Table of contents

### 

### 

[Using Google colab](#_86kr6t52mzsn)

[Matplotlib](#_o0vrap7rdcmc)

[Quandle example](#_algujcjgpc44)

[Quandl time series](#_4ydxa41ygisq)

[Quandl tables](#_grupchic75ny)

[Pandas introduction](#_wcn7yljxccyj)

### 

### [Using Google colab](http://colab.research.google.com)

수업에서 파이썬 코딩은 구글 colab에서 실시한다. Google 드라이브 계정을 가진 사람은 누구나 colab에 가입 가능하다. 아래 링크의 페이지로 이동하여 나열된 안내를 따르면 된다. 이때 브라우저는 google chrome([http://www.google.com/chrome/](https://www.google.com/chrome/)

)을 다운로드 받아서 사용한다.

<http://colab.research.google.com>

Colab에 가입 후 File -> New Notebook으로 이동한다. 여기서 파이썬 2 또는 3 모두 사용가능하다. 수업에서는 파이썬 3을 추천한다.

다음 GPU runtime을 설정한다. 경로는 다음과 같다.

Runtime -> Change runtime type -> Hardware Accelerator -> GPU

이제 코딩을 시작하면 된다.

print(“Hello World!”)

### 

### [Matplotlib](https://colab.research.google.com/drive/1BCkFX2ipve7TWb5l63GvwkHXRAaNfDZA)

<https://matplotlib.org/users/pyplot_tutorial.html>

### [Quandle example](https://colab.research.google.com/drive/1Y5d0LOTyX4oOrH8KpTKVCfFlvk9RydAF)

Quandl은 금융 전문가에게 도움이되는 정형 및 비정형 데이터를 위한 플랫폼이며 마켓플레이스다. 수백개의 업체(publishers)가 quandl에서 데이터를 제공하거나 판매하고 있다. Quandl의 모든 데이터는 API를 통해 액세스 할 수 있다.] API 액세스는 R, Python, Matlab, Maple 및 Stata를 포함한 여러 프로그래밍 언어 패키지를 통해 가능하다.

구글 Colab에서 다음과 같은 명령어로 quandl을 설치한다.

!pip install quandl

import quandl

Quandl Python 모듈 자체는 무료다. 그러나 데이터를 다운로드하려면 Quandl API 키가 필요하다. 자신의 API 키를 얻으려면 무료 Quandl 계정을 만들고 API 키를 설정하면 된다. 계정은 [http://www.quandl.com](http://www.quandl.com/) 에서 만든다.

Quandl 모듈을 가져온 후 다음 명령을 사용하여 API 키를 설정할 수 있다.

quandl.ApiConfig.api\_key = "YOURAPIKEY"

여기서 YOURAPIKEY 부분에 본인의 프로파일(<https://www.quandl.com/account/profile>)에서 볼 수 있는 API Key를 복사 후 붙여넣기(copy-and-paste)하면 된다.

### [Quandl time series](https://docs.quandl.com/docs/python-time-series)

Quandl에서 시계열자료(time series data)를 받는 방법은 다음 링크에 잘 설명이 되어 있다. <https://docs.quandl.com/docs/python-time-series>

원유 가격에 대한 대표적인 지수인 WTI 정보([WTI Crude Oil Price](https://www.quandl.com/EIA/PET_RWTC_D))를 불러보자. WTI의 quandl 코드는 EIA/PET\_RWTC\_D 이다. 데이터는 미국에너지부([US Department of Energy](https://www.quandl.com/EIA) dataset)다. 코드는 다음과 같다.

data = quandl.get("EIA/PET\_RWTC\_D")

print(data.head())

만약에 WTI의 최근 5일간 일별 % 변화를 보고 싶다면 pct\_change() 기능을 활용한다.

print(data.pct\_change().tail()\*100.)

나중에 배울 NumPy array 형식으로 다음과 같이 받을 수 있다.

data = quandl.get("EIA/PET\_RWTC\_D", returns="numpy")

월별로 데이터를 받고 싶으면 다음을 사용한다.

data = quandl.get("EIA/PET\_RWTC\_D", collapse="monthly")

세인트루이스 연방준비은행(ederal Reserve Bank of St. Louis)에서는 FRED라는 유명한 데이터베이스를 운영한다. 여기서 GDP 데이터를 받아보자. 데이터의 시작 날짜와 마지막 날짜도 설정해보자.

# GDP size

data = quandl.get("FRED/GDP", start\_date="2001-12-31", end\_date="2005-12-31")

# GDP % change

data1 = quandl.get("FRED/GDP", start\_date="2001-12-31", end\_date="2005-12-31", transformation="rdiff")

# GDP % change를 받는 또다른 방법

data2 = quandl.get("FRED/GDP", start\_date="2001-12-31", end\_date="2005-12-31").pct\_change()

#Data1과 data2를 비교해보면 동일하다.

print(data1.tail())

print(data2.tail())

인도증시 NSE(National Stock Exchange of India Ltd.)에 상장된 Oil India 주가와 NASDAQ에 상장된 Apple 의 주가를 다운로드해보자. 여기서는 Oil India 주가 정보 중 첫번째 열인 시초가(opening price)와 Apple의 종가 (closing price)를 받아보자. 시초가는 첫번째 열(column)이고 종가는 네번째 열이다. 따라서 다음과 같은 형식으로 데이터를 부른다.

data = quandl.get(["NSE/OIL.1", "WIKI/AAPL.4"])

애플 주가의 최근 5일 데이터는 다음과 같다.

data = quandl.get("WIKI/AAPL", rows=5)

데이터를 월별 수준으로 바꿀 수 있다.

data = quandl.get("EIA/PET\_RWTC\_D", collapse="monthly")

간단한 계산도 바로 수행할 수 있다 (e.g. % 변화).

data = quandl.get("FRED/GDP", transformation="rdiff")

### [Quandl tables](https://docs.quandl.com/docs/python-table)

테이블 자료를 받는 방법은 다음 링크를 참고하자.

<https://docs.quandl.com/docs/python-tables>

이하 테이블 자료에 관한 설명이다. 참고로 테이블 API는 호출 당 10,000 행으로 제한된다. 그러나 파이썬 라이브러리를 사용할 때, 인수 paginate = True를 추가하면 제한이 1,000,000 개의 행으로 확장된다. 따라서 모든 호출에 paginate = True를 사용하자.

Download data for Nokia (compnumber=39102)

data = quandl.get\_table('MER/F1', compnumber="39102", paginate=True)

Download data for Nokia (compnumber=39102) and Deutsche Bank AG (compnumber=2438)

data = quandl.get\_table('MER/F1', compnumber=["39102" , "2438"], paginate=True)

Download the compnumber column

data = quandl.get\_table('MER/F1',qopts={"columns":"compnumber"}, paginate=True)

Download the compnumber and ticker columns

data = quandl.get\_table('MER/F1',qopts={"columns":["compnumber", "ticker"]}, paginate=True)

Download the reportdate column for Nokia (compnumber=39102)

data = quandl.get\_table('MER/F1',compnumber="39102", qopts={"columns":"reportdate"}, paginate=True)

Download the reportdate, indicator, and amount columns for Nokia (compnumber=39102)

data = quandl.get\_table('MER/F1',compnumber="39102", qopts={"columns":["reportdate", "indicator", "compnumber", "ticker"]}, paginate=True)

Download the closing prices for Apple (AAPL) and Microsoft (MSFT) between 2016-01-01 and 2016-12-31.

data = quandl.get\_table('WIKI/PRICES', qopts = { 'columns': ['ticker', 'date', 'close'] }, ticker = ['AAPL', 'MSFT'], date = { 'gte': '2016-01-01', 'lte': '2016-12-31' })

print(data.head()) # 이렇게 하면 data의 첫번째 다섯행을 보여준디ㅏ.

분석하기 좋게 정렬해보자.

clean = data.set\_index('date')

table = clean.pivot(columns='ticker')

print(table.head())

주식 종가 불러오는 또다른 예는 다음과 같다.

#get adjusted closing prices of 5 selected companies with Quandl

selected = ['CNP', 'F', 'WMT', 'GE', 'TSLA'] # selected assets list

data = quandl.get\_table('WIKI/PRICES', ticker = selected,

qopts = { 'columns': ['date', 'ticker', 'adj\_close'] },

date = { 'gte': '2014-1-1', 'lte': '2018-3-15' }, paginate=True)

print(data.head())

print(data.tail())

데이터를 분석하기 쉽게 정렬해보자.

# reorganise data pulled by setting date as index with

# columns of tickers and their corresponding adjusted prices

clean = data.set\_index('date')

table = clean.pivot(columns='ticker')

print(table.head())

print(table.tail())

정렬한 데이터로 다음과 같은 간단한 계산할 수 있다: % 변화, 평균, 공분산행렬

# calculate daily and annual returns of the stocks

returns\_daily = table.pct\_change()

returns\_annual = returns\_daily.mean() \* 252 # use average instead of expected value

print(returns\_daily.tail())

print(returns\_annual)

# get daily and covariance of returns of the stock

cov\_daily = returns\_daily.cov()

cov\_annual = cov\_daily \* 252

#returns\_daily.head()

print(cov\_annual)

### 

### [Pandas introduction](https://colab.research.google.com/drive/1kwJQj7bWbEV3XUI3JzyU7lOMJTh8nbrq)

Pandas library를 불러오고 비어있는 pandas series 만들기

Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.

import pandas as pd #import the pandas library and aliasing as pd

s = pd.Series()

print(s)

ndarray 에서 pandas series 만들기

import pandas as pd

import numpy as np

data = np.array(['a','b','c','d'])

s = pd.Series(data)

print(s)

index 부여하기

data = np.array(['a','b','c','d'])

s = pd.Series(data,index=[100,101,102,103])

print(s)

Dictionary 이용하기

data = {'a' : 0., 'b' : 1., 'c' : 2.}

s = pd.Series(data)

# Observe − Dictionary keys are used to construct index.

print(s)

data = {'a' : 0., 'b' : 1., 'c' : 2.}

s = pd.Series(data,index=['b','c','d','a'])

# Observe − Index order is persisted and the missing element is filled with NaN (Not a Number).

print(s)

상수(scalar)이용 series 만들어보자.

s = pd.Series(5, index=[0, 1, 2, 3])

print(s)

위치(position)를 활용하여 series에서 데이터 불러오자.

첫번째 원소

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve the first element

print(s[0])

첫번째에서 세번째 원소

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve the first three element

print(s[:3])

마지막 세번째 원소를 불러와보자.

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve the last three element

print(s[-3:])

Index를 이용하여 데이터를 불러와보자.

s = pd.Series([1,2,3,4,5],index = ['a','b','c','d','e'])

#retrieve a single element

print(s['a'])

print('-----------------')

print(s[['a','c','d']])

DataFrame (데이터프레임)

Dataframe은 이차원 데이터구조다. 행과 열 구조의 테이블 형태로 데이터를 정렬한다. (A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.)

비어있는 dataframe 만들기

# create empty dataframe

df = pd.DataFrame()

print(df)

일차원 데이터

# Create a DataFrame from Lists

# The DataFrame can be created using a single list or a list of lists.

data = [1,2,3,4,5]

df = pd.DataFrame(data)

print(df)

문자열(string)과 정수로된 이차원 데이터에 열마다 이름을 지정한다.

data = [['Alex',10],['Bob',12],['Clarke',13]]

df = pd.DataFrame(data,columns=['Name','Age'])

print(df)

정수를 실수로 바꿀 수 있다.

data = [['Alex',10],['Bob',12],['Clarke',13]]

df = pd.DataFrame(data,columns=['Name','Age'],dtype=float)

print(df)

Dictionary와 List 기능을 활용

All the ndarrays must be of same length. If index is passed, then the length of the index should equal to the length of the arrays. If no index is passed, then by default, index will be range(n), where n is the array length. Use {} curly brackets to construct the dictionary, and [] square brackets to index it. Separate the key and value with colons : and with commas , between each pair.

딕셔너리 기능 활용

data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]}

df = pd.DataFrame(data)

print(df)

인덱스를 부여하기

#Let us now create an indexed DataFrame using arrays.

df = pd.DataFrame(data, index=['rank1','rank2','rank3','rank4'])

print(df)

# Note − Observe, the index parameter assigns an index to each row.

웹 URL에서 csv 파일 불러와서 테이블을 만들고 테이블 내부의 데이터에 접근한다.

# Reading a CSV file from a URL with pandas

df1 = pd.read\_csv("https://pythonhow.com/data/income\_data.csv")

print(df1)

# Assigning an index column to pandas dataframe

df2 = df1.set\_index("State", drop = True)

print(df2)

df2 = df1.set\_index("State", drop = False)

print(df2)

# Extracting a subset of a pandas dataframe

# format: df2.loc[startrow:endrow, startcolumn:endcolumn]

df2.loc["Alaska":"Arkansas","2005":"2007"]

# Extracting a column of a pandas dataframe

df2.loc[: , "2005"]

# To extract a column you can also do:

df2["2005"]

# Extracting a row of a pandas dataframe ¶

df2.loc["California", : ]

# Extracting specific columns of a pandas dataframe ¶

df2[["2005", "2008", "2009"]]

# Extracting specific rows of a pandas dataframe ¶

df2[1:3]

# Extracting a single cell from a pandas dataframe ¶

df2.loc["California","2013"]

# Note that you can also apply methods to the subsets:

df2.loc[:,"2005"].mean()

'''

Now, sometimes, you don’t have row or column labels.

In such case you will have to rely on position based indexing

which is implemented with iloc instead of loc.

'''

df2.iloc[0:3,0:4]

Note that when we used label based indexing both the start and the end labels were included in the subset. With position based slicing, only the start index is included. So, in this case Alabama had an index of 0, Alaska 1, and Arizona 2. Same goes for the columns.

And one more thing you should now about indexing is that when you have labels for either the rows or the columns, and you want to slice a portion of the dataframe, you wouldn’t know whether to use loc or iloc. In this case, you would want to use ix:

df2.ix[0:3,"2005":"2007"]