Lab 2: Types of Data

CPE232 Data Models

[1] CSV

```
In [90]: import csv
```

1.1 Writing new csv file

Note: Remember this example? We've already seen it in the last lab.

```
In [91]: with open("test.csv","w",newline='') as file:
                 writer = csv.writer(file)
                 writer.writerow(["Name","Surname"])
                 writer.writerow(["Alice","Johnson"])
                 writer.writerow(["Bob","Smith"])
```

1.2 Reading a csv file

```
In [92]: with open("test.csv","r") as file:
             reader = csv.reader(file)
             for row in reader:
                 print(row)
```

```
['Name', 'Surname']
['Alice', 'Johnson']
['Bob', 'Smith']
```

1.3 Use pandas to read csv file

```
In [93]: import pandas as pd

         df = pd.read_csv('test.csv')

         df
```

Out[93]:

	Name	Surname
0	Alice	Johnson
1	Bob	Smith

[Q1] Write a Python script that reads the **students.csv** file and prints the content of *the first 10 students* row by row.

```
In [94]: # Write your code here
         df = pd.read_csv('students.csv')
         df.head(10)
```

Out[94]:

	Name	Age	Grade
0	Alice	21	A
1	Bob	22	B
2	Charlie	20	C
3	David	23	A
4	Eve	19	B
5	Frank	25	C
6	Grace	22	A
7	Hank	24	B
8	Isla	18	C
9	Jack	20	A

[Q2] Load the **students.csv** file into a pandas DataFrame. Use pandas to filter the DataFrame and create a new DataFrame containing only students who received an "A" grade. Print the new DataFrame.

In [95]:

```python
# Write your code here
df = pd.read_csv('students.csv')
df_A_students = df[df['Grade'] == 'A']
with open('Astudents.csv', 'w') as file:
    writer = csv.writer(file)
    writer.writerow(["Name", "Age", "Grade"])
    for index, row in df.iterrows():
        if row['Grade'] == 'A':
            writer.writerow([row['Name'], row['Age'], row['Grade']])
df = pd.read_csv('Astudents.csv')
df
```

Out[95]:

	Name	Age	Grade
0	Alice	21	A
1	David	23	A
2	Grace	22	A
3	Jack	20	A
4	Mia	24	A
5	Paul	19	A
6	Sam	21	A
7	Victor	24	A
8	Yara	18	A
9	Adam	19	A
10	Diana	24	A
11	Gavin	20	A
12	Julia	18	A
13	Mason	22	A
14	Piper	19	A
15	Steve	22	A
16	Vera	20	A
17	Yusuf	18	A
18	Brianna	24	A
19	Ethan	20	A

[Q3] Add a new column to the DataFrame called "Passed" where the value is True if the grade is "A", and False otherwise. Print the updated DataFrame.

In [96]:
```
# Write your code here
df = pd.read_csv('students.csv')
df['Passed'] = df['Grade'].apply(lambda x: True if x == 'A' else False)
df
```

Out[96]:

	Name	Age	Grade	Passed
0	Alice	21	A	True
1	Bob	22	B	False
2	Charlie	20	C	False
3	David	23	A	True
4	Eve	19	B	False
5	Frank	25	C	False
6	Grace	22	A	True
7	Hank	24	B	False
8	Isla	18	C	False
9	Jack	20	A	True
10	Karen	21	B	False
11	Liam	22	C	False
12	Mia	24	A	True
13	Nate	23	B	False
14	Olivia	25	C	False
15	Paul	19	A	True
16	Quinn	18	B	False
17	Ruby	22	C	False
18	Sam	21	A	True
19	Tina	20	B	False
20	Uma	19	C	False
21	Victor	24	A	True
22	Wendy	23	B	False
23	Xander	22	C	False
24	Yara	18	A	True
25	Zack	20	B	False
26	Adam	19	A	True
27	Beth	22	B	False
28	Cody	21	C	False
29	Diana	24	A	True
30	Edward	23	B	False
31	Fiona	25	C	False
32	Gavin	20	A	True

	Name	Age	Grade	Passed
33	Holly	21	B	False
34	Ian	19	C	False
35	Julia	18	A	True
36	Kyle	24	B	False
37	Laura	23	C	False
38	Mason	22	A	True
39	Nina	25	B	False
40	Oscar	20	C	False
41	Piper	19	A	True
42	Quincy	18	B	False
43	Rosa	21	C	False
44	Steve	22	A	True
45	Tori	24	B	False
46	Ulysses	23	C	False
47	Vera	20	A	True
48	Will	25	B	False
49	Xenia	19	C	False
50	Yusuf	18	A	True
51	Zoe	21	B	False
52	Allen	22	C	False
53	Brianna	24	A	True
54	Caleb	23	B	False
55	Daisy	25	C	False
56	Ethan	20	A	True
57	Faith	19	B	False
58	George	18	C	False

[Q4] Calculate the average age of the students in the DataFrame.

In [97]:
```
# Write your code here
df = pd.read_csv('students.csv')
average_age = df['Age'].mean()
print(f"Average Age: {average_age:.2f} years")
```

```
Average Age: 21.39 years
```

[Q5] Calculate the average GPAX of **ALL** students in the DataFrame, where A=4, B=3, C=2, and D=1.

In [98]:
```
# Write your code here
df = pd.read_csv('students.csv')
grade_to_gpax = {'A': 4, 'B': 3, 'C': 2, 'D': 1}
df['GPAX'] = df['Grade'].map(grade_to_gpax)
grade_to_gpax = df['GPAX'].mean()
print(f"Average GPAX: {grade_to_gpax:.2f}")
```

```
Average GPAX: 3.02
```

[2] HTML

2.1 Different tags in HTML

Basic Structure Tags:

- `<!DOCTYPE html>` : Declares the document type and version of HTML.
- `<html>` : Root element of the HTML document.
- `<head>` : Contains meta-information like the title, character set, and links to external resources (CSS, scripts).
- `<title>` : Specifies the title of the webpage, visible in the browser tab.
- `<body>` : Contains the visible content of the page.

Text Formatting Tags:

- `<h1>` - `<h6>` : Header tags (h1 is the largest, h6 is the smallest).
- `<p>` : Paragraph tag, used to group text into paragraphs.
- `<blockquote>` : Defines a block of text that is a quotation from another source.
- `<code>` : Represents inline code.

Lists and Links:

- `<ul>` : Unordered list (bulleted).
- `<ol>` : Ordered list (numbered).
- `<li>` : List item, used inside `<ul>` or `<ol>` .
- `<a>` : Anchor tag, used to create hyperlinks.
- `<img>` : Image tag, used to embed images.

Tables:

- `<table>` : Defines a table.
- `<tr>` : Table row.
- `<th>` : Table header, defines header cells.
- `<td>` : Table data, defines standard cells.

and more...

In [99]:
```
# pip install bs4
```

In [100…
```
from bs4 import BeautifulSoup
```

2.2 Writing new HTML file

In [101…

```
html_temp = """
<!DOCTYPE html>
<html>
<head>
    <title>Sample Blog</title>
</head>
<body>
    <h2 class="article-title">Article 1: Introduction to Web Scraping</h2>
    <p class="article-content">This is an introduction to web scraping using Bea
    <h2 class="article-title">Article 2: Advanced Web Scraping Techniques</h2>
    <p class="article-content">Learn advanced techniques for web scraping with P
</body>
</html>
"""

with open('html_file.html', 'w') as file:
    file.write(html_temp)
```

2.3 Reading HTML file

In [102…

```
with open('html_file.html') as html_file:
    html_content = html_file.read()

# Parse the HTML content
soup = BeautifulSoup(html_content, 'html.parser')

print(soup.title.text)
print(soup.h2)
print(soup.table.text)
```

```
Sample Blog
<h2 class="article-title">Article 1: Introduction to Web Scraping</h2>
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[102], line 9
      7 print(soup.title.text)
      8 print(soup.h2)
----> 9 print(soup.table.text)

AttributeError: 'NoneType' object has no attribute 'text'
```

[Q6] Explain why the code above gives an error? Fix the code so that it runs without error.

Ans:

In [ ]:

```
# Write your code here
# สาเหตุที่เกิด error เพราะ tag table ไม่มีใน html content
# แก้โดย #print(soup.table.text) หรือการเพิ่ม tag table ใน html content
```

[Q7] You are provided an HTML file named **students.html**. Write a Python script that extracts all the data from the table (headers and rows) and prints them row by row.

In [103…

```
# Write your code here
with open('students.html') as html_file:
    html_content = html_file.read()
soup = BeautifulSoup(html_content, 'html.parser')
headers = [th.text for th in soup.find_all('th')]
print('Header :',headers)
```

```
Header : ['Name', 'Age', 'Grade']
```

In [104…

```
rows = []
for row in soup.find_all('tr')[1:]:
    cells = row.find_all('td')
    row_data = [cell.text.strip() for cell in cells]
    if row_data:
        rows.append(row_data)
for row in rows:
    print(row)
```

```
['Alice', '21', 'A']
['Bob', '22', 'B']
['Charlie', '20', 'C']
['David', '23', 'A']
['Eve', '19', 'B']
['Frank', '25', 'C']
['Grace', '22', 'A']
['Hank', '24', 'B']
['Isla', '18', 'C']
['Jack', '20', 'A']
['Karen', '21', 'B']
['Liam', '22', 'C']
['Mia', '24', 'A']
['Nate', '23', 'B']
['Olivia', '25', 'C']
['Paul', '19', 'A']
['Quinn', '18', 'B']
['Ruby', '22', 'C']
['Sam', '21', 'A']
['Tina', '20', 'B']
['Uma', '19', 'C']
['Victor', '24', 'A']
['Wendy', '23', 'B']
['Xander', '22', 'C']
['Yara', '18', 'A']
['Zack', '20', 'B']
```

[Q8] Modify the script to extract and print only the names of students who received a grade of "A".

In [105…

```
# Write your code here
soup = BeautifulSoup(html_content, 'html.parser')
print("Students who received grade 'A':")
for row in soup.find_all('tr')[1:]:
    cells = row.find_all('td')
    if len(cells) == 3:
        name = cells[0].text.strip()
        grade = cells[2].text.strip()
        if grade == 'A':
            print(name)
```

```
Students who received grade 'A':
Alice
David
Grace
Jack
Mia
Paul
Sam
Victor
Yara
```

[3] XML

In [106…

```
import xml.etree.ElementTree as ET
```

3.1 Writing new xml file

In [107…

```
root = ET.Element("data")
student = ET.SubElement(root, "student", name = "Alice")

email = ET.SubElement(student, 'email')
email.text = "alice@mail.com"

age = ET.SubElement(student, 'age')
age.text = "21"

gender = ET.SubElement(student, 'gender')
gender.text = "F"

tree = ET.ElementTree(root)
tree.write("xml_file.xml")
```

3.2 Modifying existing xml file

In [108…

```
tree = ET.parse('xml_file.xml')
root = tree.getroot()

for student in root:
    for element in student:
        if element.tag == "age":
            element.text = "22"

tree.write('xml_file.xml')
```

3.3 Reading XML file

In [109…

```
tree = ET.parse('xml_file.xml')
root = tree.getroot()

for student in root:
    print(f'name: {student.attrib["name"]}')
    for element in student:
        print(f'{element.tag}: {element.text}')

# Print the entire XML content
```

```
xml_content = ET.tostring(root, encoding='utf-8').decode('utf-8')
print(xml_content)
```

```
name: Alice
email: alice@mail.com
age: 22
gender: F
<data><student name="Alice"><email>alice@mail.com</email><age>22</age><gender>F</
gender></student></data>
```

3.4 Convert XML to List of Dictionary

In [110…

```
data_list = []
for line in root:
    name = line.attrib.get('name')
    email = line.find('email').text
    age = line.find('age').text
    gender = line.find('gender').text

    data_list.append({"Name":name, "Email":email, "Age":age, "Gender":gender})

print(data_list)
```

```
[{'Name': 'Alice', 'Email': 'alice@mail.com', 'Age': '22', 'Gender': 'F'}]
```

[Q9] Add your own data including Name, Email, Age and Gender to the XML file and put it in the existing data_list.

Note: You should show the data_list and XML file by reading the file.

In [111…

```
#Write you own code here
new_person = {
    "Name": "Kaewklaow",
    "Email": "punchaya.chan@gmail.com",
    "Age": "22",
    "Gender": "F"
}
data_list.append(new_person)
print(data_list)
```

```
[{'Name': 'Alice', 'Email': 'alice@mail.com', 'Age': '22', 'Gender': 'F'}, {'Nam
e': 'Kaewklaow', 'Email': 'punchaya.chan@gmail.com', 'Age': '22', 'Gender': 'F'}]
```

[4] JSON

In [112…

```
import json
```

4.1 Writing new json file

In [113…

```
# Data to be written to the JSON file
data_to_write = {
    "people": [
        {"name": "Alice", "age": 30, "city": "New York"},
        {"name": "Bob", "age": 25, "city": "San Francisco"},
        {"name": "Charlie", "age": 35, "city": "Los Angeles"}
    ]
}

# Open the file in write mode and write the data
with open('json_file', 'w') as json_file:
    json.dump(data_to_write, json_file, indent=2)
```

4.2 Reading json file

In [114…

```
with open('json_file', 'r') as file:
    # Load JSON data
    data = json.load(file)

print(data)

people = data['people']

# Print information about each person
for person in people:
    print(f"Name: {person['name']}, Age: {person['age']}, City: {person['city']}
```

```
{'people': [{'name': 'Alice', 'age': 30, 'city': 'New York'}, {'name': 'Bob', 'ag
e': 25, 'city': 'San Francisco'}, {'name': 'Charlie', 'age': 35, 'city': 'Los Ang
eles'}]}
Name: Alice, Age: 30, City: New York
Name: Bob, Age: 25, City: San Francisco
Name: Charlie, Age: 35, City: Los Angeles
```

[Q10] write a code to modify the existing json file so each person have a "job" data and print the result

Ans:

In [115…

```
#write your own code here
for person in data['people']:
    if person['name'] == 'Alice':
        person['job'] = 'Engineer'
    elif person['name'] == 'Bob':
        person['job'] = 'Designer'
    elif person['name'] == 'Charlie':
        person['job'] = 'Manager'
with open('json_file', 'w') as json_file:
    json.dump(data, json_file, indent=2)
with open('json_file', 'r') as file:
    updated_data = json.load(file)
    print(updated_data)
people = updated_data['people']
for person in people:
    print(f"Name: {person['name']}, Age: {person['age']}, City: {person['city']}
```

```
{'people': [{'name': 'Alice', 'age': 30, 'city': 'New York', 'job': 'Engineer'},
{'name': 'Bob', 'age': 25, 'city': 'San Francisco', 'job': 'Designer'}, {'name':
'Charlie', 'age': 35, 'city': 'Los Angeles', 'job': 'Manager'}]}
Name: Alice, Age: 30, City: New York, Job: Engineer
Name: Bob, Age: 25, City: San Francisco, Job: Designer
Name: Charlie, Age: 35, City: Los Angeles, Job: Manager
```