Collecting Job Data Using APIs

Estimated time needed: **30** minutes

Objectives

After completing this lab, you will be able to:

- Collect job data using 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

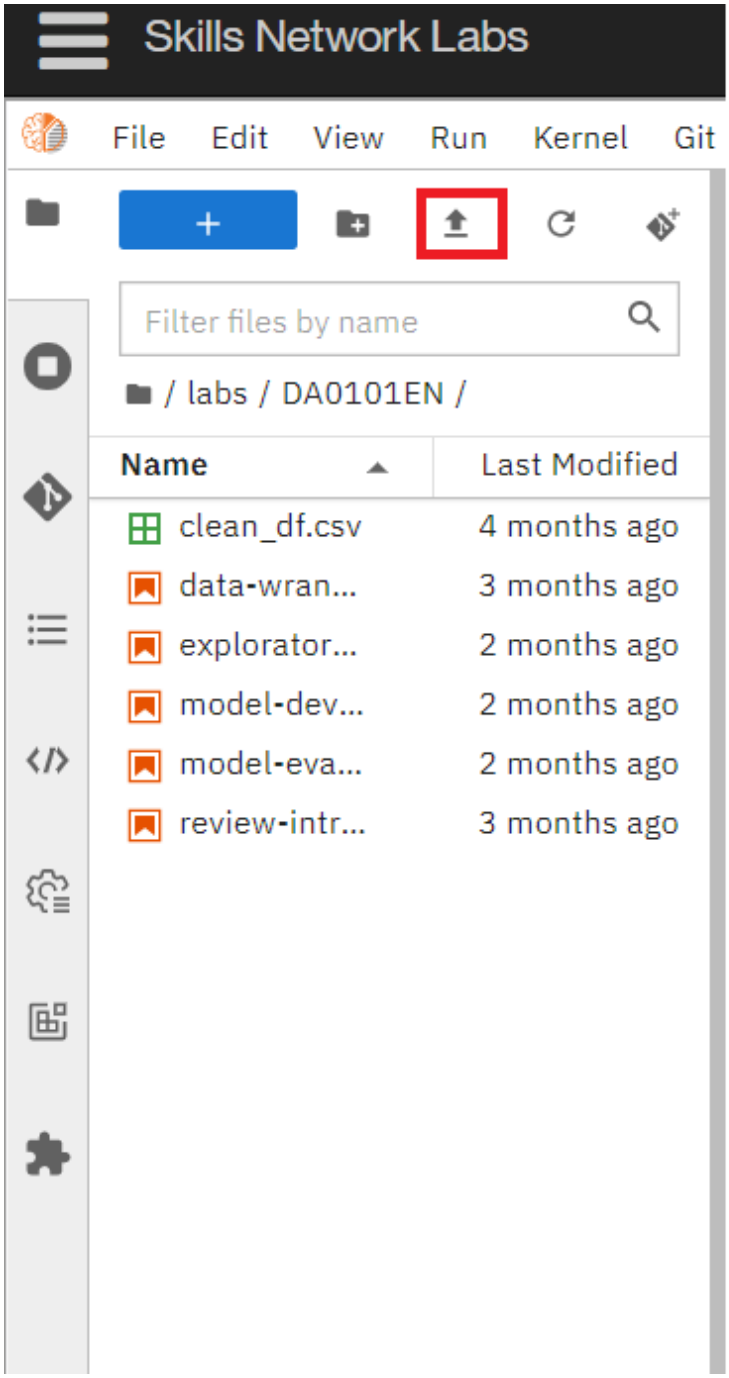
To run the actual lab, firstly you need to click on the Jobs_API notebook link. The file contains flask code which is required to run the Jobs API data.

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

Step1: Download the file.

Step2: Upload the file into your current Jupyter environment using the upload button in your Jupyter interface. Ensure that the file is in the same folder as your working .ipynb file.

Step 2: If working in a local Jupyter environment, use the "Upload" button in your Jupyter interface to upload the Jobs_API notebook into the same folder as your current .ipynb file.



Step3: Open the Jobs_API notebook, and run all the cells to start the Flask application. Once the server is running, you can access the API from the URL provided in the notebook.

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.

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 us
```

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

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

In [10]:
```
print (data)  # print the data just to check the output or for debuggi
```

```
{'people': [{'craft': 'ISS', 'name': 'Oleg Kononenko'}, {'craft': 'IS
S', 'name': 'Nikolai Chub'}, {'craft': 'ISS', 'name': 'Tracy Caldwell D
yson'}, {'craft': 'ISS', 'name': 'Matthew Dominick'}, {'craft': 'ISS',
'name': 'Michael Barratt'}, {'craft': 'ISS', 'name': 'Jeanette Epps'},
{'craft': 'ISS', 'name': 'Alexander Grebenkin'}, {'craft': 'ISS', 'nam
e': 'Butch Wilmore'}, {'craft': 'ISS', 'name': 'Sunita Williams'}, {'cr
aft': 'Tiangong', 'name': 'Li Guangsu'}, {'craft': 'Tiangong', 'name':
'Li Cong'}, {'craft': 'Tiangong', 'name': 'Ye Guangfu'}], 'number': 12,
'message': 'success'}
```

Print the number of astronauts currently on ISS.

In [11]:
```
print(data.get('number'))
```

```
12
```

Print the names of the astronauts currently on ISS.

In [12]:
```
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 12 astronauts on ISS
And their names are :
Oleg Kononenko
Nikolai Chub
Tracy Caldwell Dyson
Matthew Dominick
Michael Barratt
Jeanette Epps
Alexander Grebenkin
Butch Wilmore
Sunita Williams
Li Guangsu
Li Cong
Ye Guangfu
```

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 [13]:
```
#Import required libraries
import pandas as pd
import json
```

https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/jobs.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 [32]:
```
api_url="https://cf-courses-data.s3.us.cloud-object-storage.appdomain.

def get_number_of_jobs(technology):
    number_of_jobs = 0
    #your code goes here
    for tech in technology:
        number_of_jobs = number_of_jobs + 1
    return technology,number_of_jobs
```

Calling the function for Python and checking if it works.

In [34]:
```
print(get_number_of_jobs('python'))
```

```
('python', 6)
```

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

In [35]:
```
def get_number_of_jobs_L(location):
    payload={"Location":location}
    response=requests.get(api_url, params=payload)
    if response.ok:
        data=response.json()
        number_of_jobs = len(data)
    return location,number_of_jobs
```

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

In [36]:
```
get_number_of_jobs_L("Los Angeles")
```

Out[36]:
```
('Los Angeles', 27005)
```

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 technologies for which you need to find the number of jobs postings.

In [38]:
```
jobs=['C','C#','C++','Java','Javascript','Python','Scala','Oracle','SQ
print (jobs)
```

```
['C', 'C#', 'C++', 'Java', 'Javascript', 'Python', 'Scala', 'Oracle', '
SQL Server', 'MySQL Server', 'PostgreSQL', 'MongoDB']
```

Import libraries required to create excel spreadsheet

In [39]:
```
pip install openpyxl
```

```
Requirement already satisfied: openpyxl in /home/jupyterlab/conda/envs/
python/lib/python3.7/site-packages (3.1.3)
Requirement already satisfied: et-xmlfile in /home/jupyterlab/conda/env
s/python/lib/python3.7/site-packages (from openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

Create a workbook and select the active worksheet

In [40]:
```
from openpyxl import Workbook
```

In [44]:
```
wb=Workbook()
ws=wb.active


print ('Excel created')
```

```
Excel created
```

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

In [45]:
```
for num in tech:
    print(get_number_of_jobs(num))
    ws.append(get_number_of_jobs(num))
```

```
('C', 1)
('C#', 2)
('C++', 3)
('Java', 4)
('Javascript', 10)
('Python', 6)
('Scala', 5)
('Oracle', 6)
('SQL Server', 10)
('MySQL Server', 12)
('PostgreSQL', 10)
('MongoDB', 7)
```

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

In [46]:
```
#your code goes here
workbook.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 [80]:
```
# your code goes here
technologies = ['C', 'C#', 'C++', 'Java', 'JavaScript', 'Python', 'Sca

def get_number_of_jobs_L(technologies, number_of_jobs):
    final_list = []
    for technology in technologies:
        number_of_jobs_list = [technology]
        payload={"Key Skills": technology, "get_number_of_jobsL": get_
        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 ws.append(number_of_jobs_list)
        final_list.append(number_of_jobs_list)
    return final_list

get_number_of_jobs_L(technologies, get_number_of_jobs)
```

Out[80]:
```
[['C', 27005],
 ['C#', 27005],
 ['C++', 27005],
 ['Java', 27005],
 ['JavaScript', 27005],
 ['Python', 27005],
 ['Scala', 27005],
 ['Oracle', 27005],
 ['SQL Server', 27005],
 ['MySQL Server', 27005],
 ['PostgreSQL', 27005],
 ['MongoDB', 27005]]
```

Authors

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 |--!>