Skills Network

Collecting Job Data Using APIs

Estimated time needed: **45 to 60** minutes

Objectives

After completing this lab, you will be able to:

- Collect job data from Jobs API
- Store the collected data into an excel spreadsheet.

> **Note: Before starting with the assignment make sure to read all the instructions and then move ahead with the coding part.**

Instructions

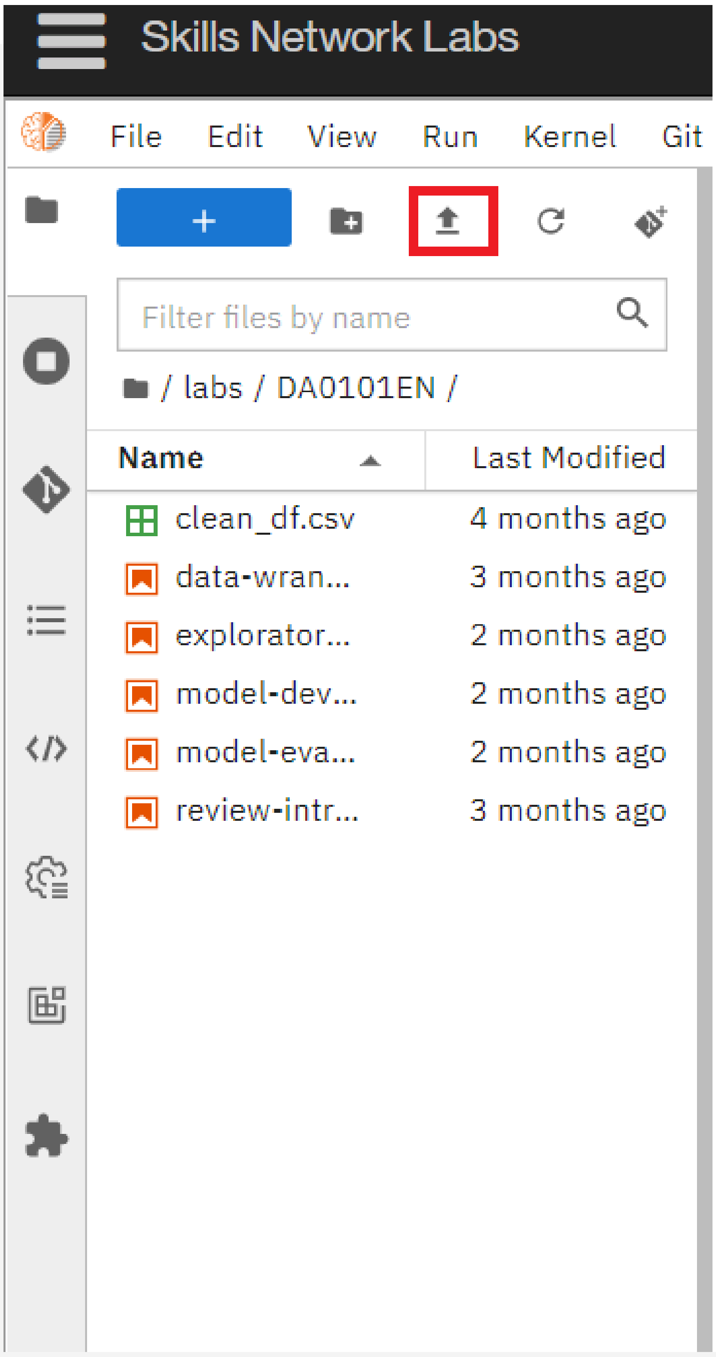
To run the actual lab, firstly you need to click on the Jobs_API notebook link. The file contains flask code which is required to run the Jobs API data.

Now, to run the code in the file that opens up follow the below steps.

Step1: Download the file.

Step2: Upload it on the IBM Watson studio. (If IBM Watson Cloud service does not work in your system, follow the alternate Step 2 below)

Step2(alternate): Upload it in your SN labs environment using the upload button which is highlighted in red in the image below: Remember to upload this Jobs_API file in the same folder as your current .ipynb file



Step3: Run all the cells of the Jobs_API file. (Even if you receive an asterik sign after running the last cell, the code works fine.)

If you want to learn more about flask, which is optional, you can click on this link here.

Once you run the flask code, you can start with your assignment.

```
In [ ]: !pip install flask
        !wget  https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-D
        import flask
        from flask import request, jsonify
        import requests
        import re

        def get_data(key,value,current):
            results = list()
            pattern_dict = {
                'C'      : '(C)',
                'C++'    : '(C\+\+)',
                'Java'   :'(Java)',
                'C#'     : '(C\#)',
                'Python' :'(Python)',
                'Scala' : '(Scala)',
                'Oracle' : '(Oracle)',
                'SQL Server': '(SQL Server)',
                'MySQL Server' :'(MySQL Server)',
                'PostgreSQL':'(PostgreSQL)',
                'MongoDB'    : '(MongoDB)',
                'JavaScript'    : '(JavaScript)',
                'Los Angeles' :'(Los Angeles)',
                'New York':'(New York)',
                'San Francisco':'(San Francisco)',
                'Washington DC':'(Washington DC)',
                'Seattle':'(Seattle)',
                'Austin':'(Austin)',
                'Detroit':'(Detroit)',





            }
            for rec in current:
                print(rec[key])
                print(type(rec[key]))
                print(rec[key].find(value))
                #if rec[key].find(value) != -1:
                import re
                #reex_str = """(C)|(C\+\+)|(JavaScript)|(Java)|(C\#)|(Python)|(Scala)|((
                if re.search(pattern_dict[value],rec[key]) != None:
                    results.append(rec)
            return results

        app = flask.Flask(__name__)

        import json
        data = None
        with open('jobs.json',encoding='utf-8') as f:
            # returns JSON object as
            # a dictionary
            data = json.load(f)



        @app.route('/', methods=['GET'])
        def home():

            return '''<h1>Welcome to flask JOB search API</p>'''


        @app.route('/data/all', methods=['GET'])
        def api_all():
            return jsonify(data)


        @app.route('/data', methods=['GET'])
        def api_id():
            # Check if keys such as Job Title,KeySkills, Role Category and others  are p
            #  Assign the keys to the corresponding variables..
            # If no key is provided, display an error in the browser.
            res = None
            for req in request.args:

                if req == 'Job Title':
                    key = 'Job Title'
                elif req == 'Job Experience Required' :
                    key='Job Experience Required'
                elif req == 'Key Skills' :
                    key='Key Skills'

                elif req == 'Role Category' :
                    key='Role Category'
                elif req == 'Location' :
                    key='Location'

                elif req == 'Functional Area' :
                    key='Functional Area'

                elif req == 'Industry' :
                    key='Industry'
                elif req == 'Role' :
                    key='Role'
                elif req=="id":
                     key="id"
                else:
                    pass

                value = request.args[key]
                if (res==None):
                    res = get_data(key,value,data)
                else:
                    res = get_data(key,value,res)

            # Use the jsonify function from Flask to convert our list of
            # Python dictionaries to the JSON format.
            return jsonify(res)

        app.run()
```

Dataset Used in this Assignment

The dataset used in this lab comes from the following source: https://www.kaggle.com/promptcloud/jobs-on-naukricom under the under a **Public Domain license**.

> Note: We are using a modified subset of that dataset for the lab, so to follow the lab instructions successfully please use the dataset provided with the lab, rather than the dataset from the original source.

The original dataset is a csv. We have converted the csv to json as per the requirement of the lab.

Warm-Up Exercise

Before you attempt the actual lab, here is a fully solved warmup exercise that will help you to learn how to access an API.

Using an API, let us find out who currently are on the International Space Station (ISS).
The API at http://api.open-notify.org/astros.json gives us the information of astronauts currently on ISS in json format.
You can read more about this API at http://open-notify.org/Open-Notify-API/People-In-Space/

In [1]:
```
import requests # you need this module to make an API call
import pandas as pd
```

In [2]:
```
api_url = "http://api.open-notify.org/astros.json" # this url gives use the astr
```

In [3]:
```
response = requests.get(api_url) # Call the API using the get method and store t
                                 # output of the API call in a variable called re
```

In [4]:
```
if response.ok:                  # if all is well() no errors, no network timeouts)
    data = response.json()  # store the result in json format in a variable cal
                            # the variable data is of type dictionary.
```

In [ ]:

In [5]:
```
print(data)   # print the data just to check the output or for debugging
```

```
{'message': 'success', 'number': 10, 'people': [{'craft': 'ISS', 'name': 'Serge
y Prokopyev'}, {'craft': 'ISS', 'name': 'Dmitry Petelin'}, {'craft': 'ISS', 'na
me': 'Frank Rubio'}, {'craft': 'Shenzhou 15', 'name': 'Fei Junlong'}, {'craft':
'Shenzhou 15', 'name': 'Deng Qingming'}, {'craft': 'Shenzhou 15', 'name': 'Zhan
g Lu'}, {'craft': 'ISS', 'name': 'Stephen Bowen'}, {'craft': 'ISS', 'name': 'Wa
rren Hoburg'}, {'craft': 'ISS', 'name': 'Sultan Alneyadi'}, {'craft': 'ISS', 'n
ame': 'Andrey Fedyaev'}]}
```

Print the number of astronauts currently on ISS.

In [6]:
```
print(data.get('number'))
```

```
10
```

Print the names of the astronauts currently on ISS.

In [7]:
```
astronauts = data.get('people')
print("There are {} astronauts on ISS".format(len(astronauts)))
print("And their names are :")
for astronaut in astronauts:
    print(astronaut.get('name'))
```

```
There are 10 astronauts on ISS
And their names are :
Sergey Prokopyev
Dmitry Petelin
Frank Rubio
Fei Junlong
Deng Qingming
Zhang Lu
Stephen Bowen
Warren Hoburg
Sultan Alneyadi
Andrey Fedyaev
```

Hope the warmup was helpful. Good luck with your next lab!

Lab: Collect Jobs Data using Jobs API

Objective: Determine the number of jobs currently open for various technologies and for various locations

Collect the number of job postings for the following locations using the API:

- Los Angeles
- New York
- San Francisco
- Washington DC
- Seattle
- Austin
- Detroit

In [42]:
```
#Import required libraries
import pandas as pd
import json
```

Write a function to get the number of jobs for the Python technology.

> Note: While using the lab you need to pass the **payload** information for the **params** attribute in the form of **key value** pairs.

Refer the ungraded **rest api lab** in the course **Python for Data Science, AI & Development** link

The keys in the json are

- Job Title
- Job Experience Required
- Key Skills
- Role Category
- Location
- Functional Area
- Industry
- Role

You can also view the json file contents from the following json URL.

In [43]:
```
api_url="http://127.0.0.1:5000/data"
def get_number_of_jobs_T(technology):
    payload={"Key Skills": technology}
    response=requests.get(api_url, params=payload)
    if response.ok:
        data=response.json()
```

```
#         print(data)
        number_of_jobs = len(data)

    return technology,number_of_jobs
```

Calling the function for Python and checking if it works.

In [32]:
```
get_number_of_jobs_T("Python")
```

Out[32]:
```
('Python', 1173)
```

Write a function to find number of jobs in US for a location of your choice

In [44]:
```
locations=["Los Angeles", "New York", "San Francisco", "Washington DC", "Seattle

def get_number_of_jobs_Loc_list(locations):
    number_of_jobs_list = []
    for location in locations:
        payload={"Location":location}
        response=requests.get(api_url, params=payload)
        if response.ok:
            data=response.json()
            number_of_jobs = len(data)
            number_of_jobs_list.append({location: number_of_jobs})

    return number_of_jobs_list
```

Call the function for Los Angeles and check if it is working.

In [35]:
```
#your code goes here
get_number_of_jobs_Loc_list(locations)
```

Out[35]:
```
[{'Los Angeles': 640},
 {'New York': 3226},
 {'San Francisco': 435},
 {'Washington DC': 5316},
 {'Seattle': 3375}]
```

Store the results in an excel file

Call the API for all the given technologies above and write the results in an excel spreadsheet.

If you do not know how create excel file using python, double click here for **hints**.

Create a python list of all locations for which you need to find the number of jobs postings.

In [45]:
```
#your code goes here
countries = ['Los Angeles', 'New York', 'San Francisco', 'Washington DC', 'Seat
```

Import libraries required to create excel spreadsheet

In [46]:
```
# your code goes here
from openpyxl import Workbook
```

Create a workbook and select the active worksheet

```
In [47]: # your code goes here
         wb=Workbook()
         ws=wb.active
         ws.append(countries)
```

Find the number of jobs postings for each of the location in the above list. Write the Location name and the number of jobs postings into the excel spreadsheet.

```
In [48]: #your code goes here
         def get_number_of_jobs_TL(technology, countries):
             number_of_jobs_list = []
             for location in countries:
                 payload={"Key Skills": technology, "Location": location}
                 response=requests.get(api_url, params=payload)
                 if response.ok:
                     data=response.json()
                     number_of_jobs = len(data)
                     number_of_jobs_list.append(number_of_jobs)
             return number_of_jobs_list
         #     return ws.append(number_of_jobs_list)

         get_number_of_jobs_TL("Python", countries)

Out[48]: [24, 143, 17, 258, 133, 15, 170]
```

Save into an excel spreadsheet named 'job-postings.xlsx'.

```
In [ ]: #your code goes here
        wb.save('job-postings.xlsx')
```

In the similar way, you can try for below given technologies and results can be stored in an excel sheet.

Collect the number of job postings for the following languages using the API:

- C
- C#
- C++
- Java
- JavaScript
- Python
- Scala
- Oracle
- SQL Server
- MySQL Server
- PostgreSQL
- MongoDB

```
In [ ]: # your code goes here
```

Author

Ayushi Jain

Other Contributors

Rav Ahuja

Lakshmi Holla

Malika

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-01-19	0.3	Lakshmi Holla	Added changes in the markdown
2021-06-25	0.2	Malika	Updated GitHub job json link
2020-10-17	0.1	Ramesh Sannareddy	Created initial version of the lab