Interest Book in Python (Jupyter Notebook)

1. Introduction

This project is part of the Unit 10 submission for the Programming and Testing module. It extends the work completed in Part 1, where the initial logic and pseudocode were implemented in PHP. For Part 2, the application has been rewritten in Python and executed in a Jupyter Notebook, in line with academic requirements for algorithm development, testing and validation.

The transition from PHP to Python was made intentionally to demonstrate the underlying logic in a different language and to take advantage of Jupyter Notebook's interactive capabilities for testing and documentation. With over 25 years of programming experience, including extensive work in PHP and Python (see linkedin.com/in/victorangelier), I am comfortable working across languages and platforms. This assignment reflects both my academic progress and professional experience in data-driven development.

The goal remains the same: to build a personal digital repository for storing, organising, and retrieving online resources using structured data and well-tested logic. All development was done without external libraries, in accordance with the assignment constraints.

2. Application Description

The Interest Book allows users to:

- Add new resource records
- Search for resources using tags or keywords
- Delete existing records
- Edit records by URI
- Sort all records by creation date
- Interact via a simple, menu-driven interface

Each record contains:

- A title (string)
- A URI (string)
- Between 1 and 5 descriptive tags (list of strings)
- A created_at timestamp
- An accessed_at timestamp
- Records are stored in memory using a Python list of dictionaries, each representing a resource. The application uses only core Python features, in line with algorithmic and data structure principles from the module (Cormen et al., 2009).

3. Data Structure and Algorithm Design

The application's core data structure is a list of records (records[]), where each record is a Python dictionary:

```
{
   "title": "Machine Learning Basics",
   "uri": "https://ml.org",
   "tags": ["ai", "ml"],
   "created_at": "2025-04-01T09:45:00",
   "accessed_at": "2025-04-01T09:45:00"
}
```

Key algorithms implemented include:

- **Bubble sort** for ordering by created_at
- **Linear search** for searching by tags, keyword, or URI
- **Validation** logic for tag limits and URI uniqueness
- **Update and deletion logic** with confirmation prompts

4. Interface Design

A menu loop is provided in the function main_menu(), which enables interaction via keyboard input. Each user action (add, search, delete, edit, sort) is accessible via a numbered menu, and users are prompted for input as required.

Before deleting or editing a record, users are asked for confirmation, aligning with the requirement to “get a prompt before an action is performed”.

5. Testing Strategy

Testing was based on the plan defined in Part 1 (Section 3), covering:

- Normal cases (happy flow) – e.g. adding valid records, retrieving known tags
- Boundary tests – e.g. exceeding the tag limit, searching for missing tags, deleting a non-existent record
- Input validation – e.g. empty fields, incorrect types

All tests were executed in Jupyter Notebook cells, each preceded by a markdown heading. Screenshots were taken of the output and are included with the submission.

6. How to Run

To run the Interest Book application:

- Install Python 3 and Jupyter Notebook (via pip install notebook)
- Launch Jupyter by running jupyter notebook in a terminal
- Open the notebook file (e.g. interest_book.ipynb)
- Run the code cells to initialise the program
- Execute main_menu() to start using the interface

7. Reflection and Design Justification

This implementation required careful attention to algorithmic thinking. Searching, sorting, and validation routines were implemented manually to demonstrate understanding of fundamental principles. Python’s in-built flexibility allowed for readable, structured code while maintaining the restrictions of no external libraries.

All code is commented for clarity. Key design decisions, such as storing timestamps using ISO format and limiting tags to five, stem from the original PHP design in Part 1. Bubble sort was chosen for simplicity and alignment with the algorithms studied (Knuth, 1997).

8. Academic Integrity

All code was written by the student, following the design principles established in Part 1. No third-party libraries or code generation tools were used. References are included below.

9. References

- Cormen, T.H., Leiserson, C.E., Rivest, R.L. and Stein, C., 2009. Introduction to Algorithms. 3rd ed. MIT Press.
- Knuth, D.E., 1997. The Art of Computer Programming, Volume 3: Sorting and Searching. 2nd ed. Addison-Wesley.
- Russell, S. and Norvig, P., 2010. Artificial Intelligence: A Modern Approach. 3rd ed. Pearson.

Screenshots

localhost:8888/notebooks/Untitled.ipynb

jupyter Untitled Last Checkpoint: 12 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[2]:
```python
from datetime import datetime

# Dataset
records = []

# Record-structure:
# {
#   "title": str,
#   "uri": str,
#   "tags": list[str],
#   "created_at": str (ISO),
#   "accessed_at": str (ISO)
# }

def add_record(title, uri, tags):
    if not title or not uri or not tags:
        print("Fout: Titel, URI en tags zijn verplicht.")
        return
    if len(tags) < 1 or len(tags) > 5:
        print("Fout: Je moet tussen de 1 en 5 tags gebruiken.")
        return

    timestamp = datetime.now().isoformat()
    record = {
        "title": title,
        "uri": uri,
        "tags": tags,
        "created_at": timestamp,
        "accessed_at": timestamp
    }
    records.append(record)
    print(f"✅ Record toegevoegd: {title}")
```

[3]:
```python
add_record("Testlink", "https://www.thecodingcompany.nl", ["ai", "datastructures"])
```

```
✅ Record toegevoegd: Testlink
```

[5]:
```python
print(records)
```

```
[{'title': 'Testlink', 'uri': 'https://www.thecodingcompany.nl', 'tags': ['ai', 'datastructures'], 'created_at': '2025-04-01T09:54:51.199519', 'accessed_
at': '2025-04-01T09:54:51.199519'}]
```

jupyter Untitled Last Checkpoint: 12 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[6]:
```python
def delete_by_uri(uri):
    global records
    found = False
    for i, record in enumerate(records):
        if record["uri"] == uri:
            confirm = input(f"Weet je zeker dat je het record '{record['title']}' wilt verwijderen? (ja/nee): ")
            if confirm.lower() == "ja":
                del records[i]
                print("🗑 Record verwijderd.")
            else:
                print("❌ Verwijderen geannuleerd.")
            found = True
            break
    if not found:
        print("⚠️ Geen record gevonden met deze URI.")
```

[7]:
```python
def search_by_tag(tag):
    results = []
    for record in records:
        if tag in record["tags"]:
            results.append(record)
    print(f"Gevonden {len(results)} record(s) met tag '{tag}':")
    for r in results:
        print(f"- {r['title']} ({r['uri']})")
```

[8]:
```python
def sort_by_created_at_desc():
    sorted_records = sorted(records, key=lambda x: x["created_at"], reverse=True)
    print("📄 Records gesorteerd op aanmaakdatum (nieuwste eerst):")
    for r in sorted_records:
        print(f"- {r['created_at']} | {r['title']} ({r['uri']})")
```

[9]:
```python
search_by_tag('ai');
```

```
Gevonden 1 record(s) met tag 'ai':
- Testlink (https://www.thecodingcompany.nl)
```

[ ]:

jupyter Untitled Last Checkpoint: 12 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[11]:
```
def search_by_tags_all(tags):
    results = []
    for record in records:
        if all(tag in record["tags"] for tag in tags):
            results.append(record)
    print(f"Found {len(results)} record(s) matching all tags:")
    for r in results:
        print(f"- {r['title']} ({r['uri']})")
```

[16]:
```
search_by_tags_any('datastructures')
```

```
No records found matching any of the given tags.
```

[13]:
```
search_by_tags_any('ai')
```

```
Found 0 record(s) matching at least one tag:
```

[14]:
```
print(records)
```

```
[{'title': 'Testlink', 'uri': 'https://www.thecodingcompany.nl', 'tags': ['ai', 'datastructures'], 'created_at': '2025-04-01T09:54:51.199519', 'accessed_
at': '2025-04-01T09:54:51.199519'}]
```

[15]:
```
def search_by_tags_any(tags):
    results = []
    for record in records:
        for tag in tags:
            if tag.lower() in [t.lower() for t in record["tags"]]:
                results.append(record)
                break  # Stop after first match
    if results:
        print(f"Found {len(results)} record(s) matching at least one tag:")
        for r in results:
            print(f"- {r['title']} ({r['uri']})")
    else:
        print("No records found matching any of the given tags.")
```

[18]:
```
search_by_tags_any(['datastructures'])
```

```
Found 1 record(s) matching at least one tag:
- Testlink (https://www.thecodingcompany.nl)
```

[19]:
```
search_by_tags_all(['ai'])
```

```
Found 1 record(s) matching all tags:
- Testlink (https://www.thecodingcompany.nl)
```

jupyter Untitled Last Checkpoint: 12 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

- Testlink (https://www.thecodingcompany.nl)

[20]:
```
def main_menu():
    while True:
        print("\n=== Interest Book Menu ===")
        print("1. Add a new record")
        print("2. Delete a record by URI")
        print("3. Search by a single tag")
        print("4. Search by multiple tags (any match)")
        print("5. Search by multiple tags (all must match)")
        print("6. Search by keyword (title or tags)")
        print("7. Sort and display by creation date (newest first)")
        print("8. Edit an existing record")
        print("9. View all records")
        print("0. Exit")

        choice = input("Select an option (0-9): ")

        if choice == "1":
            title = input("Title: ")
            uri = input("URI: ")
            tags_input = input("Enter 1-5 tags, separated by commas: ")
            tags = [tag.strip() for tag in tags_input.split(",")]
            add_record(title, uri, tags)

        elif choice == "2":
            uri = input("Enter URI of the record to delete: ")
            delete_by_uri(uri)

        elif choice == "3":
            tag = input("Enter tag to search for: ")
            search_by_tag(tag)

        elif choice == "4":
            tags = input("Enter tags (comma-separated): ").split(",")
            search_by_tags_any([tag.strip() for tag in tags])

        elif choice == "5":
            tags = input("Enter tags (comma-separated): ").split(",")
            search_by_tags_all([tag.strip() for tag in tags])

        elif choice == "6":
            keyword = input("Enter keyword to search in title or tags: ")
            search_by_keyword(keyword)

        elif choice == "7":
            sort_by_created_at_desc()

        elif choice == "8":
            uri = input("Enter URI of the record to edit: ")
            new_title = input("New title (leave blank to keep current): ")
            new_tags_input = input("Enter new tags (comma-separated, leave blank to keep current): ")
            new_tags = [t.strip() for t in new_tags_input.split(",")] if new_tags_input else []
            edit_record(uri, new_title, new_tags)

        elif choice == "9":
            print("📄 Current records:")
            for r in records:
                print(f"- {r['title']} ({r['uri']}) - Tags: {r['tags']}")

        elif choice == "0":
            print("👋 Exiting Interest Book. Goodbye!")
            break

        else:
            print("⚠️ Invalid choice. Please select a number from 0 to 9.")
```

jupyter Untitled Last Checkpoint: 12 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[*]:
```python
main_menu()
```

```
=== Interest Book Menu ===
1. Add a new record
2. Delete a record by URI
3. Search by a single tag
4. Search by multiple tags (any match)
5. Search by multiple tags (all must match)
6. Search by keyword (title or tags)
7. Sort and display by creation date (newest first)
8. Edit an existing record
9. View all records
0. Exit
Select an option (0-9):  1
Title:  Testing for Essex
URI:  https://my-course.co.uk
Enter 1-5 tags, separated by commas:  computer science, python, programming
✅ Record toegevoegd: Testing for Essex

=== Interest Book Menu ===
1. Add a new record
2. Delete a record by URI
3. Search by a single tag
4. Search by multiple tags (any match)
5. Search by multiple tags (all must match)
6. Search by keyword (title or tags)
7. Sort and display by creation date (newest first)
8. Edit an existing record
9. View all records
0. Exit
Select an option (0-9):  3
Enter tag to search for:  programming
Gevonden 1 record(s) met tag 'programming':
- Testing for Essex (https://my-course.co.uk)

=== Interest Book Menu ===
1. Add a new record
2. Delete a record by URI
3. Search by a single tag
4. Search by multiple tags (any match)
5. Search by multiple tags (all must match)
6. Search by keyword (title or tags)
7. Sort and display by creation date (newest first)
8. Edit an existing record
9. View all records
0. Exit
Select an option (0-9):  ↑↓ for history. Search history with c-↑/c-↓
```

[ ]:

[22]:
```python
print("TEST 1: Add a valid record")
add_record("Machine Learning Basics", "https://ml-basics.org", ["ai", "ml", "education"])
print("Current dataset:")
for r in records:
    print(f"- {r['title']} | Tags: {r['tags']}")
```

```
TEST 1: Add a valid record
✅ Record toegevoegd: Machine Learning Basics
Current dataset:
- Testing for Essex | Tags: ['computer science', 'python', 'programming']
- https://www.python.org | Tags: ['programming', 'python', 'computer science']
- Machine Learning Basics | Tags: ['ai', 'ml', 'education']
```

[ ]:

[23]:
```python
print("TEST 2: Add a record with more than 5 tags (should fail)")
add_record("Too Many Tags", "https://toomanytags.com", ["tag1", "tag2", "tag3", "tag4", "tag5", "tag6"])
```

```
TEST 2: Add a record with more than 5 tags (should fail)
Fout: Je moet tussen de 1 en 5 tags gebruiken.
```

[ ]:

[26]:
```python
add_record("Software Development", "https://thecodingcompany.nl", ["ai", "programming", "development"])
```

```
✅ Record toegevoegd: Software Development
```

[27]:
```python
print("TEST 3: Search by existing tag 'ai'")
search_by_tag("ai")
```

```
TEST 3: Search by existing tag 'ai'
Gevonden 2 record(s) met tag 'ai':
- Machine Learning Basics (https://ml-basics.org)
- Software Development (https://thecodingcompany.nl)
```

[ ]:

[28]:
```python
print("TEST 4: Search by non-existent tag 'quantum'")
search_by_tag("quantum")
```

```
TEST 4: Search by non-existent tag 'quantum'
Gevonden 0 record(s) met tag 'quantum':
```

[ ]:

[29]:
```python
print("TEST 5: Search by multiple tags (any match: 'ai', 'education')")
search_by_tags_any(["ai", "education"])
```

```
TEST 5: Search by multiple tags (any match: 'ai', 'education')
Found 2 record(s) matching at least one tag:
- Machine Learning Basics (https://ml-basics.org)
- Software Development (https://thecodingcompany.nl)
```

[30]:
```python
print("TEST 6: Search by multiple tags (all must match: 'ai', 'ml')")
search_by_tags_all(["ai", "ml"])
```

```
TEST 6: Search by multiple tags (all must match: 'ai', 'ml')
Found 1 record(s) matching all tags:
- Machine Learning Basics (https://ml-basics.org)
```

[ ]:

[31]:
```python
print("TEST 7: Search by keyword 'learning'")
search_by_keyword("learning")
```

```
TEST 7: Search by keyword 'learning'
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[31], line 2
      1 print("TEST 7: Search by keyword 'learning'")
----> 2 search_by_keyword("learning")

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

[ ]:

[32]:
```python
def search_by_keyword(keyword):
    results = []
    for record in records:
        in_title = keyword.lower() in record["title"].lower()
        in_tags = any(keyword.lower() in tag.lower() for tag in record["tags"])
        if in_title or in_tags:
            results.append(record)
    if results:
        print(f"Found {len(results)} record(s) containing keyword '{keyword}':")
        for r in results:
            print(f"- {r['title']} ({r['uri']})")
    else:
        print("No records found matching the keyword.")
```

[33]:
```python
print("TEST 7: Search by keyword 'learning'")
search_by_keyword("learning")
```

```
TEST 7: Search by keyword 'learning'
Found 1 record(s) containing keyword 'learning':
- Machine Learning Basics (https://ml-basics.org)
```

[ ]:

[34]:
```python
print("TEST 8: Delete record by URI")
delete_by_uri("https://ml-basics.org")
```

```
TEST 8: Delete record by URI
Weet je zeker dat je het record 'Machine Learning Basics' wilt verwijderen? (ja/nee):  ja\
❌ Verwijderen geannuleerd.
```

[35]:
```python
print("TEST 8: Delete record by URI")
delete_by_uri("https://ml-basics.org")
```

```
TEST 8: Delete record by URI
Weet je zeker dat je het record 'Machine Learning Basics' wilt verwijderen? (ja/nee):  ja
🗑️ Record verwijderd.
```

[ ]:

[36]:
```python
print("TEST 9: Delete non-existent record")
delete_by_uri("https://thisuridoesnotexist.com")
```

```
TEST 9: Delete non-existent record
⚠️ Geen record gevonden met deze URI.
```

[ ]:

[37]:
```python
print("TEST 10: Sort by creation date (newest first)")
sort_by_created_at_desc()
```

```
TEST 10: Sort by creation date (newest first)
📋 Records gesorteerd op aanmaakdatum (nieuwste eerst):
- 2025-04-01T10:18:49.961463 | Software Development (https://thecodingcompany.nl)
- 2025-04-01T10:12:54.667642 | https://www.python.org (https://www.python.org)
- 2025-04-01T10:10:26.055290 | Testing for Essex (https://my-course.co.uk)
```

[ ]:

[39]:
```python
def edit_record(uri, new_title, new_tags):
    for record in records:
        if record["uri"] == uri:
            if new_title:
                record["title"] = new_title
            if new_tags and 1 <= len(new_tags) <= 5:
                record["tags"] = new_tags
            print("✅ Record successfully updated.")
            return
    print("⚠️ Record not found.")
```

[40]:
```python
print("TEST 11: Edit record by URI")
edit_record(
    "https://www.thecodingcompany.nl",
    "The Coding Company - Updated Title",
    ["php", "development", "web"]
)
```

```
TEST 11: Edit record by URI
⚠️ Record not found.
```

[42]:
```python
print("TEST 11: Edit record by URI");
edit_record(
    "https://thecodingcompany.nl",
    "New Title Coding",
    ["tag1", "tag2", "tag5"]
)
```

```
TEST 11: Edit record by URI
✅ Record successfully updated.
```

[ ]: