

파이썬으로 배우는 알고리즘 트레이딩

<Chapter 3>

제출일자: 2018년 05월 02일 (수)

학 과: 컴퓨터공학과
지도교수: 김태석 교수님
학 번: 2011722026
성 명: 김효빈

3장 연습문제

3-1 데이터를 받아와 DataFrame 만들기

```
from pandas import Series, DataFrame
from pandas_datareader import data as pdr
import fix_yahoo_finance
import pandas_datareader.data as web
import datetime
import matplotlib.pyplot as plt
import numpy as np

fix_yahoo_finance.pdr_override()

print("딕셔너리로DataFrame 초기화")

raw_data = {'col0' : [1, 2, 3, 4],
            'col1' : [10, 20, 30, 40],
            'col2' : [100, 200, 300, 400]}
data = DataFrame(raw_data)
print("raw_data to DataFrame")
print(data)

print('')

#####
#####
print("딕셔너리로DataFrame 초기화및Index 설정")

index_reset = ['16.02.29', '16.02.26', '16.02.25', '16.02.24', '16.02.23']
daeshin = {'open' : [11650, 11100, 11200, 11100, 11000],
           'high' : [12100, 11800, 11200, 11100, 11150],
           'low' : [11600, 11050, 10900, 10950, 10900],
           'close' : [11900, 11600, 11000, 11100, 11050]}

daeshin_day = DataFrame(daeshin, columns=['open', 'high', 'low', 'close'],
                        index=index_reset)
print(daeshin_day)

print('')

#####
#####
print("pandas yahoo에서078930.KS 2017년~ 2018년4월20일까지의정보를받아옵니다.\n")
data = pdr.get_data_yahoo("078930.KS", start="2017-01-01", end="2018-04-20")
#data = pdr.get_data_yahoo(["SPY", "IWM"], start="2017-01-01",
#end="2017-04-30")
print("----data----")
print(data)
print("----data.index----")
print(data.index)
plt.plot(data.index, data)
plt.show()

print("수정종가에대해5일주가이동평균을구합니다.")
ma5 = data.rolling(window=5).mean() #수정종가에대해5일주가이동평균을구하기

print(type(ma5))
print(ma5)

#공휴일주가를제외한데이터를저장하기
print("공휴일주가를제외한데이터를저장합니다.")
new_data = data[data['Volume'] != 0]

print("\ndata 변수와new_data 변수를비교\n")
print(data.tail(5))
print(new_data.tail(5))

plt.plot(new_data.index, new_data, label="Abj Close")
plt.show()
```

```
chapter3-1
C:\Python_anaconda\python.exe C:/Python/Project_Practice_chapter3/chapter3-1.py
딕셔너리로 DataFrame 초기화
raw_data to DataFrame
  col0  col1  col2
0     1     10   100
1     2     20   200
2     3     30   300
3     4     40   400

딕셔너리로 DataFrame 초기화 및 Index 설정
      open  high  low  close
16.02.29 11650 12100 11600 11900
16.02.26 11100 11800 11050 11600
16.02.25 11200 11200 10900 11000
16.02.24 11100 11100 10950 11100
16.02.23 11000 11150 10900 11050
```

[*****100%*****] 1 of 1 downloaded

Date	Open	High	Low	Close	Adj. Close	#
2017-01-03	225.039993	225.830002	223.880005	225.240005	220.063202	
2017-01-04	225.619995	226.750000	225.610001	226.580002	221.372406	
2017-01-05	226.270004	226.580002	225.479996	226.399994	221.196548	
2017-01-06	226.529999	227.750000	225.899994	227.210007	221.987915	
2017-01-09	226.910004	227.070007	226.419998	226.460007	221.255173	
2017-01-10	226.479996	227.449997	226.009995	226.460007	221.255173	
2017-01-11	226.360001	227.100006	225.589996	227.100006	221.880432	
2017-01-12	226.500000	226.750000	224.960007	226.529999	221.323547	
2017-01-13	226.729996	227.399994	226.690002	227.050003	221.631589	
2017-01-17	226.309998	226.779999	225.800003	226.250000	221.049988	
2017-01-18	226.539993	226.800003	225.899994	226.750000	221.538483	
2017-01-19	226.839996	227.000000	225.410004	225.910004	220.717804	
2017-01-20	226.699997	227.309998	225.970001	226.740005	221.528717	
2017-01-23	226.740005	226.809998	225.270004	226.149994	220.952255	
2017-01-24	226.399994	228.080002	226.270004	227.600006	222.368958	
2017-01-25	226.699997	229.570007	228.509995	229.570007	224.293686	
2017-01-26	229.399994	229.710007	229.009995	229.330002	224.058204	
2017-01-27	229.419998	229.589996	228.759995	228.970001	223.707458	
2017-01-30	228.159998	228.199997	226.410004	227.550003	222.320099	
2017-01-31	226.979996	227.600006	226.320007	227.529999	222.300568	
2017-02-01	227.529999	228.589996	226.940002	227.619995	222.388489	
2017-02-02	227.619995	228.100006	226.820007	227.770004	222.535049	
2017-02-03	228.820007	229.550003	228.460007	229.339996	224.068954	
2017-02-06	228.869995	229.330002	228.539993	228.929993	223.668365	
2017-02-07	229.380005	229.660004	228.720001	228.940002	223.678146	
2017-02-08	228.940002	229.389999	228.309998	229.240005	223.971252	
2017-02-09	229.240005	230.949997	229.240005	230.600006	225.300003	

Date	Volume
2017-01-03	91366500
2017-01-04	78744400
2017-01-05	78379000
2017-01-06	71559900
2017-01-09	46265300
2017-01-10	63771900
2017-01-11	74650000
2017-01-12	72113200
2017-01-13	62717900
2017-01-17	59261200
2017-01-18	54793300
2017-01-19	66608800
2017-01-20	129168600
2017-01-23	75061600
2017-01-24	95555300
2017-01-25	84437700
2017-01-26	59970700
2017-01-27	59711100
2017-01-30	79737300
2017-01-31	75880800
2017-02-01	79117700
2017-02-02	69657600
2017-02-03	80563200
2017-02-06	57790100
2017-02-07	57831200
2017-02-08	51566200
2017-02-09	65955200
2017-02-10	66015900
2017-02-13	55182100
2017-02-14	67794500

chapter3-1		
↑	2017-03-17	89002100
↓	2017-03-20	52537000
↕	2017-03-21	131809300
📄	2017-03-22	97569200
📄	2017-03-23	100410300
📄	2017-03-24	112504900
🗑️	2017-03-27	87454500
	2017-03-28	93483900
	2017-03-29	61950400
	2017-03-30	56737900
	2017-03-31	73733100
	2017-04-03	85546500
	2017-04-04	56466200
	2017-04-05	108800600
	2017-04-06	69135800
	2017-04-07	74412300
	2017-04-10	67615300
	2017-04-11	88045300
	2017-04-12	81864400
	2017-04-13	92880400
	2017-04-17	68405400
	2017-04-18	83225800
	2017-04-19	68699900
	2017-04-20	92572200
	2017-04-21	110389800
	2017-04-24	119209900
	2017-04-25	76698300
	2017-04-26	84702500
	2017-04-27	57410300
	2017-04-28	53247400
[81 rows x 6 columns]		

3장에서 대부분 pandas_datareader 모듈을 이용하면 된다고 되어있지만, 최근에는 yahoo에서 주식데이터를 받아오는게 막히게 되어 사용할 수 없게되었다. 그러므로

```
from pandas_datareader import data as pdr
import fix_yahoo_finance
```

fix_yahoo_finance.pdr_override() <== 필수적으로 해야함.

위 3줄의 코드를 추가하여 진행하여야 했다.

날짜	종가	전일비	시가	고가	저가	거래량
2018-04-30	61,300	▲ 900	60,600	61,300	60,400	109,778
2018-04-27	60,400	▼ 100	60,200	60,900	60,100	100,542
2018-04-26	60,500	▲ 300	60,200	60,900	59,900	140,748
2018-04-25	60,200	▼ 400	60,600	60,800	59,600	198,205
2018-04-24	60,600	▼ 300	61,000	61,200	60,200	144,045
2018-04-23	60,900	▲ 200	60,200	61,200	60,200	155,595
2018-04-20	60,700	▼ 700	60,900	61,400	60,200	238,034
2018-04-19	61,400	0	61,200	61,700	60,600	122,691
2018-04-18	61,400	▲ 1,600	59,800	59,800	59,600	254,268
2018-04-17	59,800	▼ 100	60,000	60,200	59,400	208,447

finance.naver.com의 일별 주식 데이터는

'http://finance.naver.com/item/sise_day.nhn?code=' + codenum +
'&page=' + str(page)

으로 찾을 수 있습니다.

codenum은 종목번호, str(page)는 아래 1 2 3 4 5 6 7 8 9 10 다음 > 맨위 >>에서의 페이지 번호입니다.

3-2 차트 그리기

```
import pandas as pd
from pandas_datareader import data as pdr
from pandas import DataFrame
import fix_yahoo_finance
import pandas_datareader.data as web
import matplotlib.pyplot as plt

fix_yahoo_finance.pdr_override()

# Get GS data from yahoo finance
print("Get GS data from yahoo finance !")
gs = pdr.get_data_yahoo("078930.KS", start="2017-01-01", end="2018-04-20")

print(gs)

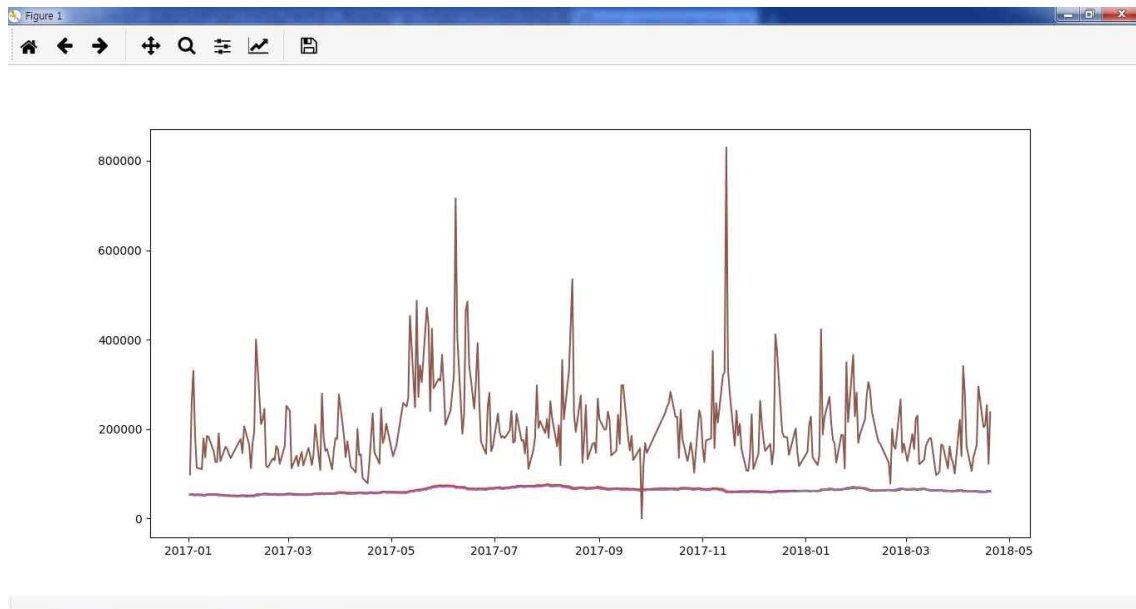
new_gs = gs[gs['Volume'] != 0]

# Moving average
ma5 = new_gs['Adj Close'].rolling(window=5).mean()
ma20 = new_gs['Adj Close'].rolling(window=20).mean()
ma60 = new_gs['Adj Close'].rolling(window=60).mean()
ma120 = new_gs['Adj Close'].rolling(window=120).mean()

# insert columns
new_gs.insert(len(new_gs.columns), "MA5", ma5)
new_gs.insert(len(new_gs.columns), "MA20", ma20)
new_gs.insert(len(new_gs.columns), "MA60", ma60)
new_gs.insert(len(new_gs.columns), "MA120", ma120)

# plot
plt.plot(new_gs.index, new_gs['Adj Close'], label='Adj Close')
plt.plot(new_gs.index, new_gs['MA5'], label='MA5')
plt.plot(new_gs.index, new_gs['MA20'], label='MA20')
plt.plot(new_gs.index, new_gs['MA60'], label='MA60')
plt.plot(new_gs.index, new_gs['MA120'], label='MA120')

plt.legend(loc="best")
plt.grid()
plt.show()
```



앞서서 받아온 GS 종목의 데이터 수는 너무 적어 그래프를 그리기에 적합하지는 않습니다. 하지만 파이썬에서 더 많은 데이터를 가져오는 것은 전혀 어렵지 않습니다. Pandas.datareader import data 모듈을 이용해 더 많은 데이터를 받아오고 이를 그래프로 그립니다.

특히, 그래프는 matplotlib 패키지의 pyplot이라는 모듈을 사용합니다.

```
<class 'pandas.core.frame.DataFrame'>
```

Date	Open	High	Low	Close	Adj Close	Volume
2018-04-09	61800.0	62420.0	60780.0	61300.0	61300.0	205683.8
2018-04-10	61300.0	61880.0	60320.0	60840.0	60840.0	205073.0
2018-04-11	60900.0	61580.0	60420.0	60940.0	60940.0	166826.0
2018-04-12	60980.0	61600.0	60500.0	61040.0	61040.0	143904.2
2018-04-13	61040.0	61460.0	60420.0	60880.0	60880.0	170605.6
2018-04-16	60940.0	61360.0	60200.0	60720.0	60720.0	190150.4
2018-04-17	60780.0	61100.0	59980.0	60460.0	60460.0	204519.4
2018-04-18	60540.0	61060.0	59760.0	60500.0	60500.0	225382.8
2018-04-19	60500.0	61080.0	59740.0	60500.0	60500.0	217051.2
2018-04-20	60460.0	61100.0	59820.0	60640.0	60640.0	205577.2

```
<class 'pandas.core.frame.DataFrame'>
```

Date	Open	High	Low	Close	Adj Close	Volume
2017-01-02	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-03	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-04	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-05	NaN	NaN	NaN	NaN	NaN	NaN
2017-01-06	53760.0	54160.0	53040.0	53900.0	52345.193750	197648.8
2017-01-09	53680.0	54040.0	53000.0	53680.0	52131.539844	199971.0
2017-01-10	53400.0	53680.0	52800.0	53280.0	51743.078125	184109.2
2017-01-11	52960.0	53540.0	52540.0	53180.0	51645.962500	145361.6
2017-01-12	52940.0	53580.0	52640.0	53260.0	51723.654687	145119.4
2017-01-13	53040.0	53640.0	52740.0	53340.0	51801.346875	158873.4
2017-01-16	53140.0	53780.0	52880.0	53520.0	51976.153906	166937.8
2017-01-17	53480.0	54160.0	53240.0	53900.0	52345.192969	156476.8
2017-01-18	54000.0	54360.0	53540.0	54040.0	52481.154688	154353.8

이번 예제에서는 이동평균선을 구합니다. 주가이동평균선을 구하는데 사용되는 값은 수정 종가(Adjusted Closing Price)입니다. 만약 5일 주가이동평균을 구한다면 주가이동평균 계산일을 기준으로 최근 5일의 수정 종가를 모두 합산한 후 그 값을 5로 나누어 계산합니다. 예시를 들어보겠습니다.

2월 19일 50,600 #20일, 21일 주말은 제외됩니다.
 2월 22일 50,400
 2월 23일 52,800
 2월 24일 53,500
 2월 25일 53,900 이라고 한다면,

2월 25일의 5일 주가이동평균은
 $(50,600 + 50,400 + 52,800 + 53,500 + 53,900) / 5 = 52,240$ 입니다.

Chapter 15.1 봉 차트 그리기

```
from pandas_datareader import data as pdr
import fix_yahoo_finance
import matplotlib.pyplot as plt
import matplotlib.finance as matfin

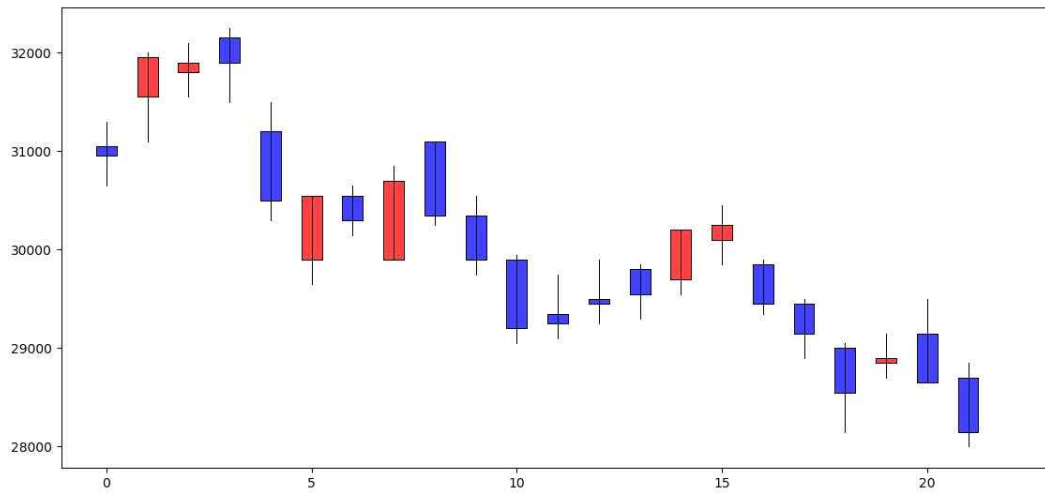
fix_yahoo_finance.pdr_override()

skhynix = pdr.get_data_yahoo("000660.KS", start="2016-03-01",
                             end="2016-03-31")
skhynix = skhynix[skhynix['Volume'] > 0]

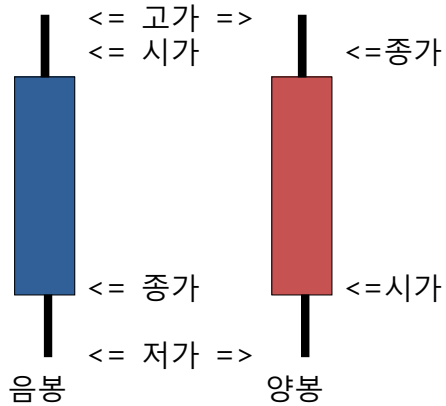
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111)

matfin.candlestick2_ohlc(ax, skhynix['Open'], skhynix['High'],
                          skhynix['Low'], skhynix['Close'], width=0.5, colorup='r', colordown='b')

plt.show()
```



matplotlib를 이용해 봉 차트를 그리는 예제를 구성해보았습니다. 먼저 위 그래프를 이해하려면 봉 차트(캔들 스틱 차트)를 이해해야 합니다.



증권사마다 다르지만 양봉은 보통 빨간색으로 표시하고 음봉은 파란색으로 표시합니다. 일봉 중 종가가 시가보다 높은 경우를 양봉이라고 하며, 반대로 종가가 시가보다 낮은 경우를 음봉이라고 합니다.

Chapter3 15.3 candlestick 수정본

```
from pandas_datareader import data as pdr
import fix_yahoo_finance
import matplotlib.pyplot as plt
import matplotlib.finance as matfin
import matplotlib.ticker as ticker

fix_yahoo_finance.pdr_override()

skhynix = pdr.get_data_yahoo("000660.KS", start="2016-03-01", end="2016-03-31")

fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111)

day_list = range(len(skhynix))
name_list = []

for day in skhynix.index :
    name_list.append(day.strftime('%d'))

ax.xaxis.set_major_locator(ticker.FixedLocator(day_list))
ax.xaxis.set_major_formatter(ticker.FixedFormatter(name_list))

matfin.candlestick2_ohlc(ax, skhynix['Open'], skhynix['High'], skhynix['Low'],
skhynix['Close'], width=0.5, colorup='r', colordown='b')

plt.show()
```



봉 차트를 조금 수정하여 x축에 날짜 정보를 출력하게 합니다.

Chapter3 15.4 bar 차트 그리기

```
from pandas_datareader import data as pdr
import fix_yahoo_finance
import matplotlib.pyplot as plt
import matplotlib.finance as matfin
import matplotlib.ticker as ticker

fix_yahoo_finance.pdr_override()

skhynix = pdr.get_data_yahoo("000660.KS", start="2016-03-01",
end="2016-03-31")

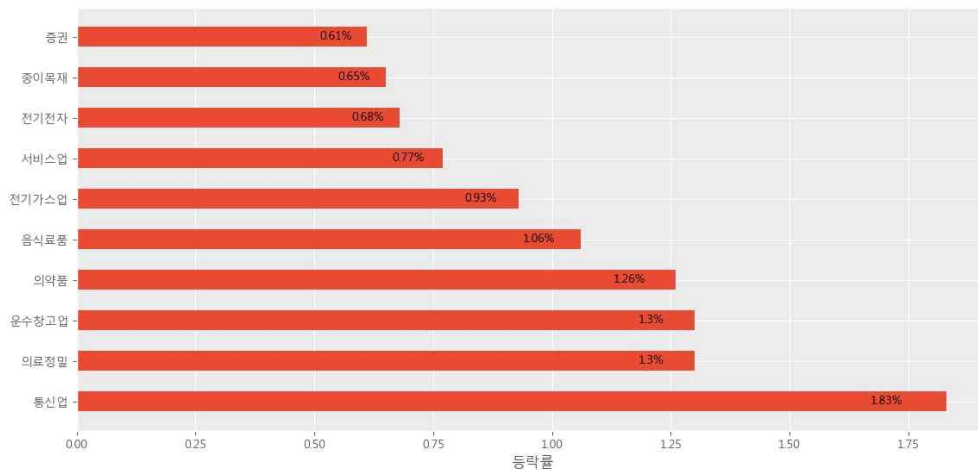
fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111)

day_list = []
name_list = []

for i, day in enumerate(skhynix.index) :
    if day.dayofweek == 0:
        day_list.append(i)
        name_list.append(day.strftime('%Y-%m-%d') + '(Mon)')

ax.xaxis.set_major_locator(ticker.FixedLocator(day_list))
ax.xaxis.set_major_formatter(ticker.FixedFormatter(name_list))

matfin.candlestick2_ohlc(ax, skhynix['Open'], skhynix['High'],
skhynix['Low'], skhynix['Close'], width=0.5, colorup='r', colordown='b')
plt.grid()
plt.show()
```



수평 방향의 bar차트는 matplotlib.pyplot 모듈의 barh 함수를 사용해 그릴 수 있습니다. barh 함수의 첫 번째 인자는 각 bar가 그려질 위치이고, 두 번째 인자는 각 bar에 대한 수치입니다. 이 값들은 파이썬 리스트 형태로 전달하면 됩니다.

ex) fig = plt.figure(figsize=(12,8))

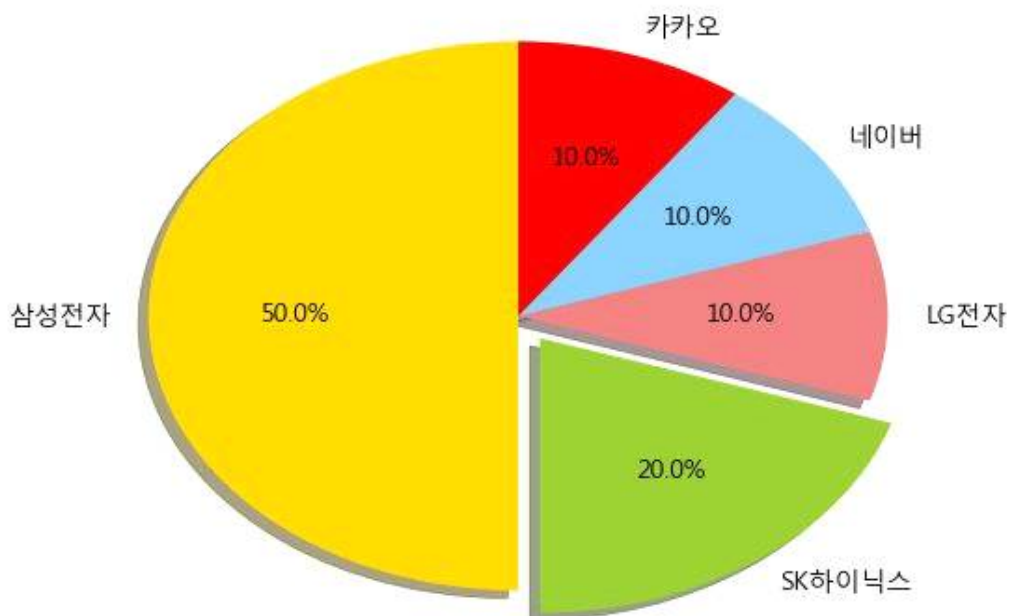
Chapter3 15.6 pie 차트

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import font_manager, rc
from matplotlib import style

font_name =
font_manager.FontProperties(fname="c:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)
style.use('ggplot')

colors = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'red']
labels = ['삼성전자', 'SK하이닉스', 'LG전자', '네이버', '카카오']
ratio = [50, 20, 10, 10, 10]
explode = (0.0, 0.1, 0.0, 0.0, 0.0)

plt.pie(ratio, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)
plt.show()
```



pie 차트 또한 matplotlib.pyplot 모듈의 pie 함수를 사용해 그릴 수 있습니다. pie 함수에 각 범주가 데이터에서 차지하는 비율을 전달하고, shadow=True로 설정해 그림자를 넣을 수 있습니다.

Chapter3 16.1 QMainWindow를 이용한 윈도우 구성

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setWindowTitle("Review")
        btn1 = QPushButton("Click me", self)
        btn1.move(20, 20)
        btn1.clicked.connect(self.btn1_clicked)

    def btn1_clicked(self):
        QMessageBox.about(self, "message", "clicked")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()
```



PyQt에서 위젯은 UI를 구성하는 핵심 요소입니다. 대부분 프로그램에서 QMainWindow나 QDialog 클래스를 사용해 윈도우를 생성합니다. clicked 이벤트는 사용자가 버튼을 클릭할 시 전달한 함수가 호출되도록 연결합니다.

Chapter3 16.4 QPushButton 사용 예

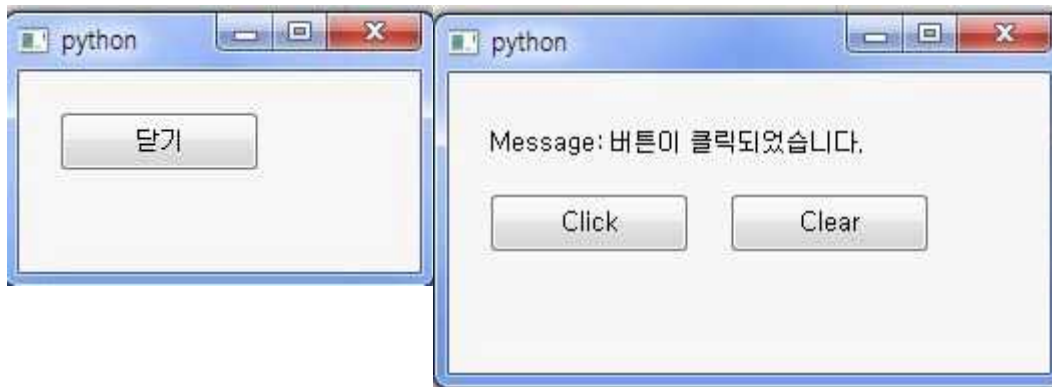
```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *

app = None

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        btn1 = QPushButton("닫기", self)
        btn1.move(20, 20)
        btn1.clicked.connect(app.quit)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()
```



QPushButton을 이용한 2가지 간단한 UI를 구성해보았습니다. 첫 번째는 보이는데로 '닫기'를 눌렀을 때 종료되는 버튼이고 2번째는 Click을 눌렀을 때 Message를 출력하는 기능을 가지고 있습니다. Clear는 말 그대로 Message를 지우는 기능을 합니다.

Chapter3 16.6 QLabel을 이용한 텍스트 출력

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 400, 300, 150)

        textLabel = QLabel("Message: ", self)
        textLabel.move(20, 20)

        self.label = QLabel("", self)
        self.label.move(80, 20)
        self.label.resize(150, 30)

        btn1 = QPushButton("Click", self)
        btn1.move(20, 60)
        btn1.clicked.connect(self.btn1_clicked)

        btn2 = QPushButton("Clear", self)
        btn2.move(140, 60)
        btn2.clicked.connect(self.btn2_clicked)

    def btn1_clicked(self):
        self.label.setText("버튼이 클릭되었습니다.")

    def btn2_clicked(self):
        self.label.clear()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()
```

Chapter3 16.7 QLine 사용 예제

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 400, 300, 150)

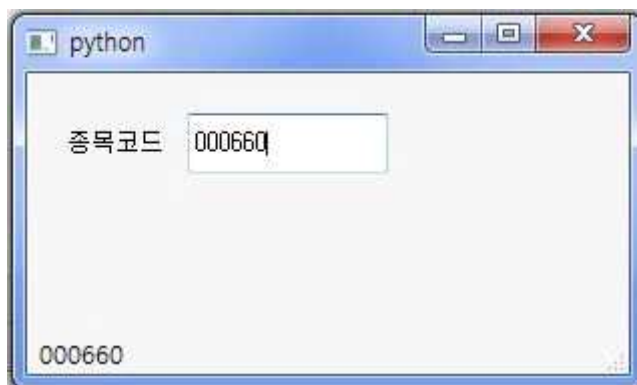
        #Label
        label = QLabel("종목코드", self)
        label.move(20, 20)

        #LineEdit
        self.lineEdit = QLineEdit("", self)
        self.lineEdit.move(80, 20)
        self.lineEdit.textChanged.connect(self.lineEditChaned)

        #StatusBar
        self.statusBar = QStatusBar(self)
        self.setStatusBar(self.statusBar)

    def lineEditChaned(self):
        self.statusBar.showMessage(self.lineEdit.text())

if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()
```



QLineEdit은 한 줄의 텍스트를 입력할 수 있는 위젯으로, 사용자로부터 간단한 텍스트를 입력받을 때 사용합니다. 위 예제는 사용자로부터 주식 종목 코드를 입력받는 프로그램입니다.

Chapter3 16-8 QRadioButton, QGroupBox

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic
from PyQt5.QtCore import *

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)

        groupBox = QGroupBox("시간단위", self)
        groupBox.move(10, 10)
        groupBox.resize(280, 80)

        self.radio1 = QRadioButton("일봉", self)
        self.radio1.move(20, 20)
        self.radio1.setChecked(True)
        self.radio1.clicked.connect(self.radioButtonClicked)

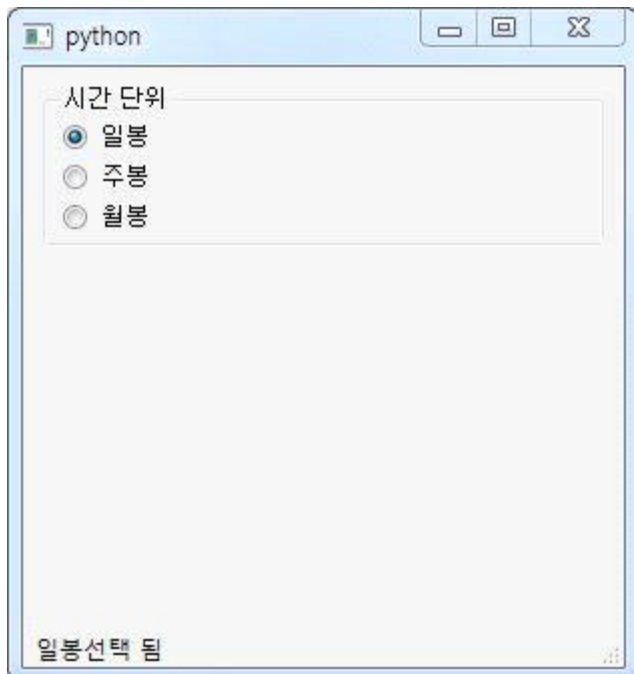
        self.radio2 = QRadioButton("주봉", self)
        self.radio2.move(20, 40)
        self.radio2.clicked.connect(self.radioButtonClicked)

        self.radio3 = QRadioButton("월봉", self)
        self.radio3.move(20, 60)
        self.radio3.clicked.connect(self.radioButtonClicked)

        self.statusBar = QStatusBar(self)
        self.setStatusBar(self.statusBar)

    def radioButtonClicked(self):
        msg = ""
        if self.radio1.isChecked():
            msg = "일봉"
        elif self.radio2.isChecked():
            msg = "주봉"
        else:
            msg = "월봉"
        self.statusBar.showMessage(msg + "선택됨")

if __name__ == "__main__":
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()
```



QRadioButton은 사용자로부터 여러 가지 옵션 중 하나를 입력받을 때 주로 사용하며, QGroupBox는 제목이 있는 네모 박스 형태의 경계선을 만드는 데 사용합니다.

Chapter3 16.9 QCheckBox

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)

        self.checkBox1 = QCheckBox("5일이동평균선", self)
        self.checkBox1.move(10, 20)
        self.checkBox1.stateChanged.connect(self.checkBoxState)

        self.checkBox2 = QCheckBox("20일이동평균선", self)
        self.checkBox2.move(10, 50)
        self.checkBox2.resize(150, 30)
        self.checkBox2.stateChanged.connect(self.checkBoxState)

        self.checkBox3 = QCheckBox("60일이동평균선", self)
        self.checkBox3.move(10, 80)
        self.checkBox3.resize(150, 30)
        self.checkBox3.stateChanged.connect(self.checkBoxState)

        self.statusBar = QStatusBar(self)
        self.setStatusBar(self.statusBar)

    def checkBoxState(self):
        msg = ""
        if self.checkBox1.isChecked() == True:
            msg += "5일"
```

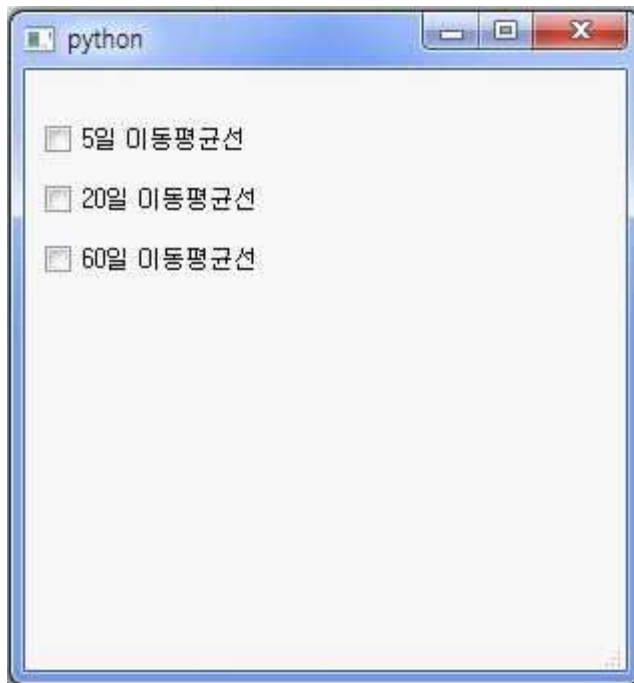


```

if self.checkBox2.isChecked() == True:
    msg += "20일"
if self.checkBox3.isChecked() == True:
    msg += "60일"
self.statusBar.showMessage(msg)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()

```



QRadioButton과 유사한 사용자로부터 옵션을 선택받을 때 사용하는 QCheckBox 위젯입니다. QCheckBox는 여러 옵션 중 하나만 선택할 수 있었던 QRadioBox와는 달리 여러 옵션을 동시에 선택받을 수도 있습니다.

Chapter3 16.10 QSpinBox

```

import sys
from PyQt5.QtWidgets import *

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)

        label = QLabel("매도수량: ", self)
        label.move(10, 20)

        self.spinBox = QSpinBox(self)
        self.spinBox.move(100, 25)

```

```

self.spinBox.resize(80, 22)
self.spinBox.valueChanged.connect(self.spinBoxChanged)

self.statusBar = QStatusBar(self)
self.setStatusBar(self.statusBar)

def spinBoxChanged(self):
    val = self.spinBox.value()
    msg = '%d 주를매도합니다.' % (val)
    self.statusBar.showMessage(msg)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()

```



QSpinBox 위젯을 보면 알다시피, 값을 증가시키는 화살표, 감소시키는 화살표, 값이 출력되는 부분으로 구성됨을 알 수 있습니다. 사용자가 직접값을 입력할 때보다 조금이나마 더 편하게 사용할 수 있습니다.

Chapter3 16.11 ~ 12 QWidget

```

import sys
from PyQt5.QtWidgets import *

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)

        self.tableWidget = QTableWidget(self)
        self.tableWidget.resize(290, 290)
        self.tableWidget.setRowCount(2)

```

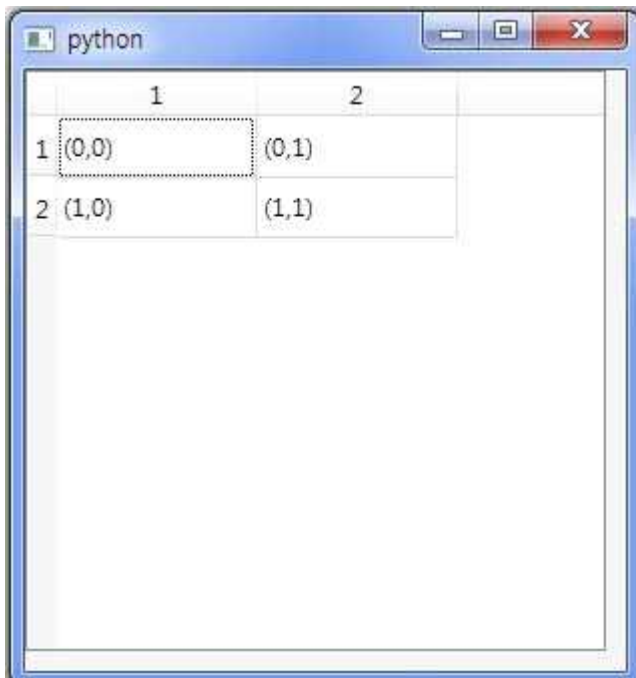
```

self.tableWidget.setColumnCount(2)
self.setTableWidgetData()

def setTableWidgetData(self):
    self.tableWidget.setItem(0, 0, QTableWidgetItem("(0,0)"))
    self.tableWidget.setItem(0, 1, QTableWidgetItem("(0,1)"))
    self.tableWidget.setItem(1, 0, QTableWidgetItem("(1,0)"))
    self.tableWidget.setItem(1, 1, QTableWidgetItem("(1,1)"))

if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()

```



```

import sys
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *

kospi_top5 = {
    'code': ['005930', '015760', '005380', '090430', '012330'],
    'name': ['삼성전자', '한국전력', '현대차', '아모레퍼시픽', '현대모비스'],
    'cprice': ['1,269,000', '60,100', '132,000', '414,500', '243,500']
}

column_idx_lookup = {'code' : 0, 'name' : 1, 'cprice' : 2}

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)

        self.tableWidget = QTableWidgetItem(self)
        self.tableWidget.resize(290, 290)
        self.tableWidget.setRowCount(5)

```

```

self.tableWidget.setColumnCount(3)
self.tableWidget.setEditTriggers(QAbstractItemView.NoEditTriggers)

self.setTableWidgetData()

def setTableWidgetData(self):
    column_headers = ['종목코드', '종목명', '종가']
    self.tableWidget.setHorizontalHeaderLabels(column_headers)

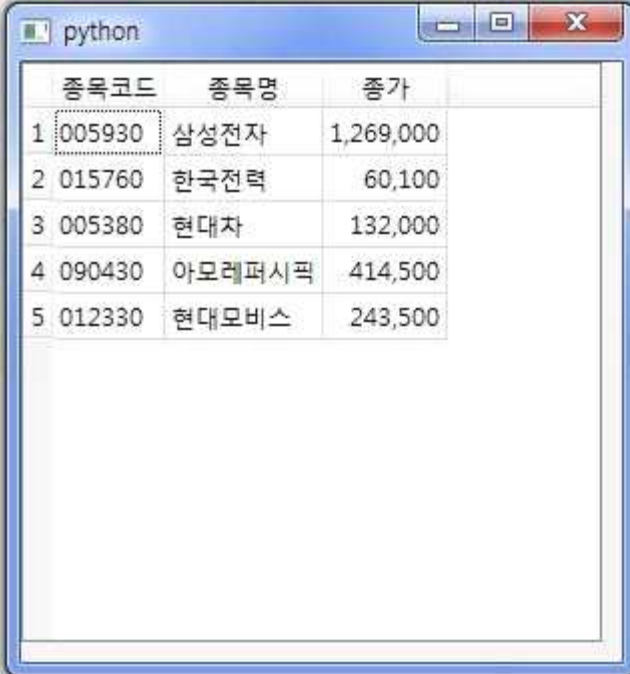
    for k, v in kospi_top5.items():
        col = column_idx_lookup[k]
        for row, val in enumerate(v):
            item = QTableWidgetItem(val)
            if col == 2:
                item.setTextAlignment(Qt.AlignVCenter | Qt.AlignRight)  # 우측정렬

    self.tableWidget.setItem(row, col, item)

self.tableWidget.resizeColumnsToContents()
self.tableWidget.resizeRowsToContents()

if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()

```



	종목코드	종목명	종가
1	005930	삼성전자	1,269,000
2	015760	한국전력	60,100
3	005380	현대차	132,000
4	090430	아모레퍼시픽	414,500
5	012330	현대모비스	243,500

QTableWidget은 2차원 포맷 형태를 표현하고자 할 때 많이 사용되는 함수입니다. 12번 예제는 setEditTriggers 메서드를 사용해 사용자가 QTableWidget의 아이템 항목을 수정할 수 없도록 설정하였고 row 방향에 대한 라벨을 설정할 때는 setVerticalHeaderLabels 메서드를 사용합니다.

Chapter3 16.13 위치 및 크기 조절을 통한 레이아웃 설정

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)

        self.textEdit = QTextEdit(self)
        self.textEdit.resize(280, 250)
        self.textEdit.move(10, 10)

        self.pushButton = QPushButton('저장', self)
        self.pushButton.resize(280, 25)
        self.pushButton.move(10, 270)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()
```



GUI 프로그래밍에서 위젯을 배치하는 것을 레이아웃이라고 한다.(Layout) 위 예제는 QTextEdit와 QPushButton 위젯의 위치와 크기를 직접 설정한다. 전에는 MyWindow 클래스를 정의할 때 QMainWindow를 사용하였지만, 이번에는 QWidget 클래스를 상속하였습니다.

Chapter3 16.14 QVBoxLayout

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)

        self.textEdit = QTextEdit()
        self.pushButton = QPushButton('저장')

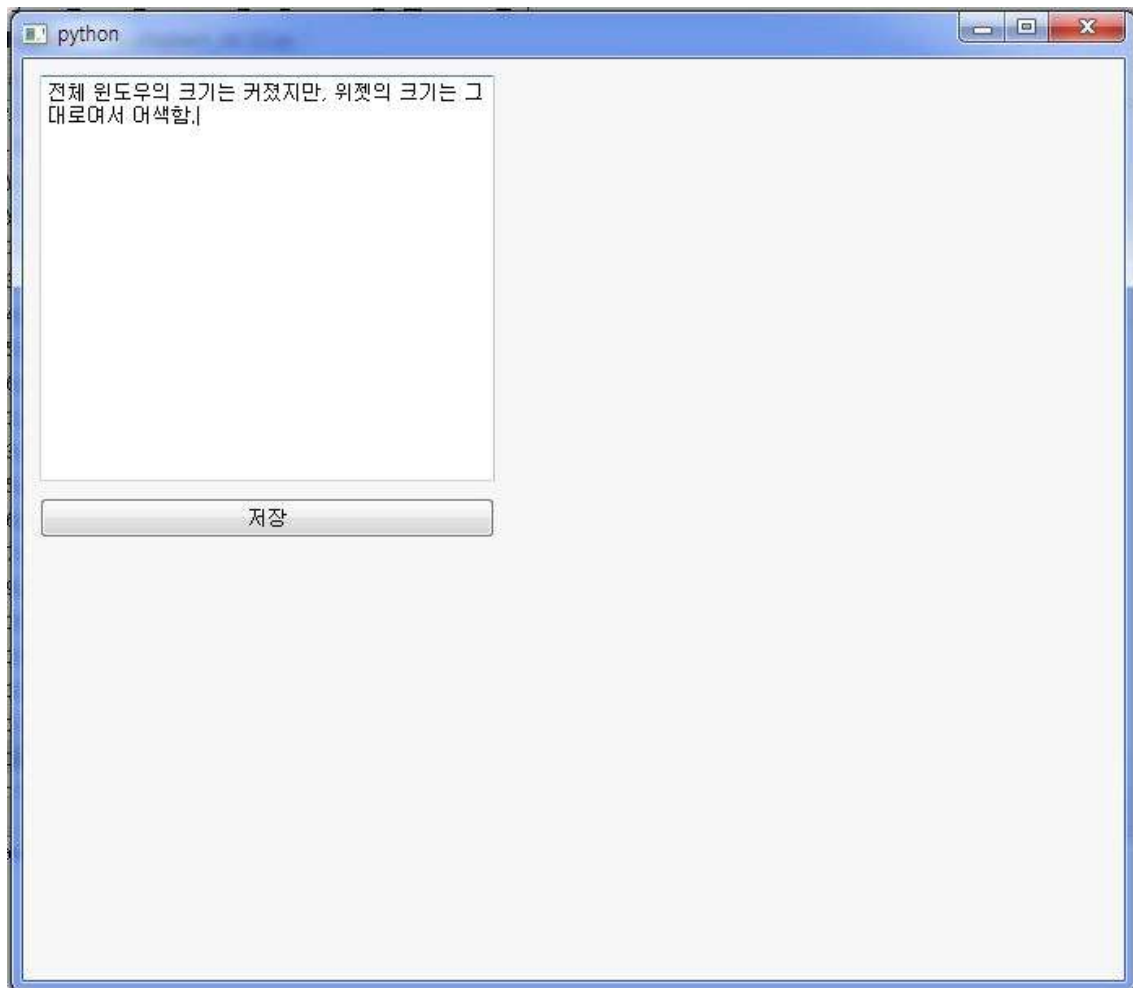
        layout = QVBoxLayout()
        layout.addWidget(self.textEdit)
        layout.addWidget(self.pushButton)

        self.setLayout(layout)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()
```



QVBoxLayout은 위젯의 크기와 출력 위치를 명시적으로 설정하는 방법에 한계가 있기 때문에 사용됩니다. 만약 이전에 사용하였던 방법 그대로 사용한다면,



위 결과이미지와 같은 문제가 발생합니다. 윈도우의 크기를 늘렸을 때, 위젯의 크기는 그대로 남아서 어색해보이게 됩니다. PyQt에서는 이러한 문제를 해결하고자 레이아웃 매니저를 제공하는데 QVBoxLayout은 그 중의 하나입니다.

Chapter3 16.15 QVBoxLayout

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 100)

        self.pushButton1 = QPushButton("Button1")
        self.pushButton2 = QPushButton("Button2")
        self.pushButton3 = QPushButton("Button3")

        layout = QVBoxLayout()
        layout.addWidget(self.pushButton1)
        layout.addWidget(self.pushButton2)
        layout.addWidget(self.pushButton3)
```

```
self.setLayout(layout)
```

```
if __name__ == '__main__':  
    app = QApplication(sys.argv)  
    mywindow = MyWindow()  
    mywindow.show()  
    app.exec_()
```



Chapter3 16.16 QGridLayout

```
import sys  
from PyQt5.QtWidgets import *  
  
class MyWindow(QWidget):  
    def __init__(self):  
        super().__init__()  
        self.setupUI()  
  
    def setupUI(self):  
        self.setGeometry(800, 200, 300, 100)  
  
        self.label1 = QLabel("ID : ")  
        self.label2 = QLabel("Password : ")  
        self.lineEdit1 = QLineEdit()  
        self.lineEdit2 = QLineEdit()  
        self.pushButton1 = QPushButton("Sign In")  
  
        layout = QGridLayout()  
  
        layout.addWidget(self.label1, 0, 0)  
        layout.addWidget(self.lineEdit1, 0, 1)  
        layout.addWidget(self.pushButton1, 0, 2)  
  
        layout.addWidget(self.label2, 1, 0)  
        layout.addWidget(self.lineEdit2, 1, 1)  
  
        self.setLayout(layout)  
  
if __name__ == '__main__':  
    app = QApplication(sys.argv)  
    mywindow = MyWindow()  
    mywindow.show()  
    app.exec_()
```




QGridLayout 클래스는 격자 형태의 UI를 구성하는데 사용합니다. 위 예제에서는 2x3 크기의 격자 구조에 위젯을 배치했습니다.

Chapter3 16.17 레이아웃 중첩을 통한 UI 구성

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 500, 300)

        groupBox = QGroupBox("검색옵션")
        checkBox1 = QCheckBox("상한가")
        checkBox2 = QCheckBox("하한가")
        checkBox3 = QCheckBox("시가총액상위")
        checkBox4 = QCheckBox("시가총액하위")
        checkBox5 = QCheckBox("회전율상위")
        checkBox6 = QCheckBox("대량거래상위")
        checkBox7 = QCheckBox("환산주가상위")
        checkBox8 = QCheckBox("외국인한도소진상위")
        checkBox9 = QCheckBox("투자자별순위")

        tableWidget = QTableWidget(10, 5)
        tableWidget.setHorizontalHeaderLabels(["종목코드", "종목명", "현재가",
        "등락률", "거래량"])
        tableWidget.resizeColumnsToContents()
        tableWidget.resizeRowsToContents()

        leftInnerLayout = QVBoxLayout()
        leftInnerLayout.addWidget(checkBox1)
        leftInnerLayout.addWidget(checkBox2)
        leftInnerLayout.addWidget(checkBox3)
        leftInnerLayout.addWidget(checkBox4)
        leftInnerLayout.addWidget(checkBox5)
        leftInnerLayout.addWidget(checkBox6)
        leftInnerLayout.addWidget(checkBox7)
        leftInnerLayout.addWidget(checkBox8)
        leftInnerLayout.addWidget(checkBox9)
        groupBox.setLayout(leftInnerLayout)

        leftLayout = QVBoxLayout()
        leftLayout.addWidget(groupBox)

        rightLayout = QVBoxLayout()
        rightLayout.addWidget(tableWidget)

        layout = QHBoxLayout()
```

```
layout.addLayout(leftLayout)
layout.addLayout(rightLayout)
```

```
self.setLayout(layout)
```

```
if __name__ == '__main__':
    app = QApplication(sys.argv)
    mywindow = MyWindow()
    mywindow.show()
    app.exec_()
```



위 예제는 QHBoxLayout과 QVBoxLayout을 중첩시켜 구성하였습니다. 설명은 아래 표에서 간단한 그림으로 하겠습니다.

QHBoxLayout	
QVBoxLayout	QVBoxLayout
QVBoxLayout x 2	QTableWidget
QCheckBox x 9	

Chapter 16.18 QFileDialog

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)
        self.setWindowTitle("PyStock v0.1")

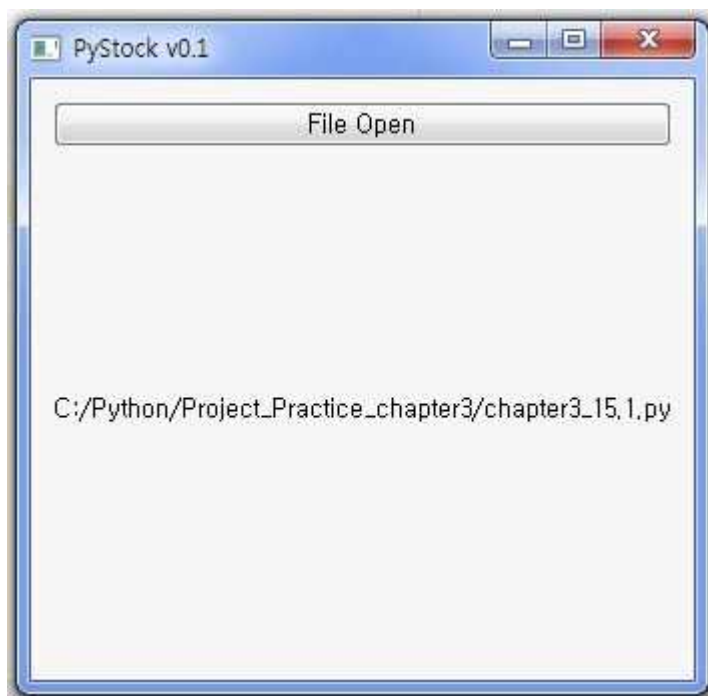
        self.pushButton = QPushButton("File Open")
        self.pushButton.clicked.connect(self.pushButtonClicked)
        self.label = QLabel()

        layout = QVBoxLayout()
        layout.addWidget(self.pushButton)
        layout.addWidget(self.label)

        self.setLayout(layout)

    def pushButtonClicked(self):
        fname = QFileDialog.getOpenFileName()
        self.label.setText(fname[0])

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = MyWindow()
    window.show()
    app.exec_()
```



QFileDialog를 사용한 예제입니다. 사용자가 File Open을 누르면 다이얼로그창이 뜨고 선택한 파일의 경로가 메인 윈도우에 출력되게 합니다.

Chapter3 16.19 QInputDialog

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)
        self.setWindowTitle("PyStock v0.1")

        self.pushButton = QPushButton("Input number")
        self.pushButton.clicked.connect(self.pushButtonClicked)
        self.label = QLabel()

        layout = QVBoxLayout()
        layout.addWidget(self.pushButton)
        layout.addWidget(self.label)

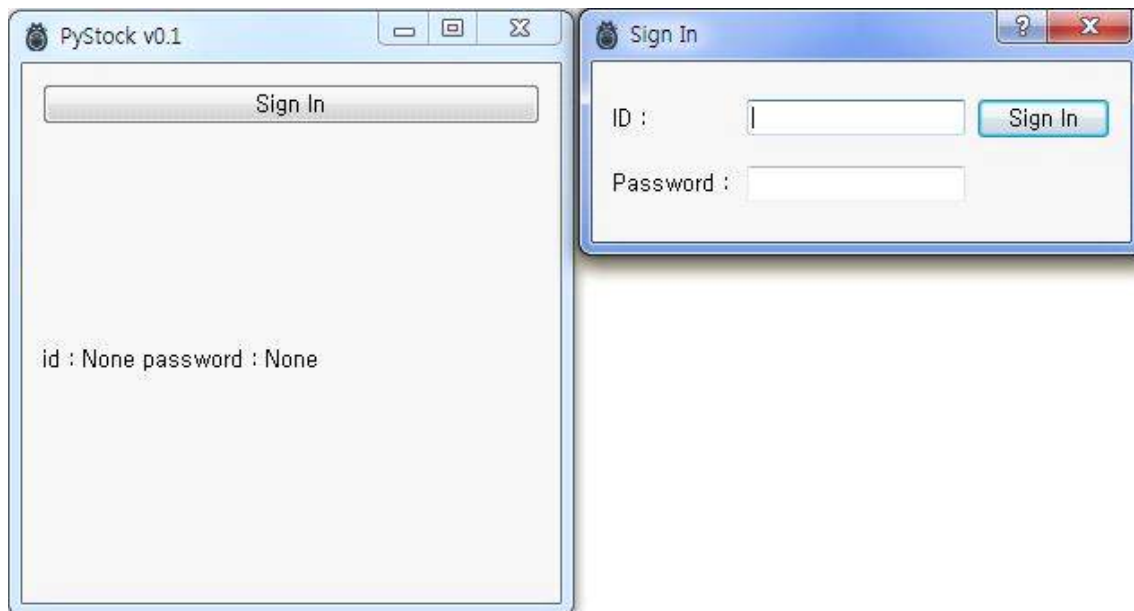
        self.setLayout(layout)

    def pushButtonClicked(self):
        text, ok = QInputDialog.getInt(self, '매수수량', '매수수량을입력하세요.')
        if ok:
            self.label.setText(str(text))

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = MyWindow()
    window.show()
    app.exec_()
```



Chapter3 16.20 메인 윈도우와 다이얼로그의 상호작용



```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *

class LogInDialog(QDialog):
    def __init__(self):
        super().__init__()
        self.setupUI()

    self.id = None
    self.password = None

    def setupUI(self):
        self.setGeometry(1100, 200, 300, 100)
        self.setWindowTitle("Sign In")
        self.setWindowIcon(QIcon('icon.png'))

        label1 = QLabel("ID : ")
        label2 = QLabel("Password : ")

        self.lineEdit1 = QLineEdit()
        self.lineEdit2 = QLineEdit()
        self.pushButton1 = QPushButton("Sign In")
        self.pushButton1.clicked.connect(self.pushButtonClicked)

        layout = QGridLayout()
        layout.addWidget(label1, 0, 0)
        layout.addWidget(self.lineEdit1, 0, 1)
        layout.addWidget(self.pushButton1, 0, 2)
        layout.addWidget(label2, 1, 0)
        layout.addWidget(self.lineEdit2, 1, 1)

        self.setLayout(layout)

    def pushButtonClicked(self):
        self.id = self.lineEdit1.text()
        self.password = self.lineEdit2.text()
        self.close()
```

```

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(800, 200, 300, 300)
        self.setWindowTitle("PyStock v0.1")
        self.setWindowIcon(QIcon('icon.png'))

        self.pushButton = QPushButton("Sign In")
        self.pushButton.clicked.connect(self.pushButtonClicked)
        self.label = QLabel()

        layout = QVBoxLayout()
        layout.addWidget(self.pushButton)
        layout.addWidget(self.label)

        self.setLayout(layout)

    def pushButtonClicked(self):
        dlg = LogInDialog()
        dlg.exec_()
        id = dlg.id
        passwd = dlg.password
        self.label.setText("id : %s password : %s"%(id, passwd))

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = MyWindow()
    window.show()
    app.exec_()

```

Chapter3 16.21 메인 레이아웃 구성

```

import sys
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import FigureCanvasQTAgg as
FigureCanvas

class MyWindow(QWidget):
    def __init__(self):
        super().__init__()
        self.setupUI()

    def setupUI(self):
        self.setGeometry(600, 200, 1200, 600)
        self.setWindowTitle("PyChart Viewwe v0.1")
        self.setWindowIcon(QIcon('icon.png'))

        self.lineEdit = QLineEdit()
        self.pushButton = QPushButton("차트그리기")
        self.pushButton.clicked.connect(self.pushButtonClicked)

        self.fig = plt.Figure()
        self.canvas = FigureCanvas(self.fig)

        leftLayout = QVBoxLayout()
        leftLayout.addWidget(self.canvas)

        #Right Layout
        rightLayout = QVBoxLayout()

```

```

rightLayout.addWidget(self.lineEdit)
rightLayout.addWidget(self.pushButton)
rightLayout.addStretch(1)

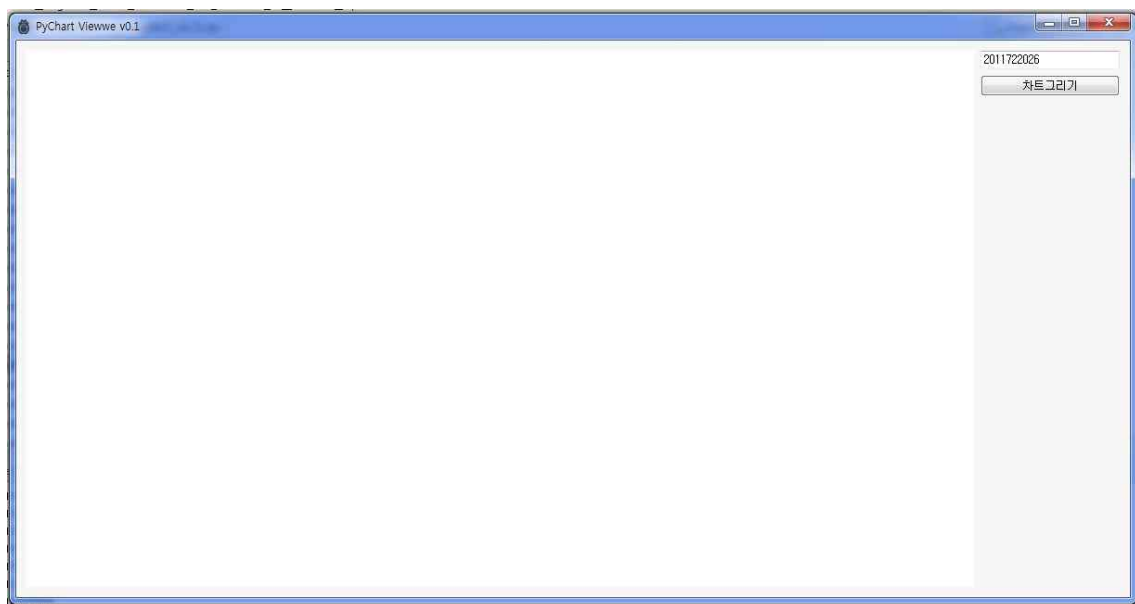
layout = QHBoxLayout()
layout.addLayout(leftLayout)
layout.addLayout(rightLayout)
layout.setStretchFactor(leftLayout, 1)
layout.setStretchFactor(rightLayout, 0)

self.setLayout(layout)

def pushButtonClicked(self):
    print(self.lineEdit.text())

if __name__ == '__main__':
    app = QApplication(sys.argv)
    window = MyWindow()
    window.show()
    app.exec_()

```



위에서 활용하였던 Layout 클래스들을 활용하여 메인 레이아웃을 구성해보는 예제입니다. 우측 위젯에 종목코드를 입력하면 좌측 레이아웃에 그래프가 출력되도록 하는 예제의 기본 코드로 활용됩니다.

졸업작품을 진행하는데 필요한 파이썬 모듈

크롤러에 사용한 모듈

```
#from pandas_datareader import data
#import fix_yahoo_finance as YF
from bs4 import BeautifulSoup
from urllib.request import urlopen
import re
import os
#from openpyxl import Workbook
from pyexcelerate import Workbook
import time
import sys
```

1. BeautifulSoup

웹 사이트를 분석하는데 필수적인 모듈이다. bs4를 이용해 웹 사이트에 있는 자료들을 크롤링할 수 있다.

2. urllib.request

웹을 열어 데이터를 가져오는데 쓰인 urllib.request 함수도 필수적인 모듈입니다.

3. re

문자열의 슬라이싱을 더욱 간편하게 하고자 할 때 쓰이는 re 모듈입니다.

4. os

수집한 자료들을 컴퓨터 내부 HDD에 저장하거나 HDD에 있는 데이터를 꺼내와 사용하고자 할 때 필요한 os 모듈입니다. 주로 경로생성 및 파일의 존재여부 확인에 사용하였습니다.

5. openpyxl, pyexcelerate

수집하고 가공한 데이터를 엑셀형식으로 저장하거나 가져올 때 필요한 모듈들입니다. 처음에는 openpyxl을 사용하였으나, pyexcelerate가 더 빠르고 사용하기 더 편하여 pyexcelerate가 제공하는 함수들을 사용하였지만 방대한 양의 데이터가 아니라면 openpyxl에서 제공하는 함수들을 사용해도 무방합니다.

6. time

말 그대로 시간에 관련된 함수들을 사용하고자 할 때 필요한 time 모듈입니다. 크롤러에서는 웹 사이트에서 데이터를 가져올 때 인터넷 연결이 불안정하여 제대로 가져오지 못하였을 경우 프로세스를 잠시 기다리게 하는 time.sleep 함수를 사용하는데 쓰였습니다.

7. sys

파이썬에서 변수나 함수들을 직접 제어하고자 할 때 쓰이는 모듈입니다. 크롤러를 코딩할 때에는 반복적으로 인터넷 연결이 불가능하다면 sys.exit(1)로 강제 프로세스 종료에 쓰였습니다.

8. oauth2

Twitter 인증 Package

9. matplotlib

그래프 생성 Package
시각화에 필수적입니다.

10. selenium

웹 브라우저 시뮬레이션 Package

11. pandas, numpy

Data Processing Package

12. jpyepl, KoNLPy, pytagcloud

자연어 처리, 워드 클라우드 그림 생성 Package

13. scikit-learn

data mining 과 data analysis를 위한 tool

NumPy, SciPy, 그리고 matplotlib와 함께 사용하기를 권장하는 모듈입니다.