Republic of the Philippines
POLYTECHNIC UNIVERSITY OF THE PHILIPPINES
BIÑAN CAMPUS
Bachelor of Science in Computer Engineering

# CINEMA BOOKING SYSTEM

In Partial Fulfillment
Of the Requirements for CMPE 102 : Programming, Logic, and Design

by:
Artes, Francesca Q.
Caravana, Ana Maricris A.
Serdeña, Arzel D.
Toquero, Jacqueline Sophia R.
Villapando, Christelle Denise C.
Group 5 (BSCpE 1 – 1)

Sir Jobert Cadiz

March 2023

## INTRODUCTION

The Cinema Booking System is a software project that aims to automate the booking process in a cinema. It opts to design and implement three programs using Java, Python, and C++ programming languages with the same output and purpose. The objective of this project is to demonstrate the versatility and functionality of these three programming languages in developing a cinema booking system. The programs developed for this project apply various programming concepts such as decision structures, Boolean logic, repetition structures, functions, input validation, arrays, files, and menus. These concepts are essential in developing a functional and reliable cinema booking system.

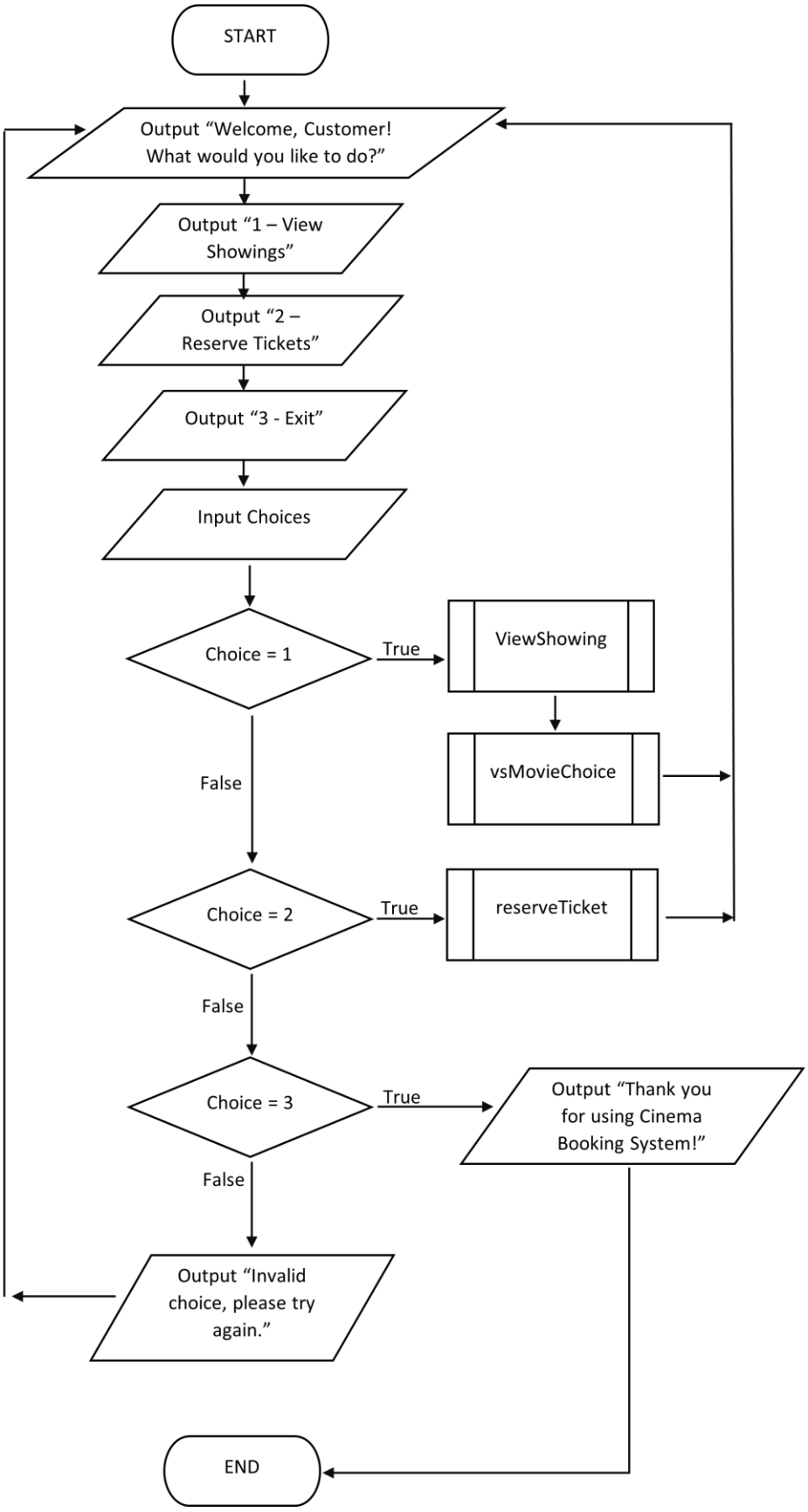
The purpose of the Cinema Booking System is to enable users to book seats for their preferred movie, date, and time slot. The program provides a user-friendly interface that allows customers to browse available movies, view their description and genre, and reserve their preferred seats. The booking system is designed to be easy to use and navigate, with clear instructions and options presented to the user.

Java, Python, and C++ were chosen for this project due to their widespread use and popularity in the software development industry. Each of these programming languages has its own unique features and benefits, making them ideal for different types of software development projects. The use of the same source code in all three languages allows for a direct comparison of their strengths and weaknesses in developing the same program.

In conclusion, the Cinema Booking System project aims to demonstrate the versatility and functionality of three popular programming languages, Java, Python, and C++, in developing a cinema booking system. The use of the same source code in all three languages allows for a direct comparison of their strengths and weaknesses in developing the same program. The program applies various programming concepts, such as decision structures, Boolean logic, repetition structures, functions, input validation, arrays, files, and menus, to provide a functional and reliable cinema booking system.

**DESIGN**

The following diagram illustrates the logic implemented in the program.



**IMPLEMENTATION**

In Python

The program starts by importing two libraries: "tabulate" and "random". The "tabulate" library is used to print out data in a neat and organized format, while "random" is not used in this program.

Next, the "openpyxl" library is imported to read data from an Excel file named "film_list.xlsx". The data from the first sheet of the Excel file is then stored in a variable named "sheet". The movies are stored in a dictionary named "movies", where the keys are the movie numbers, and the values are another dictionary containing the movie details such as title, description, genre, and price.

The "seats" dictionary stores information about the seats available for each movie. It contains sub-dictionaries for each movie, with keys "A" through "E" representing the different rows of seats, and the values being a list of integers representing the seat numbers available in each row.

The program then defines a function called "dates_movie1()" to allow the user to select a date and time slot for movie 1. This function prints out a table of available dates and prompts the user to enter a number corresponding to their desired date. If the user enters an invalid number, the program will prompt them to enter a valid number. Once the user selects a valid date, the function prints out a table of available time slots for the selected date and prompts the user to enter a number corresponding to their desired time slot. Again, if the user enters an invalid number, the program will prompt them to enter a valid number. Once the user selects a valid time slot, the function returns the selected date and time slot as a tuple.

The "view_showings()" function prints out a table of all the movies available and their corresponding numbers.

Finally, the "vs_movie_choice()" function prompts the user to choose a movie number and displays the available dates and time slots for that movie using the "dates_movie1()" function. Once the user selects a valid date and time slot, the program displays the available seats for that movie and prompts the user to choose a seat. If the user enters an invalid seat number or the seat is already taken, the program will prompt the user to enter a valid seat number. Once the user selects a valid seat, the program displays the ticket details and prompts the user to confirm the booking. If the user confirms the booking, the program updates the "seats" dictionary and writes the ticket

details to a text file. If the user cancels the booking, the program returns to the main menu.

Source Code: https://github.com/anamaricriscaravana/Final-Project/blob/main/CinemaBookingSystem.py

In Java

The program begins by importing the java.util package, which provides the Scanner class for user input, and then defines two static Map objects: "seats" and "movies". The "seats" map is a nested map containing the available seats for each movie, while the "movies" map contains information about the movies, including their title, genre, and price. Both maps are initialized using the HashMap class.

The program initializes the "movies" map using the put() method, which allows the program to add key-value pairs to the map. The keys in the map correspond to the movie numbers (1-5), and the values are nested HashMaps containing the movie title, genre, and price.

The program also initializes the "seats" map using a for loop that iterates over the keys of the "movies" map. For each movie, the program creates a new HashMap object to store the available seats for each row. The available seats for each row are stored as a List of Integers, which are added to the corresponding row in the HashMap. Finally, the row map is added to the "seats" map using the movie number as the key.

The main() method contains a while loop that repeatedly displays a menu of options to the user and waits for input. The user can choose to view the available showings, reserve a ticket for a movie, or exit the program. Depending on the user's input, the program calls the appropriate method: viewMovies() or reserveTicket().

The viewMovies() method displays a formatted list of the available movies and their corresponding details, including the title, genre, and price. It uses a for loop to iterate over the keys of the "movies" map and retrieve the corresponding values using the get() method. The retrieved values are then formatted and printed to the console using the printf() method.

The reserveTicket() method prompts the user to select a movie to book tickets for and then displays the available seats for that movie. It uses the Scanner class to read input from the user and validate the input for the movie number and number of tickets. It then retrieves the available seats for the selected movie from the "seats" map and

displays them to the user in a formatted table. The user is prompted to select a seat for each ticket, and the program validates the input and marks the selected seats as unavailable in the "seats" map.

Source Code: https://github.com/anamaricriscaravana/Final-Project/blob/main/CinemaBookingSystem.java

In C++

This program is an implementation of a cinema booking system. It has three functions: viewMovies, viewAvailableTickets, and reserveTicket. It includes several data structures like maps and vectors to store and manage information related to movies, seats, and ticket prices.

The program starts with the init function, which initializes the data structures. It creates a map called "movies," which stores movie numbers as keys and a nested map as values. The nested map stores the movie's title and price as key-value pairs. It also creates a map called "seats," which stores movie numbers as keys and a nested map as values. The nested map stores the seat rows as keys and a vector of available seat numbers as values.

The main function runs an infinite loop that displays a menu of options: reserve a ticket, view available movies, or cancel. Depending on the user's input, it calls the respective function.

The viewMovies function prints a table of available movies with their titles and prices. The viewAvailableTickets function prints a table of available seats for each movie. The reserveTicket function prompts the user to select a movie and the number of tickets to book. It then displays the available seats for that movie and prompts the user to select the seats. After the user selects the seats, the function reserves the tickets and updates the seats data structure.

Source Code: https://github.com/anamaricriscaravana/Final-Project/blob/main/CinemaBookingSystem.cpp

**Challenges Encountered**

One of the main challenges faced by the students in developing the cinema booking system in Python, Java, and C++ is the complexity of the code. They initially planned to make it very easy and simple, however as the project progressed, the code became longer, making it difficult to keep track of everything. To overcome this challenge, the students broken down the code into smaller, manageable chunks, called modules which could be tested and integrated into the system as needed. This have made it easier to debug and maintain the code.

Another issue encountered by the students was the difficulty in making the code concise. Since every value was sourced from an external file, it was challenging to ensure that the code was both concise and functional. To solve this problem, the students have used data structures, dictionaries and lists, to store the values and retrieve them when needed. This approach would made the code more concise and easier to maintain.

Syntax and logic errors were also a common issue encountered by the students during the implementation of the cinema booking system. This problem was solved by adopting good coding practices such as writing clean and well-documented code, using descriptive variable names, and testing the code regularly to ensure that it works as expected. The students also utilized tools such as debuggers and IDEs to help them identify and fix syntax and logic errors.

Lastly, the students faced the challenge of incorporating all the programming concepts such as decision structures, Boolean logic, repetition structures, functions, input validation, arrays, files, menus, and recursion. While this is an ambitious goal, it can be overwhelming, especially for them who are still learning the basics of programming. To overcome this challenge, the students focused on implementing the essential concepts first, such as decision structures and input validation, before moving on to the more advanced ones.

In conclusion, developing a cinema booking system using programming languages such as Python, Java, and C++ is a challenging task, however, by breaking down the code into modules, adopting good coding practices, and focusing on essential programming concepts, the students overcame these challenges and develop a functional cinema booking system.

**TESTING**

The following table shows the testing approach, the test cases used and their expected outcomes, and the results of testing for each program.

| Testing Approach | Description | Language | Test Case | Input | Expected Result | Actual Result | Result |
|---|---|---|---|---|---|---|---|
| Unit Testing | Test each method/function separately to check their behavior. | Python | Testing if the movie's available showings are displayed | [1] | Display List of Movies | As Expected | Pass |
| | | | Testing if the selected date and time are returned correctly | [2, 1, 1] | Display the correct date and time selected | As Expected | Pass |
| | | | Testing if an error message is displayed for an invalid time slot selection | [2, 1, 4] | Display “Invalid choice. Please try again.” | As Expected | Pass |
| | | Java | Testing if the movie's available showings are displayed | [1] | Display List of Movies | As Expected | Pass |
| | | | Testing if the selected date and time are returned correctly | [2, 1, 1] | Display the correct date and time selected | As Expected | Pass |
| | | | Testing if an error message is displayed for an invalid time slot selection | [2, 1, 4] | Display “Invalid choice. Please try again.” | As Expected | Pass |
| | | C++ | Testing if the movie's available showings are displayed | [1] | Display List of Movies | As Expected | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | Testing if the selected date and time are returned correctly | [2, 1, 1] | Display the correct date and time selected | As Expected | Pass |
| | | | Testing if an error message is displayed for an invalid time slot selection | [2, 1, 4] | Display “Invalid number. Please try again.” | As Expected | Pass |
| Integration Testing | Test all methods/functions in combination to check their compatibility with each other. | Python | Selecting a movie showing | [1, 1] | Display movie details | As Expected | Pass |
| | | | Invalid input | [2, 1, 4] | Display “Invalid number. Please try again.” | As Expected | |
| | | | Viewing all available showings | [1] | Display a table with all available movie showings | As Expected | Pass |
| | | Java | Selecting a movie showing | [1, 1] | Display movie details | As Expected | Pass |
| | | | Invalid input | [2, 1, 4] | Display “Invalid number. Please try again.” | As Expected | Pass |
| | | | Viewing all available showings | [1] | Display a table with all available movie showings | As Expected | Pass |
| | | C++ | Selecting a movie showing | [1, 1] | Display movie details | As Expected | Pass |
| | | | Invalid input | [2, 1, 4] | Display “Invalid number. Please try again.” | As Expected | Pass |
| | | | Viewing all available showings | [1] | Display a table with all available | As Expected | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | movie showings | | |
| Acceptance Testing | Test the entire program with various inputs and compare the output with expected results. | Python | Check if the program prints available dates for the first movie (Movie 1) | [2, 1] | Display a table with available dates for the selected movie | As Expected | Pass |
| | | | Check if the program prints available time slots for the selected date | [2, 1, 1] | Display a table with available time slots for the selected movie on the selected date | As Expected | Pass |
| | | | Check if the program returns the selected date and time slot. | [2, 1, 1, 1] | Return the selected date and time slot | As Expected | Pass |
| | | Java | Check if the program prints available dates for the first movie (Movie 1) | [2, 1] | Display a table with available dates for the selected movie | As Expected | Pass |
| | | | Check if the program prints available time slots for the selected date | [2, 1, 1] | Display a table with available time slots for the selected movie on the selected date | As Expected | Pass |
| | | | Check if the program returns the selected | [2, 1, 1, 1] | Return the selected date and time slot | As Expected | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | date and time slot. | | | | |
| | | C++ | Check if the program prints available dates for the first movie (Movie 1) | [2, 1] | Display a table with available dates for the selected movie | As Expected | Pass |
| | | | Check if the program prints available time slots for the selected date | [2, 1, 1] | Display a table with available time slots for the selected movie on the selected date | As Expected | Pass |
| | | | Check if the program returns the selected date and time slot. | [2, 1, 1, 1] | Return the selected date and time slot | As Expected | Pass |

**USER MANUAL**

Preparation of Program

Python

1. Install Required Libraries
   1.1. Open command prompt or terminal.
   1.2. Type "pip install tabulate" and press enter.
   1.3. Type "pip install openpyxl" and press enter.
2. Run the Program
   1.4. Open your Python IDE or editor.
   1.5. Open a new Python file.
   1.6. Copy and paste the code into the new file.
   1.7. Save the file with a .py extension.
   1.8. Run the program by clicking the "Run" button or by typing "python <filename>.py" in the terminal.

Java

1. Installation
    1.1. Download the CinemaBookingSystem.java file.
    1.2. Open a terminal or command prompt.
    1.3. Navigate to the directory where the CinemaBookingSystem.java file is located.
    1.4. Compile the program by typing "javac CinemaBookingSystem.java" and press enter.
    1.5. Run the program by typing "java CinemaBookingSystem" and press enter.
2. Run the Program
    2.1. Open the IDE.
    2.2. Create a new Java project.
    2.3. Copy the CinemaBookingSystem.java file into the project folder.
    2.4. Add the CinemaBookingSystem.java file to the project.
    2.5. Compile and run the program.

C++

Preparation of the Program

1. Open a terminal or command prompt.
    1.1. Clone the repository from GitHub using the following command: git clone https://github.com/chatgpt/cinema-booking-system.git
    1.2. Navigate to the directory where the repository is cloned.
    1.3. Compile the program using the command: g++ main.cpp -o cinema-booking-system
    1.4. Run the program using the command: ./cinema-booking-system

Program Navigation

1. The program will display a menu with the following options:
    1.1. View Showings
    1.2. Reserve Ticket
    1.3. Exit Menu
2. View Showings:
    2.1. Select option 1 from the menu to view available movie showings.
    2.2. The program will display a table with movie numbers and titles.
    2.3. Choose a movie by typing the corresponding number and pressing enter.

2.4. The program will display the details of the movie such as the title, description, genre, and standard price of the ticket.

3. Reserve Ticket:
   3.1. Select option 2 from the menu to reserve a movie ticket.
   3.2. The program will display a table with movie numbers and titles.
   3.3. Choose a movie by typing the corresponding number and pressing enter.
   3.4. The program will display the available dates and time slots for the selected movie.
   3.5. Choose a date and time slot by typing the corresponding number and pressing enter.
   3.6. The program will display a seating chart for the selected movie.
   3.7. Choose seats by typing the row and seat number (e.g. A1) and pressing enter.
   3.8. The program will display the selected options, total price, and transaction number.
   3.9. Choose whether to confirm the booking or cancel it by entering Y for yes or N for no.
   3.10. The program will confirm the booking.
   3.11. The program will return to the main menu.
4. Exit Menu
   4.1. Select option 3 to Exit the Menu
   4.2. The program will end.

Assumptions

1. The given program reads a movie list from an excel file named "film_list.xlsx" and stores it in a dictionary named 'movies'. It also creates a nested dictionary named 'seats', which stores seat information for each movie in the movie dictionary. Then, it defines some functions to view available dates and time slots for a selected movie, and to view currently showing movies.
2. The program assumes that the excel file "film_list.xlsx" exists in the same directory as the Python file. The program also assumes that the excel file has a sheet named "Sheet1" and the data is organized as follows:
   2.1. The movie title is located in column A starting from row 2 and in increments of 9 rows thereafter.
   2.2. The movie description is located in column B starting from row 2 and in increments of 9 rows thereafter.

2.3. The movie genre is located in column C starting from row 2 and in increments of 9 rows thereafter.
2.4. The movie price is located in column D starting from row 2 and in increments of 9 rows thereafter.
2.5. The available dates for each movie are located in column E starting from row 2 and in increments of 3 rows thereafter. Each movie has 3 available dates.
2.6. The available time slots for each date are located in column F starting from row 2 and in increments of 1 row thereafter. Each time slot has a unique ID.

3. The program also assumes that the tabulate library and openpyxl library are installed.
4. Furthermore, the program assumes that the user inputs valid choices when prompted for input. If the user inputs an invalid value, the program assumes the user will enter the correct value in the subsequent prompts.

**CONCLUSION**

The Cinema Booking System project has achieved the objective of demonstrating the versatility and functionality of three popular programming languages, Java, Python, and C++, in developing a cinema booking system. By using the same source code in all three languages, the project provides a direct comparison of their strengths and weaknesses in developing the same program. The project has also applied various programming concepts, such as decision structures, Boolean logic, repetition structures, functions, input validation, arrays, files, and menus, to provide a functional and reliable cinema booking system.

The project has made the students more knowledgeable about the three programming languages and their applications in solving advanced programming issues. The students have faced various challenges, such as the complexity of the code, making the code concise, syntax and logic errors, and incorporating all the programming concepts. However, they have overcome these challenges by breaking down the code into manageable chunks, using data structures to store values, adopting good coding practices, and focusing on essential programming concepts.

To enhance the project in the future, the addition of new features such as a cancellation system and user accounts could improve the user experience. The cancellation system would allow customers to cancel their reservations if necessary, freeing up seats for other customers. The user account feature would allow customers to create an account, view their booking history, and make future bookings more efficiently. The rating system would also be an

excellent addition, allowing customers to rate and review the movies they have watched, giving other customers an idea of what to expect.

Implementing these new features would require the incorporation of new programming concepts such as database management, user authentication, and user input validation. Using SQL, the database could store user information, booking history, and movie ratings. User authentication would ensure that only authorized users can access their accounts, and input validation would prevent users from entering invalid data.

In summary, the Cinema Booking System project has achieved its objective of demonstrating the versatility and functionality of three programming languages in developing a cinema booking system. The project has provided the students with valuable experience in coding, problem-solving, and project management. By overcoming the challenges encountered during the project, the students have become more effective in solving advanced programming issues.

**REFERENCES**


*Arrays (The Java™ Tutorials > Learning the Java Language > Language Basics)*. (n.d.). https://docs.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

Bayani, R. (2016, September 9). *Repetition (Looping) Control Structures in C++*. TECHNIG. https://www.technig.com/repetition-looping-control-structures/

Bhardwaj, H. (2023, February 13). *What is Recursion in C++? Types, its Working and Examples*. Simplilearn.com. https://www.simplilearn.com/tutorials/cpp-tutorial/what-is-recursion-in-cpp

*C++ Function (With Examples)*. (n.d.). https://www.programiz.com/cpp-programming/function

Cosentino, S. (2022, August 3). *Understanding Basic Decision Structures in Python - The Startup*. Medium. https://medium.com/swlh/understanding-basic-decision-structures-in-python-24dd04f28717

Dancuk, M. (2023, February 7). *File Handling in Python: Create, Open, Append, Read, Write*. Knowledge Base by phoenixNAP. https://phoenixnap.com/kb/file-handling-in-python

*Decision Making & Looping Structure in C++*. (n.d.). https://www.tutorialride.com/cpp/decision-making-looping-structure-in-c.htm

GeeksforGeeks. (2020, July 10). *How to Create a Programming Language using Python*. https://www.geeksforgeeks.org/how-to-create-a-programming-language-using-python/

Great Learning Team. (2022, August 23). *File Handling Through C++ | How to Open, Save, Read and Close*. Great Learning Blog: Free Resources What Matters to Shape Your Career! https://www.mygreatlearning.com/blog/file-handling-in-cpp/

*How to Validate user input in Python | bobbyhadz*. (n.d.). Blog - Bobby Hadz. https://bobbyhadz.com/blog/python-input-validation

International Journal of Recent Research Aspects ISSN 2349-7688. (2020, January 6). *Efficient Way of Web Development Using Python and Flask*. https://www.academia.edu/41538217/Efficient_Way_of_Web_Development_Using_Python_and_Flask

J. (2022, December 16). *Why Learn Python? Five Reasons to Start Programming With Python*. UT Austin Boot Camps. https://techbootcamps.utexas.edu/blog/why-learn-python-get-started-programming/

*Java File Class - javatpoint*. (n.d.). www.javatpoint.com. https://www.javatpoint.com/java-file-class

*Java Library Functions | Programiz*. (n.d.). https://www.programiz.com/java-programming/library

Kenneth, C. (2021, December 14). *Understanding Control Structures in Java - Analytics Vidhya*. Medium. https://medium.com/analytics-vidhya/understanding-control-structures-in-java-3eacbc294600

Mandliya, A. (2021, May 15). *Input Validation in C++*. Java2Blog. https://java2blog.com/input-validation-cpp/

Mirza, A. (2023, January 27). *Building a Website using C++. A Step-by-Step Guide to Setting Up a. . . | by Arslan Mirza | Medium | DataDrivenInvestor*. Medium. https://medium.datadriveninvestor.com/building-a-website-using-c-ef7d27ad3aa

Naik, V. (2021, December 23). *Boolean Operators in Python*. Scaler Topics. https://www.scaler.com/topics/python/boolean-operators-in-python/

Pedamkar, P. (2022, August 25). *Boolean Operators in C++*. EDUCBA. https://www.educba.com/boolean-operators-in-c-plus-plus/

*Python Booleans*. (n.d.). https://www.w3schools.com/python/python_booleans.asp

*Python Built-in Functions*. (n.d.). https://www.w3schools.com/python/python_ref_functions.asp

Python Tutorial. (2022, September 15). *Understanding Python Recursive Functions By Practical Examples*. Python Tutorial - Master Python Programming for Beginners From Scratch. https://www.pythontutorial.net/python-basics/python-recursive-functions/

*decision-making-in-java*. (n.d.). techvidvan.com. https://techvidvan.com/tutorials/decision making-in-java/

S, R. A. (2023, February 20). *Everything You Need to Know About Python Arrays*. Simplilearn.com. https://www.simplilearn.com/tutorials/python-tutorial/python-arrays

S, R. A. (2023a, February 13). *An Easy Guide to Understand the C++ Array*. Simplilearn.com. https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-array

Tomić, J. (2020, February 6). *Stork: How to Make a Programming Language in C++*. Toptal Engineering Blog. https://www.toptal.com/c-plus-plus/creating-programming-language-in-c-plus-plus

*What are control flow statements in Python?* (n.d.). Educative: Interactive Courses for Software Developers. https://www.educative.io/answers/what-are-control-flow-statements-in-python

Winter, R. (2023, January 21). *How Recursion Works in Java*. https://blog.hubspot.com/website/recursion-java