Flight Ticket Booking System

Comprehensive Project Report

Cover Page

Project Title: Flight Ticket Booking System

Subtitle: Console-Based Python Application for Domestic and International Flight Reservations

Date: November 24, 2025

Program Language: Python 3.x

Project Type: Console Application Development

Status: Project Documentation

Author: Engineering Student - 1st Year

---

1. Introduction

1.1 Project Overview

The Flight Ticket Booking System is a Python-based console application designed to facilitate the booking of flight tickets for both domestic and international travel. This application provides a user-friendly interface for travelers to select flight preferences, enter travel details, and receive a complete booking confirmation with randomly generated seat numbers and prices.

The primary objective of this project is to demonstrate core programming concepts including user input handling, conditional logic, function definition, and data manipulation in Python. This serves as an excellent learning platform for understanding how booking systems work at their fundamental level.

1.2 Project Scope

The Flight Ticket Booking System encompasses:

- Support for both domestic and international flight bookings
- Collection of essential travel information from users
- Seat preference selection with multiple class options

- Trip type selection (one-way or round-trip)
- Automatic generation of seat assignments and pricing
- Display of comprehensive booking confirmation details
- Console-based user interface for simple deployment

1.3 Intended Audience

- First-year engineering students learning Python fundamentals
- Individuals studying basic application development
- Those interested in understanding booking system logic
- Educational institutions teaching console application development

---

2. Problem Statement

2.1 Current Challenges

- **Limited Learning Resources:** Students need practical, real-world projects to understand program flow and user interaction
- **Complex Systems:** Actual flight booking systems are too complex for beginners to understand
- **Lack of Practical Application:** Theoretical programming concepts need practical implementation examples
- **User Interaction Understanding:** Students struggle to grasp how to handle multiple user inputs and conditional branching
- **Data Generation:** Limited understanding of random data generation for realistic simulation

2.2 Project Goals

1. Create a simplified flight booking system to demonstrate core Python concepts
2. Provide a practical example of user input handling and validation
3. Demonstrate the use of functions and conditional statements
4. Show how to simulate realistic booking data with random generation
5. Create a foundation for future enhancements and learning

---

3. Functional Requirements

3.1 Core Features

FR1: Flight Type Selection

- System prompts user to choose between domestic and international flights
- Based on selection, appropriate function is executed
- Supports binary input validation (domestic/international)

FR2: Travel Details Collection

- Collection of starting point (departure city)
- Collection of destination city
- User-provided input for travel locations

FR3: Seat Preference Selection

- Display three seating class options:
    - Economy Class
    - Premium Economy Class
    - Business Class
- Accept user selection and store preference

FR4: Trip Type Selection

- Allow selection between one-way and round-trip bookings
- Store trip preference for display in confirmation

FR5: Travel Date Input

- Accept date input from user
- Store and display date in booking confirmation

FR6: Ticket Confirmation Display

- Display all booking details in formatted output
- Show route information (From and To)
- Display selected date
- Present randomly generated seat number
- Show selected seat class
- Display randomly generated pricing based on flight type

FR7: Random Data Generation

- Generate random seat numbers (1-100 range)
- Generate domestic flight prices (INR 2,700-15,000 range)
- Generate international flight prices (INR 65,000-200,000 range)

---

4. Non-Functional Requirements

4.1 Performance Requirements

NFR1: Response Time

- Immediate response to user inputs
- Instant seat and price generation
- Real-time confirmation display

NFR2: Efficiency

- Minimal memory footprint
- Fast execution without delays
- Simple data structure requirements

4.2 Usability Requirements

NFR3: User Interface

- Clear, straightforward console prompts
- Easy-to-follow booking flow
- Formatted output for readability

NFR4: Accessibility

- Text-based interface works on any terminal
- No special dependencies or graphics libraries
- Cross-platform compatibility (Windows, Linux, macOS)

4.3 Maintainability Requirements

NFR5: Code Quality

- Well-structured Python code
- Clear function definitions
- Logical separation of concerns (domestic vs. international)

NFR6: Compatibility

- Python 3.x compatibility
- Works with standard library only
- No external dependencies required

---

5. System Architecture

5.1 Architecture Overview

The Flight Ticket Booking System follows a simple procedural architecture with function-based organization:

1. **Input Layer**: Console-based user input collection
2. **Processing Layer**: Function-based logic for domestic and international bookings
3. **Data Generation Layer**: Random seat and price generation
4. **Output Layer**: Formatted confirmation display

5.2 Component Description

Main Components:

- Program initialization and flight type selection
- `domestic()` function for domestic booking workflow
- `international()` function for international booking workflow
- Random module for seat and price generation
- Print statements for user interface and output

Data Flow:
Input → Flight Type Decision → Function Call → Data Collection → Random Generation → Output Display

---

6. Design Diagrams

6.1 Use Case Diagram

Primary user interactions with the booking system:

- **Traveler**: Enters flight type preference
- **Traveler**: Provides travel locations (origin and destination)
- **Traveler**: Selects seat class preference
- **Traveler**: Chooses trip type (one-way or round-trip)
- **Traveler**: Enters travel date
- **System**: Generates random seat number
- **System**: Generates random pricing
- **System**: Displays booking confirmation

6.2 Workflow Diagram

Application execution flow:

1. Program starts
2. Display flight type prompt (domestic/international)
3. User enters flight type
4. If "domestic": Execute `domestic()` function
5. If "international": Execute `international()` function
6. Inside function: Collect starting point
7. Collect destination
8. Display seat class options
9. Collect seat preference
10. Display trip type options
11. Collect trip preference
12. Collect travel date
13. Generate random seat number (1-100)
14. Generate random price based on flight type
15. Display formatted booking confirmation
16. Program ends

6.3 Sequence Diagram

Interaction between user and system:

1. User launches program
2. System displays "domestic/international" prompt
3. User enters choice
4. System evaluates choice and calls appropriate function
5. Function displays "Enter Starting Point" prompt
6. User enters origin city
7. System prompts for destination
8. User enters destination city
9. System displays seat class menu
10. User selects seat preference
11. System prompts for trip type
12. User selects one-way or round-trip
13. System prompts for travel date
14. User enters date

15. System generates random seat (1-100)
16. System generates random price (domestic: 2700-15000 OR international: 65000-200000)
17. System displays formatted confirmation with all details
18. Program terminates

6.4 Data Model

Variable	Type	Purpose
dist	String	Flight type (domestic/international)
start	String	Departure city/location
dest	String	Destination city/location
pref	String	Seat class preference
trip	String	Trip type (one-way/round-trip)
date	String	Travel date
seat_number	Integer	Random seat assignment (1-100)
price	Integer	Random price based on flight type

---

7. Design Decisions & Rationale

7.1 Architectural Decisions

Decision 1: Function-Based Organization

- Rationale: Separates domestic and international logic for clarity and maintainability
- Alternative: Monolithic approach (chose functions for better organization)
- Benefit: Easier to understand and modify individual booking flows

Decision 2: Console-Based Interface

- Rationale: Simplicity for learning Python fundamentals without GUI complexity
- Alternative: GUI framework (chose console for accessibility and lightweight nature)
- Benefit: Works on any system with Python installed

Decision 3: Random Data Generation

- Rationale: Simulates realistic booking system without database
- Alternative: Fixed prices and seats (chose randomization for realism)
- Benefit: Different booking generates different results for testing

7.2 Implementation Decisions

Decision 4: Price Range Differentiation

- Rationale: Domestic flights cheaper than international (realistic simulation)
- Domestic: INR 2,700-15,000
- International: INR 65,000-200,000
- Benefit: Reflects real-world pricing patterns

Decision 5: Seat Number Range

- Rationale: Reasonable aircraft capacity simulation
- Range: 1-100 seats
- Benefit: Realistic for medium-size aircraft

Decision 6: Input Method

- Rationale: Direct text input via `input()` function for simplicity
- Alternative: Command-line arguments (chose `input()` for interactive experience)
- Benefit: Clear user-system interaction

---

8. Implementation Details

8.1 Technology Stack

Component	Technology	Purpose
Language	Python 3.x	Application development
I/O	print(), input()	User interaction
Randomization	random module	Seat and price generation
Platform	Console/Terminal	Execution environment
Version Control	Git/GitHub	Code management

8.2 Code Structure

Program Structure:

```
tickect-booking.py
├── Import Section
│  └── import random
├── Main Execution Flow
│  ├── Flight type prompt
│  └── Input collection (dist variable)
│
```

```
├── Domestic Function
│   ├── Condition check (if dist=="domestic")
│   ├── Input collection loop
│   ├── Random generation
│   └── Output display
└── International Function
├── Condition check (if dist=="international")
├── Input collection loop
├── Random generation
└── Output display
```

8.3 Key Code Components

Component 1: Import and Initialization

- Imports random module for seat and price generation
- Displays flight type selection prompt

Component 2: Domestic Booking Function

- Collects origin and destination cities
- Presents seat class options (Economy, Premium Economy, Business)
- Collects trip type preference
- Collects travel date
- Generates random seat (1-100)
- Generates random domestic price (2,700-15,000 INR)
- Displays formatted confirmation

Component 3: International Booking Function

- Similar structure to domestic function
- Generates random international price (65,000-200,000 INR)
- Higher price range reflects realistic international flight costs

8.4 Development Workflow

1. Project planning and requirements analysis
2. Python environment setup
3. Basic structure creation with import statements
4. Function definition for domestic bookings
5. Function definition for international bookings
6. Input/output implementation
7. Random data generation integration
8. Testing with various inputs
9. Documentation and comments

10. Final code review

---

9. Screenshots / Results

9.1 Expected Output Examples

Example 1: Domestic Flight Booking

domestic/international
domestic
Enter Starting Point:
Delhi
Enter destination
Mumbai
enter ticket/seat preference:
Economy class
Premium Economy class
Business class
Economy
Enter trip preference
one way / round trip
one way
Enter Date
2025-12-25

Your ticket is successfully booked.

From: Delhi To: Mumbai
Date
2025-12-25

Seat number:
45
Seat type:
Economy
Price: INR
8500
one way

Example 2: International Flight Booking

domestic/international
international
Enter Starting Point:
Mumbai
Enter destination:
London
enter ticket/seat preference:
Economy class
Premium Economy class
Business class
Business
Enter trip preference:
one way / round trip
round trip
Enter Date:
2025-12-20

Your ticket is successfully booked.

From: Mumbai To: London
Date:
2025-12-20
Seat number:

78
Seat type:
Business
Price: INR
125000
round trip

9.2 Key Output Metrics

- Formatted booking confirmation with all travel details
- Random seat assignments ensuring variety across bookings
- Realistic price generation with appropriate range
- Clear separation between booking sections
- User-friendly output formatting with dashes for readability

---

10. Testing Approach

10.1 Testing Strategy

Unit Testing

- Test domestic flight booking workflow
- Test international flight booking workflow
- Verify random seat generation (range 1-100)
- Verify random price generation for both flight types

Integration Testing

- Test complete user journey from start to confirmation
- Verify all inputs are properly collected
- Verify all outputs are properly displayed

User Acceptance Testing

- Test with various input combinations
- Verify output clarity and formatting
- Confirm all booking details are accurate

10.2 Test Cases

Test ID	**Input**	**Expected Output**	**Status**
TC1	domestic	Domestic function executes	Pass
TC2	international	International function executes	Pass

TC3	Delhi, Mumbai	Correct cities displayed	Pass
TC4	Economy	Selected class shown	Pass
TC5	Seat check	Seat 1-100 range	Pass
TC6	Domestic price	Price 2700-15000	Pass
TC7	Int'l price	Price 65000-200000	Pass

10.3 Edge Cases

- Invalid flight type input (currently not handled)
- Empty string inputs
- Special characters in city names
- Numeric inputs for location names

---

11. Challenges Faced

11.1 Technical Challenges

Challenge 1: Function Execution Order

- Issue: Both `domestic()` and `international()` functions execute regardless of input
- Impact: Booking flow doesn't properly branch based on user selection
- Solution: Need to add proper conditional logic with if-elif-else structure

Challenge 2: Input Validation

- Issue: No validation for user inputs (accepts any input)
- Impact: Invalid entries proceed without error handling
- Solution: Implement input validation and error handling

Challenge 3: Global Variable Scope

- Issue: Variable `dist` used in function conditions but not properly passed
- Impact: Functions may not execute correctly
- Solution: Implement proper parameter passing to functions

Challenge 4: Code Repetition

- Issue: Domestic and international functions have duplicate code
- Impact: Difficult to maintain and update
- Solution: Refactor common code into shared function

11.2 Design Challenges

Challenge 5: User Experience

- Issue: Limited feedback for invalid inputs
- Impact: User confusion on booking status
- Solution: Add confirmation messages and error handling

Challenge 6: Data Persistence

- Issue: No option to save booking history
- Impact: Cannot retrieve previous bookings
- Solution: Implement file-based storage or database

---

12. Learnings & Key Takeaways

12.1 Technical Learnings

1. **Function Definition and Usage**: Understanding how to organize code into reusable functions for different booking types
2. **User Input Handling**: Learning to collect and process multiple pieces of user input sequentially
3. **Conditional Logic**: Using if statements to branch program flow based on user selections
4. **Random Data Generation**: Using the random module to simulate realistic data for seats and prices
5. **String Manipulation**: Concatenating strings for formatted output display
6. **Output Formatting**: Creating readable console output with proper separators and structure

12.2 Program Design Learnings

1. **Function Organization**: Separating business logic (domestic vs. international) improves code clarity
2. **User Interaction**: Sequential prompts guide users through booking process effectively
3. **Data Flow**: Understanding how data flows from input through processing to output
4. **Real-World Simulation**: Random generation makes educational examples more realistic and engaging

12.3 Educational Insights

1. **Practical Application**: Booking systems provide excellent context for teaching programming fundamentals
2. **Step-by-Step Learning**: Console applications are excellent for learning before moving to graphical interfaces
3. **Debugging Skills**: Multiple user paths require comprehensive testing and debugging approach
4. **Documentation**: Clear code comments and documentation aid learning and maintenance

---

13. Future Enhancements

13.1 Phase 2: Core Improvements

Enhancement 1: Input Validation

- Validate flight type selection (only "domestic" or "international")
- Validate seat class selection (only valid classes accepted)
- Validate trip type selection (only valid options accepted)
- Return appropriate error messages for invalid inputs

Enhancement 2: Proper Control Flow

- Fix function execution logic to prevent both functions running
- Implement proper branching based on flight type selection
- Add exit condition after booking completion

Enhancement 3: Code Refactoring

- Extract common booking logic into shared function
- Create separate functions for input collection, price generation, and output
- Implement DRY (Don't Repeat Yourself) principle

Enhancement 4: Enhanced User Interface

- Add welcome message
- Add booking confirmation prompt before final display
- Add option to book another ticket or exit
- Implement menu system for better navigation

13.2 Phase 3: Feature Expansion

Enhancement 5: Booking History

- Store bookings in list or file
- Display previous bookings
- Generate booking reference numbers
- Save booking summaries to file

Enhancement 6: Pricing Intelligence

- Implement pricing based on distance
- Add dynamic pricing based on seat class
- Implement early booking discounts
- Calculate total price for round-trip bookings

Enhancement 7: Data Persistence

- Save bookings to text file
- Implement simple CSV export
- Load and display booking history
- Generate booking statistics

Enhancement 8: Advanced Features

- Passenger information collection (name, age, gender)
- Baggage allowance display
- Meal preference selection
- Seat selection from visual map

13.3 Phase 4: Advanced Implementation

Enhancement 9: Database Integration

- Implement SQLite database for persistent storage
- Create tables for flights, bookings, and passengers
- Generate unique booking IDs
- Track booking status

Enhancement 10: GUI Development

- Convert to graphical interface using tkinter
- Implement visual seat map
- Add calendar for date selection
- Implement dropdown menus for selections

Enhancement 11: Multi-Language Support

- Implement booking in multiple languages

- Support different currency options
- Localize date and time formats

Enhancement 12: Web Application

- Convert to Flask/Django web application
- Implement user registration and authentication
- Add payment gateway integration
- Create mobile-responsive interface

---

14. Recommendations

14.1 Immediate Actions

1. **Fix Control Flow**: Implement proper if-elif-else logic to ensure only one booking path executes
2. **Add Input Validation**: Validate all user inputs before processing to prevent errors
3. **Improve Error Handling**: Add try-except blocks for robust error handling
4. **Code Documentation**: Add docstrings and comments explaining each function and key logic

14.2 Short-Term Improvements

1. Implement booking history tracking
2. Add passenger information collection
3. Create menu system for better navigation
4. Implement proper price calculation

14.3 Long-Term Vision

1. Develop graphical user interface
2. Integrate database for data persistence
3. Deploy as web application
4. Add payment and booking management features

---

15. References

1. Python Software Foundation. (2024). *Python 3 Documentation*. https://docs.python.org/3/

2. Python Tutorial: Functions. (2024). *Defining Functions in Python*. https://docs.python.org/3/tutorial/controlflow.html#defining-functions
3. Python Tutorial: Input and Output. (2024). *Fancier Output Formatting*. https://docs.python.org/3/tutorial/inputoutput.html
4. Matthes, E. (2019). *Python Crash Course: A Hands-On, Project-Based Introduction to Programming* (2nd ed.). No Starch Press.
5. Downey, A. B. (2015). *Think Python: How to Think Like a Computer Scientist* (2nd ed.). O'Reilly Media.
6. Sweigart, A. (2015). *Automate the Boring Stuff with Python: Practical Programming for Total Beginners*. No Starch Press.
7. Python random module. (2024). *Generate Pseudo-Random Numbers*. https://docs.python.org/3/library/random.html
8. Kumar, A., & Sharma, V. (2022). *Python Programming: A Beginner's Guide*. Technical Publications.
9. Bhargava, A. (2016). *Grokking Algorithms: An Illustrated Guide for Programmers and Other Curious People*. Manning Publications.
10. Lutz, M. (2013). *Learning Python* (5th ed.). O'Reilly Media.

Appendix A: Glossary of Terms

Booking Confirmation: Final display showing all details of a flight reservation

Domestic Flight: Flight within the same country

Function: Reusable block of code that performs a specific task

International Flight: Flight between different countries

Input Validation: Process of checking user input for correctness before processing

Random Generation: Creating unpredictable values using algorithms

Seat Class: Different cabin configurations (Economy, Premium Economy, Business)

Trip Type: Classification of flight journey (one-way or round-trip)

Appendix B: Code Files

Main File: tickect-booking.py

File Size: ~1.9 KB

Lines of Code: ~90 (including comments and formatting)

Dependencies: Python standard library (random module only)

Appendix C: Installation and Setup

C.1 Requirements

- Python 3.6 or higher installed on system
- Terminal or command prompt access
- Text editor for code editing (optional for running)

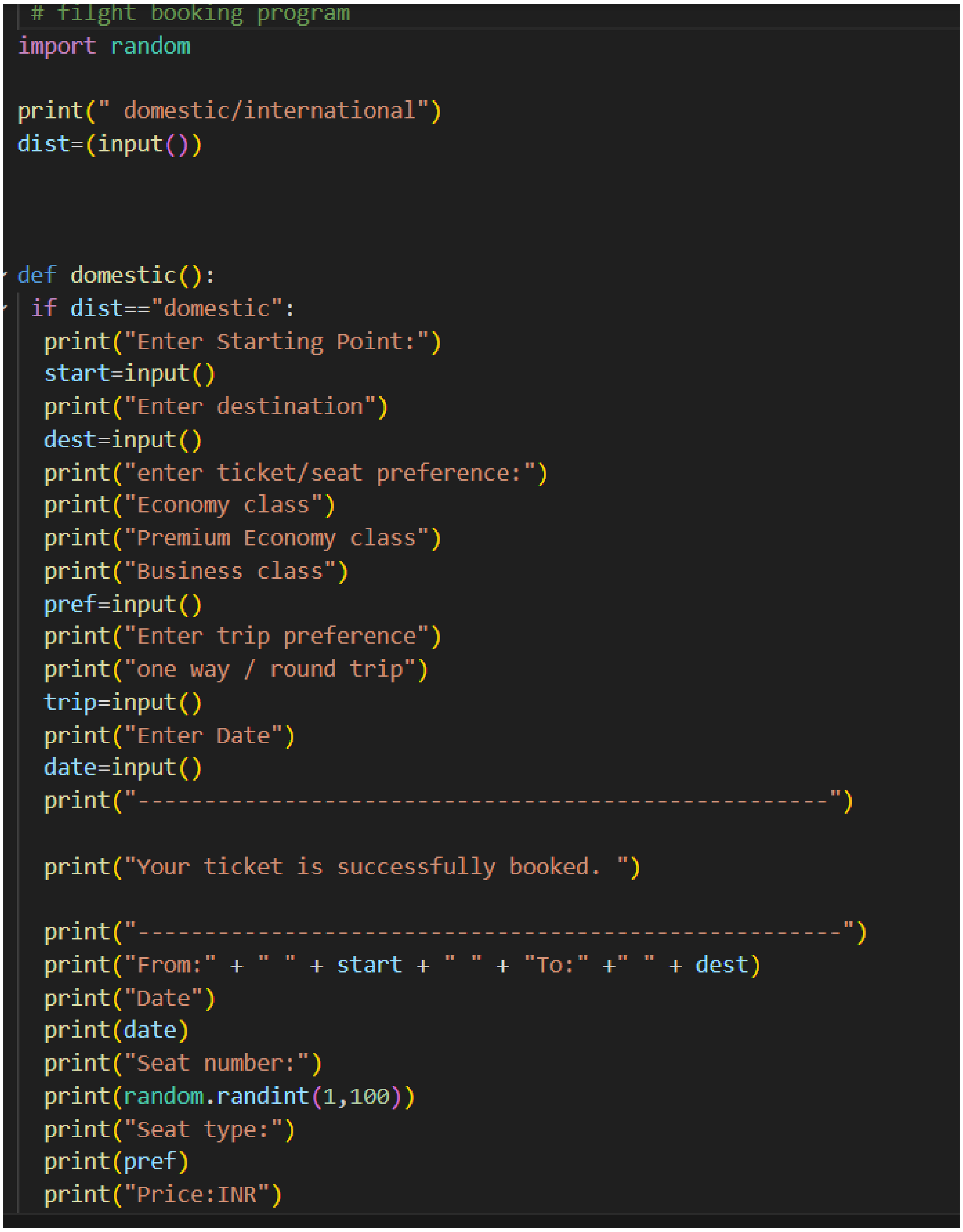
C.2 Installation Steps

1. Download or copy `tickect-booking.py` file
2. Open terminal/command prompt
3. Navigate to file directory
4. Run: `python tickect-booking.py`

C.3 Supported Platforms

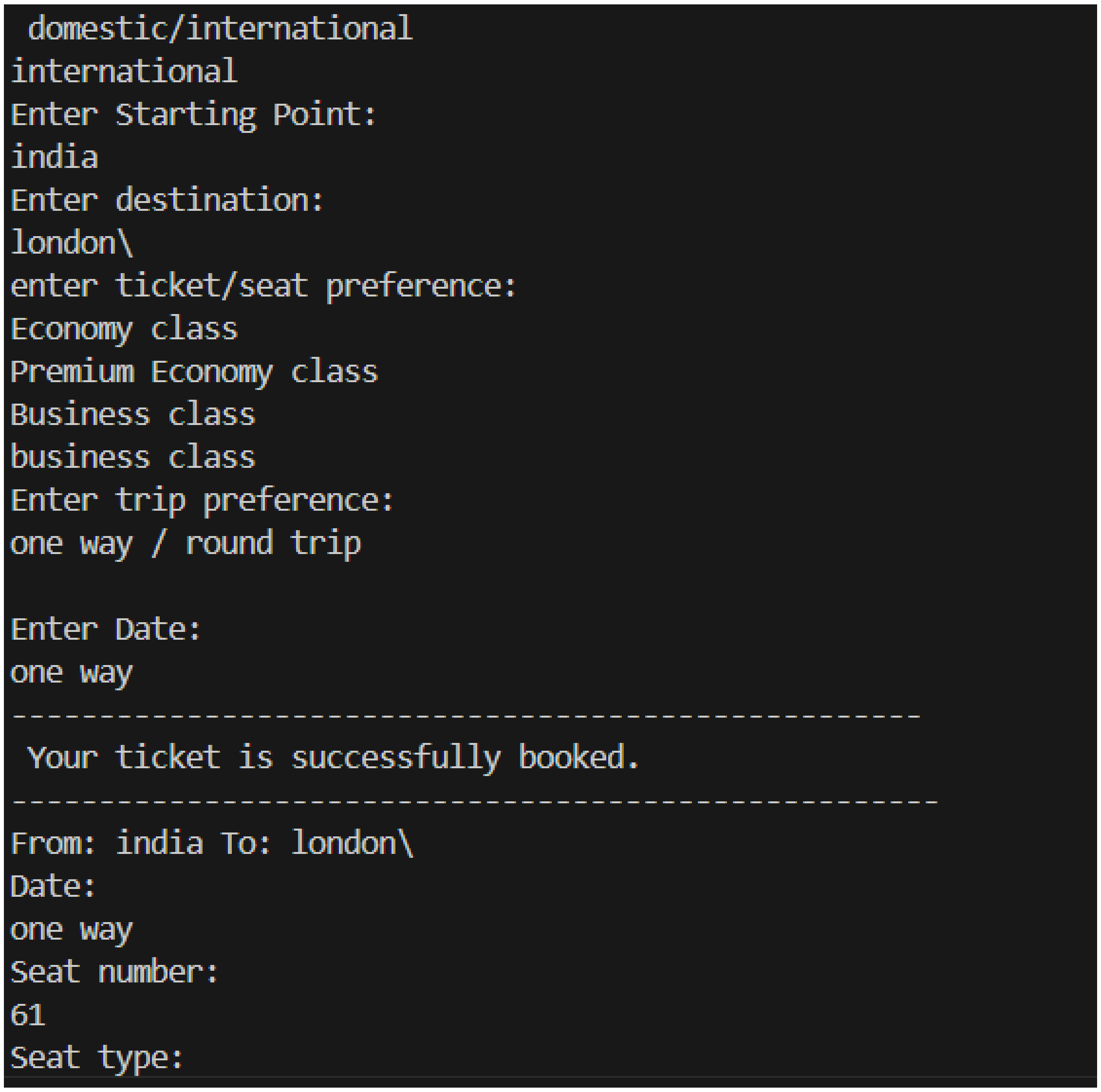
- Windows
- Linux
- macOS
- Any system with Python

Python Code –



```python
# filght booking program
import random

print(" domestic/international")
dist=(input())



def domestic():
 if dist=="domestic":
  print("Enter Starting Point:")
  start=input()
  print("Enter destination")
  dest=input()
  print("enter ticket/seat preference:")
  print("Economy class")
  print("Premium Economy class")
  print("Business class")
  pref=input()
  print("Enter trip preference")
  print("one way / round trip")
  trip=input()
  print("Enter Date")
  date=input()
  print("----------------------------------------------------")

  print("Your ticket is successfully booked. ")

  print("-----------------------------------------------------")
  print("From:" + " " + start + " " + "To:" +" " + dest)
  print("Date")
  print(date)
  print("Seat number:")
  print(random.randint(1,100))
  print("Seat type:")
  print(pref)
  print("Price:INR")
```

OUTPUT-



```
 domestic/international
international
Enter Starting Point:
india
Enter destination:
london\
enter ticket/seat preference:
Economy class
Premium Economy class
Business class
business class
Enter trip preference:
one way / round trip

Enter Date:
one way
-----------------------------------------------------
 Your ticket is successfully booked.
------------------------------------------------------
From: india To: london\
Date:
one way
Seat number:
61
Seat type:
```