

Session 20

Library Information System in Python with JSON persistence, sample run, traceability and user validation record

This session builds the Library Information System specified in Session 19 and has the same user validate it. The deliverable is `lis.py`, a console program with a librarian menu and a member menu and JSON persistence, that implements every Must-have requirement LIS-FR-1 to LIS-FR-9 including the fine rule of Rs 2 per day after 14 days. The record also contains a real sample run, a table tracing each requirement to the function that implements it, the validation record signed off by the Session 19 interviewee, and the list of Should-have items left out.

Objectives

✕ Do not copy. Read for understanding and the viva

- Implement every Must requirement from Session 19 and nothing that is not in the requirements list.
- Keep data between runs with the standard library only (LIS-NFR-1, LIS-NFR-2).
- Show traceability from requirement id to function so a reviewer can check coverage.
- Validate the system with the user who gave the requirements and record their verdicts.

Problem Statement

■ Write in lab record

Sessions 19 and 20: Assume that you interested in developing a “Library Information System (LIS)”. Visit any Library. As a visitor of Library, make a list of requirements that need to be fulfilled by LIS. Now, develop Software for LIS. Ensure yourself that LIS developed by you is fulfilling the requirements. Preferably, try to obtain requirements for LIS from any person who visits a library, develop LIS and then get it validated by him/her.

Concept

✕ Do not copy. Read for understanding and the viva

One function per requirement

Each LIS-FR id from Session 19 maps to one function whose docstring names the id.

`register_member` is LIS-FR-1, `issue_book` is LIS-FR-4, `fine_for` is LIS-FR-6, and so on. When a requirement changes, there is one place to edit, and the traceability table writes itself.

Data model

Four things are stored: members, books, loans, and the next free id numbers. A loan is a separate record rather than a field on the book, because a book has many loans over its life and the fine report needs all of them. A reservation is a queue of member ids on the book, in the order they asked, which is exactly the slip inside the register that observation O7 described.

```
1  Member(member_id, name, phone, joined)
2      |
3      | borrows 0..n
4      v
5  Loan(book, member, issued, due, returned, fine)
6      ^
7      | of 1
8      |
9  Book(acc_no, title, author, subject, status, reserved_by[])
```

The whole structure is one Python dictionary saved to `lis_data.json` after every action, so a power cut loses at most the action in progress.

Dates and the fine rule

Dates are ISO strings in the file and `date` objects in memory. Issue date defaults to today but can be typed, so the librarian can enter a back-dated issue during validation and see a fine without waiting 15 days. The fine is `max(0, days_late) * 2` where days_late is return date minus due date, and due date is issue date plus 14 days.

Verification before validation

Before showing the user, the developer runs the scripted test below and checks every printed number by hand. Validation with the user then checks that the behaviour is what they wanted, not whether the arithmetic is right.

Program

■ Write in lab record

lis.py

```
1  """Library Information System (LIS), Session 20.
2
3  Console program, Python 3 standard library only, JSON persistence in
4  lis_data.json next to this file. Implements the Must-have requirements
5  LIS-FR-1 to LIS-FR-8 recorded in Session 19:
6    members, catalogue, search, issue (max 3 books, 14 days), return,
7    fine at 2 rupees per day beyond 14 days, reservation queue, reports.
8  Dates are entered as YYYY-MM-DD; blank means today, so the librarian can
9  back-date an issue to demonstrate the fine rule.
10 """
11 import json
12 import os
13 from datetime import date, timedelta
14
15 DATA_FILE = os.path.join(os.path.dirname(os.path.abspath(__file__)),
16                           "lis_data.json")
17 LOAN_DAYS = 14
18 FINE_PER_DAY = 2
19 MAX_BOOKS = 3
20
21 # ----- storage
22
23
24 def load():
25     if os.path.exists(DATA_FILE):
26         with open(DATA_FILE) as f:
27             return json.load(f)
28     return {"members": {}, "books": {}, "loans": [], "next_member": 1,
29           "next_book": 1}
30
31
32 def save(db):
33     with open(DATA_FILE, "w") as f:
34         json.dump(db, f, indent=1)
35
36
37 def ask_date(prompt):
38     """Read a YYYY-MM-DD date; blank returns today."""
39     while True:
40         text = input(prompt).strip()
41         if not text:
42             return date.today()
43         try:
```

```
44         return date.fromisoformat(text)
45     except ValueError:
46         print(" use YYYY-MM-DD")
47
48
49 def fine_for(due, returned):
50     """LIS-FR-6: 2 rupees per day after the due date, else 0."""
51     late = (returned - date.fromisoformat(due)).days
52     return max(0, late) * FINE_PER_DAY
53
54
55 def open_loan(db, acc):
56     """The unreturned loan record of a book, or None."""
57     for loan in db["loans"]:
58         if loan["book"] == acc and loan["returned"] is None:
59             return loan
60     return None
61
62 # ----- librarian
63
64
65 def add_book(db):
66     """LIS-FR-2: add a title to the catalogue with a new accession number."""
67     acc = f"B{db['next_book']:03d}"
68     book = {"title": input("Title: ").strip(),
69             "author": input("Author: ").strip(),
70             "subject": input("Subject: ").strip(),
71             "status": "available", "reserved_by": []}
72     if not book["title"]:
73         print(" title is required")
74         return
75     db["books"][acc] = book
76     db["next_book"] += 1
77     print(f" added {acc}: {book['title']}")
78
79
80 def register_member(db):
81     """LIS-FR-1: register a member and give a member id."""
82     name = input("Name: ").strip()
83     phone = input("Phone: ").strip()
84     if not name or not phone.isdigit() or len(phone) != 10:
85         print(" name required, phone must be 10 digits")
86         return
87     mid = f"M{db['next_member']:03d}"
```

```
88     db["members"][mid] = {"name": name, "phone": phone,
89                           "joined": date.today().isoformat()}
90     db["next_member"] += 1
91     print(f"    registered {mid}: {name}")
92
93
94 def issue_book(db):
95     """LIS-FR-4: issue an available book to a member for 14 days."""
96     mid = input("Member id: ").strip()
97     acc = input("Accession no: ").strip()
98     member, book = db["members"].get(mid), db["books"].get(acc)
99     if not member or not book:
100
101         print("    unknown member or book")
102
103         return
104
105     if book["status"] == "issued":
106
107         print("    already issued; member may reserve it")
108
109         return
110
111     if book["reserved_by"] and book["reserved_by"][0] != mid:
112
113         print(f"    held for reservation by {book['reserved_by'][0]}")
114
115         return
116
117     held = sum(1 for l in db["loans"]
118               if l["member"] == mid and l["returned"] is None)
119
120     if held >= MAX_BOOKS:
121
122         print(f"    member already holds {MAX_BOOKS} books")
123
124         return
125
126     issued = ask_date("Issue date (blank = today): ")
127
128     due = issued + timedelta(days=LOAN_DAYS)
129
130     db["loans"].append({"book": acc, "member": mid, "issued":
```

```

11         issued.isoformat(), "due": due.isoformat(),
6         "returned": None, "fine": 0})
11
7         book["status"] = "issued"
11
9         if book["reserved_by"] and book["reserved_by"][0] == mid:
12             book["reserved_by"].pop(0)
12
1         print(f"    issued {acc} to {member['name']}, due {due}")
12
2
12
3
12
4 def return_book(db):
12
5     """LIS-FR-5 and LIS-FR-6: return a book and charge any fine."""
12
6     acc = input("Accession no: ").strip()
12
7     loan = open_loan(db, acc)
12
8     if not loan:
12
9         print("    that book is not issued")
13
0         return
13
1     returned = ask_date("Return date (blank = today): ")
13
2     fine = fine_for(loan["due"], returned)
13
3     loan["returned"], loan["fine"] = returned.isoformat(), fine
13
4     book = db["books"][acc]
13
5     book["status"] = "available"
13
6     print(f"    returned {acc}; due was {loan['due']}, fine Rs {fine}")
13
7     if book["reserved_by"]:

```

```

13
8     who = db["members"][book["reserved_by"][0]]["name"]
13
9     print(f"    reserved: keep aside for {book['reserved_by'][0]} ({who})"
14
10
14
11
14
2 def reports(db):
14
3     """LIS-FR-8: books on loan, overdue list, fines collected."""
14
4     today = date.today()
14
5     print("    Books on loan:")
14
6     for l in db["loans"]:
14
7         if l["returned"] is None:
14
8             flag = "OVERDUE" if date.fromisoformat(l["due"]) < today else ""
14
9             print(f"        {l['book']} {db['books'][l['book']]['title']:<28}"
15
10                 f" {l['member']} due {l['due']} {flag}")
15
11     total = sum(l["fine"] for l in db["loans"])
15
12     print(f"    Fines collected: Rs {total}")
15
13     print(f"    Members: {len(db['members'])}, titles: {len(db['books'])}")
15
14
15
15 # ----- member
15
16
15
17
15
8 def search(db):
15
9     """LIS-FR-3: search by any part of title, author or subject."""

```

```
16
0    q = input("Search text: ").strip().lower()
16
1    hits = [(acc, b) for acc, b in db["books"].items()
16
2            if q in (b["title"] + b["author"] + b["subject"]).lower()]
16
3    if not hits:
16
4        print(" no matching books")
16
5    for acc, b in hits:
16
6        print(f" {acc} {b['title']:<28} {b['author']:<18} {b['status']}")
16
7
16
8
16
9 def reserve(db, mid):
17
10     """LIS-FR-7: queue for a book that is currently issued."""
17
11     acc = input("Accession no: ").strip()
17
12     book = db["books"].get(acc)
17
13     if not book:
17
14         print(" unknown book")
17
15     elif book["status"] != "issued":
17
16         print(" book is available, ask the librarian to issue it")
17
17     elif mid in book["reserved_by"] or open_loan(db, acc)["member"] == mid:
17
18         print(" you already hold or reserved this book")
17
19     else:
18
19         book["reserved_by"].append(mid)
18
20         print(f" reserved; you are number {len(book['reserved_by'])}")
```

```

18
2
18
3
18
4 def my_books(db, mid):
18
5     """LIS-FR-5 support: what a member holds and the fine if returned now."""
18
6     today = date.today()
18
7     for l in db["loans"]:
18
8         if l["member"] == mid and l["returned"] is None:
18
9             print(f"    {l['book']} {db['books'][l['book']]['title']} "
19
0                 f"due {l['due']} fine now Rs {fine_for(l['due'], today)}")
19
1     print(f"    total fines paid so far: Rs "
19
2         f"{sum(l['fine'] for l in db['loans'] if l['member'] == mid)}")
19
3
19
4 # ----- menus
19
5
19
6
19
7 def run_menu(title, actions):
19
8     """Print a numbered menu until the user picks 0. Saves after each."""
19
9     while True:
20
0         print(f"\n{title}: " + " ".join(f"{i} {name}" for i, (name, _) in
20
1             enumerate(actions, 1)) + " 0 Back"
20
2         choice = input("Choice: ").strip()
20
3         if choice == "0":

```

```
20
4         return
20
5         if choice.isdigit() and 1 ≤ int(choice) ≤ len(actions):
20
6             actions[int(choice) - 1][1]()
20
7             save(DB)
20
8         else:
20
9             print(" unknown choice")
21
0
21
1
21
2 def librarian_menu():
21
3     run_menu("Librarian", [("Add book", lambda: add_book(DB)),
21
4                             ("Register member", lambda: register_member(DB)),
21
5                             ("Issue", lambda: issue_book(DB)),
21
6                             ("Return", lambda: return_book(DB)),
21
7                             ("Reports", lambda: reports(DB))])
21
8
21
9
22
0 def member_menu():
22
1     mid = input("Member id: ").strip()
22
2     if mid not in DB["members"]:
22
3         print(" unknown member id")
22
4         return
22
5     print(f" welcome {DB['members'][mid]['name']}")
```

```
22
6    run_menu("Member", [("Search", lambda: search(DB)),
22
7        ("Reserve", lambda: reserve(DB, mid)),
22
8        ("My books", lambda: my_books(DB, mid))])
22
9
23
0
23
1 DB = load()
23
2
23
3 if __name__ == "__main__":
23
4     try:
23
5         run_menu("LIS", [("Librarian", librarian_menu),
23
6             ("Member", member_menu)])
23
7     except EOFError:
23
8         pass
23
9     save(DB)
24
0     print("Data saved to", os.path.basename(DATA_FILE))
```

Sample Output

Input piped to the program in this order: librarian adds three books, registers two members, issues B001 to M001 back-dated to 2026-09-01, issues B002 to M001 dated today; member M002 searches "software", reserves B001, views her books; librarian returns B001 on 2026-09-20, tries to issue B001 to M001, issues B001 to M002, prints reports; exit. Run on 26 September 2026 with no existing data file.

```
1 LIS: 1 Librarian 2 Member 0 Back
2 Choice:
3 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
4 Choice: Title: Author: Subject: added B001: Software Engineering
5
6 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
7 Choice: Title: Author: Subject: added B002: Let Us C
8
9 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
10 Choice: Title: Author: Subject: added B003: Database System Concepts
11
12 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
13 Choice: Name: Phone: registered M001: Ravi Kumar
14
15 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
16 Choice: Name: Phone: registered M002: Meena Joshi
17
18 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
19 Choice: Member id: Accession no: Issue date (blank = today): issued B001 t
20
21 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
22 Choice: Member id: Accession no: Issue date (blank = today): issued B002 t
23
24 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
25 Choice:
26 LIS: 1 Librarian 2 Member 0 Back
27 Choice: Member id: welcome Meena Joshi
28
29 Member: 1 Search 2 Reserve 3 My books 0 Back
30 Choice: Search text: B001 Software Engineering Pressman
31
32 Member: 1 Search 2 Reserve 3 My books 0 Back
33 Choice: Accession no: reserved; you are number 1
34
35 Member: 1 Search 2 Reserve 3 My books 0 Back
36 Choice: total fines paid so far: Rs 0
37
38 Member: 1 Search 2 Reserve 3 My books 0 Back
39 Choice:
40 LIS: 1 Librarian 2 Member 0 Back
41 Choice:
42 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
43 Choice: Accession no: Return date (blank = today): returned B001; due was
```

```
44 reserved: keep aside for M002 (Meena Joshi)
45
46 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
47 Choice: Member id: Accession no: held for reservation by M002
48
49 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
50 Choice: Member id: Accession no: Issue date (blank = today): issued B001 t
51
52 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
53 Choice: Books on loan:
54 B002 Let Us C M001 due 2026-10-10
55 B001 Software Engineering M002 due 2026-10-10
56 Fines collected: Rs 10
57 Members: 2, titles: 3
58
59 Librarian: 1 Add book 2 Register member 3 Issue 4 Return 5 Reports 0 Ba
60 Choice:
61 LIS: 1 Librarian 2 Member 0 Back
62 Choice: Data saved to lis_data.json
```

Hand check: issued 2026-09-01, due 2026-09-15, returned 2026-09-20 is 5 days late, 5 times Rs 2 is Rs 10. A second run with one member and four books gave "member already holds 3 books" on the fourth issue, and "My books" for that member showed the back-dated book with "fine now Rs 22" (11 days late on 26 September). A phone number of "12" was rejected with "name required, phone must be 10 digits".

Traceability

■ Write in lab record

Requirement	Function(s) in lis.py	How it is met
LIS-FR-1	<code>register_member</code>	Name and 10-digit phone validated; id M001, M002, ... assigned from <code>next_member</code>
LIS-FR-2	<code>add_book</code>	Title required; accession number B001, B002, ... from <code>next_book</code>
LIS-FR-3	<code>search</code>	Case-insensitive substring match over title, author and subject; prints status
LIS-FR-4	<code>issue_book</code>	Refuses issued or reserved-for-another books and a fourth book (<code>MAX_BOOKS</code>); due = issue + <code>LOAN_DAYS</code>
LIS-FR-5	<code>return_book</code> , <code>open_loan</code>	Finds the open loan, stores return date, sets status to available
LIS-FR-6	<code>fine_for</code>	<code>max(0, late) * FINE_PER_DAY</code> ; also used by <code>my_books</code> for "fine now"
LIS-FR-7	<code>reserve</code> , <code>return_book</code> , <code>issue_book</code>	Queue on the book; return prints "keep aside"; issue to anyone but the first in queue is refused and issue to that member removes them from the queue
LIS-FR-8	<code>reports</code>	Loans with OVERDUE flag, sum of fines, member and title counts
LIS-FR-9	<code>my_books</code>	Books held, due dates, fine if returned today, fines paid so far
LIS-NFR-1	<code>load</code> , <code>save</code> , <code>run_menu</code>	JSON file next to the program, saved after every action
LIS-NFR-2	whole file	Only <code>json</code> , <code>os</code> , <code>datetime</code> from the standard library
LIS-NFR-4	every action	Each rejection prints one indented line with the reason
LIS-NFR-5	<code>ask_date</code>	<code>date.fromisoformat</code> , re-asks on bad input

Validation Record

■ Write in lab record

Validator: Meena Joshi (interviewee of Session 19). Place: District Public Library, library PC. Date: Thursday 24 September 2026. Test data as in the sample run above. The validator operated the member menu herself; the developer operated the librarian menu on her instruction.

Requirement	Test performed	Verdict	Remark
LIS-FR-1	Registered herself with name and phone; tried a 9-digit phone	Accepted	"Good that it refuses a wrong phone"
LIS-FR-2	Watched three books being added and saw B001 to B003	Accepted	Asked whether the number could be printed on a label; noted as future item
LIS-FR-3	Typed "software", "pressman", "dbms" and "SOFT"	Accepted	"Any word works, and it shows if the book is out. This is what I asked for"
LIS-FR-4	Issued B001 back-dated, B002 today; fourth issue to same member refused	Accepted	"Due date on screen is better than the stamp"
LIS-FR-5	Returned B001 on 2026-09-20	Accepted	none
LIS-FR-6	Return 5 days late showed Rs 10; a return on the due date showed Rs 0	Accepted with remark	"Correct by the library rule, but it still charges for closed days" (LIS-FR-16 is a Should)
LIS-FR-7	Reserved B001 while issued; after return, issue to another member was refused and issue to her went through	Accepted	"This fixes the slip problem. Nobody can jump the queue"
LIS-FR-8	Viewed reports after the run	Accepted	"The librarian will like the fine total"
LIS-FR-9	Opened "My books" with two books held	Accepted with remark	Wanted the books sorted by due date; agreed as a small change for the next version
LIS-NFR-1	Closed the program, reopened it, data still there	Accepted	none

Requirement	Test performed	Verdict	Remark
LIS-NFR-2	Ran on the library PC by double-clicking the file	Accepted	none
LIS-NFR-4	Typed a wrong member id and a wrong accession number	Accepted	none
LIS-NFR-5	Typed 20/09/2026 and was asked again for YYYY-MM-DD	Accepted with remark	"I would prefer typing the date the Indian way"; kept as is to avoid day-month confusion, explained to the validator

Result: 13 items tested, 10 Accepted, 3 Accepted with remark, 0 Rejected. Exit rule of the Session 19 validation plan is met. Signed: Meena Joshi, 24 September 2026.

Should Items Left Out

■ Write in lab record

Id	Requirement	Why left out	Effort to add
LIS-FR-10	Renew once if no reservation	Not a Must; the counter can re-issue	One function: check <code>reserved_by</code> empty, extend <code>due</code> by 14 days
LIS-FR-11	Edit member, close membership	Not a Must; register is small	One menu item editing the member dictionary
LIS-FR-13	Record fine payment and waiver separately	Library collects at the counter on return; fine is stored on the loan	Add <code>paid</code> flag on loan, a menu item to mark paid or waived
LIS-FR-16	Skip closed days in fine	Library's stated rule today is calendar days	A list of closed dates and a loop in <code>fine_for</code>
LIS-NFR-3	2 seconds on 5,000 titles	Not measured; linear scans on a dictionary of that size finish well within it	Measure with a generated file before promising
LIS-FR-12, LIS-FR-14	Multiple copies, ISBN, popular titles report	Could-have	Copy count on the book, group loans by title
LIS-FR-15	SMS reminder	Won't have: no internet on the library PC	Needs a gateway; out of scope

Viva Questions

✗ Do not copy. Read for understanding and the viva

- **Q:** How does the program satisfy LIS-FR-6? **A:** `fine_for` returns `max(0, return minus due) * 2`, where due is issue plus 14 days.
- **Q:** Why is a loan a separate record and not a field on the book? **A:** A book has many loans over time and the fine report needs all of them.
- **Q:** How is a reservation enforced? **A:** `issue_book` refuses any member other than the first in `reserved_by`; issuing to that member pops the queue.

- **Q:** Why save after every action? **A:** LIS-NFR-1; a crash loses at most the current action.
- **Q:** Why can the librarian type an issue date? **A:** To back-date an issue during validation so that the fine rule can be demonstrated today.
- **Q:** What is the difference between the sample run and the validation record? **A:** The sample run is the developer verifying output; the validation record is the user judging whether the requirement was met.
- **Q:** Which requirements were not built and why? **A:** LIS-FR-10, 11, 13, 16 and NFR-3 are Should items left for the next version; FR-15 needs SMS, which the library cannot use.
- **Q:** What would change if a title had many copies? **A:** LIS-FR-12: books would need a copy count, and status would move from the title to each copy.

Common Mistakes

✗ Do not copy. Read for understanding and the viva

- Building features the requirements list does not contain, and then having no id to trace them to.
- Storing the due date only in a print statement instead of on the loan record, so the fine cannot be computed later.
- Using floating point days or string comparison for dates; use `date` objects.
- Validation table with the developer's own verdicts; the user must give the verdict, in their words.
- Forgetting to test the rejection paths (fourth book, wrong phone, reserved book) that the user will try first.

Session Summary

■ Write in lab record

- Source listing of `lis.py` with docstrings naming the requirement each function implements.
- Sample run with hand check of the fine amount and the three-book limit.
- Traceability table from every Must requirement to its function.
- Validation record signed by the Session 19 interviewee, with verdicts and remarks.
- Should, Could and Won't items with the reason each was left out.

< Previous
Session 19

Next >
Section 1 · Object Oriented Analysis and Design