Skills Network

(https://skills.network/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=1000
SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork928-2022-01-01)

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.

In [1]:
```
!pip install flask
```

```
Requirement already satisfied: flask in c:\users\administrator\anaconda3\lib
\site-packages (2.2.2)
Requirement already satisfied: Werkzeug>=2.2.2 in c:\users\administrator\anac
onda3\lib\site-packages (from flask) (2.2.3)
Requirement already satisfied: Jinja2>=3.0 in c:\users\administrator\anaconda
3\lib\site-packages (from flask) (3.1.2)
Requirement already satisfied: itsdangerous>=2.0 in c:\users\administrator\an
aconda3\lib\site-packages (from flask) (2.0.1)
Requirement already satisfied: click>=8.0 in c:\users\administrator\anaconda3
\lib\site-packages (from flask) (8.0.4)
Requirement already satisfied: colorama in c:\users\administrator\anaconda3\l
ib\site-packages (from click>=8.0->flask) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\administrator\anac
onda3\lib\site-packages (from Jinja2>=3.0->flask) (2.1.1)
```

In [2]:
```
!wget  https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-I
```

```
'wget' is not recognized as an internal or external command,
operable program or batch file.
```

In [3]:
```
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
    #  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()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[3], line 49
     47 import json
     48 data = None
---> 49 with open('jobs.json',encoding='utf-8') as f:
     50     # returns JSON object as
     51     # a dictionary
     52     data = json.load(f)
     56 @app.route('/', methods=['GET'])
     57 def home():

File ~\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:286, in _
modified_open(file, *args, **kwargs)
    279 if file in {0, 1, 2}:
    280     raise ValueError(
    281         f"IPython won't let you open fd={file} by default "
    282         "as it is likely to crash IPython. If you know what you are d
oing, "
    283         "you can use builtins' open."
    284     )
--> 286 return io_open(file, *args, **kwargs)

FileNotFoundError: [Errno 2] No such file or directory: 'jobs.json'
```

In [ ]:

In [ ]:

Instructions

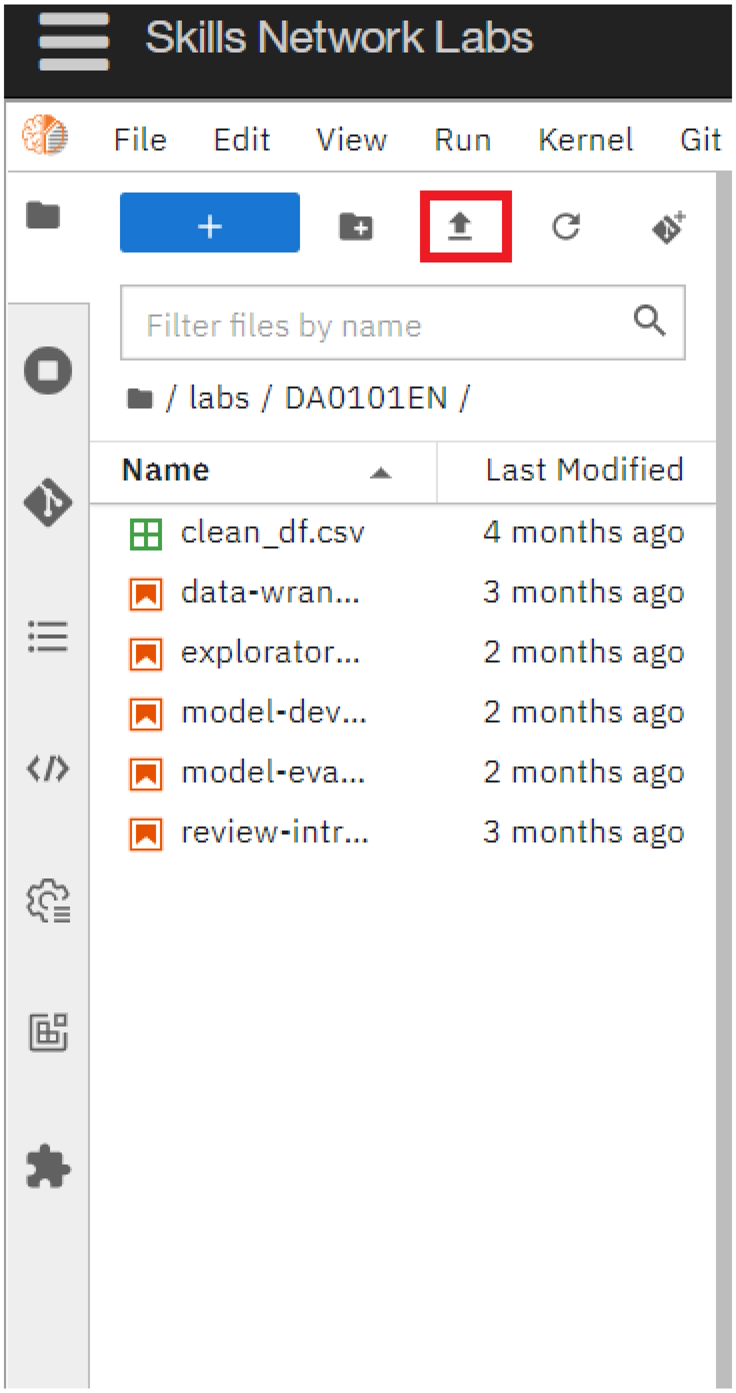
To run the actual lab, firstly you need to click on the Jobs_API (https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/Jobs_API.ipynb) notebook link. The file contains flask code which is required to run the Jobs 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.)

Dataset Used in this Assignment

The dataset used in this lab comes from the following source: https://www.kaggle.com/promptcloud/jobs-on-naukricom (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.

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 (http://api.open-notify.org/astros.json?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006 SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork21426264-2021-01-01&cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBM-DA0321EN-SkillsNetwork-21426264&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=en
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/ (http://open-notify.org/Open-Notify-API/People-In-Space?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_term=10006 SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork21426264-2021-01-01&cm_mmc=Email_Newsletter-_-Developer_Ed%2BTech-_-WW_WW-_-SkillsNetwork-Courses-IBM-DA0321EN-SkillsNetwork-21426264&cm_mmca1=000026UJ&cm_mmca2=10006555&cm_mmca3=M12345678&cvosrc=en

In [4]:
```
import requests # you need this module to make an API call
import pandas as pd
```

```
C:\Users\Administrator\anaconda3\Lib\site-packages\pandas\core\arrays\masked.
py:60: UserWarning: Pandas requires version '1.3.6' or newer of 'bottleneck'
(version '1.3.5' currently installed).
  from pandas.core import (
```

In [5]:
```
api_url = "http://api.open-notify.org/astros.json" # this url gives use the as
```

In [7]:
```
response = requests.get(api_url) # Call the API using the get method and store
                                 # output of the API call in a variable called 
```

In [8]:
```
if response.ok:             # if all is well() no errors, no network timeouts)
    data = response.json()  # store the result in json format in a variable ca
                            # the variable data is of type dictionary.
```

In [12]:
```
df = pd.DataFrame(data)
df
```

Out[12]:

	message	people	number
0	success	{'name': 'Jasmin Moghbeli', 'craft': 'ISS'}	7
1	success	{'name': 'Andreas Mogensen', 'craft': 'ISS'}	7
2	success	{'name': 'Satoshi Furukawa', 'craft': 'ISS'}	7
3	success	{'name': 'Konstantin Borisov', 'craft': 'ISS'}	7
4	success	{'name': 'Oleg Kononenko', 'craft': 'ISS'}	7
5	success	{'name': 'Nikolai Chub', 'craft': 'ISS'}	7
6	success	{'name': 'Loral O'Hara', 'craft': 'ISS'}	7

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

print(type(data))
```

```
{'message': 'success', 'people': [{'name': 'Jasmin Moghbeli', 'craft': 'IS
S'}, {'name': 'Andreas Mogensen', 'craft': 'ISS'}, {'name': 'Satoshi Furukaw
a', 'craft': 'ISS'}, {'name': 'Konstantin Borisov', 'craft': 'ISS'}, {'name':
'Oleg Kononenko', 'craft': 'ISS'}, {'name': 'Nikolai Chub', 'craft': 'ISS'},
{'name': "Loral O'Hara", 'craft': 'ISS'}], 'number': 7}
<class 'dict'>
```

Print the number of astronauts currently on ISS.

In [12]:
```
print(data.get('number'))
```

```
7
```

Print the names of the astronauts currently on ISS.

In [13]:
```
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 7 astronauts on ISS
And their names are :
Jasmin Moghbeli
Andreas Mogensen
Satoshi Furukawa
Konstantin Borisov
Oleg Kononenko
Nikolai Chub
Loral O'Hara
```

In [14]:
```
astronauts
```

Out[14]:
```
[{'name': 'Jasmin Moghbeli', 'craft': 'ISS'},
 {'name': 'Andreas Mogensen', 'craft': 'ISS'},
 {'name': 'Satoshi Furukawa', 'craft': 'ISS'},
 {'name': 'Konstantin Borisov', 'craft': 'ISS'},
 {'name': 'Oleg Kononenko', 'craft': 'ISS'},
 {'name': 'Nikolai Chub', 'craft': 'ISS'},
 {'name': "Loral O'Hara", 'craft': 'ISS'}]
```

In [ ]:
```

```

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 [15]:
```
#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 (https://www.coursera.org/learn/python-for-applied-data-science-ai/ungradedLti/P6sW8/hands-on-lab-access-rest-apis-request-http?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=000026UJ&utm_terr SkillsNetwork-Channel-SkillsNetworkCoursesIBMDA0321ENSkillsNetwork928-2022-01-01)

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 (https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/jobs.json) URL.

In [26]:
```
api_url = requests.get("https://cf-courses-data.s3.us.cloud-object-storage.appd
api_url.url
```

Out[26]:
```
'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA032
1EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Using%20APIs/jobs.json'
```

In [46]:
```
len(api_url.json())
```

Out[46]:
```
27005
```

In [93]:
```
df_python =(df['Location']== 'Los Angeles').sum()
df_python
```

Out[93]:
```
640
```

In [150]:
```
def num_of_jobs(location):
    return location, (df['Location'] == 'Los Angeles').sum()

num_of_jobs('Los Angeles')
```

Out[150]:
```
('Los Angeles', 640)
```

In [86]:
```
df_loc = df['Location'].value_counts()
df_loc
```

Out[86]:
```
Location
Washington DC    5316
Detroit          3945
Seattle          3375
Houston          3339
New York         3226
Boston           2966
Baltimore        1263
Dallas           1208
New Orleons       817
Los Angeles       640
San Francisco     435
Austin            434
Philadelphia       41
Name: count, dtype: int64
```

In [91]:
```
(df['Role'] == 'c++').value_counts()
```

Out[91]:
```
Role
False    27005
Name: count, dtype: int64
```

In [48]:
```
if api_url.ok:
    data = api_url.json()
df = pd.DataFrame(data)
df.head()
```

Out[48]:

	Id	Job Title	Job Experience Required	Key Skills	Role Category	Location
0	0	Digital Media Planner	5 - 10 yrs	Media Planning| Digital Media	Advertising	Los Angeles
1	1	Online Bidding Executive	2 - 5 yrs	pre sales| closing| software knowledge| client...	Retail Sales	New York
2	2	Trainee Research/ Research Executive- Hi- Tech...	0 - 1 yrs	Computer science| Fabrication| Quality check| ...	R&D	San Francisco
3	3	Technical Support	0 - 5 yrs	Technical Support	Admin/Maintenance/Security/Datawarehousing	Washington DC
4	4	Software Test Engineer - hyderabad	2 - 5 yrs	manual testing| test engineering| test cases| ...	Programming & Design	Boston

```
In [177]: url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-D


          r=requests.get(url,payload)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[177], line 4
      1 url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomai
n.cloud/IBM-DA0321EN-SkillsNetwork/labs/module%201/Accessing%20Data%20Usin
g%20APIs/jobs.json'
----> 4 r=requests.get(url,payload)

File ~\anaconda3\Lib\site-packages\requests\api.py:73, in get(url, params,
**kwargs)
     62 def get(url, params=None, **kwargs):
     63     r"""Sends a GET request.
     64 
     65     :param url: URL for the new :class:`Request` object.
   (...)
     70     :rtype: requests.Response
     71     """
---> 73     return request("get", url, params=params, **kwargs)
```

In [53]:
```
#api_url="http://127.0.0.1:5000/data"
def get_number_of_jobs_T(technology):

    #your code goes here
    number_of_jobs = 0
    #your code goes here
    page=1
    new_results=1
    while new_results>0:
        payload={"description":technology,"page":page}
        r=requests.get(url,payload)
        new_results =len(r.json())
        page+=1
        number_of_jobs+=(len(r.json()))

get_number_of_jobs_T("Python")
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[53], line 16
     13         page+=1
     14         number_of_jobs+=(len(r.json()))
---> 16 get_number_of_jobs_T("Python")

Cell In[53], line 11, in get_number_of_jobs_T(technology)
      9 while new_results>0:
     10     payload={"description":technology,"page":page}
---> 11     r=requests.get(url,payload)
     12     new_results =len(r.json())
     13     page+=1

File ~\anaconda3\Lib\site-packages\requests\api.py:73, in get(url, params,
**kwargs)
     62 def get(url, params=None, **kwargs):
     63     r"""Sends a GET request.
     64
```

In [4]:
```
api_url="http://127.0.0.1:5000/data"
def get_number_of_jobs_T(technology):

    #your code goes here
    number_of_jobs = 0
    for tech in technology:
        number_of_jobs = number_of_jobs + 1
    return technology,number_of_jobs

get_number_of_jobs_T("c++")
```

Out[4]:
```
('c++', 3)
```

```
In [17]: #api_url="http://127.0.0.1:5000/data"
         def get_number_of_jobs_T(technology):
             
             #your code goes here
             return technology,number_of_jobs
```

Calling the function for Python and checking if it works.

```
In [76]: get_number_of_jobs_T("Python")

Out[76]: ('Python', 6)
```

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

```
In [158]: def no_of_jobs(location):
              return location, (df.Location == location).sum()
          
          no_of_jobs("Los Angeles")

Out[158]: ('Los Angeles', 640)
```

```
In [122]: loc = df.Location.unique()
          loc

Out[122]: array(['Los Angeles', 'New York', 'San Francisco', 'Washington DC',
                 'Boston', 'Seattle', 'Detroit', 'Austin', 'Houston',
                 'Philadelphia', 'New Orleons', 'Baltimore', 'Dallas'], dtype=object)
```

```
In [141]: location = df.Location.value_counts().index
          location

Out[141]: Index(['Washington DC', 'Detroit', 'Seattle', 'Houston', 'New York', 'Bosto
          n',
                 'Baltimore', 'Dallas', 'New Orleons', 'Los Angeles', 'San Francisco',
                 'Austin', 'Philadelphia'],
                dtype='object', name='Location')
```

```
In [127]: def get_number_of_jobs_T(location):
              
              #your code goes here
              number_of_jobs = 0
              for location in df:
                  number_of_jobs = number_of_jobs + 1
              return location,number_of_jobs
          get_number_of_jobs_T('Dallas')

Out[127]: ('Role', 9)
```

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

In [146]:
```
def get_number_of_jobs(loc):

    l = []
    for loc in df.Location:
        if df.Location == loc:
            l.append(loc)
        return loc,len(l)
get_number_of_jobs('Los Angeles')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14508\257017808.py in ?()
      4     for loc in df.Location:
      5         if df.Location == loc:
      6             l.append(loc)
      7         return loc,len(l)
----> 8 get_number_of_jobs('Los Angeles')

~\AppData\Local\Temp\ipykernel_14508\257017808.py in ?(loc)
      1 def get_number_of_jobs(loc):
      2 
      3     l = []
      4     for loc in df.Location:
----> 5         if df.Location == loc:
      6             l.append(loc)
      7         return loc,len(l)

~\anaconda3\Lib\site-packages\pandas\core\generic.py in ?(self)
   1574     @final
   1575     def __nonzero__(self) -> NoReturn:
-> 1576         raise ValueError(
   1577             f"The truth value of a {type(self).__name__} is ambiguou
s. "
   1578             "Use a.empty, a.bool(), a.item(), a.any() or a.all()."
   1579         )

ValueError: The truth value of a Series is ambiguous. Use a.empty, a.bool(),
a.item(), a.any() or a.all().
```

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 [99]:
```
#your code goes here
loc_list = ['Los Angeles', 'New York', 'San Francisco', 'Washington DC', 'Seat
```

In [156]:
```
for loc in loc_list:
    print([loc, (df.Location == loc).sum()])
```

```
['Los Angeles', 640]
['New York', 3226]
['San Francisco', 435]
['Washington DC', 5316]
['Seattle', 3375]
['Austin', 434]
['Detroit', 3945]
```

In [162]:
```
def job(location):
    for items in location:
        print([items, (df.Location == items).sum()])
job(loc_list)
```

```
['Los Angeles', 640]
['New York', 3226]
['San Francisco', 435]
['Washington DC', 5316]
['Seattle', 3375]
['Austin', 434]
['Detroit', 3945]
```

Import libraries required to create excel spreadsheet

In [164]:
```
# your code goes here

from openpyxl import Workbook
```

Create a workbook and select the active worksheet

In [165]:
```
# your code goes here
wb = Workbook()
ws = wb.active
```

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 [168]:
```
for loc in loc_list:
     ws.append([loc, (df.Location == loc).sum()])
ws
```

Out[168]:
```
<Worksheet "Sheet">
```

In [167]:
```
```

Out[167]:
```
<openpyxl.workbook.workbook.Workbook at 0x26d01015650>
```

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

In [169]:
```
#your code goes here
wb.save('Job Location')
```

In [173]:
```
pip install openpyxl
```

```
Requirement already satisfied: openpyxl in c:\users\administrator\anaconda3\l
ib\site-packages (3.0.10)
Requirement already satisfied: et_xmlfile in c:\users\administrator\anaconda3
\lib\site-packages (from openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

```
In [174]: job = pd.read_excel('Job Location')
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[174], line 1
----> 1 job = pd.read_excel('Job Location')

File ~\anaconda3\Lib\site-packages\pandas\io\excel\_base.py:495, in read_exce
l(io, sheet_name, header, names, index_col, usecols, dtype, engine, converter
s, true_values, false_values, skiprows, nrows, na_values, keep_default_na, na
_filter, verbose, parse_dates, date_parser, date_format, thousands, decimal,
comment, skipfooter, storage_options, dtype_backend, engine_kwargs)
    493 if not isinstance(io, ExcelFile):
    494     should_close = True
--> 495     io = ExcelFile(
    496         io,
    497         storage_options=storage_options,
    498         engine=engine,
    499         engine_kwargs=engine_kwargs,
    500     )
    501 elif engine and engine != io.engine:
    502     raise ValueError(
    503         "Engine should not be specified when passing "
    504         "an ExcelFile - ExcelFile already has the engine set"
    505     )

File ~\anaconda3\Lib\site-packages\pandas\io\excel\_base.py:1567, in ExcelFil
e.__init__(self, path_or_buffer, engine, storage_options, engine_kwargs)
   1564 self.engine = engine
   1565 self.storage_options = storage_options
-> 1567 self._reader = self._engines[engine](
   1568     self._io,
   1569     storage_options=storage_options,
   1570     engine_kwargs=engine_kwargs,
   1571 )

File ~\anaconda3\Lib\site-packages\pandas\io\excel\_openpyxl.py:552, in Openp
yxlReader.__init__(self, filepath_or_buffer, storage_options, engine_kwargs)
    534 @doc(storage_options=_shared_docs["storage_options"])
    535 def __init__(
    536     self,
   (...)
    539     engine_kwargs: dict | None = None,
    540 ) -> None:
    541     """
    542     Reader using openpyxl engine.
    543 
   (...)
    550         Arbitrary keyword arguments passed to excel engine.
    551     """
--> 552     import_optional_dependency("openpyxl")
    553     super().__init__(
    554         filepath_or_buffer,
    555         storage_options=storage_options,
    556         engine_kwargs=engine_kwargs,
    557     )

File ~\anaconda3\Lib\site-packages\pandas\compat\_optional.py:164, in import_
optional_dependency(name, extra, errors, min_version)
```

```
    162     return None
    163 elif errors == "raise":
--> 164     raise ImportError(msg)
    165 else:
    166     return None

ImportError: Pandas requires version '3.1.0' or newer of 'openpyxl' (version
'3.0.10' currently installed).
```

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

Copyright © 2022 IBM Corporation. All rights reserved.