

ISLAMIC UNIVERSITY OF TECHNOLOGY



RELATIONAL DATABASE MANAGEMENT SYSTEM LAB

CSE 4508

Lab 5: Complex Data Types

Author:

Ahmed M. S. Albreem (210041258)

November 5, 2024

Task 1: Parse JSON Data

Objective: Parse the provided JSON data and print the store's "name" and its "city" location.

Explanation

This task involves reading a JSON object containing details about a store, such as its name, location, and departments. The goal is to extract and display the store's name and the city where it is located. JSON (JavaScript Object Notation) is a lightweight data interchange format that's easy for humans to read and write, and easy for machines to parse and generate.

Code

```
1 import json
2
3 # Load the JSON data
4 json_data = '''{
5     "store": {
6         "name": "Tech Hub",
7         "location": {
8             "city": "San Francisco",
9             "address": {
10                 "street": "123 Market St",
11                 "postalCode": "94103"
12             }
13         },
14         "departments": [
15             {
16                 "name": "Electronics",
17                 "products": [
18                     {
19                         "id": 201,
20                         "name": "Laptop",
21                         "brand": "BrandX",
22                         "price": 999.99,
23                         "specifications": {
24                             "processor": "Intel i7",
25                             "memory": "16GB",
26                             "storage": "512GB SSD"
27                         }
28                     },
29                 ],
30             }
31         ]
32     }
33 }'''
```

```

29      {
30          "id": 202,
31          "name": "Smartphone",
32          "brand": "BrandY",
33          "price": 799.99,
34          "specifications": {
35              "screenSize": "6.1 inches",
36              "battery": "4000mAh",
37              "camera": "12MP"
38          }
39      }
40  ],
41 },
42 {
43     "name": "Accessories",
44     "products": [
45         {
46             "id": 301,
47             "name": "Wireless Mouse",
48             "brand": "BrandZ",
49             "price": 29.99,
50             "specifications": {
51                 "batteryLife": "12 months",
52                 "connectivity": "Bluetooth"
53             }
54         },
55         {
56             "id": 302,
57             "name": "Keyboard",
58             "brand": "BrandX",
59             "price": 49.99,
60             "specifications": {
61                 "layout": "QWERTY",
62                 "connectivity": "Wireless"
63             }
64         }
65     ]
66 },
67 ],
68 }
69 },
70 ,
71
72 data = json.loads(json_data)
73
74 # Extract the store's name and city location

```

```

75 store_name = data["store"]["name"]
76 city = data["store"]["location"]["city"]
77
78 print(f"Store Name: {store_name}")
79 print(f"City: {city}")
80
81
82 # what i learn at all the tasks deoumented here in initalization of
# the tasks
83 # i just have been rembered by teh python strucutre on interaction
# with json
84 # which the head of the stucture of the json is the key of the json
# object presented as [] key
85 # if i want to extract one of the instead head i can use [] the
# looped on the structured
86 # else if i can't expect the strucutre i can make the biggest loop
# to check the value on the strucutere as next on the heads of it
87 # the heads present with it's name with for the next one
88 # next() is from the json imported on the pyhton
89 # product defined as the strucure as variable to hold the data for
# the each next operation as check condition and so on

```

Output

```

PS C:\Users\ahmed> & C:/Users/ahmed/AppData/Local/Microsoft/
WindowsApps/python3.12.exe "c:/Users/ahmed/Desktop/lab tasks/210041258_task1_.py"
Store Name: Tech Hub
City: San Francisco
PS C:\Users\ahmed>

```

Task 2: Print Product Names and Prices

Objective: Iterate through the products in the JSON data and print each product's name and price.

Explanation

The task requires iterating through a list of products contained in the JSON data to extract and print the name and price of each product. This demonstrates handling nested data structures within JSON.

Code

```
1 import json
2
3 json_data = '''{
4     "store": {
5         "name": "Tech Hub",
6         "location": {
7             "city": "San Francisco",
8             "address": {
9                 "street": "123 Market St",
10                "postalCode": "94103"
11            }
12        },
13        "departments": [
14            {
15                "name": "Electronics",
16                "products": [
17                    {
18                        "id": 201,
19                        "name": "Laptop",
20                        "brand": "BrandX",
21                        "price": 999.99,
22                        "specifications": {
23                            "processor": "Intel i7",
24                            "memory": "16GB",
25                            "storage": "512GB SSD"
26                        }
27                    },
28                    {
29                        "id": 202,
30                        "name": "Smartphone",
31                        "brand": "Brandy",
32                        "price": 799.99,
33                        "specifications": {
34                            "screenSize": "6.1 inches",
35                            "battery": "4000mAh",
36                            "camera": "12MP"
37                        }
38                    }
39                ]
40            },
41            {
42                "name": "Accessories",
43                "products": [
44                    {
```

```

45         "id": 301,
46         "name": "Wireless Mouse",
47         "brand": "BrandZ",
48         "price": 29.99,
49         "specifications": {
50             "batteryLife": "12 months",
51             "connectivity": "Bluetooth"
52         }
53     },
54     {
55         "id": 302,
56         "name": "Keyboard",
57         "brand": "BrandX",
58         "price": 49.99,
59         "specifications": {
60             "layout": "QWERTY",
61             "connectivity": "Wireless"
62         }
63     }
64 ]
65 ]
66 }
67 }
68 }
69 '',
70 data = json.loads(json_data)
71
72 #Task 2: Access and print the names and prices of all products in
73 # the "Electronics" department.
74 # Extract the Electronics department
75 electronics = next(dept for dept in data["store"]["departments"] if
76     dept["name"] == "Electronics")
77
78 # Print names and prices of products in Electronics
79 for product in electronics["products"]:
80     print(f"Product Name: {product['name']}, Price: {product['price']}")
```

Output

```

PS C:\Users\ahmed> & C:/Users/ahmed/AppData/Local/Microsoft/
WindowsApps/python3.12.exe "c:/Users/ahmed/Desktop/lab tasks/210041258_task2.py"
Product Name: Laptop, Price: 999.99
Product Name: Smartphone, Price: 799.99
```

PS C:\Users\ahmed>

Task 3: Print Specific Product Details

Objective: Print details of the "Wireless Mouse" product, including brand and specifications.

Explanation

This task focuses on retrieving and displaying detailed information about a specific product. The details include the product's brand and specifications, such as battery life and connectivity options.

Code

```
1 import json
2
3
4 json_data = '''{
5     "store": {
6         "name": "Tech Hub",
7         "location": {
8             "city": "San Francisco",
9             "address": {
10                 "street": "123 Market St",
11                 "postalCode": "94103"
12             }
13         },
14         "departments": [
15             {
16                 "name": "Electronics",
17                 "products": [
18                     {
19                         "id": 201,
20                         "name": "Laptop",
21                         "brand": "BrandX",
22                         "price": 999.99,
23                         "specifications": {
24                             "processor": "Intel i7",
25                             "memory": "16GB",
26                             "storage": "512GB SSD"
27                         }
28                     }
29                 ]
30             }
31         ]
32     }
33 }
```

```

28     },
29     {
30         "id": 202,
31         "name": "Smartphone",
32         "brand": "BrandY",
33         "price": 799.99,
34         "specifications": {
35             "screenSize": "6.1 inches",
36             "battery": "4000mAh",
37             "camera": "12MP"
38         }
39     }
40 ],
41 },
42 {
43     "name": "Accessories",
44     "products": [
45         {
46             "id": 301,
47             "name": "Wireless Mouse",
48             "brand": "BrandZ",
49             "price": 29.99,
50             "specifications": {
51                 "batteryLife": "12 months",
52                 "connectivity": "Bluetooth"
53             }
54         },
55         {
56             "id": 302,
57             "name": "Keyboard",
58             "brand": "BrandX",
59             "price": 49.99,
60             "specifications": {
61                 "layout": "QWERTY",
62                 "connectivity": "Wireless"
63             }
64         }
65     ]
66 },
67 ],
68 }
69 '',
70 data = json.loads(json_data)
71
72

```

```

73 #Task 3: Extract and print the "brand" and "specifications" of the
74     product with the name "Wireless Mouse".
75
75 # Extract the Accessories department
76 accessories = next(dept for dept in data["store"]["departments"] if
77     dept["name"] == "Accessories")
78
78 # Find the Wireless Mouse
79 wireless_mouse = next(product for product in accessories["products"]
80     if product["name"] == "Wireless Mouse")
81
81 # Print brand and specifications
82 print(f"Brand: {wireless_mouse['brand']}")  

83 print("Specifications:")
84 for key, value in wireless_mouse["specifications"].items():
85     print(f"  {key}: {value}")

```

Output

```

PS C:\Users\ahmed> & C:/Users/ahmed/AppData/Local/
Microsoft/WindowsApps/python3.12.exe
"c:/Users/ahmed/Desktop/lab tasks/210041258_task3.py"
Brand: BrandZ
Specifications:
  batteryLife: 12 months
  connectivity: Bluetooth
PS C:\Users\ahmed>

```

Task 4: Print the JSON Data

Objective: Display the entire JSON structure of the store.

Explanation

The aim of this task is to print the complete JSON structure, showcasing all the store's details, including name, location, departments, and products. This helps in understanding the overall structure and content of the JSON data.

Code

```

1 import json
2
3
4 json_data = '''{
5     "store": {
6         "name": "Tech Hub",
7         "location": {
8             "city": "San Francisco",
9             "address": {
10                 "street": "123 Market St",
11                 "postalCode": "94103"
12             }
13         },
14         "departments": [
15             {
16                 "name": "Electronics",
17                 "products": [
18                     {
19                         "id": 201,
20                         "name": "Laptop",
21                         "brand": "BrandX",
22                         "price": 999.99,
23                         "specifications": {
24                             "processor": "Intel i7",
25                             "memory": "16GB",
26                             "storage": "512GB SSD"
27                         }
28                     },
29                     {
30                         "id": 202,
31                         "name": "Smartphone",
32                         "brand": "Brandy",
33                         "price": 799.99,
34                         "specifications": {
35                             "screenSize": "6.1 inches",
36                             "battery": "4000mAh",
37                             "camera": "12MP"
38                         }
39                     }
40                 ]
41             },
42             {
43                 "name": "Accessories",
44                 "products": [
45                     {

```

```

46         "id": 301,
47         "name": "Wireless Mouse",
48         "brand": "BrandZ",
49         "price": 29.99,
50         "specifications": {
51             "batteryLife": "12 months",
52             "connectivity": "Bluetooth"
53         }
54     },
55     {
56         "id": 302,
57         "name": "Keyboard",
58         "brand": "BrandX",
59         "price": 49.99,
60         "specifications": {
61             "layout": "QWERTY",
62             "connectivity": "Wireless"
63         }
64     }
65 ]
66 ]
67 }
68 }
69 }
70 ,,
71 # Load the JSON data
72 data = json.loads(json_data)
73
74 # Find the Electronics department
75 electronics = next(department for department in data["store"]["departments"] if department["name"] == "Electronics")
76
77 # Find the Smartphone in Electronics
78 smartphone = next(product for product in electronics["products"] if product["name"] == "Smartphone")
79
80 # Update the price
81 smartphone["price"] = 749.99
82
83 # Print the updated JSON
84 updated_json = json.dumps(data, indent=4)
85 print(updated_json)

```

Output

```
PS C:\Users\ahmed> & C:/Users/ahmed/AppData/Local/Microsoft/WindowsApps/python3.12.exe
{
    "store": {
        "name": "Tech Hub",
        "location": {
            "city": "San Francisco",
            "address": {
                "street": "123 Market St",
                "postalCode": "94103"
            }
        },
        "departments": [
            {
                "name": "Electronics",
                "products": [
                    {
                        "id": 201,
                        "name": "Laptop",
                        "brand": "BrandX",
                        "price": 999.99,
                        "specifications": {
                            "processor": "Intel i7",
                            "memory": "16GB",
                            "storage": "512GB SSD"
                        }
                    },
                    {
                        "id": 202,
                        "name": "Smartphone",
                        "brand": "BrandY",
                        "price": 749.99,
                        "specifications": {
                            "screenSize": "6.1 inches",
                            "battery": "4000mAh",
                            "camera": "12MP"
                        }
                    }
                ]
            }
        ]
    }
}
```

```

        ],
    },
    {
        "name": "Accessories",
        "products": [
            {
                "id": 301,
                "name": "Wireless Mouse",
                "brand": "BrandZ",
                "price": 29.99,
                "specifications": {
                    "batteryLife": "12 months",
                    "connectivity": "Bluetooth"
                }
            },
            {
                "id": 302,
                "name": "Keyboard",
                "brand": "BrandX",
                "price": 49.99,
                "specifications": {
                    "layout": "QWERTY",
                    "connectivity": "Wireless"
                }
            }
        ]
    }
}
PS C:\Users\ahmed>

```

SQLite Task

Objective: Perform database operations using SQLAlchemy and log the output.

Explanation

This task involves creating a database using SQLite and performing various operations such as inserting, updating, selecting, and deleting records. SQLAlchemy is used to interact with the SQLite database in Python. Logging is enabled to monitor these operations.

Code

```
1 from sqlalchemy.orm import sessionmaker
2 from sqlalchemy import create_engine, Column, Integer, String, Float
3 from sqlalchemy.ext.declarative import declarative_base
4
5 engine = create_engine('sqlite:///books.db', echo=True)
6 Base = declarative_base()
7
8
9 class Book(Base):
10     __tablename__ = 'books'
11
12     id = Column(Integer, primary_key=True)
13     title = Column(String, nullable=False)
14     author = Column(String, nullable=False)
15     published_year = Column(Integer, nullable=False)
16     price = Column(Float, nullable=False)
17
18 Base.metadata.create_all(engine)
19
20 Session = sessionmaker(bind=engine)
21 session = Session()
22
23 # Tracking the status of each book
24 book_status = {}
25
26 # Add a new book
27 new_book = Book(title="The Great Gatsby", author="F. Scott
28     Fitzgerald", published_year=1925, price=10.99)
29 session.add(new_book)
30 session.commit()
31 book_status[new_book.title] = 'added'
32
33 books = session.query(Book).all()
34 for book in books:
35     status = book_status.get(book.title, 'unknown')
```

```

35     print(f"Id : {book.id}, Title: {book.title}, Author: {book.author}
36         }, Year: {book.published_year}, Price: {book.price}, Status:
37         {status})")
38
39 book_to_update = session.query(Book).filter_by(title="The Great
40     Gatsby").first()
41 if book_to_update:
42     book_to_update.price = 12.99
43     session.commit()
44     book_status[book_to_update.title] = 'updated'
45
46 books = session.query(Book).all()
47 for book in books:
48     status = book_status.get(book.title, 'unknown')
49     print(f"Id : {book.id}, Title: {book.title}, Author: {book.author}
50         }, Year: {book.published_year}, Price: {book.price}, Status:
51         {status})")
52
53 book_to_delete = session.query(Book).filter_by(title="The Great
54     Gatsby").first()
55 if book_to_delete:
56     session.delete(book_to_delete)
57     session.commit()
58     book_status[book_to_delete.title] = 'deleted'
59
60 books = session.query(Book).all()
61 for book in books:
62     status = book_status.get(book.title, 'unknown')
63     print(f"Id : {book.id}, Title: {book.title}, Author: {book.author}
64         }, Year: {book.published_year}, Price: {book.price}, Status:
65         {status})")
66
67 if "The Great Gatsby" in book_status:
68     print(f"Title: The Great Gatsby, Status: {book_status['The Great
69         Gatsby']}")
70
71
72
73
74
75 # creation of the session is important
76 # i see that the session is created and commit each time
77 # i see also that table on createion with attributte initialization on
78     __tablename__ = 'books' as branch on the database
79 # i relise the use of the declarative_base() function to create a
80     base class for declarative class definitions

```

```

69 # i see that the use of the Column() function to define the columns
    of the table
70 # i see that the use of the Integer, String, Float to define the
    type of the columns and of curse the nullable attrubute and the
    nulability of each column defined
71 # i also make the stamts of the book to check the operation of the
    book to confirm the operation side becuase on sqllite i configure
    that
72 # it depneds on extra data shonen up on it which its musn't be
    appered on the output
73 # i added on it to be shonen for any one will see the code so on

```

Output

```

PS C:\Users\ahmed> & C:/Users/ahmed/AppData/Local/Microsoft/WindowsApps/python3.12.exe
c:\Users\ahmed\Desktop\lab tasks\210041258_sqlite.py:6: MovedIn20Warning: The ``declarative_base`` is deprecated. Please use ``declarative_base`` instead.
  Base = declarative_base()
2024-11-05 09:11:34,601 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-05 09:11:34,601 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("books")
2024-11-05 09:11:34,601 INFO sqlalchemy.engine.Engine [raw sql] ()
2024-11-05 09:11:34,602 INFO sqlalchemy.engine.Engine COMMIT
2024-11-05 09:11:34,603 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-05 09:11:34,608 INFO sqlalchemy.engine.Engine INSERT INTO books (title, author, year, price) VALUES (?, ?, ?, ?)
2024-11-05 09:11:34,608 INFO sqlalchemy.engine.Engine [generated in 0.00075s] ('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 10.99)
2024-11-05 09:11:34,610 INFO sqlalchemy.engine.Engine COMMIT
2024-11-05 09:11:34,614 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-05 09:11:34,618 INFO sqlalchemy.engine.Engine SELECT books.id AS books_id, books.title AS books_title, books.author AS books_author, books.year AS books_year, books.price AS books_price
FROM books
WHERE books.id = ?
2024-11-05 09:11:34,618 INFO sqlalchemy.engine.Engine [generated in 0.00100s] (1,)
2024-11-05 09:11:34,619 INFO sqlalchemy.engine.Engine SELECT books.id AS books_id, books.title AS books_title, books.author AS books_author, books.year AS books_year, books.price AS books_price
FROM books
2024-11-05 09:11:34,621 INFO sqlalchemy.engine.Engine [generated in 0.00058s] ()
Id : 1,Title: The Great Gatsby, Author: F. Scott Fitzgerald, Year: 1925, Price: 10.99
2024-11-05 09:11:34,622 INFO sqlalchemy.engine.Engine SELECT books.id AS books_id, books.title AS books_title, books.author AS books_author, books.year AS books_year, books.price AS books_price
FROM books

```

```
WHERE books.title = ?
    LIMIT ? OFFSET ?
2024-11-05 09:11:34,623 INFO sqlalchemy.engine.Engine [generated in 0.00051s] ('The Great Gatsby', 12.99)
2024-11-05 09:11:34,624 INFO sqlalchemy.engine.Engine UPDATE books SET price=? WHERE books.id = ?
2024-11-05 09:11:34,625 INFO sqlalchemy.engine.Engine [generated in 0.00060s] (12.99, 1)
2024-11-05 09:11:34,626 INFO sqlalchemy.engine.Engine COMMIT
2024-11-05 09:11:34,629 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-05 09:11:34,632 INFO sqlalchemy.engine.Engine SELECT books.id AS books_id, books.title AS books_title, books.year AS books_year, books.price AS books_price, books.status AS books_status FROM books
WHERE books.id = ?
2024-11-05 09:11:34,632 INFO sqlalchemy.engine.Engine [cached since 0.01468s ago] (1,)
2024-11-05 09:11:34,633 INFO sqlalchemy.engine.Engine SELECT books.id AS books_id, books.title AS books_title, books.year AS books_year, books.price AS books_price, books.status AS books_status FROM books
WHERE books.id = ?
2024-11-05 09:11:34,633 INFO sqlalchemy.engine.Engine [cached since 0.01398s ago] ()
Id : 1,Title: The Great Gatsby, Author: F. Scott Fitzgerald, Year: 1925, Price: 12.99
2024-11-05 09:11:34,634 INFO sqlalchemy.engine.Engine SELECT books.id AS books_id, books.title AS books_title, books.year AS books_year, books.price AS books_price, books.status AS books_status FROM books
WHERE books.title = ?
    LIMIT ? OFFSET ?
2024-11-05 09:11:34,634 INFO sqlalchemy.engine.Engine [cached since 0.01334s ago] ('The Great Gatsby')
2024-11-05 09:11:34,634 INFO sqlalchemy.engine.Engine DELETE FROM books WHERE books.id = ?
2024-11-05 09:11:34,634 INFO sqlalchemy.engine.Engine [generated in 0.00032s] (1,)
2024-11-05 09:11:34,634 INFO sqlalchemy.engine.Engine COMMIT
2024-11-05 09:11:34,640 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-05 09:11:34,641 INFO sqlalchemy.engine.Engine SELECT books.id AS books_id, books.title AS books_title, books.year AS books_year, books.price AS books_price, books.status AS books_status FROM books
WHERE books.id = ?
2024-11-05 09:11:34,641 INFO sqlalchemy.engine.Engine [cached since 0.02129s ago] ()
Title: The Great Gatsby, Status: deleted
PS C:\Users\ahmed>
* History restored
```