Lab2-RomainBlanchot

January 31, 2025

1 Lab 2: Types of Data

1.1 #### CPE232 Data Models

1.2 [1] CSV

[1]:
```
import csv
```

1.2.1 1.1 Writing new csv file

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

[2]:
```
with open("test.csv","w",newline='') as file:
        writer = csv.writer(file)
        writer.writerow(["Name","Surname"])
        writer.writerow(["Alice","Johnson"])
        writer.writerow(["Bob","Smith"])
```

1.2.2 1.2 Reading a csv file

[3]:
```
with open("test.csv","r") as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

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

1.2.3 1.3 Use pandas to read csv file

[4]:
```
import pandas as pd

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

df
```

[4]:

	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.

[6]:
```
# Lire le fichier students.csv avec pandas
df_students = pd.read_csv('students.csv')

# Afficher les 10 premières lignes
print(df_students.head(10))
```

```
      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.

[7]:
```
# Write your code here
df_students = pd.read_csv('students.csv')

df_students_a = df_students[df_students['Grade'] == 'A']

print(df_students_a)
```

```
       Name  Age Grade
0     Alice   21     A
3     David   23     A
6     Grace   22     A
9      Jack   20     A
12      Mia   24     A
15     Paul   19     A
18      Sam   21     A
21   Victor   24     A
24     Yara   18     A
26     Adam   19     A
29    Diana   24     A
32    Gavin   20     A
35    Julia   18     A
38    Mason   22     A
```

```
41    Piper   19     A
44    Steve   22     A
47     Vera   20     A
50    Yusuf   18     A
53  Brianna   24     A
56    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.

[11]:
```
# Write your code here
df_students['Passed'] = df_students['Grade'] == 'A'
df_students
```

[11]:

	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

```
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.

[15]:
```
# Write your code here
df_students['Age'].mean()
```

[15]:
```
21.389830508474578
```

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

[14]:
```
# Write your code here
df_students['GPAX'] = df_students['Grade'].map({'A': 4, 'B': 3, 'C': 2, 'D': 1})
df_students
df_students['GPAX'].mean()
```

[14]:
```
3.016949152542373
```

1.3 [2] HTML

1.3.1 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...

[16]:
```
from bs4 import BeautifulSoup
```

1.3.2 2.2 Writing new HTML file

[7]:
```
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␣
 ↪BeautifulSoup.</p>
    <h2 class="article-title">Article 2: Advanced Web Scraping Techniques</h2>
    <p class="article-content">Learn advanced techniques for web scraping with␣
 ↪Python.</p>
</body>
</html>
"""

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

1.3.3 2.3 Reading HTML file

[26]:
```
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)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[26], line 5
      2     html_content = html_file.read()
      4 # Parse the HTML content
----> 5 soup = BeautifulSoup(html_content, 'html.parser')
      7 print(soup.title.text)
      8 print(soup.h2)

NameError: name 'BeautifulSoup' is not defined
```

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

Ans: The error occurs because the code tries to access soup.table.text, but there is no

element in the HTML document. As a result, soup.table returns None, and attempting to access text on None raises an AttributeError.

[27]:
```
# Import nécessaire
from bs4 import BeautifulSoup

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

# Parse le contenu HTML
soup = BeautifulSoup(html_content, 'html.parser')

# Affiche le contenu du titre
print("Title:", soup.title.text)

# Affiche tout le contenu du body
print("\nBody content:")
print(soup.body.get_text(strip=True))
```

```
Title: Sample Blog

Body content:
```

```
Article 1: Introduction to Web ScrapingThis is an introduction to web scraping
using BeautifulSoup.Article 2: Advanced Web Scraping TechniquesLearn advanced
techniques for web scraping with Python.
```

[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.

[29]:
```python
# Write your code here
# Import nécessaire
from bs4 import BeautifulSoup

# Ouvre et lit le fichier HTML
with open('students.html') as html_file:
    html_content = html_file.read()

# Parse le contenu HTML
soup = BeautifulSoup(html_content, 'html.parser')

# Trouve la table
table = soup.find('table')

# Extrait les en-têtes
headers = []
for th in table.find_all('th'):
    headers.append(th.text.strip())

# Affiche les en-têtes
print("En-têtes:", headers)
print("\nDonnées:")

# Extrait et affiche chaque ligne
for row in table.find_all('tr')[1:]:  # Skip header row
    row_data = []
    for td in row.find_all('td'):
        row_data.append(td.text.strip())
    print(row_data)
```

```
En-têtes: ['Name', 'Age', 'Grade']

Données:
['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".

[30]:
```python
# Write your code here
# Ouvre et lit le fichier HTML
with open('students.html') as html_file:
    html_content = html_file.read()

# Parse le contenu HTML
soup = BeautifulSoup(html_content, 'html.parser')

# Trouve la table
table = soup.find('table')

print("Étudiants avec la note 'A':")

# Parcourt chaque ligne de la table (sauf l'en-tête)
for row in table.find_all('tr')[1:]:
    # Extrait les cellules de la ligne
    cells = row.find_all('td')
    if cells:  # Vérifie que la ligne contient des cellules
        name = cells[0].text.strip()  # Première colonne = nom
        grade = cells[2].text.strip() # Troisième colonne = note

        # Affiche le nom si la note est 'A'
        if grade == 'A':
            print(name)
```

```
Étudiants avec la note 'A':
Alice
David
Grace
```

```
Jack
Mia
Paul
Sam
Victor
Yara
```

1.4 [3] XML

[16]:
```
import xml.etree.ElementTree as ET
```

1.4.1 3.1 Writing new xml file

[17]:
```
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")
```

1.4.2 3.2 Modifying existing xml file

[18]:
```
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')
```

1.4.3 3.3 Reading XML file

[19]:
```
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>
```

1.4.4 3.4 Convert XML to List of Dictionary

[20]:
```
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.

[25]:
```
#Write you own code here
import xml.etree.ElementTree as ET

file_name = "xml_file.xml"

tree = ET.parse(file_name)
root = tree.getroot()

new_student = ET.Element("student", name="Romain")

email = ET.SubElement(new_student, "email")
email.text = "blanchot@et.esiea.fr"

age = ET.SubElement(new_student, "age")
age.text = "20"
```

```
gender = ET.SubElement(new_student, "gender")
gender.text = "Male"

root.append(new_student)
tree.write(file_name)

data_list = []

for student in root.findall("student"):
    student_data = {
        "Name": student.get("name"),
        "Email": student.find("email").text,
        "Age": student.find("age").text,
        "Gender": student.find("gender").text
    }
    data_list.append(student_data)

print(data_list)
```

```
[{'Name': 'Alice', 'Email': 'alice@mail.com', 'Age': '22', 'Gender': 'F'},
{'Name': 'Romain', 'Email': 'blanchot@et.esiea.fr', 'Age': '20', 'Gender':
'Male'}]
```

1.5 [4] JSON

[12]:
```
import json
```

1.5.1 4.1 Writing new json file

[13]:
```
# 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)
```

1.5.2 4.2 Reading json file

[14]:
```
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',
'age': 25, 'city': 'San Francisco'}, {'name': 'Charlie', 'age': 35, 'city': 'Los
Angeles'}]}
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:

[15]:
```
import json

with open("json_file", "r", encoding="utf-8") as file :

    data = json.load(file)

for person in data['people']:
    person["job"] = "a job"

with open("json_file", "w", encoding="utf-8") as file :

    json.dump(data, file, indent=4)

print(json.dumps(data, indent=4))
```

```
{
    "people": [
        {
            "name": "Alice",
            "age": 30,
            "city": "New York",
            "job": "a job"
        },
```

```
        {
            "name": "Bob",
            "age": 25,
            "city": "San Francisco",
            "job": "a job"
        },
        {
            "name": "Charlie",
            "age": 35,
            "city": "Los Angeles",
            "job": "a job"
        }
    ]
}
```