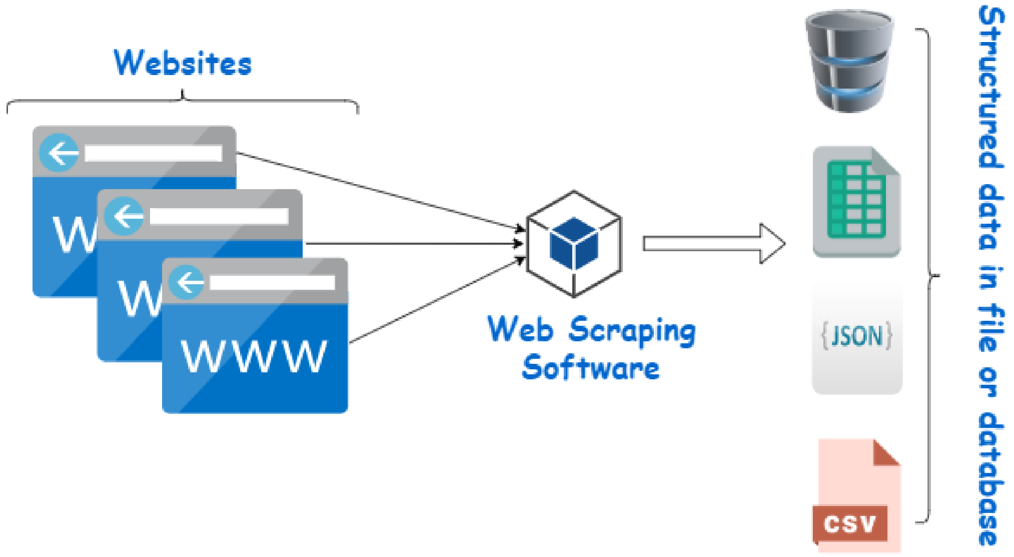
Web Scraping

DR. VEENA R S
ASSOCIATE PROFESSOR
DEPARTMENT OF ISE
DSATM, BENGALURU

Introduction

- Web data extraction is data scraping used for extracting data from websites.
- Web scraping software may access the World Wide Web directly using the Hypertext Transfer Protocol, or through a web browser.
- Web scraping is the term for using a program to download and process content from the Web.



- Several modules that make it easy to scrape web pages in Python.
- **webbrowser**. Comes with Python and opens a browser to a specific page.
- **Requests**. Downloads files and web pages from the Internet.
- **Beautiful Soup**. Parses HTML, the format that web pages are written in.
- **Selenium**. Launches and controls a web browser. Selenium is able to fill in forms and simulate mouse clicks in this browser.

HTML

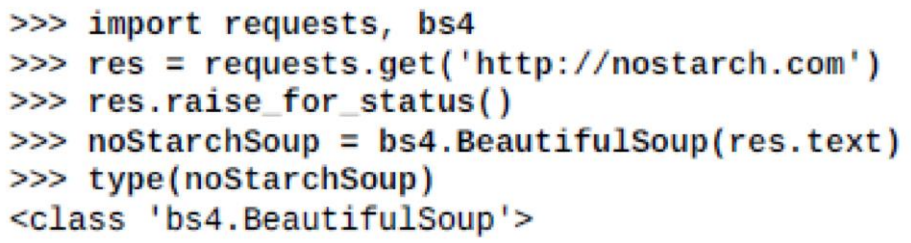
- *Hypertext Markup Language (HTML)* is the format that web pages are written in.
- Sample program example

Parsing HTML with BeautifulSoup Module

- Beautiful Soup is a module for extracting information from an HTML page.
- The BeautifulSoup module's name is bs4 (for Beautiful Soup, version 4).
- To install it, you will need to run pip install beautifulsoup4 from the command line.

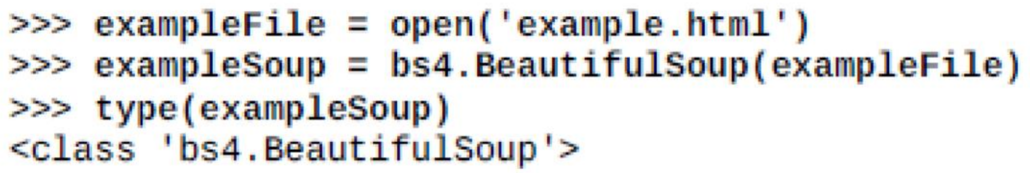
Creating a BeautifulSoup Object from HTML

- The bs4.BeautifulSoup() function needs to be called with a string containing the HTML it will parse.
- The bs4.BeautifulSoup() function returns is a BeautifulSoup object.
- From downloaded file



```
>>> import requests, bs4
>>> res = requests.get('http://nostarch.com')
>>> res.raise_for_status()
>>> noStarchSoup = bs4.BeautifulSoup(res.text)
>>> type(noStarchSoup)
<class 'bs4.BeautifulSoup'>
```

- From existing file



```
>>> exampleFile = open('example.html')
>>> exampleSoup = bs4.BeautifulSoup(exampleFile)
>>> type(exampleSoup)
<class 'bs4.BeautifulSoup'>
```

Project:
Downloading All XKCD Comics

- “Blogs and other regularly updating websites usually have a front page with the most recent post as well as a Previous button on the page that takes you to the previous post. Then that post will also have a Previous button, and so on, creating a trail from the most recent page to the first post on the site” #

- Manually navigate over every page and save each one

Here’s what your program does:

- Loads the XKCD home page.
- Saves the comic image on that page.
- Follows the Previous Comic link.
- Repeats until it reaches the first comic.

- This means your code will need to do the following:
- Download pages with the request's module.
- Find the URL of the comic image for a page using Beautiful Soup.
- Download and save the comic image to the hard drive with iter_content().
- Find the URL of the Previous Comic link and repeat.

Step 1: Design the Program

```
#! python3
# downloadXkcd.py - Downloads every single XKCD comic.

import requests, os, bs4

url = 'http://xkcd.com'               # starting url
os.makedirs('xkcd', exist_ok=True)    # store comics in ./xkcd
while not url.endswith('#'):
    # TODO: Download the page.

    # TODO: Find the URL of the comic image.

    # TODO: Download the image.

    # TODO: Save the image to ./xkcd.

    # TODO: Get the Prev button's url.

print('Done.')
```

Step 2: Download the Web Page

```
#! python3
# downloadXkcd.py - Downloads every single XKCD comic.

import requests, os, bs4

url = 'http://xkcd.com'               # starting url
os.makedirs('xkcd', exist_ok=True)    # store comics in ./xkcd
while not url.endswith('#'):
    # Download the page.
    print('Downloading page %s…' % url)
    res = requests.get(url)
    res.raise_for_status()

    soup = bs4.BeautifulSoup(res.text)

    # TODO: Find the URL of the comic image.

    # TODO: Download the image.

    # TODO: Save the image to ./xkcd.

    # TODO: Get the Prev button's url.

print('Done.')
```

Step 3: Find and Download the Comic Image

```
<p id=#comic >
<img  src="john">

</img>
</p>
```

```
#! python3
# downloadXkcd.py - Downloads every single XKCD comic.

import requests, os, bs4

--snip--

    # Find the URL of the comic image.
    comicElem = soup.select('#comic img')
    if comicElem == []:
         print('Could not find comic image.')
    else:
         comicUrl = comicElem[0].get('src')
         # Download the image.
         print('Downloading image %s…' % (comicUrl))
         res = requests.get(comicUrl)
         res.raise_for_status()

    # TODO: Save the image to ./xkcd.

    # TODO: Get the Prev button's url.

print('Done.')
```

Step 4: Save the Image & Find the Previous Comic

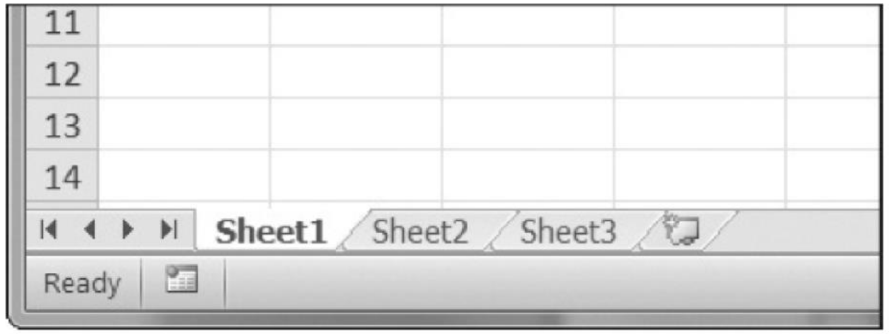
```
#! python3
# downloadXkcd.py - Downloads every single XKCD comic.

import requests, os, bs4


--snip--

        # Save the image to ./xkcd.
        imageFile = open(os.path.join('xkcd', os.path.basename(comicUrl)), 'wb')
        for chunk in res.iter_content(100000):
            imageFile.write(chunk)
        imageFile.close()

    # Get the Prev button's url.
    prevLink = soup.select('a[rel="prev"]')[0]
    url = 'http://xkcd.com' + prevLink.get('href')

print('Done.')
```

The output of this program will look like this:

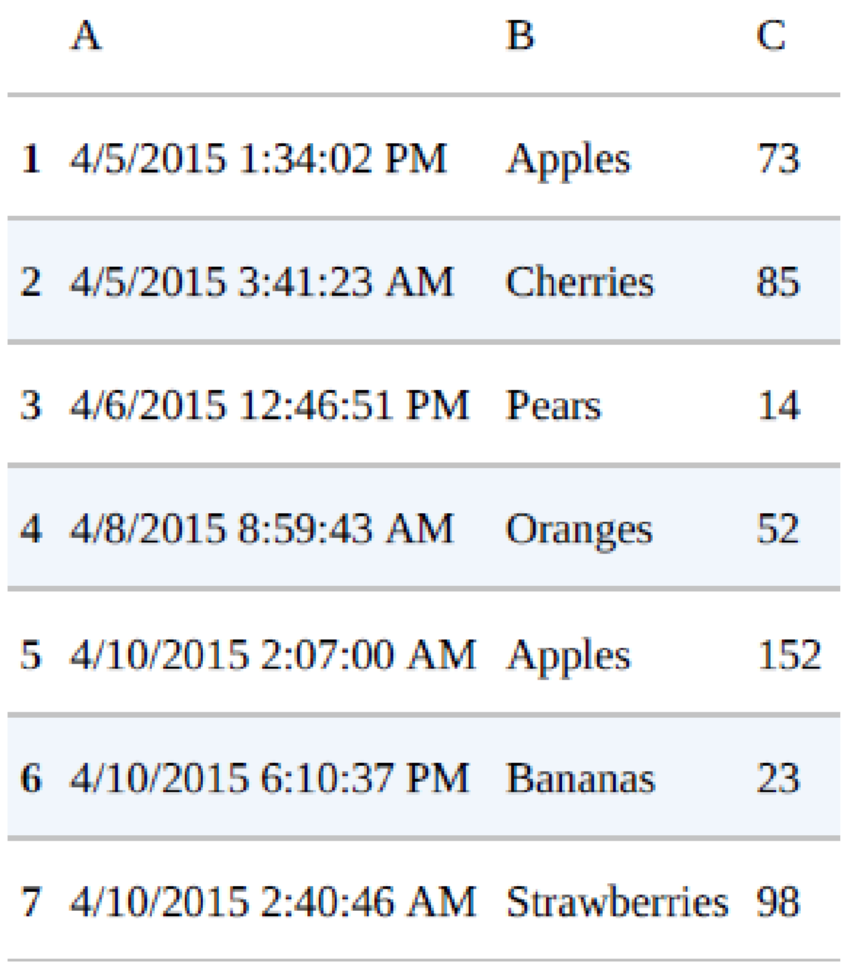
```
Downloading page http://xkcd.com…
Downloading image http://imgs.xkcd.com/comics/phone_alarm.png…
Downloading page http://xkcd.com/1358/...
Downloading image http://imgs.xkcd.com/comics/nro.png…
Downloading page http://xkcd.com/1357/...
Downloading image http://imgs.xkcd.com/comics/free_speech.png…
Downloading page http://xkcd.com/1356/...
Downloading image http://imgs.xkcd.com/comics/orbital_mechanics.png…
Downloading page http://xkcd.com/1355/...
Downloading image http://imgs.xkcd.com/comics/airplane_message.png…
Downloading page http://xkcd.com/1354/...
Downloading image http://imgs.xkcd.com/comics/heartbleed_explanation.png…
--snip--
```

Working with Excel Spreadsheets

- The openpyxl module allows your Python programs to read and modify Excel spreadsheet files.
- Installing the openpyxl Module
- >>> import openpyxl
- Reading Excel Documents
- http://nostarch.com/automatestuff/.



- Book example

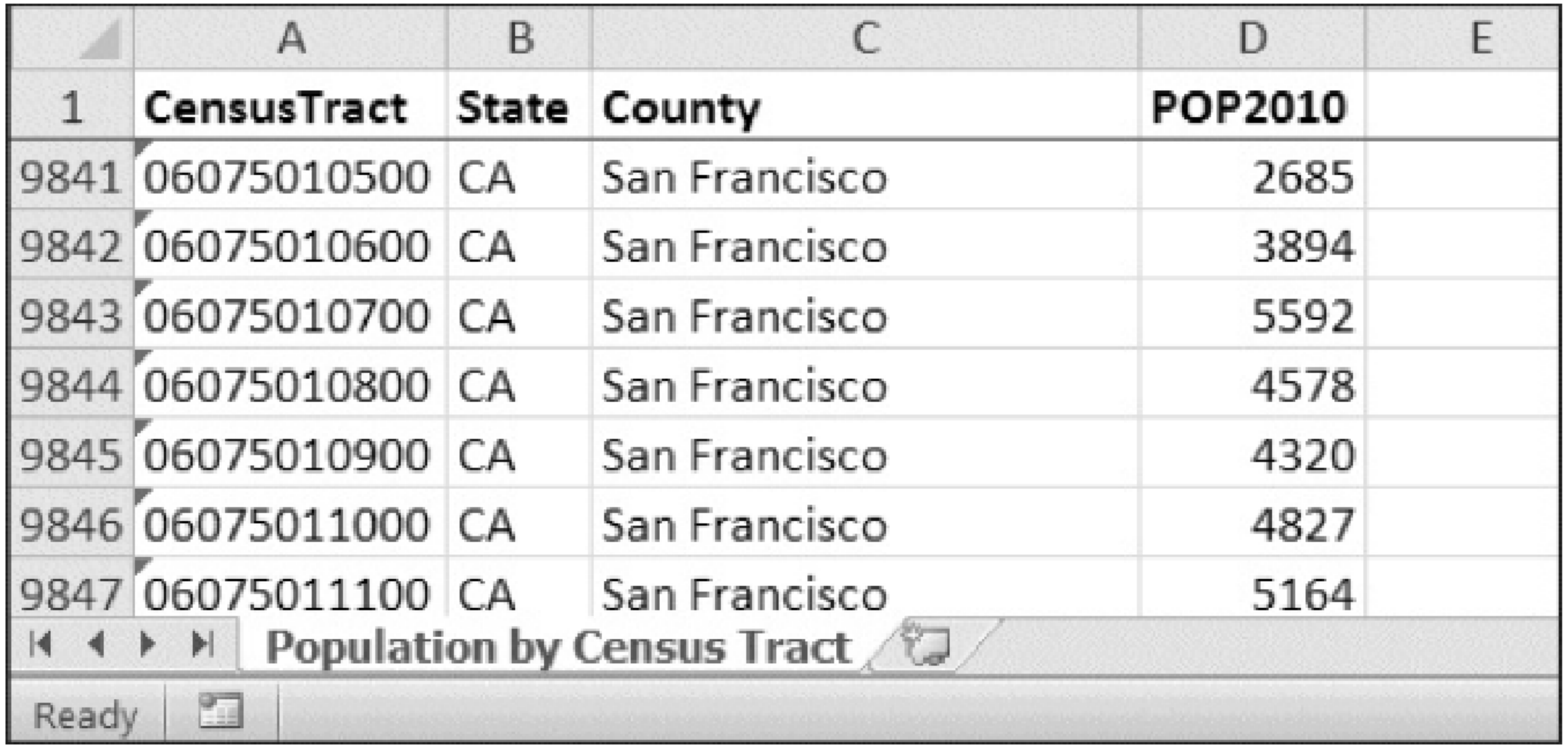


	A	B	C
1	4/5/2015 1:34:02 PM	Apples	73
2	4/5/2015 3:41:23 AM	Cherries	85
3	4/6/2015 12:46:51 PM	Pears	14
4	4/8/2015 8:59:43 AM	Oranges	52
5	4/10/2015 2:07:00 AM	Apples	152
6	4/10/2015 6:10:37 PM	Bananas	23
7	4/10/2015 2:40:46 AM	Strawberries	98

Project:
Reading Data from a Spreadsheet

- This is what your program does:
- Reads the data from the Excel spreadsheet.
- Counts the number of census tracts in each county.
- Counts the total population of each county.
- Prints the results.

- This means your code will need to do the following:
- Open and read the cells of an Excel document with the openpyxl module.
- Calculate all the tract and population data and store it in a data structure.
- Write the data structure to a text file with the .py extension using the pprint module.



	A	B	C	D	E
1	**CensusTract**	**State**	**County**	**POP2010**	
9841	06075010500	CA	San Francisco	2685	
9842	06075010600	CA	San Francisco	3894	
9843	06075010700	CA	San Francisco	5592	
9844	06075010800	CA	San Francisco	4578	
9845	06075010900	CA	San Francisco	4320	
9846	06075011000	CA	San Francisco	4827	
9847	06075011100	CA	San Francisco	5164	

Figure 12-2. The censuspopdata.xlsx spreadsheet

Step 1: Read the Spreadsheet Data

```
   #! python3
   # readCensusExcel.py - Tabulates population and number of census tracts for
   # each county.

❶ import openpyxl, pprint
   print('Opening workbook…')
❷ wb = openpyxl.load_workbook('censuspopdata.xlsx')
❸ sheet = wb.get_sheet_by_name('Population by Census Tract')
   countyData = {}

   # TODO: Fill in countyData with each county's population and tracts.
   print('Reading rows…')
❹ for row in range(2, sheet.get_highest_row() + 1):
       # Each row in the spreadsheet has data for one census tract.
       state  = sheet['B' + str(row)].value
       county = sheet['C' + str(row)].value
       pop    = sheet['D' + str(row)].value

   # TODO: Open a new text file and write the contents of countyData to it.
```

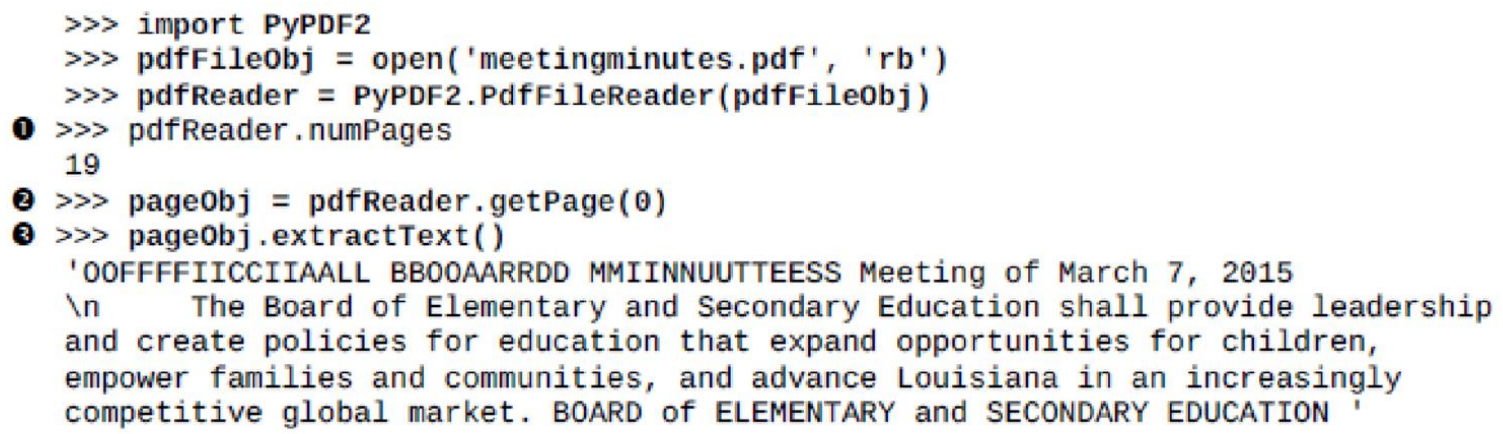
Step 2: Populate the Data Structure

Working with PDF and word documents

- PDF and Word documents are binary files, which makes them much more complex than plaintext files.
- PDF Documents
- PDF stands for Portable Document Format and uses the .pdf file extension.
- To install it, run pip install PyPDF2 from the command line.

Extracting Text from PDFs

- PyPDF2 does not have a way to extract images, charts, or other media from PDF documents, but it can extract text and return it as a Python string.



```
  >>> import PyPDF2
  >>> pdfFileObj = open('meetingminutes.pdf', 'rb')
  >>> pdfReader = PyPDF2.PdfFileReader(pdfFileObj)
❶ >>> pdfReader.numPages
  19
❷ >>> pageObj = pdfReader.getPage(0)
❸ >>> pageObj.extractText()
  'OOFFFFIICCIIAALL BBOOAARRDD MMIINNUUTTEESS Meeting of March 7, 2015
  \n     The Board of Elementary and Secondary Education shall provide leadership
  and create policies for education that expand opportunities for children,
  empower families and communities, and advance Louisiana in an increasingly
  competitive global market. BOARD of ELEMENTARY and SECONDARY EDUCATION '
```

Working with CSV Files and JSON Data

- CSV stands for "comma-separated values," and CSV files are simplified spreadsheets stored as plaintext files.
- Python's csv module makes it easy to parse CSV files.
- JSON-is a format that stores information as JavaScript source code in plaintext files.

JSON and APIs

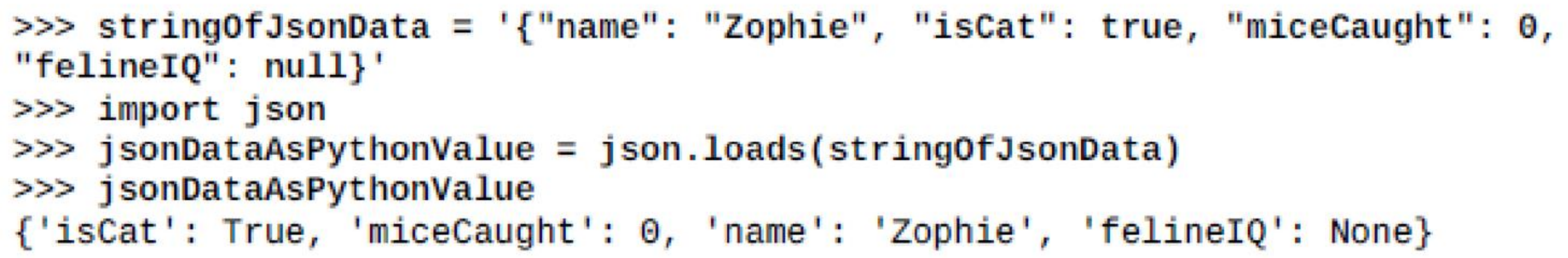
- JavaScript Object Notation is a popular way to format data as a single human-readable string.
- JSON is useful to know, because many websites offer JSON content as a way for programs to interact with the website.
- Many websites make their data available in JSON format. Facebook, Twitter, Yahoo, Google, Tumblr, Wikipedia, Flickr, Data.gov, Reddit, IMDb, Rotten Tomatoes, LinkedIn, and many other popular sites offer APIs for programs to use.

The JSON Module

- Python’s json module handles all the details of translating between a string with JSON data.
- It can contain values of only the following data types:
- strings, integers, floats, Booleans, lists, dictionaries, and NoneType.
- JSON cannot represent Python-specific objects, such as File objects, CSV Reader or Writer objects, Regex objects, or Selenium WebElement objects.

Reading JSON with loads() Function

- json.loads()



```
>>> stringOfJsonData = '{"name": "Zophie", "isCat": true, "miceCaught": 0,
"felineIQ": null}'
>>> import json
>>> jsonDataAsPythonValue = json.loads(stringOfJsonData)
>>> jsonDataAsPythonValue
{'isCat': True, 'miceCaught': 0, 'name': 'Zophie', 'felineIQ': None}
```

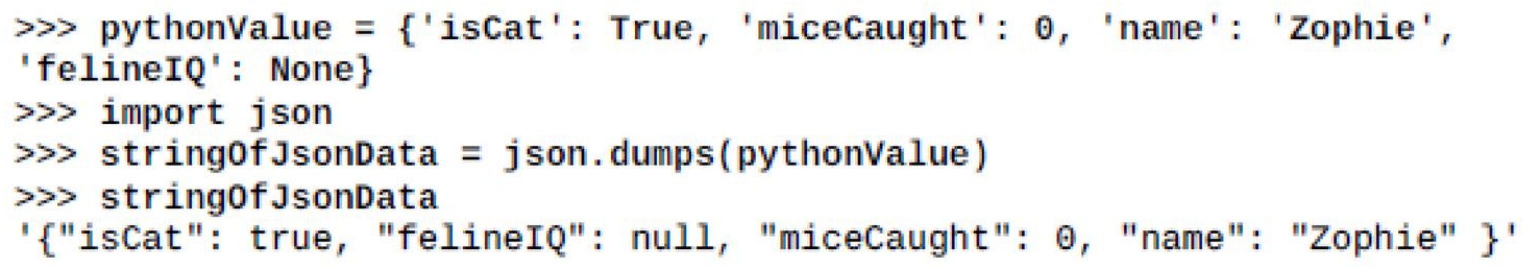
Writing JSON with dumps() Function

- json.dumps()

```
>>> pythonValue = {'isCat': True, 'miceCaught': 0, 'name': 'Zophie',
'felineIQ': None}
>>> import json
>>> stringOfJsonData = json.dumps(pythonValue)
>>> stringOfJsonData
'{"isCat": true, "felineIQ": null, "miceCaught": 0, "name": "Zophie" }'
```

Writing JSON with dumps() Function

- json.dumps()



```
>>> pythonValue = {'isCat': True, 'miceCaught': 0, 'name': 'Zophie',
'felineIQ': None}
>>> import json
>>> stringOfJsonData = json.dumps(pythonValue)
>>> stringOfJsonData
'{"isCat": true, "felineIQ": null, "miceCaught": 0, "name": "Zophie" }'
```

9.a) Write a python program to download the all XKCD comics

- XKCD is a webcomic of the varied genre that consists of sarcasm, mathematics, language, Python, and many more.
- This website consists of many curious comics and sometimes user wants to save that comic image on their local devices.
- Doing so manually is a very exhausting process because to download the comic images of “XKCD Comics” a user has to visit every page of the comic website “https://xkcd.com/”
- To make it easy we are going to create a Python program that can download the page of the comic by entering their page number.

Required Modules

To download XKCD comics pages using Python, we need to install the beautifulsoup4 and requests module. To do so run the following commands in the command prompt.

- pip install beautifulsoup4
- pip install requests
- Requests Module

The requests module is used to deal with HTTP requests to a specified URL. Whether it be Web Scrapping or REST APIs, this module must be learned to work with these technologies

Beautifulsoup4 Module

- The beautifulsoup4 module is used to scrape information from web pages. It helps to organize the unorganized web data by improving HTML and presenting it in an easily-traversable XML structure.

Stepwise implementation:

- Step 1: Import all the required libraries and modules

```
# Importing required modules
import requests as req
import os,bs4
```

- Step 2: Store the URL of the XKCD website from where we have to download our comic page. Using the os.makedirs() method create a folder for storing the images in our local folder and also check for the folder if it already exists then store it in the same folder.

```
URL url = 'https://xkcd.com/‘# Storing website
os.makedirs('xkcd', exist_ok=True) # Make Directory to store image
```

- Step 3: Take the input from the user of “comic image number” and append it to the last of the URL declared in step 1 after that store the information in the ‘res’ variable of that URL using requests.get() after that store HTML page in variable ‘soup’ using bs4.beautifulSoup() method and then store the URL of the image using the soup.select() method that we have to download and at last make that URL complete by appending it with ‘http:’ to be used in the next step. We can also see that image URL by using inspecting the element in the web browser as shown in the below image.

- **Step 4:** Requesting the information from the URL that we had made in the previous step and save it in the directory folder and store the binary mode file in the declared file and folder by using file handling in Python and methods of the os module

9.a) Write a python program to download the all XKCD comics

```
import requests
import os
from bs4 import BeautifulSoup
# Set the URL of the first XKCD comic
url = 'https://xkcd.com/1/‘
# Create a folder to store the comics
if not os.path.exists('xkcd_comics'):
    os.makedirs('xkcd_comics')
```

```
# Loop through all the comics
while True:
    # Download the page content
                    res = requests.get(url)
                    res.raise_for_status()
 # Parse the page content using BeautifulSoup
          soup = BeautifulSoup(res.text, 'html.parser')
    # Find the URL of the comic image
          comic_elem = soup.select('#comic img')
           if comic_elem == []:
                     print('Could not find comic image.')
           else:
                    comic_url = 'https:' + comic_elem[0].get('src')
 # Download the comic image
            print(f'Downloading {comic_url}...')
            res = requests.get(comic_url)
           res.raise_for_status()
  # Save the comic image to the xkcd_comics folder
 image_file = open(os.path.join('xkcd_comics',
os.path.basename(comic_url)), 'wb')
           for chunk in res.iter_content(100000):
                      image_file.write(chunk)
                     image_file.close()
```

```
# Get the URL of the previous comic
        prev_link = soup.select('a[rel="prev"]')[0]
        if not prev_link:
            break
        url = 'https://xkcd.com' + prev_link.get('href')
        print('All comics downloaded.')
```

9b. Demonstrate python program to read the data from the spreadsheet and write the data in to the Spreadsheet.

- Openpyxl is a Python library that provides various methods to interact with Excel Files using Python. It allows operations like reading, writing, arithmetic operations, plotting graphs, etc.
- This module does not come in-built with Python. To install this type the below command in the terminal. pip install openpyxl

Reading from Spreadsheets

- To read an Excel file you have to open the spreadsheet using the load_workbook() method. After that, you can use the active to select the first sheet available and the cell attribute to select the cell by passing the row and column parameter. The value attribute prints the value of the particular cell. See the below example to get a better understanding.

Note: The first row or column integer is 1, not 0.

Writing to Spreadsheets

- To create a new spreadsheet, and then we will write some data to the newly created file. An empty spreadsheet can be created using the Workbook() method.
- After creating an empty file, add some data to it using Python. To add data first we need to select the active sheet and then using the cell() method we can select any particular cell by passing the row and column number as its parameter. We can also write using cell names.

Program

```
import openpyxl
#open the Workbook
workbook = openpyxl.load_workbook("example.xlsx")
 # Select the worksheet
 worksheet = workbook["Sheet1"] # Read data from the spreadsheet
for row in worksheet.iter_rows(min_row=2, max_col=3,
values_only=True): name, age, email = row
print(f"Name: {name}, Age: {age}, Email: {email}")
```

```
# Write data to the spreadsheet
new_data = [("Alice", 25, "alice@example.com"), ("Bob", 30,
"bob@example.com"),
("Charlie", 35, "charlie@example.com")]
for i, data in enumerate(new_data, start=2):
worksheet.cell(row=i, column=1, value=data[0])
worksheet.cell(row=i, column=2, value=data[1])
worksheet.cell(row=i, column=3, value=data[2])
# Save the workbook
workbook.save("example.xlsx")
```