Python libraries

Andy Kim

COM (from Microsoft)

COM (Component Object Model) enables the code to access to external instances of other programs

```
import win23com.client

excel = win32com.client.Dispatch("Excel.Application")
excel.Visible = True
wb = excel.Workbooks.Add()
ws = wb.Worksheets("Sheet 1")
ws.Cells(1,1).Value = "hello world"
wb.SaveAs('C"\\Users\\andy.kim\\test.xlsx')
excel.quit()
```

- Pandas can make it easier to communicate with Excel
- Daesin / eBest

OCX (from Microsoft too)

OCX (Object Linking and Embedding Custom Contrl) can be accessed using QAxContainer Module from PyQt package

- Kiwoom

PyQt

PyQt is a GUI framework binding of Qt (written in C++)

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

```
app = QApplications(sys.argv)  # sys.argv contains path to current code
label = QPushButton("Quit")
label.show()
app.exec_()   # Event loop
```

Another example:

```
import sys
from PyQt5.QtWidgets import *

class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()  # calling 'init' of parent class; note no 'self'
        self.setWindowTitle("PyStock")
        self.setGeometry(300,300,300,400)

        btn1 = QPushButton("Click me", self)
        btn1.move(20, 20)
        btn1.clicked.connect(self.btn1_clicked)
        # 'clicked' event is generated, so it needs to define event handler

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



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

Kiwoom API calling

```
# ... inside of __init__(self) ... above
        self.kiwoom = QAxWidget("KHOPENAPI.KHOpenAPICtrl.1")
        # CLSID or ProgID to be found in Windows registry

        self.kiwoom.dynamicCall("CommConnect()")

        ret = self.kiwoom.dynamicCall("GetConnectStatus()")
```

```
            self.kiwoom.OnEventConnect.connect(self.event_connect)

            account_num = self.kiwoom.dynamicCall("GetLoginInfo(QString)", ["ACCNO"])
            # "ACCNO" is input to QString
            # It has to be the list format ["ACCNO"] for GetLoginInfo()
            # QString is defined in QtCore, so QtCore.QString has to be imported if not

    # Event Handler of OnEventConnect
    def event_connect(self, err_code):
        if err_code == 0:
            pass
```

Or, another method:

```
    class Kiwoom(QAxWidget):
        def __init__(self):
            super().__init()__()
            self.setControl("KHOPENAPI.KHOpenAPICtrl.1")

        def comm_connect(self)
            self.dynamicCall("CommConnect()")
            self.login_event_loop = QEventLoop()  # creates an event loop
            self.login_event_loop.exec_()

        # After Login... the following has to be executed... through event handler

            self.login_event_loop.exit()
```

QtDesigner

QtDesigner is a GUI design tool, and creates .ui file that can be included into the code and reffered to

```
    import sys
    from PyQt5.QWidgets import *
    from PyQt5 import uic

    form_class = uic.loadUiType("certainUI.ui")[0] # return is a tuple, so specify [0]

    class MyWindow(QMainWindow, form_class):
```

```
    def __init__(self)
        super().__init__()
        self.setupUi(self)
```

Pandas

Data structure library; mainly used structures are Series, DataFrame

- Series: Python list with index

  ```
  from pandas import Series

  a = Series([10, 20, 30], index=['a', 'b', 'c'])
  b = Series([100, 200, 300], index=['c', 'b', 'a'])
  ```

- DataFrame: 2x2 Matrix with column and row index

  ```
  from pandas import DataFrame

  dict1 = {'a': [1, 2, 3]
           'b': [4, 5, 6]
           'c': [7, 8, 9]}

  df1 = DataFrame(dict1, index=['x', 'y', 'z'])
  df1.ix[0] # access to row 1
  ```

Refer to manual for various operators of Series and DataFrame

pandas_datareader

pandas_datareader is a datareader that returns DataFrame. It supports numerous financial institutioins - refer to its documentation on the web

matplotlib

matplotlib is a Python 2D plotting library; similar to Matlab

```
import matplotlib.pyplot as plt
plt.plot(df1.index, df1['a'])
plt.show()
```

zipline

Zipline is an open-source algorithmic trading simulator written in Python - refer to doc

SQLite

```
import sqlite3
con = sqlite3.connect("path.db") # if not exists, creates a new file; otherwise read
cursor = con.cursor()
cursor.execute("CREATE TABLE tname(Date text, Value1 int)")
cursor.execute("INSERT INTO tname VALUES('19.01.01', 100)")
cursor.execute("INSERT INTO tname VALUES('19.01.02', 200)")

con.commit()

cursor.execute("SELECT * From tname")
cursor.fetchone()

cursor.execute("SELECT * From tname")
cursor.fetchall()

cursor.execute("SELECT * From tname")
tname_var = cursor.fetchall() # list type

con.close()
```

Below code is to connect between SQLite and Pandas

```
# continue from the above
import pandas as pd
from pandas import Series, DataFrame
raw_data = { "some dictionary data" }
df = DataFrame(raw_data)
con = sqlite3.connect("path.db")

df.to_sql('table_name', con) # save raw_data to db with given table_name
df.to_sql('table_name', con, chunksize=100) # if error occurs due to packet size limit

df2 = pd.read_sql("SELECT * From tname", con, index_col=None) # this loads db into df2
```

```
con.close()
```

pywinauto

pywinauto package can send signals (e.g., mouse click or key strokes) to Windows applications without actual clicks or strokes

```
from pywinauto import application
from pywinauto import timings
import time
import os

app = application.Application()
app.start("path to program.exe")

title = "..."
dlg = timings.WaitUntilPasses(20, 0.5, lambda: app.window_(title=title))
# returns handler of the window;
# window_() returns a window of the application and might be outdated (older version)

pass_ctrl = dlg.Edit2
pass_ctrl.setFocus()
pass_ctrl.TypeKeys('xxxx')

cert_ctrl = dlg.Edit3
cert_ctrl.setFocus()
cert_ctrl.TypeKeys('xxxx')

btn_ctrl = dlg.Button0
btn_ctrl.Click()

time.sleep(50)
os.system("taskkill /im program.exe") # may not work in linux (tested in Windows)
```

SWAPY

SWAPY can be used to find-out the window element/control names (github.com/pywinauto/SWAPY)

OS Scheduler

Windows scheduler could be used to automate running programs; pythonw.exe does not open console window, so more proper for automation (Linux may have different process)

requests module and scrapping using pandas

Use request module to download html from the Internet Then, use pandas.read_html() to parse the html

```
import requests
import pandas as pd

html = request.get(url).text
df = pd.read_html(html, index_col="  ")
```

BeautifulSoup

BeautifulSoup provides scrapping from the Internet

```
import requests
from bs4 import BeautifulSoup

url = "url address"
html = requestsi.get(url).text
soup = BeautifulSoup(html, ’lxml’)  # lxml is XML and HTML parser in Python
tr_data = soup.find_all(’tr’, id=’tr_id’)
# tr_data = soup.find_all(’tr’, {’class’: ’class_name’}) # using dictionary format

td_data = tr_data[0].find_all(’td’)

# if multiple tds exist, each td data can be accessed through the following
td_data[0].text
td_data[1].text
td_data[2].text

i = 0
for x in td_data:
    result_list[i] = x.text
```

```
        i += 1
```

numpy

NumPy is a package for scientific computing and graphs