KOKEYDIO

- 1) 현업의 코딩 학습
- 2) 코드 스니펫 공유
- 3) 개발 지원 (팀 단위, 전사 단위)

결론, "기계가 잘하는 일, 사람이 잘하는 일"



- 업무의 흐름과 예외를 가장 잘 아는 사람은 "현업"
- 코딩은 매우 효과적인 수단 (비용의 문제가 아니라)
- "자동화"는 기술과 비용의 문제가 아님! 협업과 공유의 문제
- 가장 빠르게 배워 자동화에 써먹을 수 있는 실용 언어: "파이썬"

人生苦短, 使用 Python

Thanks,

Lee, Sean-June
FinanceData.KR fb.com/financedata