Exercises: Data Formats

This document defines the **exercise assignments** for the [Back-End Technologies Basics @ SoftUni](#)
You can check your solutions in [Judge](#)

JSON

1. Books

1.1. Extract information and create a JSON

You are given a **table of five books.**

Each book has the following attributes: **title (string), author (string), released (int), pages (int), ISBN (string).**

Title	Author	Released	Pages	ISBN
In Search of Lost Time	Marcel Proust	1913	4215	978-0-307-70075-2
Ulysses	James Joyce	1922	730	978-0-679-72276-2
Pride and Prejudice	Jane Austen	1813	432	978-1-85326-000-2
Moby Dick	Herman Melville	1851	635	978-0-14-243724-7
Harry Potter and the Sorcerer's Stone	J.K. Rowling	1997	309	978-0-590-35342-7

Convert the table of books data into a **structured JSON format manually:**

- **Use a text or a code editor** to write the JSON document. We recommend **Notepad++ or VS Code**.
- **Extract relevant details** from each book's description.
- **Organize the data** into a structured JSON format:
  - **Each book** should be a **separate object within an array**.
  - **Include** the following keys: **title**, **author**, **released**, **pages**, **ISBN**.

Example:

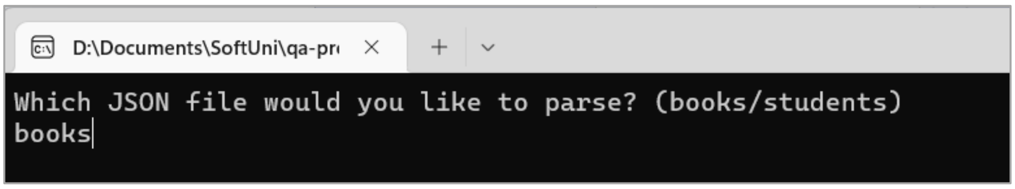
Books.json

```
[
    {
        "title": "In Search of Lost Time",
        "author": "Marcel Proust",
        "released": 1913,
        "pages": 4215,
        "ISBN": "978-0-307-70075-2"
    },
    {
        "title": "Ulysses",
        "author": "James Joyce",
        "released": 1922,
        "pages": 730,
        "ISBN": "978-0-679-72276-2"
    }
    // ... other books ...
]
```

1.2. Use the Provided JSON Parser to Parse the Books

You are provided with a [JSON parser application](#). Use it to **parse and validate** the JSON file you have created.

- **Open** the **parser** application **using Visual Studio**. This application is pre-configured to read JSON files from a specific directory.
- **Within the parser project**, locate the **Datasets folder**. You will find **empty Books.json** file here.
- Open the **existing Books.json** file.
- **Replace the content of Books.json** with the JSON data you created.
- After pasting your JSON data into the coresponding JSON file, **make sure to save any changes**.
- **Run the parser** application within your IDE.
- **The application will ask you which file you would like to use. Type books.**



```
D:\Documents\SoftUni\qa-pr
Which JSON file would you like to parse? (books/students)
books
```

- **The parser will process the chosen JSON file** and display the extracted data **in the console**.
- Carefully review the output in the console.
- If the parser displays an error message, check your JSON file for any syntax errors or formatting issues.
- Ensure all required keys are present and correctly named.
- Confirm that your JSON structure aligns with the examples provided in the assignment.
- **Copy the results from the console into the Judge System (Problem 01. Books)**.

  *Use Ctrl + C to copy from the console.

2. Students

2.1. Extract information and create a JSON

You are given a **list of 5 students**, each described with details like **name**, **age**, and a **list of courses** they are enrolled in. The details are **presented in a sentence format**:

1. "**Alice Johnson**, **20** years old, is enrolled in **Introduction to Computer Science** and **Web Development**."
2. "**Brian Smith**, **22** years old, takes courses in **Machine Learning**, **Artificial Intelligence**, **Computational Theory**, and **Robotics**."
3. "**Charlotte Brown**, **19** years old, studies **Graphic Design** and **Digital Marketing**."
4. "**David Wilson**, **21** years old, focuses on **Cybersecurity**, **Network Infrastructure**, **Cloud Computing**, and **Data Privacy**."
5. "**Ella Davis**, **23** years old, is pursuing **Advanced Mathematics** and **Quantum Mechanics**."

Each student has the following attributes: **name (string), age (int), courses (list of courses);**

Each course has a **name (string).**

Convert the list of students' data into a structured JSON format manually:

- **Use a text or code editor** to create your JSON document. We recommend using **Notepad++ or Visual Studio Code** for better formatting and syntax highlighting.
- **Extract** relevant details **from each student's description**.
- **Organize the data** into a structured JSON format:
  - **Each student** should be a **separate object within an array**.
  - **Include** the following keys: **name**, **age**, **courses**.

- **Each course** should be represented as an **object with key: name,** within the **courses array**.

Example:

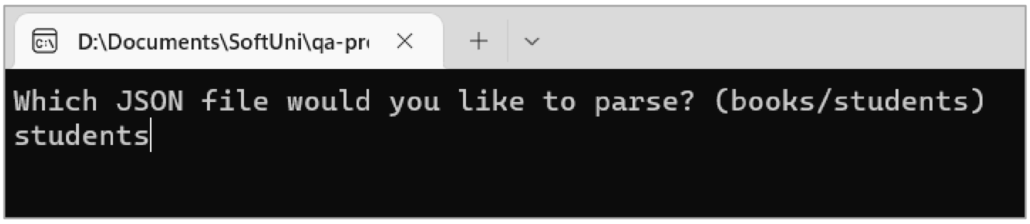
Students.json

```
[
    {
        "name": "Alice Johnson",
        "age": 20,
        "courses": [
            {"name": "Introduction to Computer Science"},
            {"name": "Web Development"}
        ]
    },
    {
        "name": "Brian Smith",
        "age": 22,
        "courses": [
            {"name": "Machine Learning"},
            {"name": "Artificial Intelligence"},
            {"name": "Computational Theory"},
            {"name": "Robotics"}
        ]
    }
    ... other students ...
]
```

2.2. Use the Provided JSON Parser to Parse the Students

Using the same **parser application**. **Parse and validate** the JSON data you have created.

- **Open** the **parser** application **using Visual Studio**. This application is pre-configured to read JSON files from a specific directory.
- **Within the parser project**, locate the **Datasets folder**. You will find an **empty Students.json** file here.
- Open the **existing Students.json** file.
- **Replace the contents of Students.json** with the JSON data you created.
- After pasting your JSON data into the corresponding JSON file, **make sure to save any changes**.
- **Run the parser** application within your IDE.
- The application will ask you **which file you would like to use**. **Type students**.



```
Which JSON file would you like to parse? (books/students)
students
```

- The parser will process the chosen JSON file and display the extracted data in the console.
- Carefully review the output in the console.
- If the parser displays an error message, check your JSON file for any syntax errors or formatting issues.
- Ensure all required keys are present and correctly named.
- Confirm that your JSON structure aligns with the examples provided in the assignment.
- **Copy the results from the console into the Judge System (Problem 02. Students).**
  *Use Ctrl + C to copy from the console.

YAML

3. Orders

3.1. Extract information and create a YAML file

You are given a **table of six orders**, each with **order_id (int)**, **customer (string)**, **item (string)**, **quantity (int)**, and **total_amount (float)**.

Order ID	Customer	Item	Quantity	Total Amount
1001	John Doe	Wireless Mouse	3	29.97
1002	Emily Clark	16GB USB Drives	2	31.96
1003	Alex Johnson	External Hard Drive	1	89.99
1004	Sarah Smith	Smartphone Cases	4	39.96
1005	Michael Lee	Digital Camera	1	120.50
1006	Karen Thompson	Bluetooth Speakers	2	58.00

Convert the table of orders data into a **structured YAML format manually:**

- **Use a text or a code editor** to write the YAML. We recommend **Notepad++ or Visual Studio Code**.
- **Extract relevant details** from each order's description.
  - **Organize** the data into a **structured YAML format**. **Each order** should be a **separate entry in the list**.
  - Include **keys**: **order_id**, **customer**, **item**, **quantity**, and **total_amount**.

Example:

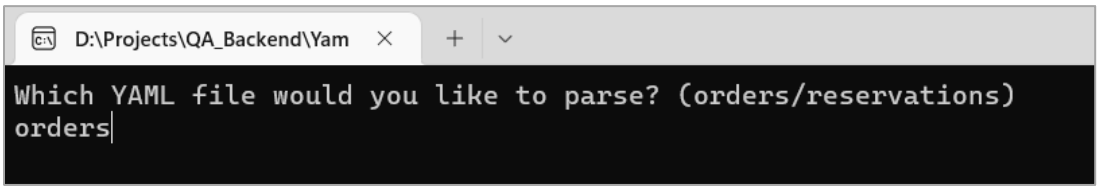
Orders.yaml

```yaml
- order_id: 1001
  customer: John Doe
  item: Wireless Mouse
  quantity: 3
  total_amount: 29.97
- order_id: 1002
  customer: Emily Clark
  #continue with the rest
```

3.2. Parse the Orders from YAML to HTML

You are provided with YAML to HTML parser application. **Parse and validate** the YAML data you have created.

- **Open** the **parser** application **using Visual Studio**. This application is pre-configured to read YAML files from a specific directory.
- **Within the parser project**, locate the **Datasets folder**. You will find the **empty Orders.yaml** file here.
- Open the **existing Orders.yaml** file.
- **Replace the content of Orders.yaml** with the YAML data that you created.
- After pasting your YAML data into the corresponding YAML file, **make sure to save any changes**.
- **Run the parser** application within your IDE.
- The application will ask you **which file you would like to use**. **Type orders**.



```
Which YAML file would you like to parse? (orders/reservations)
orders
```

- The parser **will process the chosen YAML file** and display the extracted data in **your default browser**.
  *** If asked if you're allowing to open the output in the browser, choose yes.**
- If the parser displays an error message, check your YAML file for any syntax errors or formatting issues.
- Ensure all required keys are present and correctly named.
- Confirm that your YAML structure aligns with the examples provided in the assignment.
- Carefully review the output in the browser.
- **Copy the results from the browser into the Judge System (Problem 03. Orders).**

4. Reservations

4.1. Extract information and create a YAML file

You are given a **table of 5 reservations**, each with **reservation_id**, **guest_name**, **and** list of **services**. Each service has **type, date and time**.

Each reservation has **reservation_id (int)**, **guest_name (string)**, list of **services**.

Each service has **type (string), date (string), time (string)**

Reservation ID	Guest Name	Services
101	Emma Johnson	**Spa** June 15th 2 PM **Dinner** June 16th 8 PM
102	John Davis	**Golf** June 17th 10 AM **Wine Tasting** June 18th 5 PM
103	Sophia Lee	**Yoga Class** June 19th 8 AM **Brunch** June 20th 11 AM
104	Michael Brown	**Cooking Workshop** June 21st 4 PM **Movie Night** June 22nd 9 PM
105	Olivia Smith	**Deep Sea Fishing**

		June 23rd 7 AM **Evening Cruise** June 24th 6 PM

Convert the list of reservations data into a **structured YAML format manually:**

- **Use a text or a code editor** to write the YAML. We recommend **Notepad++ or Visual Studio Code**.
- **Extract relevant details** from each reservation's description.
  - **Organize** the data into a **structured YAML format**. **Each reservation** should be a **separate entry in the list**.
  - Include **keys**: **reservation_id**, **guest_name**, and **services**.
  - **Each service** should be represented as an **object within the services array**.
  - Each service has the following keys: **type, date, time**

Example:

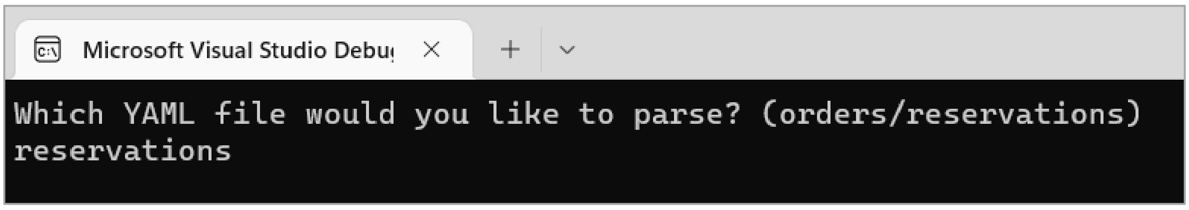
Reservations.yaml

```yaml
- reservation_id: 101
  guest_name: Emma Johnson
  services:
    - type: Spa
      date: June 15
      time: 2 PM
    - type: Dinner
      date: June 16
      time: 8 PM
- reservation_id: 102
  guest_name: John Davis
  #continue with the rest
```

4.2. Parse the Reservations from YAML to HTML

Using the same YAML to HTML parser application. **Parse and validate the YAML data** you have created.

- **Open** the **parser** application using **Visual Studio**. This application is pre-configured to read YAML files from a specific directory.
- **Within** the **parser project**, locate the **Datasets folder**. You will find the empty **Reservations.yaml** file here.
- Open the existing **Reservations.yaml**.
- Replace the content of **Reservations.yaml** with the YAML data you created. Be sure to overwrite any existing content if the files are not empty.
- After pasting your YAML data into the corresponding YAML file, make sure to **save any changes**.
- Run the parser application within your IDE.
- The application will ask you which file you would like to use. **Type reservations.**



```
Which YAML file would you like to parse? (orders/reservations)
reservations
```

- The **parser will process** the chosen **YAML file** and display the extracted data **in your default browser.**
  *** If asked if you're allowing to open the output in the browser, choose yes.**
- If the parser displays an error message, check your JSON file for any syntax errors or formatting issues.
- Ensure all required keys are present and correctly named.
- Confirm that your YAML structure aligns with the examples provided in the assignment.
- Carefully review the output in the browser.
- **Copy the results from the browser into the Judge System (Problem 04. Reservations).**

XML

5. Devices

5.1. Extract information and create an XML

You are given a **table with five devices**, each with **type**, **brand**, **specs**, and **price.**

Type	Brand	Specs	Price
Laptop	Dell XPS 13	13.4-inch display	1200
Smartphone	Apple iPhone 12	64GB storage	799
Tablet	Samsung Galaxy Tab S7	11-inch screen	650
Headphones	Bose QuietComfort 35 II	Noise-cancelling	299
Camera	Canon EOS Rebel T7 DSLR	24.1 MP	449

Convert the list of orders data into a **structured XML format manually:**

- Write the **XML document** using a text or a code editor. We recommend using **Notepad++ or Visual Studio Code** for better formatting and syntax highlighting.
- Carefully read each device's description and **identify key information.** You should **extract** the **following details:**
- **Type of device** (e.g., Laptop, Smartphone)
- **Brand** (e.g., Dell XPS 13, Apple iPhone 12)
- **Specs** (e.g., screen size, processor, storage)
- **Price**
- **Create an XML document** where **each device** is represented as a **separate entry.** Structure your XML with **appropriate tags.**

Example:

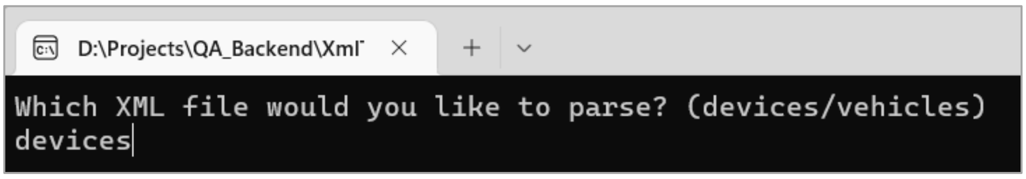
Devices.xml

```xml
<devices>
    <device>
        <type>Laptop</type>
        <brand>Dell XPS 13</brand>
        <specs>13.4-inch display </specs>
        <price>200</price>
    </device>
    <!-- More device entries here -->
</devices>
```

5.2. Parse Devices from XML to JSON

You are provided with [XML to JSON parser](). **Parse and validate** the XML data you have created.

- **Open** the **parser** application **using Visual Studio**. This application is pre-configured to read XML documents from a specific directory.
- **Within the parser project**, locate the **Datasets folder**. You will find an **empty Devices.xml** file here.
- Open the **existing Devices.xml**.
- **Replace the content of Devices.xml** with the XML data you created.
- After pasting your XML data into the corresponding XML file, **make sure to save any changes**.
- **Run the parser** application within your IDE.
- The application will ask you **which file you would like to use**. **Type devices**.



```
Which XML file would you like to parse? (devices/vehicles)
devices
```

- The parser **will process the chosen XML file** and display the extracted data in **JSON format on the console.**
- If the parser displays an error message, check your XML for any syntax errors or formatting issues.
- Ensure the required XML structure and the **appropriate tags.**
- Confirm that your XML structure aligns with the examples provided in the assignment.
- Carefully review the output in the console.
- **Copy the results from the console into the Judge System (Problem 05. Devices).**

6. Vehicles

6.1. Extract information and create an XML

You are given a **list of five vehicles**, each with **type**, **model**, **specs**, and **color**.

1. "**Car**: **Tesla Model 3; Electric sedan**; **red**"
2. "**Motorcycle**: **Harley-Davidson**; **V-twin engine**; **black**"
3. "**Bicycle**: **Giant Escape 3**; **Aluminum frame**; **black**"
4. "**Scooter**: **Vespa Primavera**; **50cc engine**; **white**"
5. "**Boat**: **Bayliner Element**; **18-foot length**; **black**"

Convert the list of orders data into a **structured <u>XML format manually:</u>**

- Write the **XML document** using a text or a code editor. We recommend using **Notepad++ or Visual Studio Code** for better formatting and syntax highlighting.
- Carefully read each device's description and **identify key information.** You should **extract** the **following details:**
- **Type** (e.g., Car, Bicycle)
- **Model** (e.g., Harley-Davidson)
- **Specs** (e.g., Electric sedan)

- **Color** (e.g., red)
- **Create an XML document** where **each device** is represented as a **separate entry.** Structure your XML with **appropriate tags.**

Example:

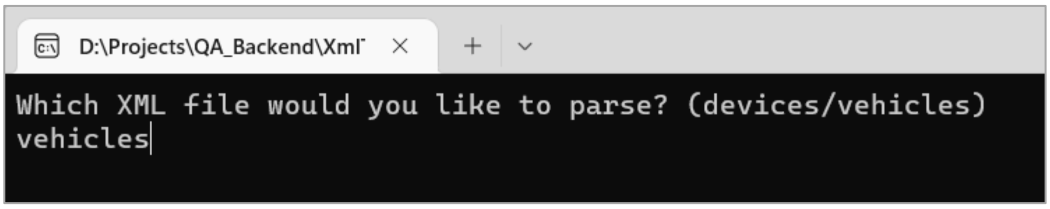
Vehicles.xml

`<vehicles>`
`    <vehicle>`
`        <type>Car</type>`
`        <model>Tesla Model 3</model>`
`        <specs>Electric sedan</specs>`
`        <color>red</color>`
`    </vehicle>`
`<!-- More vehicle entries here -->`
`</vehicles>`

6.2. Parse the Vehicles from XML to JSON

Using the same **XML to JSON parser**. **Parse and validate** the XML data you have created.

- **Open** the **parser** application **using Visual Studio**. This application is pre-configured to read XML documents from a specific directory.
- **Within the parser project**, locate the **Datasets folder**. You will find **Vehicles.xml** file here.
- Open the **existing Vehicles.xml** file.
- **Replace the content of Vehicles.xml** with the XML data you created.
- After pasting your XML data into the corresponding XML file, **make sure to save any changes**.
- **Run the parser** application within your IDE.
- The application will ask you **which file you would like to use**. **Type vehicles**.



```
Which XML file would you like to parse? (devices/vehicles)
vehicles
```

- The parser **will process the chosen XML file** and display the extracted data in **JSON format on the console.**
- If the parser displays an error message, check your XML for any syntax errors or formatting issues.
- Ensure the required XML structure and the **appropriate tags.**
- Confirm that your XML structure aligns with the examples provided in the assignment.
- Carefully review the output in the console.
- **Copy the results from the console into the Judge System (Problem 06. Vehicles).**