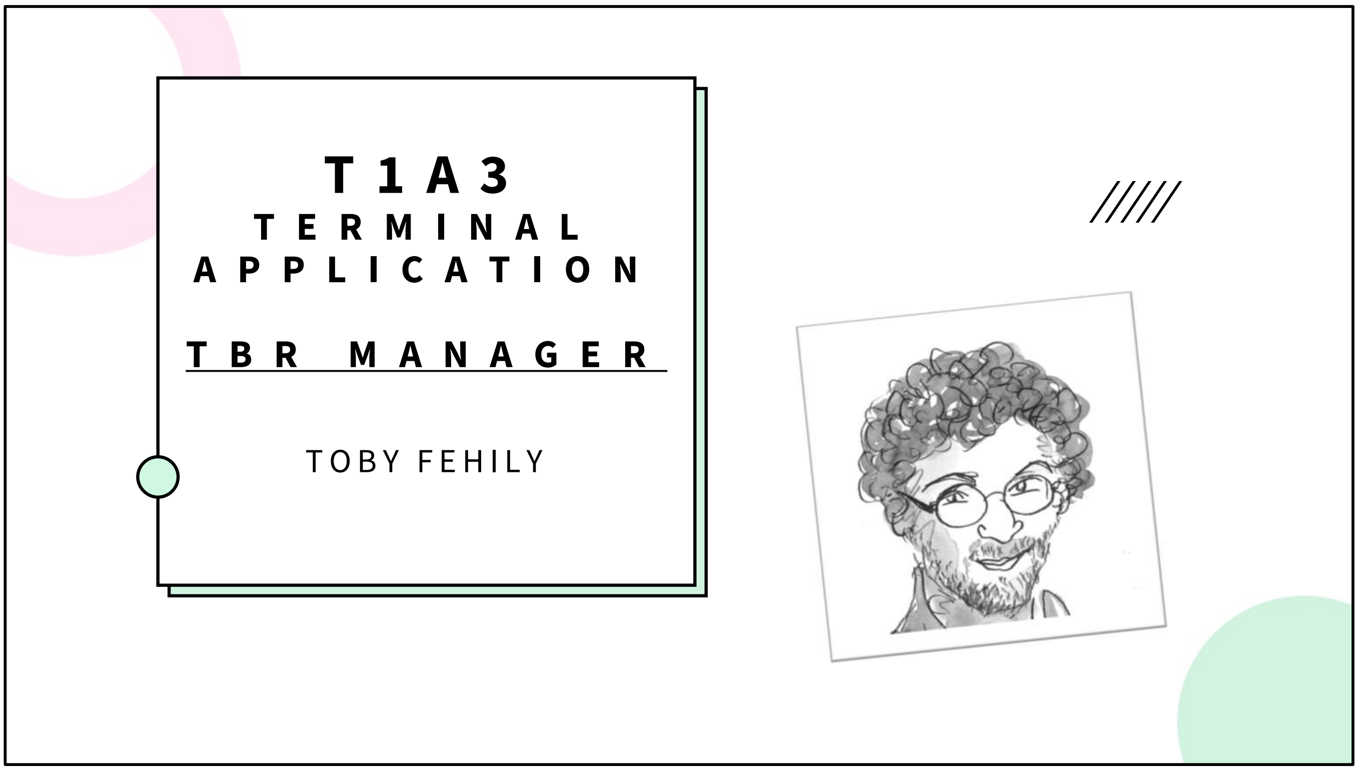
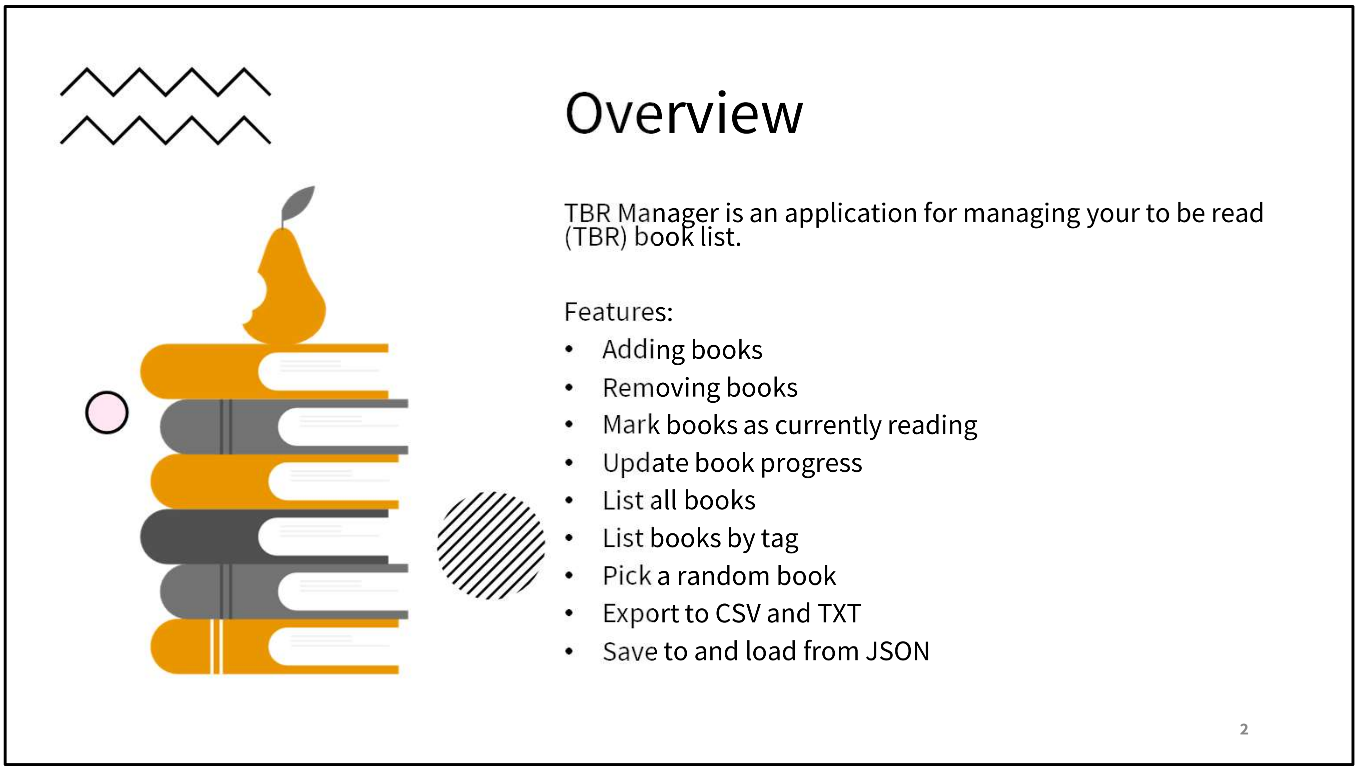


T1A3 TERMINAL APPLICATION

TBR MANAGER

TOBY FEHILY

- Welcome, everyone.
- I begin today by acknowledging the Traditional Custodians of the land on which we meet today, and pay my respects to their Elders past and present. I extend that respect to Aboriginal and Torres Strait Islander peoples here today.



Overview

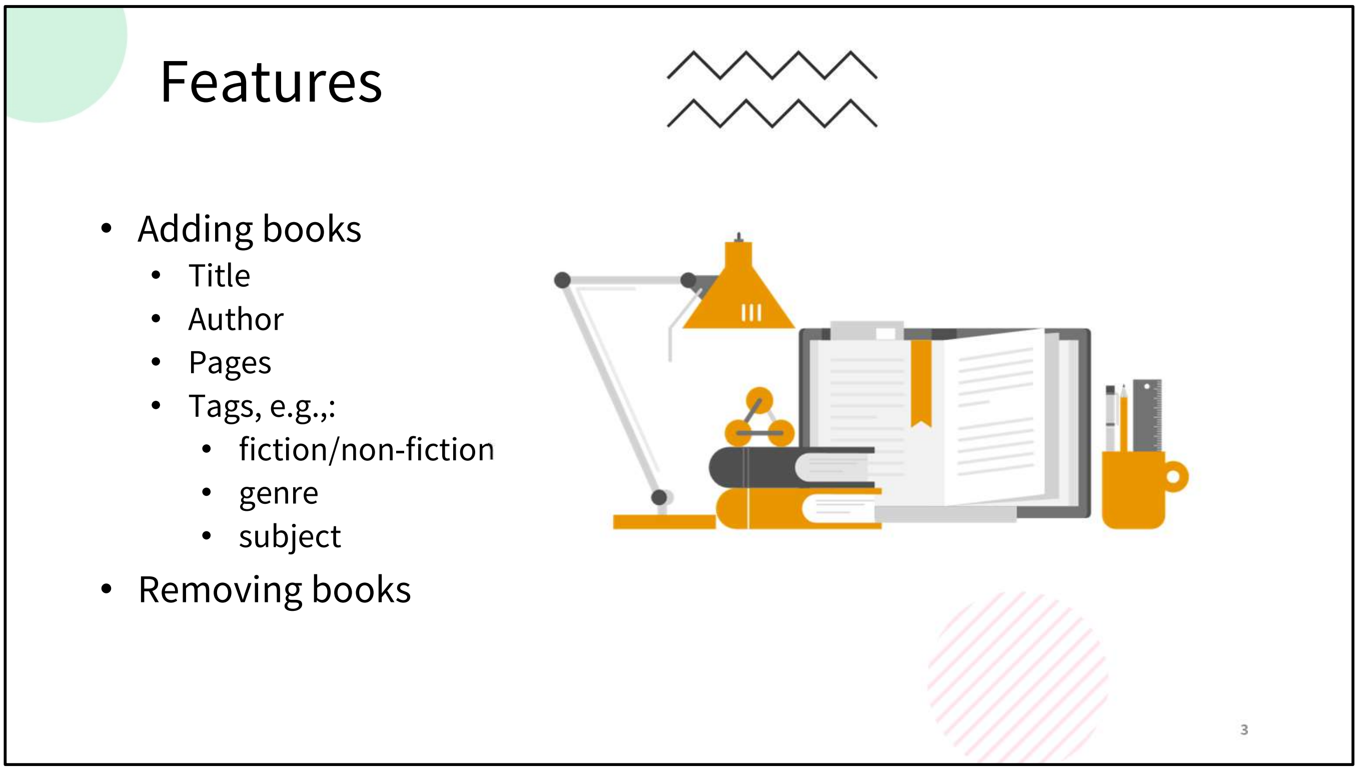
TBR Manager is an application for managing your to be read (TBR) book list.

Features:

- Adding books
- Removing books
- Mark books as currently reading
- Update book progress
- List all books
- List books by tag
- Pick a random book
- Export to CSV and TXT
- Save to and load from JSON

2

- TBR Manager is an application for managing your to be read (TBR) book list.
- The main features of the application include:
    - Adding books
    - Removing books
    - Mark books as currently reading
    - Update book progress
    - List all books
    - List books by tag
    - Pick a random book
    - Export to CSV and TXT
    - Save to and load from JSON

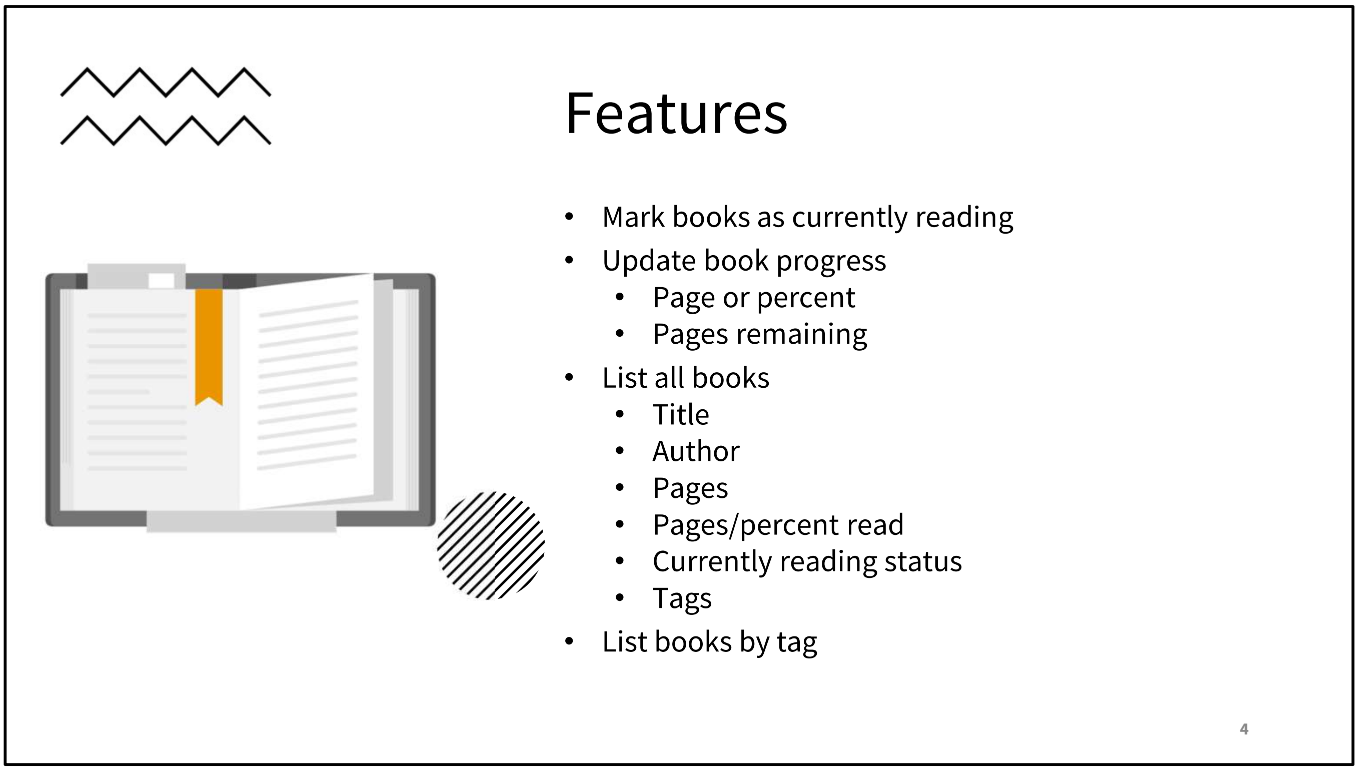


Features

- Adding books
    - Title
    - Author
    - Pages
    - Tags, e.g.,:
        - fiction/non-fiction
        - genre
        - subject
- Removing books

3

- The user can add books to the database to keep track of what they want to read.
    - The user provides the following information when adding a book:
        - Title
        - Author
        - Pages
        - Tags, e.g., fiction/non-fiction, genre, subject
            - Tags are optional, and users can enter as many or as few tags as they want
- The user can also remove a book from the database if they have finished it or no longer want to read it.

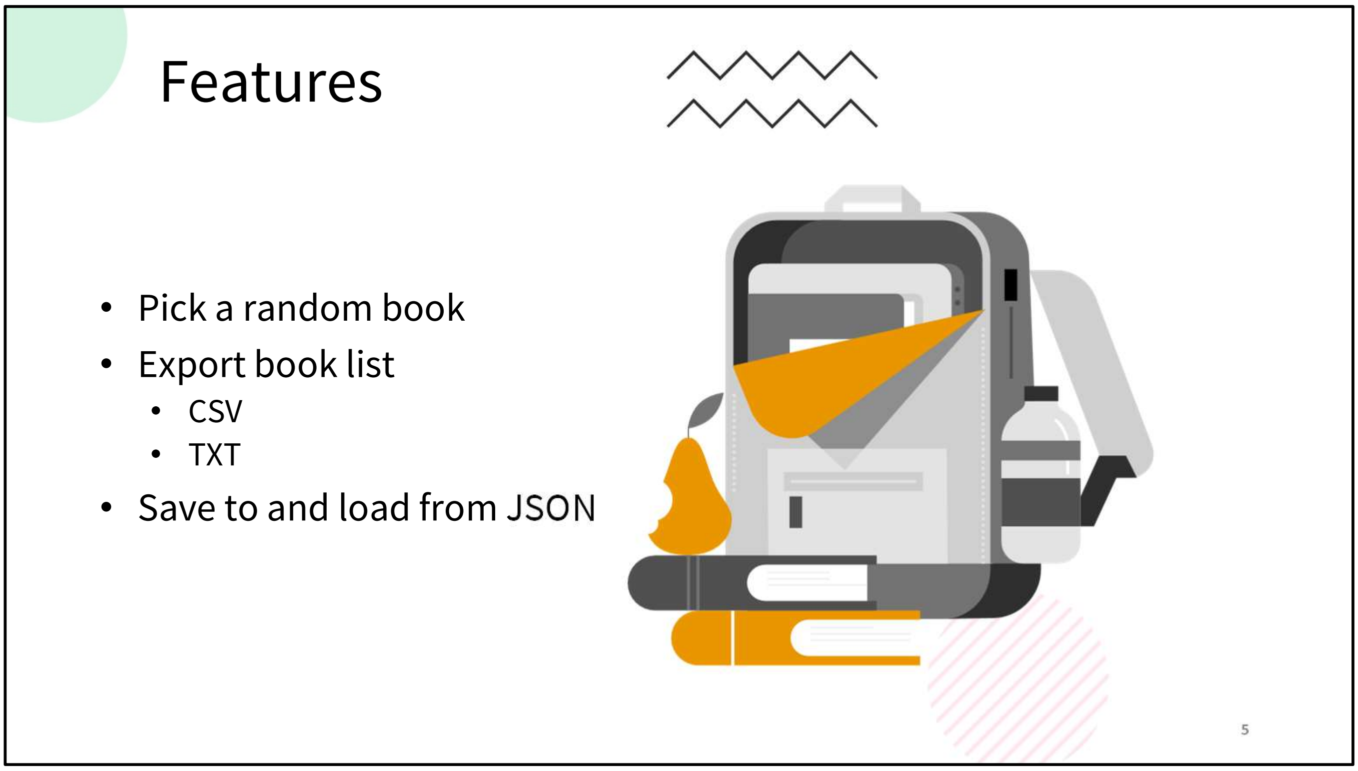


Features

- Mark books as currently reading
- Update book progress
  - Page or percent
  - Pages remaining
- List all books
  - Title
  - Author
  - Pages
  - Pages/percent read
  - Currently reading status
  - Tags
- List books by tag

4

- The user can mark books as currently reading to distinguish between books they are reading and books they want to read.
- The user can update their progress with a book to stay informed about how long they have left to go.
  - They can update progress by providing either the page number they are up to or the percentage read. The application will notify the user about how many pages they have remaining.
- The user can list all the books in the database to see what they want to read at a glance.
  - The list provides the following information about each book:
    - Title
    - Author
    - Pages
    - Pages/percent read
    - Currently reading status
    - Tags
- The user can list books by tag to filter results by criteria of their choice

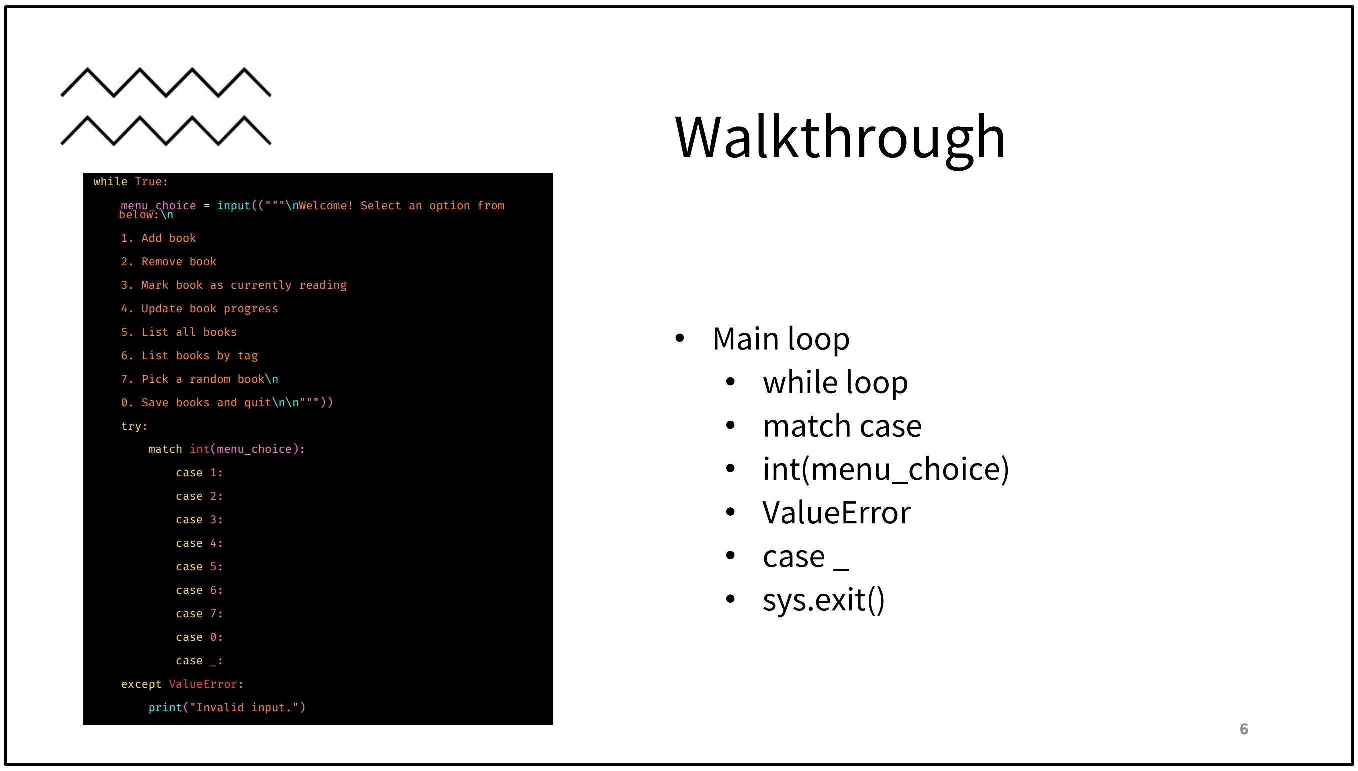


Features

- Pick a random book
- Export book list
  - CSV
  - TXT
- Save to and load from JSON

5

- The user can randomly generate a book if they can not decide what they want to read.
- On exiting, the application automatically exports the database to a CSV and a TXT file.
  - The user can use the CSV with spreadsheet software such as Microsoft Excel, and use the TXT file as an easy-to-read shopping list.
- On starting, the application loads the book list from a JSON file, and on exiting, saves to the JSON file. This preserves the book list between sessions.



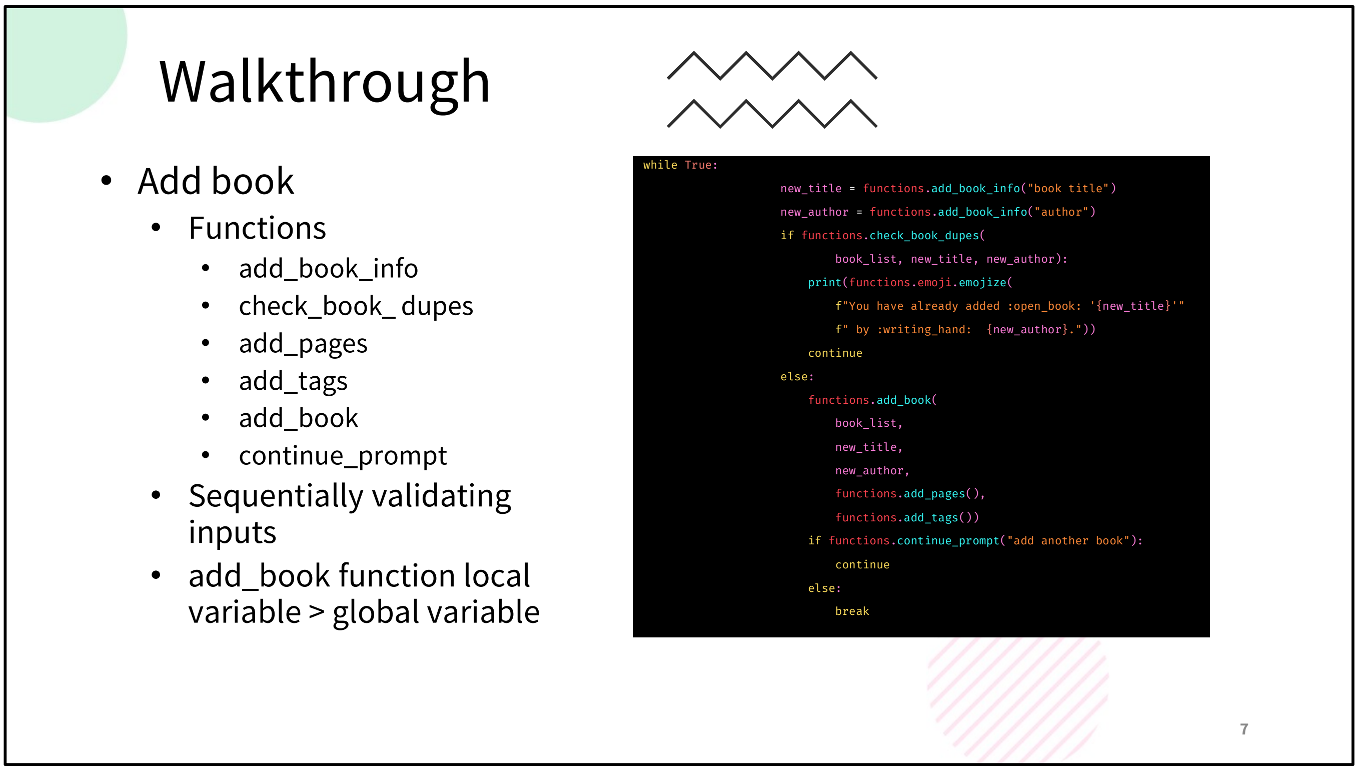
Walkthrough

```
while True:
    menu_choice = input(("""\nWelcome! Select an option from below:\n
    1. Add book
    2. Remove book
    3. Mark book as currently reading
    4. Update book progress
    5. List all books
    6. List books by tag
    7. Pick a random book\n
    0. Save books and quit\n\n"""))
    try:
        match int(menu_choice):
            case 1:
            case 2:
            case 3:
            case 4:
            case 5:
            case 6:
            case 7:
            case 0:
            case _:
    except ValueError:
        print("Invalid input.")
```

- Main loop
  - while loop
  - match case
  - int(menu_choice)
  - ValueError
  - case _
  - sys.exit()

6

- The main loop features a while True loop to keep the user within the loop until they quit the program.
  - This is especially important given the save to JSON feature, which is triggered when users choose to save books and quit.
- Using a match case pairing numbered menu items with numbered choices makes the code easily readable and maintainable.
- Forcing int(menu_choice) raises a ValueError if the user enters a string that cannot be converted into an integer, i.e., text
- The wildcard '_' case captures any number inputs that do not match a menu choice number.
- sys.exit() enables a clean and graceful exit from the program



Walkthrough

- Add book
  - Functions
    - add_book_info
    - check_book_ dupes
    - add_pages
    - add_tags
    - add_book
    - continue_prompt
  - Sequentially validating inputs
  - add_book function local variable > global variable

```
while True:
                    new_title = functions.add_book_info("book title")
                    new_author = functions.add_book_info("author")
                    if functions.check_book_dupes(
                            book_list, new_title, new_author):
                        print(functions.emoji.emojize(
                            f"You have already added :open_book: '{new_title}'"
                            f" by :writing_hand:  {new_author}."))
                        continue
                    else:
                        functions.add_book(
                            book_list,
                            new_title,
                            new_author,
                            functions.add_pages(),
                            functions.add_tags())
                        if functions.continue_prompt("add another book"):
                            continue
                        else:
                            break
```

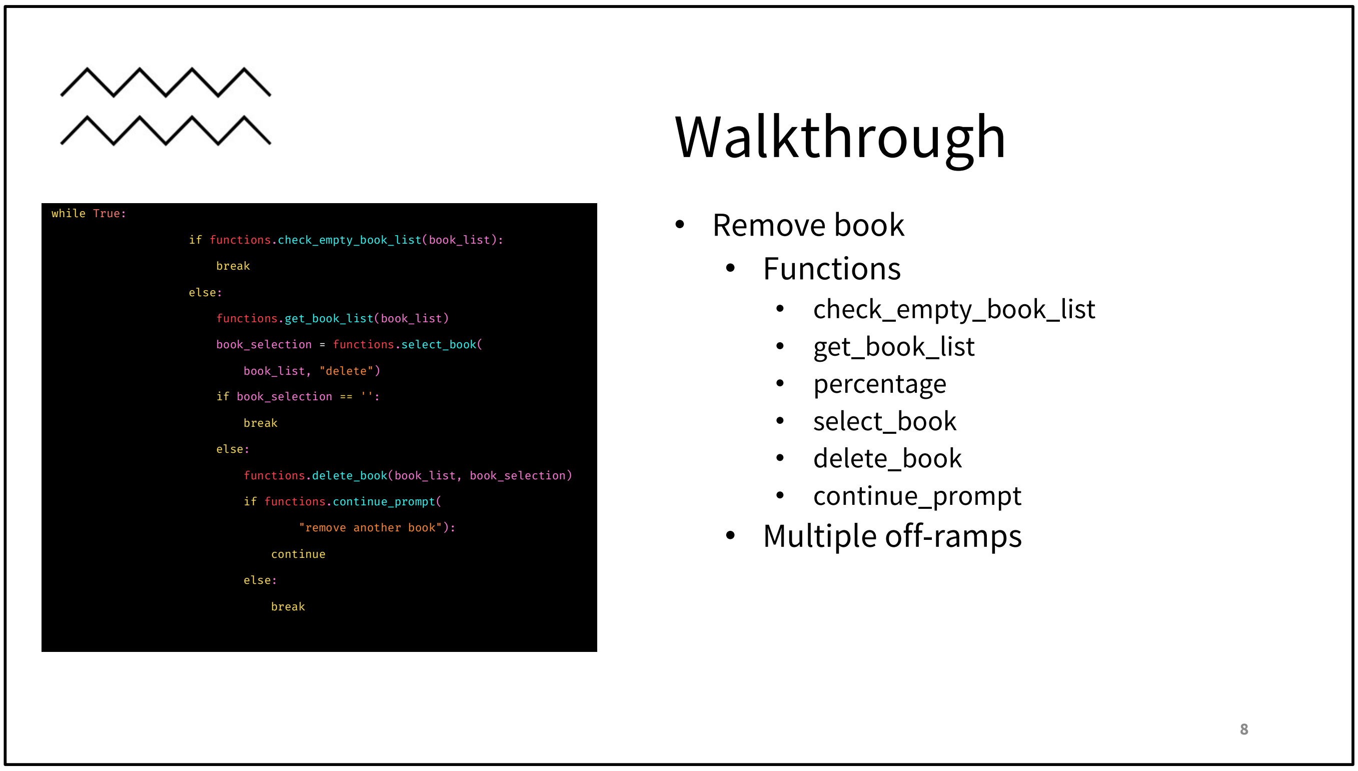
7

- The add book feature involves a number of steps requiring a sequential validation of inputs according to different criteria.
- As a result, this feature uses many functions, including nested functions, to perform these steps in an easy-to-follow separation of concerns.
- Add book info handles both the book title and author, as these have the same requirements (e.g., can be any string as long as it is empty).
- Check book dupes makes sure that the book list does not have any books where the title *and* author matches.
  - It is important to check for both title and author matches, as some books may share titles or authors.
- Add pages ensures the page number is a number, not a string, and that the number is higher than zero.
- Add tags allows the user to enter no tags or as many tags as they would like, separated by commas
  - With so many options available to the user here, the function has a number of functions to prevent PEBKAC:
    - The set function catches any duplicate tags entered by the user
    - The strip function deletes white space if the user opts to add spaces after the commas
  - The split function removes the commas when assembling the list of tags
- Add book appends the new book to the book list, ensures the function’s local book

list variable is applied globally, and prints a confirmation message.

  - Book values are stored as dictionaries in a list for convenient retrieval.
  - Title and author are strings, pages are integers, tags are a list of strings.
  - The function’s default arguments set currently_reading to False, pages_read to 0 and tags to an empty list.

- The continue prompt asks the user if they would like to repeat the action to enable quick inputting.
  - As the continue prompt is used elsewhere, it passes an action (e.g., “add another book”) to the function for printing.



Walkthrough

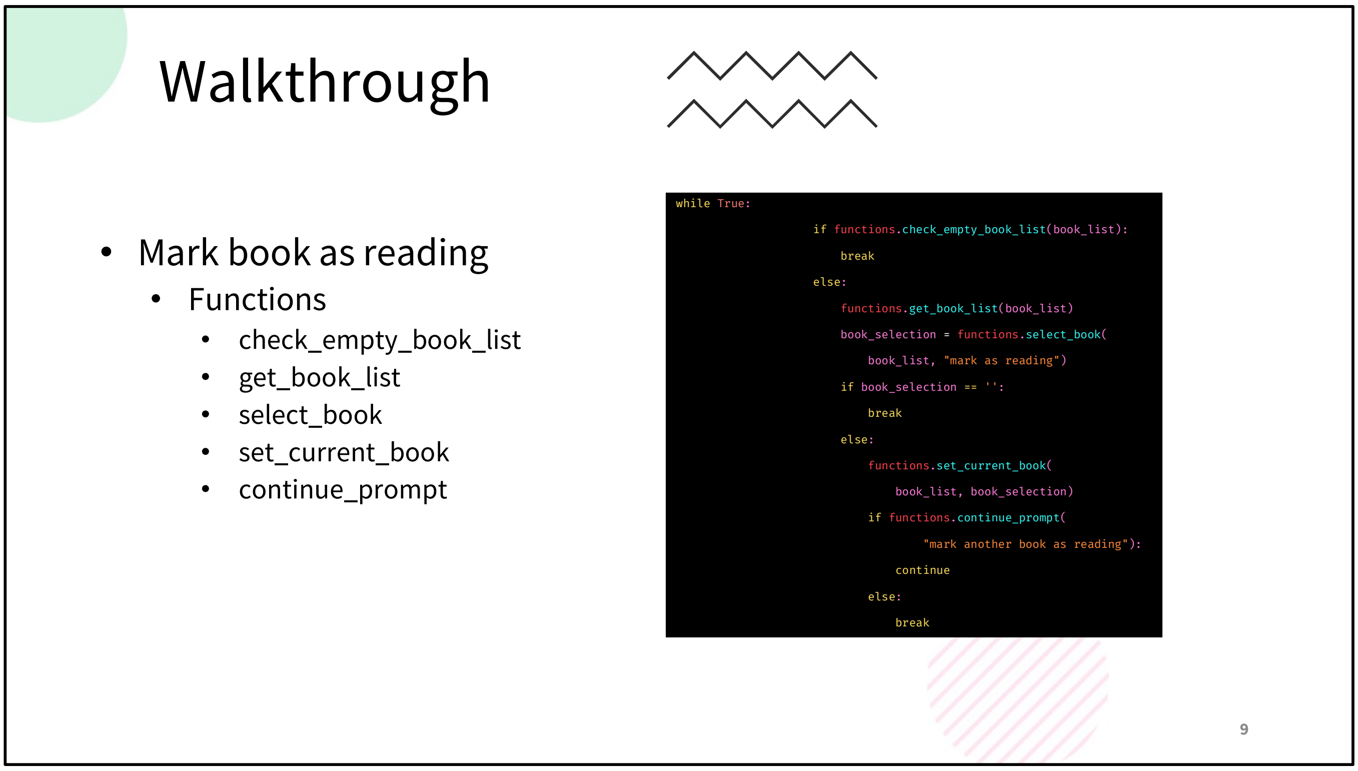
```
while True:
                    if functions.check_empty_book_list(book_list):
                        break
                    else:
                        functions.get_book_list(book_list)
                        book_selection = functions.select_book(
                            book_list, "delete")
                        if book_selection == '':
                            break
                        else:
                            functions.delete_book(book_list, book_selection)
                            if functions.continue_prompt(
                                    "remove another book"):
                                continue
                            else:
                                break
```

- Remove book
  - Functions
    - check_empty_book_list
    - get_book_list
    - percentage
    - select_book
    - delete_book
    - continue_prompt
  - Multiple off-ramps

8

- The remove book function sets the framework for features that involve retrieving an item from the book list and performing an action on it.
- As such, a lot of these functions are reused throughout the code.
- If there are no books in the book list, the check_empty_book_list will notify the user and prompt them to return to the menu.
  - To ensure the user gets back to the menu, the function will assess whether the book list is empty and, if so, request an input.
  - Regardless of the input, the function returns a True value, which will break the loop.
  - The intention is for the user to simply press Enter to exit, but provides additional safeguards if the user types something before pressing Enter.
- The get book list function prints an enumerated list of books to allow easily matching the book number with the book list index to remove a book.
  - This function will be explained further when discussing the list all books feature.
- The select book function adjusts the index value accordingly to correctly retrieve from list, catches common errors (e.g., ValueErrors from strings when expecting integers, IndexErrors when trying to access index out of range, and prompts when 0 or negative numbers would retrieve books from the back of the list, e.g., 0 – 1 = -1 (last item in list), -1 – 1 = -2 (second-last item in list)), and allows user to press Enter to cancel.

- The delete function removes the selected book from the book list and displays a confirmation message.
- The continue prompt asks the user if they would like to repeat the action to enable quick inputting.
- To prevent user error, especially with something as important as removing books, there are multiple off-ramps throughout, with the user able to cancel by pressing Enter.



Walkthrough

- Mark book as reading
  - Functions
    - check_empty_book_list
    - get_book_list
    - select_book
    - set_current_book
    - continue_prompt

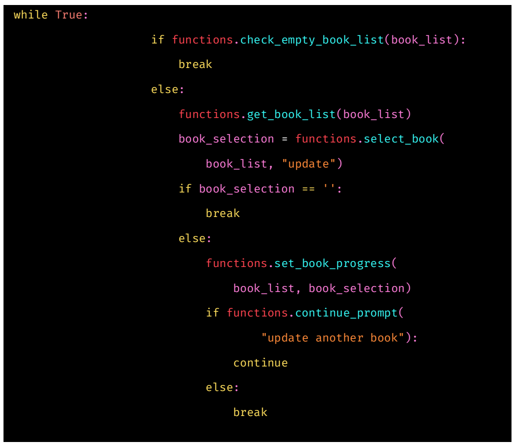
```
while True:
                    if functions.check_empty_book_list(book_list):
                        break
                    else:
                        functions.get_book_list(book_list)
                        book_selection = functions.select_book(
                            book_list, "mark as reading")
                        if book_selection == '':
                            break
                        else:
                            functions.set_current_book(
                                book_list, book_selection)
                            if functions.continue_prompt(
                                    "mark another book as reading"):
                                continue
                            else:
                                break
```

9

- To keep the code DRY and maintain separation of concerns, mark book as reading uses the same functions as removing a book, except calls set current book instead of delete book after a book has been selected.
- A Boolean value was chosen for the currently reading value, as it is either True or False.
- Using dictionary key-value pairs, set current book tells the user if a book is already being read, and if it isn't, changes the currently read value to True and displays a confirmation message.

Walkthrough

- Update book progress
  - Functions
    - check_empty_book_list
    - get_book_list
    - select_book
    - set_book_progress
    - set_book_pages
    - set_book_percent
    - delete_book
    - reverse_percentage
    - continue_prompt



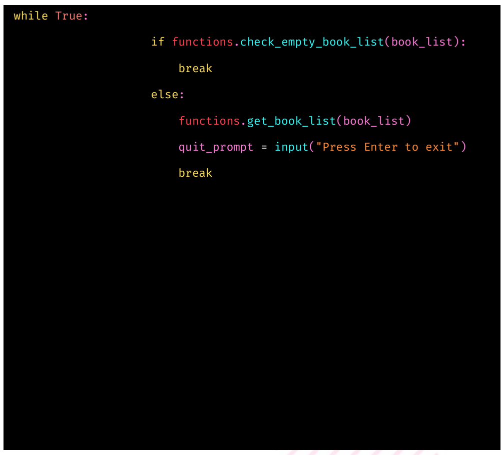
```
while True:
                    if functions.check_empty_book_list(book_list):
                        break
                    else:
                        functions.get_book_list(book_list)
                        book_selection = functions.select_book(
                            book_list, "update")
                        if book_selection == '':
                            break
                        else:
                            functions.set_book_progress(
                                book_list, book_selection)
                            if functions.continue_prompt(
                                    "update another book"):
                                continue
                            else:
                                break
```

10

- Update book progress also uses the same approach, but with set book progress calling either set book pages or set book percent.
- Set book progress ensures a valid number is received first, and forces the user to choose ‘pages’ or ‘progress’ with match cases and a case _ wildcard.
- Set book pages checks that the pages read don’t exceed the pages of the book and, if the pages read match the pages of the book, considers the book finished and deletes it; otherwise, it updates the book’s pages read value and displays the pages remaining
- Set book pages ensures a valid percentage is received (> 0, <= 100) and, if the percent read is 100, considers the book finished and deletes it; otherwise, it invokes the reverse percentage to adjust the book’s pages read value and displays the pages remaining.

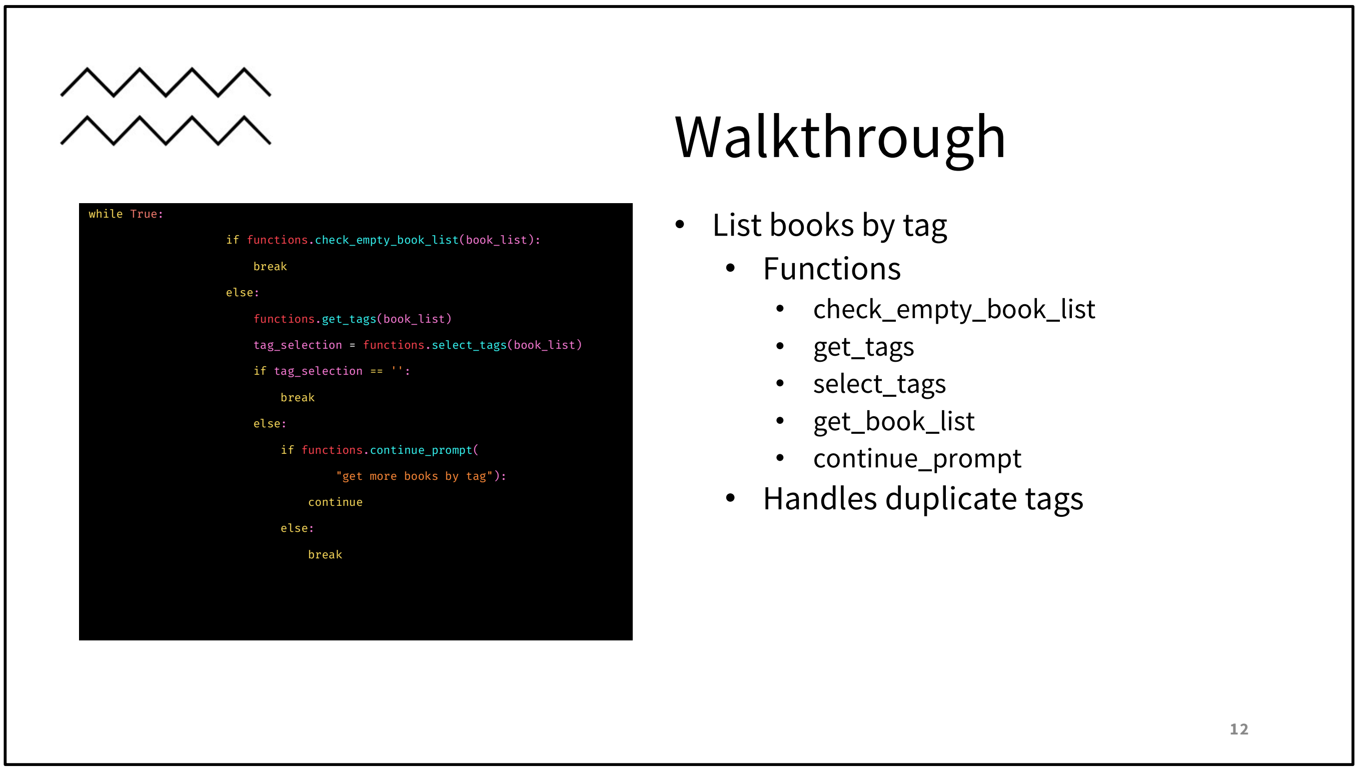
Walkthrough

- List all books
  - Functions
    - check_empty_book_list
    - get_book_list
    - percentage
  - Displays:
    - Title
    - Author
    - Pages
    - Pages read
    - Percent read
    - Currently reading (if applicable)



```
while True:
                    if functions.check_empty_book_list(book_list):
                        break
                    else:
                        functions.get_book_list(book_list)
                        quit_prompt = input("Press Enter to exit")
                        break
```

- The get book list function displays the book’s title, author, pages, pages read and, if and only if the book is marked as currently reading, an indication that the book is now being read by checking a bool value.
- This function also passes the book’s pages and pages read to the percentage function to calculate what percent of the book has been read.
- Instead of the continue prompt, this feature simply offers a quit prompt similar to the check empty book list’s, as no further action is required.



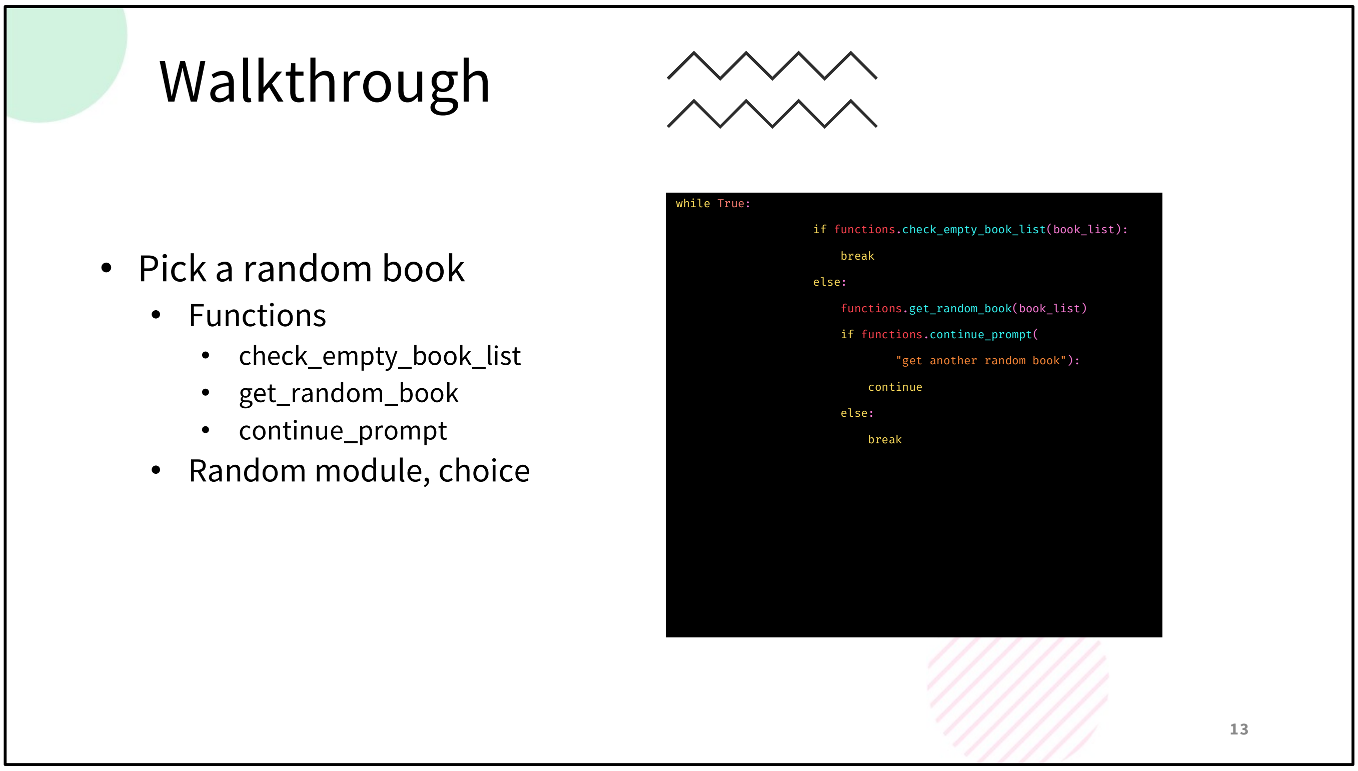
Walkthrough

```
while True:
                    if functions.check_empty_book_list(book_list):
                        break
                    else:
                        functions.get_tags(book_list)
                        tag_selection = functions.select_tags(book_list)
                        if tag_selection == '':
                            break
                        else:
                            if functions.continue_prompt(
                                    "get more books by tag"):
                                continue
                            else:
                                break
```

- List books by tag
    - Functions
        - check_empty_book_list
        - get_tags
        - select_tags
        - get_book_list
        - continue_prompt
    - Handles duplicate tags

12

- The list books by tag feature has to contend with the fact that, when retrieving a list tags, they may be duplicates due to different books sharing the same tags.
- To circumvent this, the get tags function uses the set function on a list comprehension of tags to retrieve unique tags, before the list function converts it to a list.
- It then iterates over the unique tags and displays them to the user.
- The select tags function allows the user to press Enter to exit and alerts the user if there are no matching books for their tag.
- Matching books are added to an empty tag book list by retrieving dictionary values through the tags key, and this list is passed to the book list function to keep the code DRY.



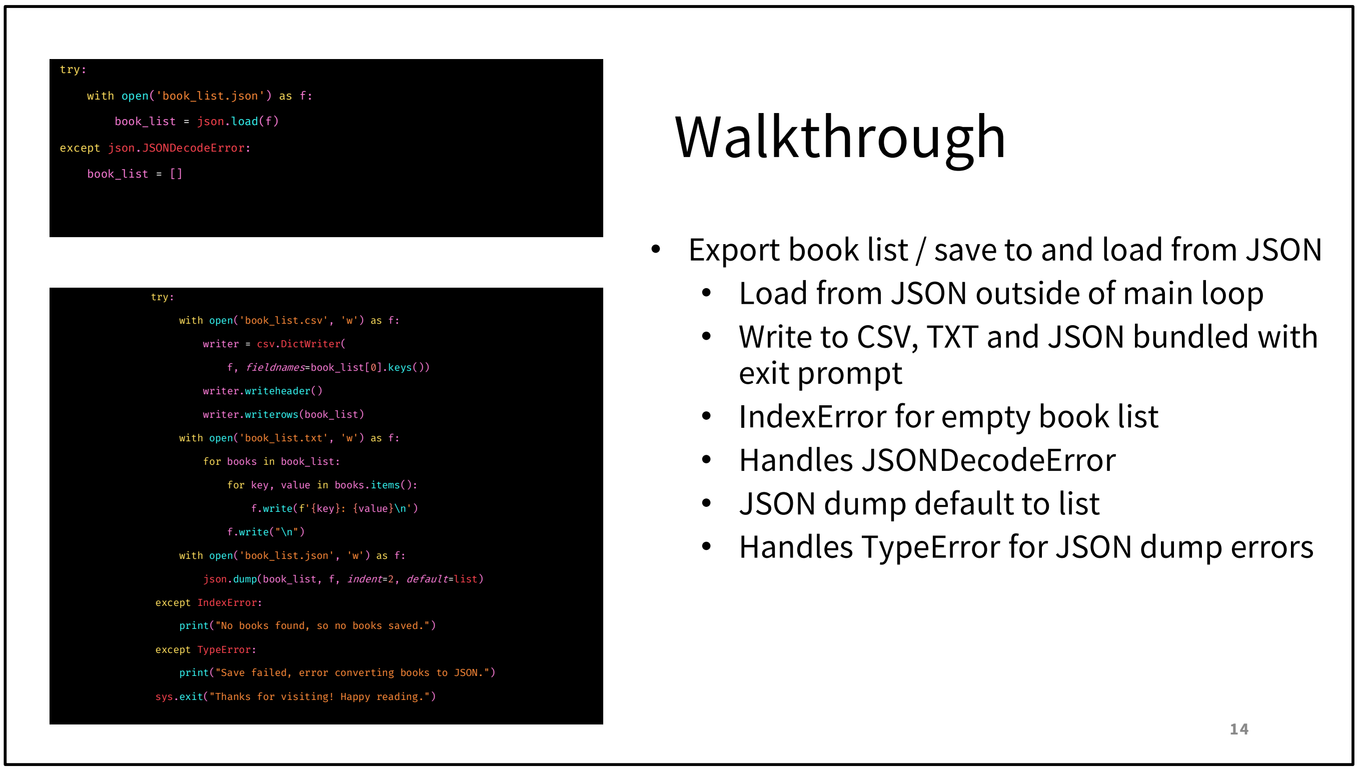
Walkthrough

- Pick a random book
  - Functions
    - check_empty_book_list
    - get_random_book
    - continue_prompt
  - Random module, choice

```
while True:
                    if functions.check_empty_book_list(book_list):
                        break
                    else:
                        functions.get_random_book(book_list)
                        if functions.continue_prompt(
                                "get another random book"):
                            continue
                        else:
                            break
```

13

- The pick a random book feature uses the random module from Python’s standard library.
- The choice function picks a random book from the book list.



```
try:
    with open('book_list.json') as f:
        book_list = json.load(f)
except json.JSONDecodeError:
    book_list = []
```

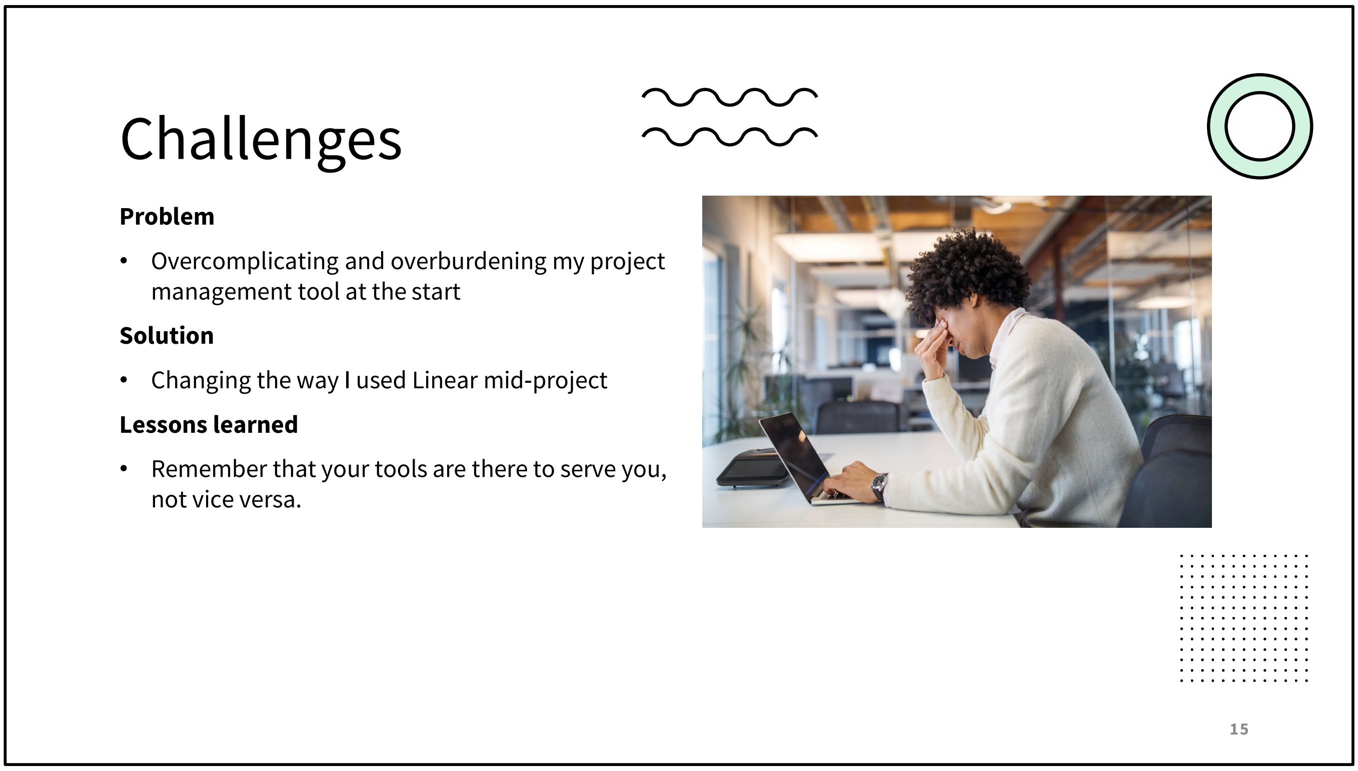
```
try:
    with open('book_list.csv', 'w') as f:
        writer = csv.DictWriter(
            f, fieldnames=book_list[0].keys())
        writer.writeheader()
        writer.writerows(book_list)
    with open('book_list.txt', 'w') as f:
        for books in book_list:
            for key, value in books.items():
                f.write(f'{key}: {value}\n')
            f.write("\n")
    with open('book_list.json', 'w') as f:
        json.dump(book_list, f, indent=2, default=list)
except IndexError:
    print("No books found, so no books saved.")
except TypeError:
    print("Save failed, error converting books to JSON.")
sys.exit("Thanks for visiting! Happy reading.")
```

Walkthrough

- Export book list / save to and load from JSON
  - Load from JSON outside of main loop
  - Write to CSV, TXT and JSON bundled with exit prompt
  - IndexError for empty book list
  - Handles JSONDecodeError
  - JSON dump default to list
  - Handles TypeError for JSON dump errors

14

- The application loads the JSON file into the book list before the main program loop initiates.
- In the event of an issue loading the JSON file, the application raises a JSONDecodeError to prevent a crash and simply sets the book list to a blank list.
- To ensure the book list is saved, the code for writing to CSV, TXT and JSON runs after the user selects the menu choice to quit but before the sys.exit() function is executed.
- If the book list is empty, writing to CSV, TXT and JSON would raise an error, so an IndexError catches that exception.
- Setting the JSON dump default to list obviates a TypeError caused by trying to dump the tag list, but a TypeError exception is provided in case of any JSON conversion errors, both to prevent crashing the program and to notify the user that the export was unsuccessful.



Challenges

Problem

- Overcomplicating and overburdening my project management tool at the start

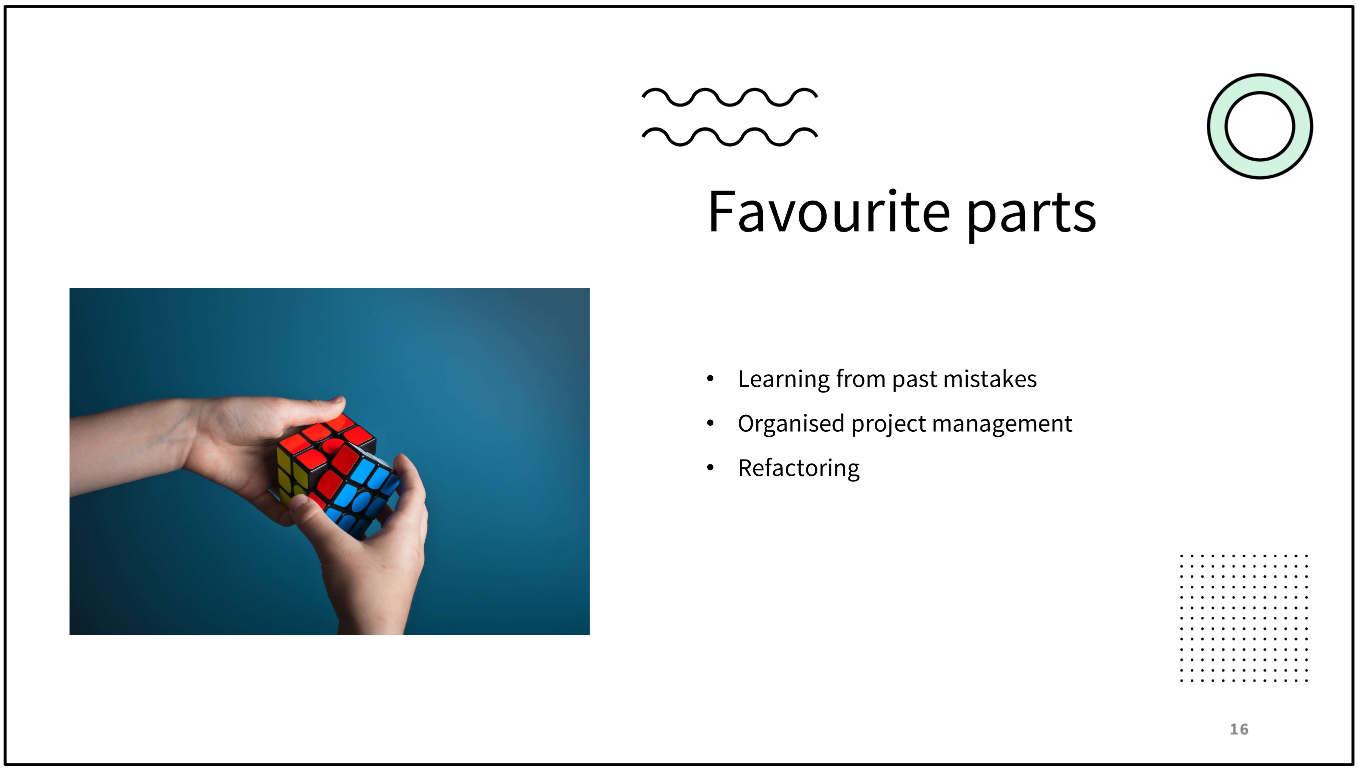
Solution

- Changing the way I used Linear mid-project

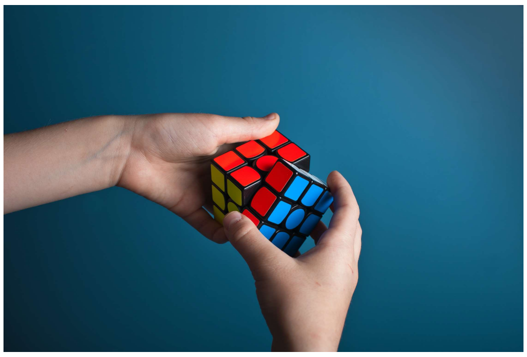
Lessons learned

- Remember that your tools are there to serve you, not vice versa.

- A major challenge was overcomplicating and overburdening my project management tool at the outset of the project.
- Lured by all the bells and whistles offered by Linear, I availed myself of all the options for managing my tasks.
- I soon discovered that I was micromanaging myself, and the features were causing more work for me while not providing enough benefit.
- My solution involved changing the way I used Linear mid-project.
- I realised that I didn’t have to log every task with every possible variable offered by Linear.
- Going forwards, I will remember that my tools are there to serve me, not vice versa – if they are bogging me down, then they are not working and I should shift my approach.



Favourite parts



- Learning from past mistakes
- Organised project management
- Refactoring

16

- My favourite parts of this application development included the following:
  - Learning from past mistakes.
    - During my last project, I identified a failure to correctly estimate the scope and complex interactivity required.
    - I resolved to be more organised from the get-go this time.
    - By learning from this past mistake, I was not as overwhelmed this time, and it was a lot easier to rework my code.
  - Organised project management.
    - Using the project management tool Linear made a marked different to my experience.
    - Having everything organised made everything more manageable and tangible.
    - Source control through git provided peace of mind and enabled me to revert changes when problems arose.
  - Refactoring.
    - I was surprised at how much I enjoyed refactoring my code.
    - At first, it seemed daunting, but by breaking it up into small, discrete steps, I was able to refactor sections without affecting the rest of the code.
    - The problem solving involved in refactoring my code was fun, but even better, having simpler, DRYer code made it easier to maintain

and update later on.

THANK YOU

LINKEDIN.COM/TOBYFEHILY

GITHUB.COM/TOBYFEHILY

TOBYFEHILY@GMAIL.COM

17

- Thank you!