Copyright © 2021 Quandatics Academy Sdn Bhd

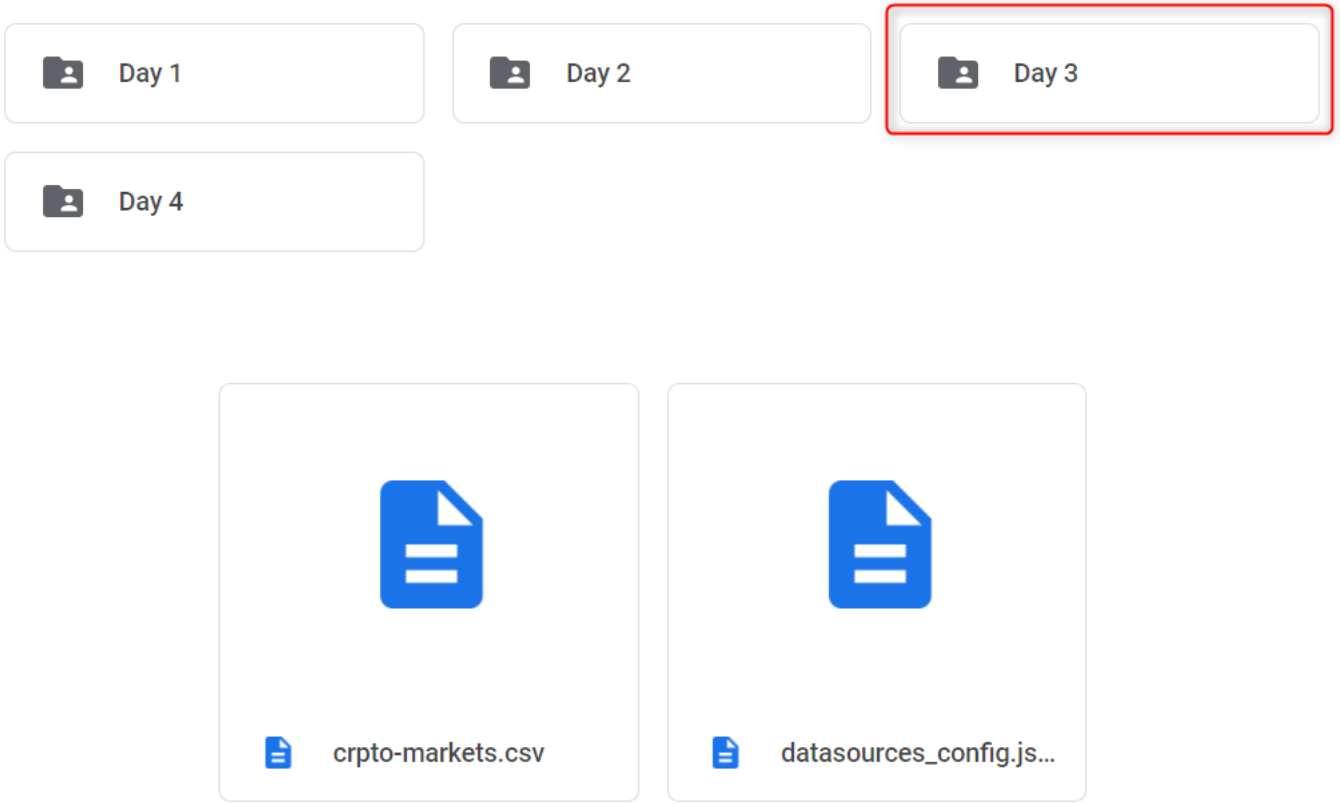
QUANDATICS ACADEMY

ETL in Python

Trainer : Foo Chee Chuan

QUANDATICS ACADEMY

Data Sources



QUANDATICS ACADEMY

Json Config File

```
{
    "datasources":{
        "api":{
            "Pollution": "https://api.openaq.org/v1/latest?country=IN&limit=10000",
            "Economy": "https://api.data.gov.in/resource/07d49df4-233f-4898-92db-e6855d4dd94c?api-key=579b464db66ec23
        },
        "csv":{
            "CryptoMarkets": "crypto-markets.csv"
        }
    }
}
```

QUANDATICS ACADEMY

APIs

```
# Load api information from json config file
data_sources = json.load(open('datasources_config.json'))
api = data_sources['datasources']['api']
api_url = api['Economy']

print(api_url)
```

https://api.data.gov.in/resource/07d49df4-233f-4898-92db-e6855d4dd94c?api-key=579b464db66ec23bdd000001cdd3946e44ce4aad7209ff7b23ac571b&format=json&offset=0&limit=100

Click to open URL

```
{
    "created": 1496921004,
    "updated": 1543520100,
    "title": "GDP of India and major Sectors of Economy, Share of each sector to GDP and Growth rate
    "source": "data.gov.in",
    "org_type": "Central",
    "org": [
        "NITI Aayog"
    ],
    "sector": [
        "Macro Economy",
        "Economy"
    ],
    "visualizable": "1",
    "index_name": "07d49df4-233f-4898-92db-e6855d4dd94c",
    "catalog_uuid": "9e964439-fb05-446c-9903-a6bf14899f26",
    "status": "ok",
    "field": [
        {
            "id": "financial_year",
            "name": "Financial Year",
            "type": "keyword"
        },
        {
            "id": "gross_domestic_product_in_rs_cr_at_2004_05_prices",
            "name": "Gross Domestic Product (in Rs. Cr) at 2004-05 Prices",
            "type": "double"
        },
```


← Data in json format

```
"records": [
    {
        "financial_year": "1951-52",
        "gross_domestic_product_in_rs_cr_at_2004_05_prices": "286147",
        "agriculture_allied_services_in_rs_cr_at_2004_05_prices": "147216",
        "agriculture_in_rs_cr_at_2004_05_prices": "118877",
        "industry_in_rs_cr_at_2004_05_prices": "47768",
        "mining_and_quarrying_in_rs_cr_at_2004_05_prices": "5772",
        "manufacturing_in_rs_cr_at_2004_05_prices": "25889",
        "services_in_rs_cr_at_2004_05_prices": "84799",
        "agri_culture_allied_services__share_to_total_gdp": "5145",
        "agriculture___share_to_total_gdp": "4154",
        "industry___share_to_total_gdp": "1669",
        "mining_and_quarrying___share_to_total_gdp": "202",
        "manufacturing___share_to_total_gdp": "905",
        "services___share_to_total_gdp": "2963",
        "gross_domestic_product___growth_rate_yoy_": "233",
        "agri_culture_allied_services___growth_rate_yoy_": "149",
        "agriculture___growth_rate_yoy_": "164",
        "industry___growth_rate_yoy_": "550",
        "mining_and_quarrying___growth_rate_yoy_": "1233",
        "manufacturing___growth_rate_yoy_": "316",
        "services___growth_rate_yoy_": "267"
    },
```


- **records** is our data
- **Key** – Column Name
- **Value** – Data

QUANDATICS ACADEMY

Financial Year	Gross Domestic Product (In Rs. Cr) At 2004-05 Prices	Agriculture & Allied Services (In Rs. Cr.) At 2004-05 Prices	Agriculture (In Rs. Cr.) At 2004-05 Prices	Industry (In Rs. Cr.) At 2004-05 Prices	Mining And Quarrying (In Rs. Cr.) At 2004-05 Prices	Manufacturing (In Rs. Cr.) At 2004-05 Prices	Services (In Rs. Cr.) At 2004-05 Prices	Agri-Culture & Allied Services -Share To Total GDP	Agriculture - Share To Total GDP	Industry - Share To Total GDP	
1951-52	286147	147216	118877	47768	5772	25889	84799	5145	4154	1669	2
1952-53	294267	151859	123822	47739	5905	26789	87438	5161	4208	1622	2
1953-54	312177	163553	134958	50409	5993	28863	89940	5239	4323	1615	1
1954-55	325431	168361	138731	54574	6250	30885	94172	5173	4263	1677	1
1955-56	333766	166906	136679	60311	6348	33304	98845	5001	4095	1807	1
1956-57	352766	175980	144859	65480	6671	35804	103391	4989	4106	1856	1
1957-58	348500	168075	137497	64842	7105	37184	107275	4823	3945	1861	2
1958-59	374948	185010	152961	69335	7327	39026	111690	4934	4080	1849	1
1959-60	383153	183147	150730	74081	7704	41676	117232	4780	3934	1933	2
1960-61	410279	195482	161708	82413	8857	45134	123872	4765	3941	2009	2

Powered by visualize.data.gov.in

Data in Table Form

```
# Use requests.get to get information from website
response = requests.get(api_url)

# Store data in json format
apiData = response.json()

print(type(apiData))
```

```
<class 'dict'>
```

All information from API is stored in a dictionary

QUANDATICS ACADEMY

```
1 print(apiData["records"])
```

```
[{'financial_year': '1951-52', 'gross_domestic_product_in_rs_cr_at_2004_05_prices
s_cr_at_2004_05_prices': '147216', 'agriculture_in_rs_cr_at_2004_05_prices': '118
'47768', 'mining_and_quarrying_in_rs_cr_at_2004_05_prices': '5772', 'manufacturin
ces_in_rs_cr_at_2004_05_prices': '84799', 'agri_culture_allied_services__share_to
otal_gdp': '4154', 'industry___share_to_total_gdp': '1669', 'mining_and_quarrying
__share_to_total_gdp': '905', 'services___share_to_total_gdp': '2963', 'gross_dom
i_culture_allied_services___growth_rate_yoy_': '149', 'agriculture___growth_rate_
'550', 'mining_and_quarrying___growth_rate_yoy_': '1233', 'manufacturing___growth
y_': '267'}, {'financial_year': '1952-53', 'gross_domestic_product_in_rs_cr_at_20
ervices_in_rs_cr_at_2004_05_prices': '151859', 'agriculture_in_rs_cr_at_2004_05_p
5_prices': '47739', 'mining_and_quarrying_in_rs_cr_at_2004_05_prices': '5905', 'm
89', 'services_in_rs_cr_at_2004_05_prices': '87438', 'agri_culture_allied_service
_share_to_total_gdp': '4208', 'industry___share_to_total_gdp': '1622', 'mining_an
ufacturing___share_to_total_gdp': '910', 'services___share_to_total_gdp': '2971',
'284', 'agri_culture_allied_services___growth_rate_yoy_': '315', 'agriculture___g
te yoy ': '-006', 'mining and quarrying  growth rate yoy ': '231', 'manufacturin
```

records is a **list** that contains multiple **dictionaries**

QUANDATICS ACADEMY

Convert "records" into pandas data frame

```
# print(apiData["records"])

df = pd.json_normalize(apiData["records"])

df.head()
```

	financial_year	gross_domestic_product_in_rs_cr_at_2004_05_prices	agriculture_allied_services_in_rs_cr_at_2004_0
0	1951-52	286147	
1	1952-53	294267	
2	1953-54	312177	
3	1954-55	325431	
4	1955-56	333766	

```
df_GDP = df[['gross_domestic_product_in_rs_cr_at_2004_05_prices','gross_domestic_product___growth_rate_yoy_']]
df_GDP.head()
```

	gross_domestic_product_in_rs_cr_at_2004_05_prices	gross_domestic_product___growth_rate_yoy_
0	286147	233
1	294267	284
2	312177	609
3	325431	425
4	333766	256

Take 2 columns from the data frame

```
df_GDP['CalcGDPGrowth'] = df_GDP['gross_domestic_product_in_rs_cr_at_2004_05_prices'].astype(str).astype(int).pct_change()

df_GDP['CalcGDPGrowth']*100

df_GDP.head()
```

	gross_domestic_product_in_rs_cr_at_2004_05_prices	gross_domestic_product___growth_rate_yoy_	CalcGDPGrowth
0	286147	233	NaN
1	294267	284	0.028377
2	312177	609	0.060863
3	325431	425	0.042457
4	333766	256	0.025612

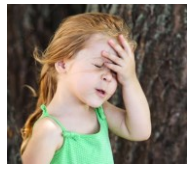
Add a column to calculate GDP growth rate

QUANDATICS ACADEMY

Pollution API

```
1  # Load api information from json config file
2  data_sources = json.load(open('datasources_config.json'))
3  api = data_sources['datasources']['api']
4  api_urlPollution = api['Pollution']
5
6  print(api_urlPollution)
7
8  # Use requests.get to get information from website
9  response = requests.get(api_urlPollution)
10
11 # Store data in json format
12 apiDataPollution = response.json()
13
14 print(type(apiDataPollution))
```

```
https://api.openaq.org/v1/latest?country=IN&limit=10000
<class 'dict'>
```



We have to repeat the same code all over again

```python
# Load api information from json config file
data_sources = json.load(open('datasources_config.json'))
api = data_sources['datasources']['api']
api_urlPollution = api['Pollution']

print(api_urlPollution)

# Use requests.get to get information from website
response = requests.get(api_urlPollution)

# Store data in json format
apiDataPollution = response.json()

print(type(apiDataPollution))
```

Have to use different name to not overwrite

```
https://api.openaq.org/v1/latest?country=IN&limit=10000
<class 'dict'>
```

QUANDATICS ACADEMY

Class

UTM (Johor)

Students

- StudentID
- Name

QUANDATICS ACADEMY

Initialization __init__(self)

QUANDATICS ACADEMY

Identify each students

In [252]:

```
1  class Student:
2      def __init__(self,name):
3          self.name = name
4
5      def stu_name(self):
6          return self.name
7
8  student_180756 = Student("Andrew")
9  student_180757 = Student("Sarah")
10 print(student_180756.stu_name())
11 print(student_180757.stu_name())
```

```
Andrew
Sarah
```

- **Self : StudentID**
- **Self.name : Name of the student belong to this studentID**
- **1 object = 1 student = 1 student ID**

QUANDATICS ACADEMY

Instance Variables

- StudentID : 180756
- Name : Andrew
- Age : 25
- Gender : Male

- StudentID : 180757
- Name : Sarah
- Age : 21
- Gender : Female

In [ ]:

```
1  class Student:
2      def __init__(self,name,age,gender):
3          self.name = name
4          self.age = age
5          self.gender = gender
6
7      def stu_name(self):
8          return self.name
9      def stu_age(self):
10         return self.age
11     def stu_gender(self):
12         return self.gender
13
14 student_180756 = Student("Andrew",25,"Male")
15 student_180757 = Student("Sarah",21,"Female")
16
```

QUANDATICS ACADEMY

Practice 1: Print Test Scores

- Test_01 : 60%
- Test_02 : 78%

- Test_01 : 75%
- Test_02 : 88%

Create an instance variable for test_01 & test_02 respectively

```
15 student_180756 = Student("Andrew",60,78)
16 print(student_180756.stu_name(),"'s test scores:")
17 print("Test 1 score: ",student_180756.test1_score())
18 print("Test 2 score: ",student_180756.test2_score())
19 print()
20 student_180757 = Student("Sarah",75,88)
21 print(student_180757.stu_name(),"'s test scores:")
22 print("Test 1 score: ",student_180757.test1_score())
23 print("Test 2 score: ",student_180757.test2_score())
```

```
Andrew 's test scores:
Test 1 score:  60
Test 2 score:  78

Sarah 's test scores:
Test 1 score:  75
Test 2 score:  88
```

Print this output!

Calculate test score

- **Mid-term : test 1 + test 2 = (20%)**
- **Assignment : (20%)**
- **Final : (60%)**

QUANDATICS ACADEMY

Mid-term = (test 1 + test 2)/200*20

In [267]:

```
class Student:

    def __init__(self,name,test_01,test_02):
        self.name = name
        self.test_01 = test_01
        self.test_02 = test_02

    def stu_name(self):
        return self.name
    def test1_score(self):
        return self.test_01
    def test2_score(self):
        return self.test_02

student_180756 = Student("Andrew",60,78)
test_01 = student_180756.test1_score()
test_02 = student_180756.test2_score()
mid_term = round((test_01 + test_02)/200*20,1)
print(student_180756.stu_name(),"'s mid-term score is: ",mid_term,"%")
student_180757 = Student("Sarah",75,88)
test_01 = student_180757.test1_score()
test_02 = student_180757.test2_score()
mid_term = round((test_01 + test_02)/200*20,1)
print(student_180757.stu_name(),"'s mid-term score is: ",mid_term,"%")
```

```
Andrew 's mid-term score is:  13.8 %
Sarah 's mid-term score is:  16.3 %
```

QUANDATICS ACADEMY

Instance Method

QUANDATICS ACADEMY

Midterm score Method

In [276]:

```
class Student:

    def __init__(self,name,test_01,test_02):
        self.name = name
        self.test_01 = test_01
        self.test_02 = test_02

    def stu_name(self):
        return self.name
    def test1_score(self):
        return self.test_01
    def test2_score(self):
        return self.test_02
    def mid_term(self):
        return round((self.test_01 + self.test_02)/200*20,1)

student_180756 = Student("Andrew",60,78)
print(student_180756.stu_name(),"'s mid-term score:")
print(student_180756.mid_term())
student_180757 = Student("Sarah",75,88)
print(student_180757.stu_name(),"'s mid-term score:")
print(student_180757.mid_term())
```

```
Andrew 's mid-term score:
13.8
Sarah 's mid-term score:
16.3
```

QUANDATICS ACADEMY

Define extract class

```
class extract:
```

```
class extract:
    # Load configuration at initiation
    def __init__(self,configfile):
        self.configfile = configfile
        self.data_sources = json.load(open(self.configfile))['datasources']
        self.api = self.data_sources['api']
        self.csv_path = self.data_sources['csv']


    def getAPIsData(self,api_name):
        # Initiate Parameter
        self.api_name = api_name
        # We have multiple APIs
        # Use their name to identify
        api_url = self.api[self.api_name]
        # Show api name when connected
        print("API URL:",api_url)
        response = requests.get(api_url)
            # response.json() will convert json data into Python dictionary
        return response.json()
```

In [116]:

```
1 trainingDataSources = extract('datasources_config.json')
2 pollution = trainingDataSources.getAPIsData("Pollution")
```

```
API URL: https://api.openaq.org/v1/latest?country=IN&limit=10000
```

```python
# Load api information from json config file
data_sources = json.load(open('datasources_config.json'))
api = data_sources['datasources']['api']
api_urlPollution = api['Pollution']

print(api_urlPollution)

# Use requests.get to get information from website
response = requests.get(api_urlPollution)

# Store data in json format
apiDataPollution = response.json()

print(type(apiDataPollution))
```

Procedural Approach

```
https://api.openaq.org/v1/latest?country=IN&limit=10000
<class 'dict'>
```

OOP Approach

Extract class for API

```
17]: trainingDataSources = extract('datasources_config.json')
     pollution = trainingDataSources.getAPIsData("Pollution")
```

```
API URL: https://api.openaq.org/v1/latest?country=IN&limit=10000
```

```
18]: economy = trainingDataSources.getAPIsData("Economy")
```

```
API URL: https://api.data.gov.in/resource/07d49df4-233f-4898-92db-e6855d4dd94c?api-key=579b464db66ec23bdd000001cdd3946e44ce4aad7209ff7b23ac571b&format=json&offset=0&limit=100
```

QUANDATICS ACADEMY

Does it mean I have to memorize all Data Source Names?

Extract class for API

In [117]:
```
1 trainingDataSources = extract('datasources_config.json')
2 pollution = trainingDataSources.getAPIsData("Pollution")
```

```
API URL: https://api.openaq.org/v1/latest?country=IN&limit=10000
```

In [118]:
```
1 economy = trainingDataSources.getAPIsData("Economy")
```

```
API URL: https://api.data.gov.in/resource/07d49df4-233f-4898-92db-
7209ff7b23ac571b&format=json&offset=0&limit=100
```

We can create a method to show **all data sources available** in the config file

```
In [121]:  1  trainingDataSources.showDataSources()
```

Out[121]:

	Name	Type	File Name
0	Pollution	api	https://api.openaq.org/v1/latest?country=IN&li...
1	Economy	api	https://api.data.gov.in/resource/07d49df4-233f...
2	CryptoMarkets	csv	crypto-markets.csv

QUANDATICS ACADEMY

Directly convert into pandas Data Frame

In [123]:
```
1 datasources = json.load(open('datasources_config.json'))['datasources']
2 # print(datasources)
3 pd.DataFrame(datasources)
```

Out[123]:

	api	csv
Pollution	https://api.openaq.org/v1/latest?country=IN&li...	NaN
Economy	https://api.data.gov.in/resource/07d49df4-233f...	NaN
CryptoMarkets	NaN	crypto-markets.csv

QUANDATICS ACADEMY

Convert json into pandas Data Frame

In [124]:
```
1 datasources = json.load(open('datasources_config.json'))['datasources']
2 # print(datasources)
3 pd.json_normalize(datasources)
```

Out[124]:

	api.Pollution	api.Economy	csv.CryptoMarkets
0	https://api.openaq.org/v1/latest?country=IN&li...	https://api.data.gov.in/resource/07d49df4-233f...	crypto-markets.csv

Not the format we want

QUANDATICS ACADEMY

This is the format we want

	Name	Type	File Name
0	Pollution	api	https://api.openaq.org/v1/latest?country=IN&li...
1	Economy	api	https://api.data.gov.in/resource/07d49df4-233f...
2	CryptoMarkets	csv	crypto-markets.csv

```
datasources = json.load(open('datasources_config.json'))['datasources']
print(datasources)
```

```
{'api': {'Pollution': 'https://api.openaq.org/v1/latest?country=IN&limit=10000', 'Economy': 'https://api.data.gov.in/resource/0
7d49df4-233f-4898-92db-e6855d4dd94c?api-key=579b464db66ec23bdd000001cdd3946e44ce4aad7209ff7b23ac571b&format=json&offset=0&limit
=100'}, 'csv': {'CryptoMarkets': 'crypto-markets.csv'}}
```

	Name	Type	File Name
0	Pollution	api	https://api.openaq.org/v1/latest?country=IN&li...
1	Economy	api	https://api.data.gov.in/resource/07d49df4-233f...
2	CryptoMarkets	csv	crypto-markets.csv

How do we convert the dictionary into this Data Frame

QUANDATICS ACADEMY

How Dictionary works:

value

Dictionary = {'key': 'value'}

Nested Dictionary = {'key': '{'key': 'value'}'}

How Pandas works:

```
# append columns to an empty DataFrame
df['Name'] = ['Ankit', 'Ankita', 'Yashvardhan']
df['Articles'] = [97, 600, 200]
df['Improved'] = [2200, 75, 100]
```

	Name	Articles	Improved
0	Ankit	97	2200
1	Ankita	600	75
2	Yashvardhan	200	100

```
1  datasources = json.load(open('datasources_config.json'))['datasources']
2  # print(datasources)
3
4  LoadedDataSources = {}
5  for key, value in datasources.items():
6  #     print("Key:" ,key)
7  #     print("Values:" ,value)
8      for key1, value1 in datasources[key].items():
9  #         print("Nested Key:" ,key1)
10 #         print("Nested Values:" ,value1)
11 #         print("")
12         LoadedDataSources[key1] = {value1:key}
13 #     else:
14 #         print(LoadedDataSources)
```

Loop the Nested Dictionary, store in another order

QUANDATICS ACADEMY

Always keep your target in mind

	Name	Type	File Name
0	Pollution	api	https://api.openaq.org/v1/latest?country=IN&li...
1	Economy	api	https://api.data.gov.in/resource/07d49df4-233f...
2	CryptoMarkets	csv	crypto-markets.csv

```
16 # Create Empty Data Frame
17 DataSourceTable = pd.DataFrame()
18 # Create empty list for each column
19 DataSourceName = []
20 DataSourceType = []
21 DataSourceFile = []
22 # Loop the values into the list
23 for key, value in LoadedDataSources.items():
24     print("Key:", key)
25     print("Values:", value,"\n")
26     DataSourceName.append(key)
27     for key1, value1 in LoadedDataSources[key].items():
28         DataSourceFile.append(key1)
29         DataSourceType.append(value1)
30 # After completing the loop, append rows to the Data Frame columns
31 else:
32     DataSourceTable['Name'] = DataSourceName
33     DataSourceTable['Type'] = DataSourceType
34     DataSourceTable['File Name'] = DataSourceFile
35
36 # Show Result
37 DataSourceTable.head()
```

QUANDATICS ACADEMY

Add method into the class

```
def showDataSources(self):
    # Print all available data sources
    LoadedDataSources = {}
    for key, value in self.data_sources.items():
        for key1, value1 in self.data_sources[key].items():
            LoadedDataSources[key1] = {value1:key}


    DataSourceTable = pd.DataFrame()
    DataSourceName = []
    DataSourceType = []
    DataSourceFile = []
    for key, value in LoadedDataSources.items():
        DataSourceName.append(key)
        for key1, value1 in LoadedDataSources[key].items():
            DataSourceFile.append(key1)
            DataSourceType.append(value1)


    else:
        DataSourceTable['Name'] = DataSourceName
        DataSourceTable['Type'] = DataSourceType
        DataSourceTable['File Name'] = DataSourceFile

    return DataSourceTable
```

How do we use it?

QUANDATICS ACADEMY

Step 1: What Data Sources do we have?

Extract class for API

In [121]:
```
1 trainingDataSources.showDataSources()
```

Out[121]:

	Name	Type	File Name
0	Pollution	api	https://api.openaq.org/v1/latest?country=IN&li...
1	Economy	api	https://api.data.gov.in/resource/07d49df4-233f...
2	CryptoMarkets	csv	crypto-markets.csv

QUANDATICS ACADEMY

Step 2: Initiate the class

```
1 trainingDataSources = extract('datasources_config.json')
```

QUANDATICS ACADEMY

Step 3: Get the data

```
1  Pollution = trainingDataSources.getAPIsData("Pollution")
2  Economy = trainingDataSources.getAPIsData("Economy")
```

```
API URL: https://api.openaq.org/v1/latest?country=IN&limit=10000
API URL: https://api.data.gov.in/resource/07d49df4-233f-4898-92db-e6855d
7209ff7b23ac571b&format=json&offset=0&limit=100
```

Transform

Transform Economy Data

```
"records": [
    {
        "financial_year": "1951-52",
        "gross_domestic_product_in_rs_cr_at_2004_05_prices": "286147",
        "agriculture_allied_services_in_rs_cr_at_2004_05_prices": "147216",
        "agriculture_in_rs_cr_at_2004_05_prices": "118877",
        "industry_in_rs_cr_at_2004_05_prices": "47768",
        "mining_and_quarrying_in_rs_cr_at_2004_05_prices": "5772",
        "manufacturing_in_rs_cr_at_2004_05_prices": "25889",
        "services_in_rs_cr_at_2004_05_prices": "84799",
        "agri_culture_allied_services__share_to_total_gdp": "5145",
        "agriculture___share_to_total_gdp": "4154",
        "industry___share_to_total_gdp": "1669",
        "mining_and_quarrying___share_to_total_gdp": "202",
        "manufacturing___share_to_total_gdp": "905",
        "services___share_to_total_gdp": "2963",
        "gross_domestic_product___growth_rate_yoy_": "233",
        "agri_culture_allied_services___growth_rate_yoy_": "149",
        "agriculture___growth_rate_yoy_": "164",
        "industry___growth_rate_yoy_": "550",
        "mining_and_quarrying___growth_rate_yoy_": "1233",
        "manufacturing___growth_rate_yoy_": "316",
        "services___growth_rate_yoy_": "267"
    },
```


	gross_domestic_product_in_rs_cr_at_2004_05_prices	gross_domestic_product___growth_rate_yoy_	CalcGDPGrowth
0	286147	233	NaN
1	294267	284	0.028377
2	312177	609	0.060863
3	325431	425	0.042457
4	333766	256	0.025612

```
class transform():
    def __init__(self,data):
        self.data = data
    def gdpGrowth(self):
        self.df = pd.DataFrame(self.data["records"])
        self.GDP = self.df[['gross_domestic_product_in_rs_cr_at_2004_05_prices','gross_domestic_product___growth_rate_yoy_']
        self.GDP['CalcGDPGrowth'] = self.GDP['gross_domestic_product_in_rs_cr_at_2004_05_prices'].astype(str).astype(int).pc
        return self.GDP
```

```
Economy_df = transform(Economy).gdpGrowth()
Economy_df
```

	gross_domestic_product_in_rs_cr_at_2004_05_prices	gross_domestic_product___growth_rate_yoy_	CalcGDPGrowth
0	286147	233	NaN
1	294267	284	0.028377
2	312177	609	0.060863
3	325431	425	0.042457
4	333766	256	0.025612
5	352766	569	0.056926
6	348500	-121	-0.012093
7	374948	759	0.075891
8	383153	219	0.021883
9	410279	708	0.070797

QUANDATICS ACADEMY

Transform Pollution Data

QUANDATICS ACADEMY

Transform Pollution

```
In [46]: 1 pd.DataFrame(Pollution['results'])
```

Out[46]:

	location	city	country	coordinates	measurements
0	Burari Crossing, Delhi - IMD	Delhi	IN	{'latitude': 28.7256504, 'longitude': 77.2011573}	[{'parameter': 'pm25', 'value': 133.69, 'lastU...
1	Aya Nagar, Delhi - IMD	Delhi	IN	{'latitude': 28.4706914, 'longitude': 77.1099364}	[{'parameter': 'co', 'value': 0, 'lastUpdated'...
2	Sector - 62, Noida - IMD	Noida	IN	{'latitude': 28.6245479, 'longitude': 77.3577104}	[{'parameter': 'co', 'value': 4190, 'lastUpdat...
3	Lodhi Road, Delhi - IMD	Delhi	IN	{'latitude': 28.5918245, 'longitude': 77.2273074}	[{'parameter': 'pm10', 'value': 232.25, 'lastU...
4	IGI Airport (T3), Delhi - IMD	Delhi	IN	{'latitude': 28.5627763, 'longitude': 77.1180053}	[{'parameter': 'co', 'value': 3740, 'lastUpdat...
...	...	...	...	...	...
533	Opposite GPO Civil Lines Nagpur -MPCB	Nagpur	IN	{}	[{'parameter': 'pm10', 'value': 93.83, 'lastUp...
534	East Arjun Nagar	Delhi	IN	{}	[{'parameter': 'no2', 'value': 5.08, 'lastUpda...
535	Civil Lines	Delhi	IN	{'latitude': 28.6787, 'longitude': 77.2262}	[{'parameter': 'no2', 'value': 97.29, 'lastUpd...
536	IGI Airport	Delhi	IN	{'latitude': 28.56, 'longitude': 77.094}	[{'parameter': 'pm10', 'value': 1.05, 'lastUpd...
537	SPARTAN - IIT Kanpur	Kanpur	IN	{'latitude': 26.519, 'longitude': 80.233}	[{'parameter': 'pm25', 'value': 27.8, 'lastUpd...

538 rows × 5 columns

We cannot convert into pandas data frame directly!

QUANDATICS ACADEMY

Expected Output:

Out[152]:

	location	city	country	latitude	longitude	parameter	pollution	lastUpdated	sourceName	averagingPeriod
0	Aya Nagar, Delhi - IMD	Delhi	IN	28.470691	77.109936	pm10	278.12 µg/m³	2021-11-18T15:45:00Z	caaqm	3600 seconds
1	Sector - 62, Noida - IMD	Noida	IN	28.624548	77.357710	o3	27.34 µg/m³	2021-11-18T15:45:00Z	caaqm	3600 seconds
2	CRRI Mathura Road, Delhi - IMD	Delhi	IN	28.551201	77.273574	o3	19.4 µg/m³	2021-11-18T15:45:00Z	caaqm	3600 seconds
3	Lodhi Road, Delhi - IMD	Delhi	IN	28.591825	77.227307	pm25	142.12 µg/m³	2021-11-18T15:45:00Z	caaqm	3600 seconds
4	Pusa, Delhi - IMD	Delhi	IN	28.639645	77.146263	pm25	164.84 µg/m³	2021-11-18T15:45:00Z	caaqm	900 seconds
...	...	...	...	...	...	...	...	...	...	...
533	Opposite GPO Civil Lines Nagpur - MPCB	Nagpur	IN	NaN	NaN	o3	69.73 µg/m³	2016-04-05T10:30:00Z	CPCB	900 seconds
534	East Arjun Nagar	Delhi	IN	NaN	NaN	no2	5.08 µg/m³	2016-04-05T09:00:00Z	CPCB	900 seconds
535	Civil Lines	Delhi	IN	28.678700	77.226200	no2	97.29 µg/m³	2015-07-10T08:15:00Z	CPCB	900 seconds
536	IGI Airport	Delhi	IN	28.560000	77.094000	no2	39.18 µg/m³	2015-07-10T06:30:00Z	CPCB	900 seconds
537	SPARTAN - IIT Kanpur	Kanpur	IN	26.519000	80.233000	pm25	27.8 µg/m³	2014-09-26T00:30:00Z	Spartan	3600 seconds

538 rows × 10 columns

QUANDATICS ACADEMY

Hint:

```
1  pollutionResult = Pollution['results']
2  type(pollutionResult)
```

```
list
```

```
In [69]:
1  for data in pollutionResult:
2      print(type(data))
```

```
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
<class 'dict'>
```

```
air_dict['country'] = data['country']

air_dict['parameter'] = measurement['parameter']
```

QUANDATICS ACADEMY

Answer:

```
air_list = []
for data in pollutionResult:
    air_dict = {}
    air_dict['location'] = data['location']
    air_dict['city'] = data['city']
    air_dict['country'] = data['country']
    for key, value in data['coordinates'].items():
        air_dict[key] = value
    for measurement in data['measurements']:
        for key0, value0 in measurement.items():
            if key0 == 'averagingPeriod':
                avgperiod = []
                for key1, value1 in value0.items():
                    avgperiod.append(value1)
                else:
                    air_dict['averagingPeriod'] = str(avgperiod[0]) +" "+ avgperiod[1]
            elif (key0 == 'value' or key0 == 'unit'):
                air_dict['pollution'] = str(measurement['value']) +" "+ measurement['unit']
            else:
                air_dict[key0] = value0
    air_list.append(air_dict)

# Convert into pandas data frame
Pollution_df = pd.DataFrame(air_list, columns=air_dict.keys())
Pollution_df
```

QUANDATICS ACADEMY

Add into transform class as a method

```
def transformPollution(self):
    self.pollutionResult = self.data['results']
    air_list = []
    for data in self.pollutionResult:
        air_dict = {}
        air_dict['location'] = data['location']
        air_dict['city'] = data['city']
        air_dict['country'] = data['country']
        for key, value in data['coordinates'].items():
            air_dict[key] = value
        for measurement in data['measurements']:
            for key0, value0 in measurement.items():
                if key0 == 'averagingPeriod':
                    avgperiod = []
                    for key1, value1 in value0.items():
                        avgperiod.append(value1)
                    else:
                        air_dict['averagingPeriod'] = str(avgperiod[0]) +" "+ avgperiod[1]
                elif (key0 == 'value' or key0 == 'unit'):
                    air_dict['pollution'] = str(measurement['value']) +" "+ measurement['unit']
                else:
                    air_dict[key0] = value0
        air_list.append(air_dict)

    # Convert into pandas data frame
    self.Pollution_df = pd.DataFrame(air_list, columns=air_dict.keys())
    return self.Pollution_df
```

In [164]:
```
1 PollutionData = transform(Pollution).transformPollution()
2 PollutionData
```

Out[164]:

	location	city	country	latitude	longitude	parameter	pollution	lastUpdated	sourceName	averagingPeriod
0	Aya Nagar, Delhi - IMD	Delhi	IN	28.470691	77.109936	pm10	278.12 µg/m³	2021-11-18T15:45:00Z	caaqm	3600 seconds
1	Sector - 62, Noida - IMD	Noida	IN	28.624548	77.357710	o3	27.34 µg/m³	2021-11-18T15:45:00Z	caaqm	3600 seconds
2	CRRI Mathura Road, Delhi - IMD	Delhi	IN	28.551201	77.273574	o3	19.4 µg/m³	2021-11-18T15:45:00Z	caaqm	3600 seconds
3	Lodhi Road, Delhi - IMD	Delhi	IN	28.591825	77.227307	pm25	142.12 µg/m³	2021-11-18T15:45:00Z	caaqm	3600 seconds
4	Pusa, Delhi - IMD	Delhi	IN	28.639645	77.146263	pm25	164.84 µg/m³	2021-11-18T15:45:00Z	caaqm	900 seconds
...	...	...	...	...	...	...	...	...	...	...
533	Opposite GPO Civil Lines Nagpur - MPCB	Nagpur	IN	NaN	NaN	o3	69.73 µg/m³	2016-04-05T10:30:00Z	CPCB	900 seconds
534	East Arjun Nagar	Delhi	IN	NaN	NaN	no2	5.08 µg/m³	2016-04-05T09:00:00Z	CPCB	900 seconds
535	Civil Lines	Delhi	IN	28.678700	77.226200	no2	97.29 µg/m³	2015-07-10T08:15:00Z	CPCB	900 seconds
536	IGI Airport	Delhi	IN	28.560000	77.094000	no2	39.18 µg/m³	2015-07-10T06:30:00Z	CPCB	900 seconds
537	SPARTAN - IIT Kanpur	Kanpur	IN	26.519000	80.233000	pm25	27.8 µg/m³	2014-09-26T00:30:00Z	Spartan	3600 seconds

538 rows × 10 columns

QUANDATICS ACADEMY

List all methods in a class

```
1 method_list = [method for method in dir(transform) if method.startswith('_') is False]
2 print(method_list)
```

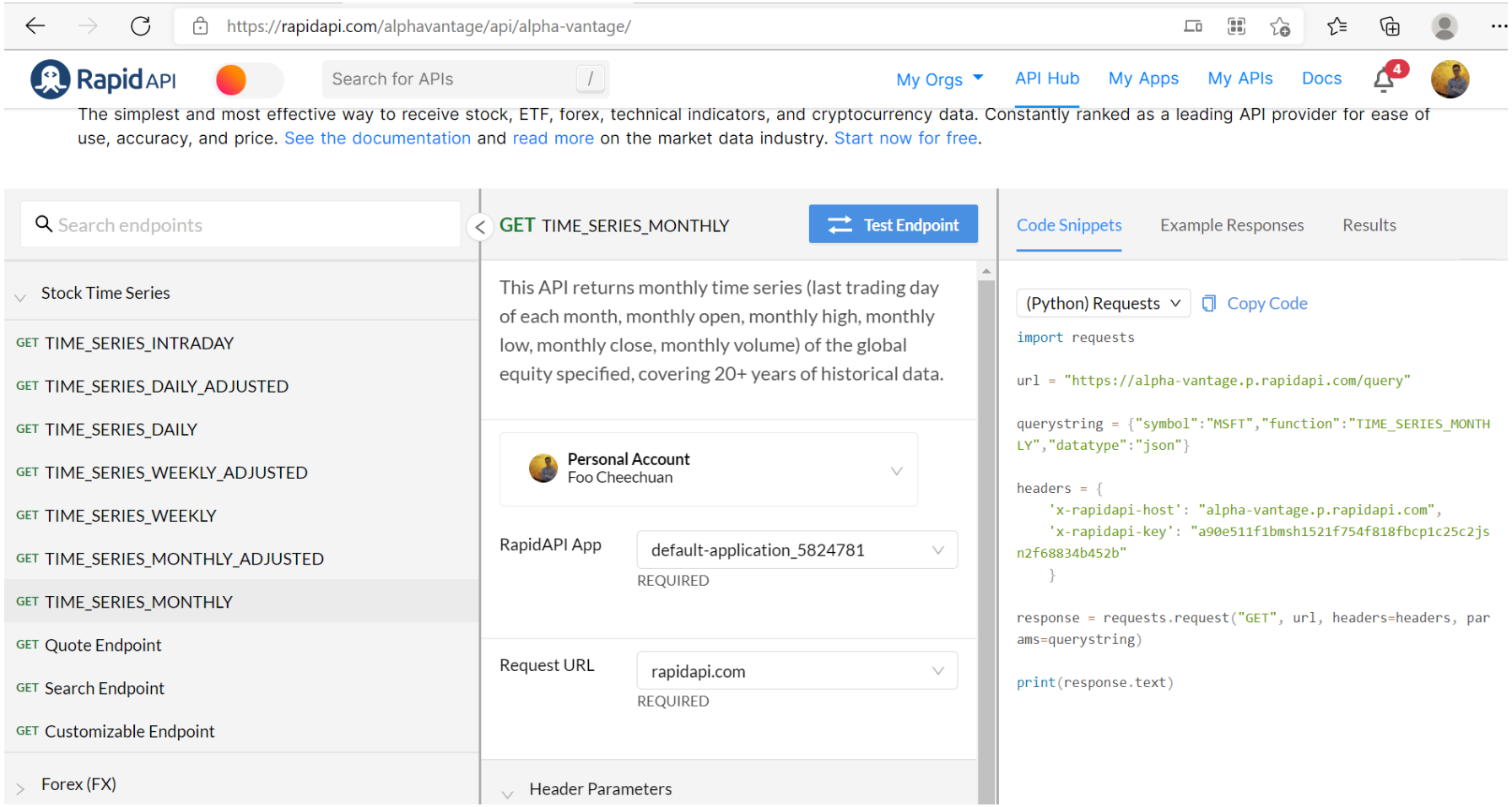
```
['gdpGrowth', 'transformPollution']
```

QUANDATICS ACADEMY

Retrieve Stock Prices, Write to CSV

QUANDATICS ACADEMY

Alpha Vantage API Documentation (alphavantage) | RapidAPI



Register to get apiKey

- Add a method in extract()
- When user insert a stock ticker, it will retrieve monthly data for that stock
- Convert into Pandas Data Frame

```
def stockPriceData(self,ticker):
    self.ticker = ticker
    import requests

    url = "https://alpha-vantage.p.rapidapi.com/query"

    querystring = {"symbol": self.ticker ,"function":"TIME_SERIES_MONTHLY","datatype":"json"}

    headers = {
        'x-rapidapi-host': "alpha-vantage.p.rapidapi.com",
        'x-rapidapi-key': "a90e511f1bmsh1521f754f818fbcp1c25c2jsn2f68834b452b"
        }

    response = requests.request("GET", url, headers=headers, params=querystring)
    monthlyPrices = response.json()['Monthly Time Series']
    histPrice = []

    for date,prices in monthlyPrices.items():
        priceDict = {}
        priceDict['Date'] = date
        for key, value in prices.items():
            priceDict[key] = value
        histPrice.append(priceDict)
    histPrice
    histPricedf = pd.DataFrame(histPrice)
    return histPricedf
```

- Add a method in transform()
- Write the pandas data frame into CSV without index
- Name the CSV in this format '{ticker}-{year}-Monthly-Stock-Price.csv'

In [353]:
```
1 aaplPrice = Extract.stockPriceData("AAPL")
2 transform(aaplPrice).transformStockPrice("AAPL")
```

Out[353]:

	Date	1. open	2. high	3. low	4. close	5. volume
263	1999-12-31	101.0000	118.0000	91.0600	102.8100	84091200

In [354]:
```
1 msftPrice = Extract.stockPriceData("MSFT")
2 transform(msftPrice).transformStockPrice("MSFT")
```

Out[354]:

	Date	1. open	2. high	3. low	4. close	5. volume
263	1999-12-31	91.0600	119.9400	90.8700	116.7500	630488900

In [355]:
```
1 amznPrice = Extract.stockPriceData("AMZN")
2 transform(amznPrice).transformStockPrice("AMZN")
```

Out[355]:

	Date	1. open	2. high	3. low	4. close	5. volume
263	1999-12-31	87.2500	113.0000	76.0000	76.1300	246476100

QUANDATICS ACADEMY

Date	1. open	2. high	3. low	4. close	5. volume
12/29/2017	1172.05	1194.78	1124.74	1169.47	56130525
11/30/2017	1105.4	1213.41	1086.87	1176.75	76272767
10/31/2017	964	1122.79	950.37	1105.28	80304293
9/29/2017	984.2	1000	931.75	961.35	58038449
8/31/2017	990.65	1003.21	936.33	980.6	75779478
7/31/2017	972.79	1083.31	951.0001	987.78	76564735
6/30/2017	998.59	1017	927	968	94729701
5/31/2017	927.8	1001.2	927.8	994.62	75223469
4/28/2017	888	949.59	884.49	924.99	72336605
3/31/2017	853.05	890.35	833.5	886.54	60722752
2/28/2017	829.21	860.86	803	845.04	71748231
1/31/2017	757.92	843.84	747.7	823.48	70714587

Sample Code

```
def transformStockPrice(self,ticker):
    # Get unique years
    priceDate = self.data['Date'].tolist()
    year = []
    for data in priceDate:
        substring = data[0:4]
        year.append(substring)
    uniqueYear = list(dict.fromkeys(year))
    # uniqueYearList

    # Create csv for each year
    for year in uniqueYear:
        self.subset = self.data.loc[msftPrice['Date'].str.contains(year)]
        self.subset.to_csv(f'writeCSV/{ticker}-{year}-Monthly-Stock-Price.csv',index=False)
    return self.subset
```

QUANDATICS ACADEMY

Add folder path

```
{
    "datasources":{
        "api":{
            "TestPollution": "https://api.openaq.org/v1/latest?country=IN&limit=10000",
            "TestEconomy": "https://api.data.gov.in/resource/07d49df4-233f-4898-92db-e6855d4dd94c?api-key=579b464db66
        },
        "csv":{
            "CryptoMarkets": "crypto-markets.csv"
            "StockPrices": "writeCSV"
        }
    }
}
```

QUANDATICS ACADEMY

Load the name of all files in the given path into a list

```
1  # Loop folder
2  import os
3  import re
4
5  # Read filenames from the given path
6  data_files = os.listdir('writeCSV')
7  data_files
```

QUANDATICS ACADEMY

Define a Generator to loop all files in a specific folder

```
def load_files(filenames):
    for filename in filenames:
        yield pd.read_csv(f'Data/All/{filename}')
```

QUANDATICS ACADEMY

Define a Generator to loop all files in a specific folder

```
1  def load_files(filenames):
2      for filename in filenames:
3          yield pd.read_csv(f'writeCSV/{filename}')
```

QUANDATICS ACADEMY

Read files with certain pattern

In [436]:

```
def read_data():
    pattern = re.compile(f'^....-....-Monthly-Stock-Price')
    data = [x for x in data_files if pattern.match(x)]
    # Concatenate data frames generated by load_files
    stockPrice = pd.concat(load_files(data))
    # Reset index
    stockPrice.reset_index(drop=True, inplace=True)
    # Removes last column
    stockPrice = stockPrice.iloc[: , :-1]
    return stockPrice
```

QUANDATICS ACADEMY

Regex Cheat Sheet

Anchors

^	Start of string, or start of line in multi-line pattern
\A	Start of string
$	End of string, or end of line in multi-line pattern
\Z	End of string
\b	Word boundary
\B	Not word boundary
\<	Start of word
\>	End of word

Character Classes

\c	Control character
\s	White space

Quantifiers

*	0 or more	{3}	Exactly 3
+	1 or more	{3,}	3 or more
?	0 or 1	{3,5}	3, 4 or 5

Add a ? to a quantifier to make it ungreedy.

Escape Sequences

\	Escape following character
\Q	Begin literal sequence
\E	End literal sequence

"Escaping" is a way of treating characters which have a special meaning in regular expressions literally, rather than as special characters.

Groups and Ranges

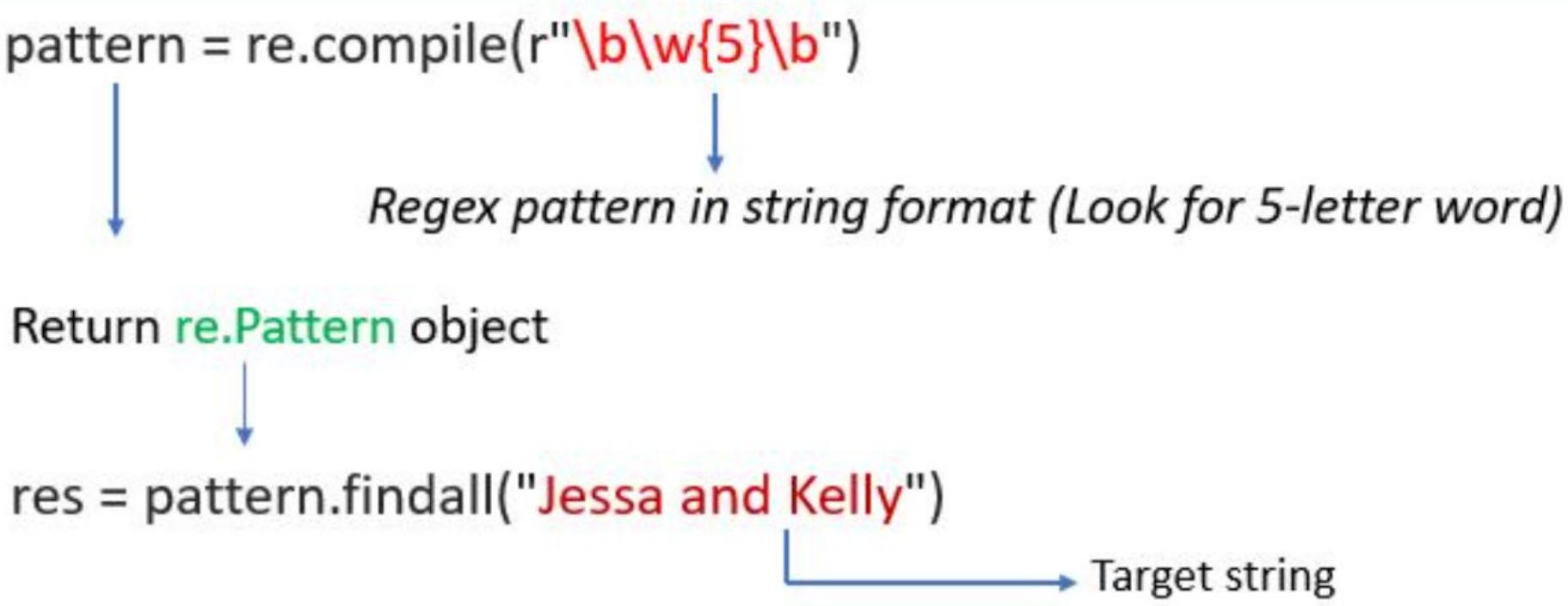
.	Any character except new line (\n)
(a|b)	a or b
(...)	Group
(?:...)	Passive (non-capturing) group
[abc]	Range (a or b or c)
[^abc]	Not (a or b or c)
[a-q]	Lower case letter from a to q
[A-Q]	Upper case letter from A to Q
[0-7]	Digit from 0 to 7
\x	Group/subpattern number "x"

Ranges are inclusive.

Pattern Modifiers

QUANDATICS ACADEMY

re.compile



Result: 2 matches [Jessa, Kelly]

Using regular expression methods

The “re” package provides several methods to actually perform queries on an input string. We will see the methods of re in Python:

- re.match()
- re.search()
- re.findall()

```
pattern = re.compile(f'^....-.....-Monthly-Stock-Price')

data = [x for x in data_files if pattern.match(x)]
```

Generate file names in data_files

```
patternYear = re.compile(f'^ssc8_{year}-..-..\.csv$')
```

```
[x for x in data_files if patternYear.match(x)]
```

Generate file names in data_files

If file name matches the pattern

```
1  paTTern = re.compile(f'^ssc8_{year}-..-..\.csv$')
2  print(type(a))
3
4  [x for x in data_files]
```

```
<class 're.Pattern'>

['ssc8_2020-09-09.csv',
 'ssc8_2020-09-10.csv',
 'ssc8_2020-09-11.csv',
 'ssc8_2020-09-12.csv',
 'ssc8_2020-09-13.csv',
```

Generate file names in data_files

QUANDATICS ACADEMY

List Comprehension

Create a list to store the squares of numbers (x_i^2)

```
1  # Create a list
2  squares = []
3  for i in range(1,11):
4      squares.append(i**2)
5
6  print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

[*expression* for *value* in *collection*]

```
In [20]:
1  squareComp = [i**2 for i in range(1,11)]
2  print(squareComp)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

expression : ***i**2***

value : ***i***

collection : ***range(1,11)***

[*expression* for *value* in *collection* if <*test*>]

```
1  numbers = [i for i in range(1,21)]
2  print("The range given is:",numbers)
3
4  # Conventional approach
5  even_num = []
6  for num in numbers:
7      if num%2==0:
8          even_num.append(num)
9  print("Even numbers in the range given are:",even_num)
10
11 # List comprehension approach
12 even_numComp = [num for num in numbers if num%2==0]
13 print("List Comprehension:",even_numComp)
```

```
The range given is: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
Even numbers in the range given are: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
List Comprehension: [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```
paTTern = re.compile(f'^ssc8_{year}-..-..\.csv$')

# paTTern.match = None when doesn't match
print(paTTern.match("abcd1234"))

# paTTern.match will return a match object when matched
print(paTTern.match("ssc8_2020-09-09.csv"))

# re.Match class type
print(type(paTTern.match("ssc8_2020-09-09.csv")))
```

```
None
<re.Match object; span=(0, 19), match='ssc8_2020-09-09.csv'>
<class 're.Match'>
```

```
1  # Return true when matched, False when not matched
2  pattern = re.compile(f'^....-....-Monthly-Stock-Price')
3  print(bool(pattern.match("abcd-1234-Monthly-Stock-Price")))
4  print(bool(pattern.match("abcd1234")))
5
6  if pattern.match("abcd-1234-Monthly-Stock-Price"):
7      print("Yay! It matched")
8  else:
9      print("Oops! Doesn't match")
```

→ True
False

→ Yay! It matched

```
True
False
Yay! It matched
```

```
print("Loaded", len(data),'files')
print("Total Rows =", len(stockPrice))
columns = []
for column in stockPrice:
    columns.append(column)
print("Total Columns =", len(columns))
return stockPrice
```

← **Check Number of Files, Rows & Columns**

```
Loaded 69 files
Total Rows = 792
Total Columns = 8
```

Out[459]:

	Date	1. open	2. high	3. low	4. close	5. volume	Ticker	Year
0	1999-12-31	101.00	118.00	91.0600	102.81	84091200	AAPL	1999
1	2000-12-29	17.00	17.50	13.6300	14.88	158195900	AAPL	2000
2	2000-11-30	19.44	23.00	16.1200	16.50	151578100	AAPL	2000
3	2000-10-31	26.69	26.75	17.5000	19.56	391174800	AAPL	2000
4	2000-09-29	61.31	64.12	25.3700	25.75	259230900	AAPL	2000
...	...	...	...	...	...	...	...	...
787	2021-05-28	253.40	254.35	238.0700	249.68	495089530	MSFT	2021
788	2021-04-30	238.47	263.19	238.0501	252.18	568698272	MSFT	2021
789	2021-03-31	235.90	241.05	224.2600	235.77	724981506	MSFT	2021
790	2021-02-26	235.06	246.13	227.8800	232.38	489396912	MSFT	2021
791	2021-01-29	222.53	242.64	211.9400	231.96	647897096	MSFT	2021

792 rows × 8 columns

QUANDATICS ACADEMY

loopcsv
Sample Code

```
def loopcsv(self):
    # Loop folder
    import os
    import re

    # Read filenames from the given path
    data_files = os.listdir('writeCSV')
    data_files
    def load_files(filenames):
        for filename in filenames:
            df = pd.read_csv(f'writeCSV/{filename}')
            csvTicker = filename[0:4]
            df["Ticker"] = csvTicker
            csvYear = filename[5:9]
            df["Year"] = csvYear
            yield df
    pattern = re.compile(f'^....-....-Monthly-Stock-Price')
    data = [x for x in data_files if pattern.match(x)]
    # Concatenate data frames generated by load_files
    stockPrice = pd.concat(load_files(data))
    # Reset index
    stockPrice.reset_index(drop=True, inplace=True)
    # Removes last column
    stockPrice = stockPrice.iloc[: , :-1]
    print("Loaded", len(data),'files')
    print("Total Rows =", len(stockPrice))
    columns = []
    for column in stockPrice:
        columns.append(column)
    print("Total Columns =", len(columns))
    return stockPrice
```

QUANDATICS ACADEMY

loopcsv
Sample Output

In [459]:
```
1 Extract = extract('datasources_config.json')
2 stockPrice = Extract.loopcsv()
3 stockPrice
```

```
Loaded 69 files
Total Rows = 792
Total Columns = 8
```

Out[459]:

	Date	1. open	2. high	3. low	4. close	5. volume	Ticker	Year
0	1999-12-31	101.00	118.00	91.0600	102.81	84091200	AAPL	1999
1	2000-12-29	17.00	17.50	13.6300	14.88	158195900	AAPL	2000
2	2000-11-30	19.44	23.00	16.1200	16.50	151578100	AAPL	2000
3	2000-10-31	26.69	26.75	17.5000	19.56	391174800	AAPL	2000
4	2000-09-29	61.31	64.12	25.3700	25.75	259230900	AAPL	2000
...	...	...	...	...	...	...	...	...
787	2021-05-28	253.40	254.35	238.0700	249.68	495089530	MSFT	2021
788	2021-04-30	238.47	263.19	238.0501	252.18	568698272	MSFT	2021
789	2021-03-31	235.90	241.05	224.2600	235.77	724981506	MSFT	2021
790	2021-02-26	235.06	246.13	227.8800	232.38	489396912	MSFT	2021
791	2021-01-29	222.53	242.64	211.9400	231.96	647897096	MSFT	2021

792 rows × 8 columns

QUANDATICS ACADEMY

Final Code

QUANDATICS ACADEMY

extract()

```
class extract:
    # Load configuration at initiation
    def __init__(self,configfile):
        import pandas as pd
        import requests # For APIs
        import json # For config file
        self.configfile = configfile
        self.data_sources = json.load(open(self.configfile))['datasources']
        self.api = self.data_sources['api']
        self.csv_path = self.data_sources['csv']
```

```
    def showDataSources(self):
        # Print all available data sources
        LoadedDataSources = {}
        for key, value in self.data_sources.items():
            for key1, value1 in self.data_sources[key].items():
                LoadedDataSources[key1] = {value1:key}


        DataSourceTable = pd.DataFrame()
        DataSourceName = []
        DataSourceType = []
        DataSourceFile = []
        for key, value in LoadedDataSources.items():
            DataSourceName.append(key)
            for key1, value1 in LoadedDataSources[key].items():
                DataSourceFile.append(key1)
                DataSourceType.append(value1)


        else:
            DataSourceTable['Name'] = DataSourceName
            DataSourceTable['Type'] = DataSourceType
            DataSourceTable['File Name'] = DataSourceFile

        return DataSourceTable
```

```
def getAPIsData(self,api_name):
    # Initiate Parameter
    self.api_name = api_name
    # We have multiple APIs
    # Use their name to identify
    api_url = self.api[self.api_name]
    # Show api name when connected
    print("API URL:",api_url)
    response = requests.get(api_url)
        # response.json() will convert json data into Python dictionary
    return response.json()

def getCSVData(self,csv_name):
    # Similarly, use csv name as identifier
    df = pd.read_csv(self.csv_path[csv_name])
    return df
```

```
def stockPriceData(self,ticker):
    self.ticker = ticker
    import requests

    url = "https://alpha-vantage.p.rapidapi.com/query"

    querystring = {"symbol": self.ticker ,"function":"TIME_SERIES_MONTHLY","datatype":"json"}

    headers = {
        'x-rapidapi-host': "alpha-vantage.p.rapidapi.com",
        'x-rapidapi-key': "a90e511f1bmsh1521f754f818fbcp1c25c2jsn2f68834b452b"
        }

    response = requests.request("GET", url, headers=headers, params=querystring)
    monthlyPrices = response.json()['Monthly Time Series']
    histPrice = []

    for date,prices in monthlyPrices.items():
        priceDict = {}
        priceDict['Date'] = date
        for key, value in prices.items():
            priceDict[key] = value
        histPrice.append(priceDict)
    histPrice
    histPricedf = pd.DataFrame(histPrice)
    return histPricedf
```

```
def loopcsv(self):
    # Loop folder
    import os
    import re

    # Read filenames from the given path
    data_files = os.listdir('writeCSV')
    data_files
    def load_files(filenames):
        for filename in filenames:
            df = pd.read_csv(f'writeCSV/{filename}')
            csvTicker = filename[0:4]
            df["Ticker"] = csvTicker
            csvYear = filename[5:9]
            df["Year"] = csvYear
            yield df
    pattern = re.compile(f'^....-....-Monthly-Stock-Price')
    data = [x for x in data_files if pattern.match(x)]
    # Concatenate data frames generated by load_files
    stockPrice = pd.concat(load_files(data))
    # Reset index
    stockPrice.reset_index(drop=True, inplace=True)
    # Removes last column
    stockPrice = stockPrice.iloc[: , :-1]
    print("Loaded", len(data),'files')
    print("Total Rows =", len(stockPrice))
    columns = []
    for column in stockPrice:
        columns.append(column)
    print("Total Columns =", len(columns))
    return stockPrice
```

QUANDATICS ACADEMY

transform()

```
class transform():
    def __init__(self,data):
        self.data = data
    def gdpGrowth(self):
        self.df = pd.DataFrame(self.data["records"])
        self.GDP = self.df[['gross_domestic_product_in_rs_cr_at_2004_05_prices',
                            'gross_domestic_product___growth_rate_yoy_']]
        self.GDP['CalcGDPGrowth'] = self.GDP['gross_domestic_product_in_rs_cr_at_2004_05_prices']\
        .astype(str).astype(int).pct_change()
        return self.GDP
```

```
def transformPollution(self):
    self.pollutionResult = self.data['results']
    air_list = []
    for data in self.pollutionResult:
        air_dict = {}
        air_dict['location'] = data['location']
        air_dict['city'] = data['city']
        air_dict['country'] = data['country']
        for key, value in data['coordinates'].items():
            air_dict[key] = value
        for measurement in data['measurements']:
            for key0, value0 in measurement.items():
                if key0 == 'averagingPeriod':
                    avgperiod = []
                    for key1, value1 in value0.items():
                        avgperiod.append(value1)
                    else:
                        air_dict['averagingPeriod'] = str(avgperiod[0]) +" "+ avgperiod[1]
                elif (key0 == 'value' or key0 == 'unit'):
                    air_dict['pollution'] = str(measurement['value']) +" "+ measurement['unit']
                else:
                    air_dict[key0] = value0
        air_list.append(air_dict)

    # Convert into pandas data frame
    self.Pollution_df = pd.DataFrame(air_list, columns=air_dict.keys())
    return self.Pollution_df
```

```python
def transformStockPrice(self,ticker):
    # Get unique years
    priceDate = self.data['Date'].tolist()
    year = []
    for data in priceDate:
        substring = data[0:4]
        year.append(substring)
    uniqueYear = list(dict.fromkeys(year))
    # uniqueYearList

    # Create csv for each year
    for year in uniqueYear:
        self.subset = self.data.loc[msftPrice['Date'].str.contains(year)]
        self.subset.to_csv(f'writeCSV/{ticker}-{year}-Monthly-Stock-Price.csv',index=False)
    return self.subset
```

QUANDATICS ACADEMY

Thank you