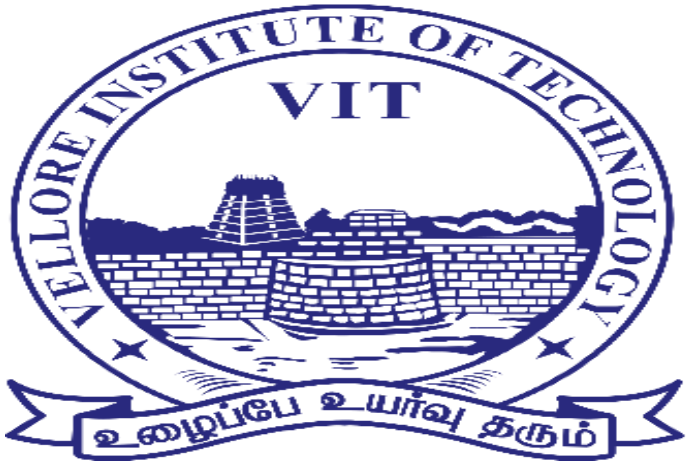


VIT®

BHOPAL

- Project Title: Code Snippet Organiser
- Submitted by: Kirtika Singh
- Roll No.- 25BCE10533

Vityarthi Project

Introduction

- The Code Snippet Organiser is a simple and efficient tool designed to store, manage, search, and delete code snippets. It helps programmers maintain an organised collection of frequently used code fragments, improving productivity and reducing repetition.

Problem Statement

- Developers often struggle with organising useful code snippets. Searching for previous code becomes time-consuming and inefficient. A simple, user-friendly solution is needed to store and retrieve code snippets quickly.

Functional Requirements

- 1. Add a new code snippet
- 2. Search for existing snippets
- 3. Display all snippets
- 4. Delete a snippet
- 5. Data stored in dictionary or file
- 6. Exit option

Non-functional Requirements

- 1. Easy to use interface
- 2. Quick search performance
- 3. Reliable storage of snippets
- 4. Minimal memory usage
- 5. Maintainability and readability of code

System Architecture

- User interacts with the console menu → Program handles input → Dictionary stores snippets → Operations performed (Add, Search, Delete, Display).

SYSTEM WORKFLOW

1. User opens the program and sees the main menu.

2. User selects an option: Add, View, Search, Delete, or Exit.

3. Program takes the input and performs the selected operation.

4. Snippets are read from or written to the storage file.

5. Output is shown to the user.

Design Decisions & Rationale

- Chose dictionary for fast lookup
- Made menu-based interface for simplicity
- Functions used for modularity
- Text-based storage avoided to keep project simple

Implementation Details

- The project is implemented in Python using functions for add, search, delete, and display operations. Snippets are stored in a dictionary with titles as keys.

Program

```
def add_snippet():
    title = input("Enter snippet title: ")
    code = input("Enter the code snippet: ")

    with open("snippets.txt", "a") as file:
        file.write(f"{title}:::{code}\n")
    print("Snippet added successfully!\n")



def view_snippets():
    try:
        with open("snippets.txt", "r") as file:
            snippets = file.readlines()

        if not snippets:
            print("No snippets found!\n")
```

```
        if not snippets:
            print("No snippets found!\n")
            return

        print("--- All Snippets ---")
        for line in snippets:
            title, code = line.strip().split(":::")
            print(f"Title: {title}\nCode: {code}\n")

    except FileNotFoundError:
        print("Storage file not found!\n")



def search_snippet():
    search_title = input("Enter title to search: ")
```

```
def search_snippet():
    search_title = input("Enter title to search: ")

    try:
        with open("snippets.txt", "r") as file:
            snippets = file.readlines()

        found = False
        for line in snippets:
            title, code = line.strip().split(":::")
            if title.lower() == search_title.lower():
                print(f"Snippet found!\nTitle: {title}\nCode:
{code}\n")
                found = True
                break
```

```
                found = True
                break

        if not found:
            print("Snippet not found.\n")

    except FileNotFoundError:
        print("Storage file not found!\n")


def delete_snippet():
    del_title = input("Enter the title to delete: ")

    try:
        with open("snippets.txt", "r") as file:
            snippets = file.readlines()
```

```
updated = []
deleted = False

for line in snippets:
    title, code = line.strip().split(":::")
    if title.lower() != del_title.lower():
        updated.append(line)
    else:
        deleted = True

with open("snippets.txt", "w") as file:
    file.writelines(updated)

if deleted:
    print("Snippet deleted successfully.\n")
else:
```

```
        else:
            print("Snippet not found.\n")

    except FileNotFoundError:
        print("Storage file not found!\n")


def main():
    while True:
        print("\n--- Code Snippet Organiser ---")
        print("1. Add Snippet")
        print("2. View All Snippets")
        print("3. Search Snippet")
        print("4. Delete Snippet")
        print("5. Exit")
```

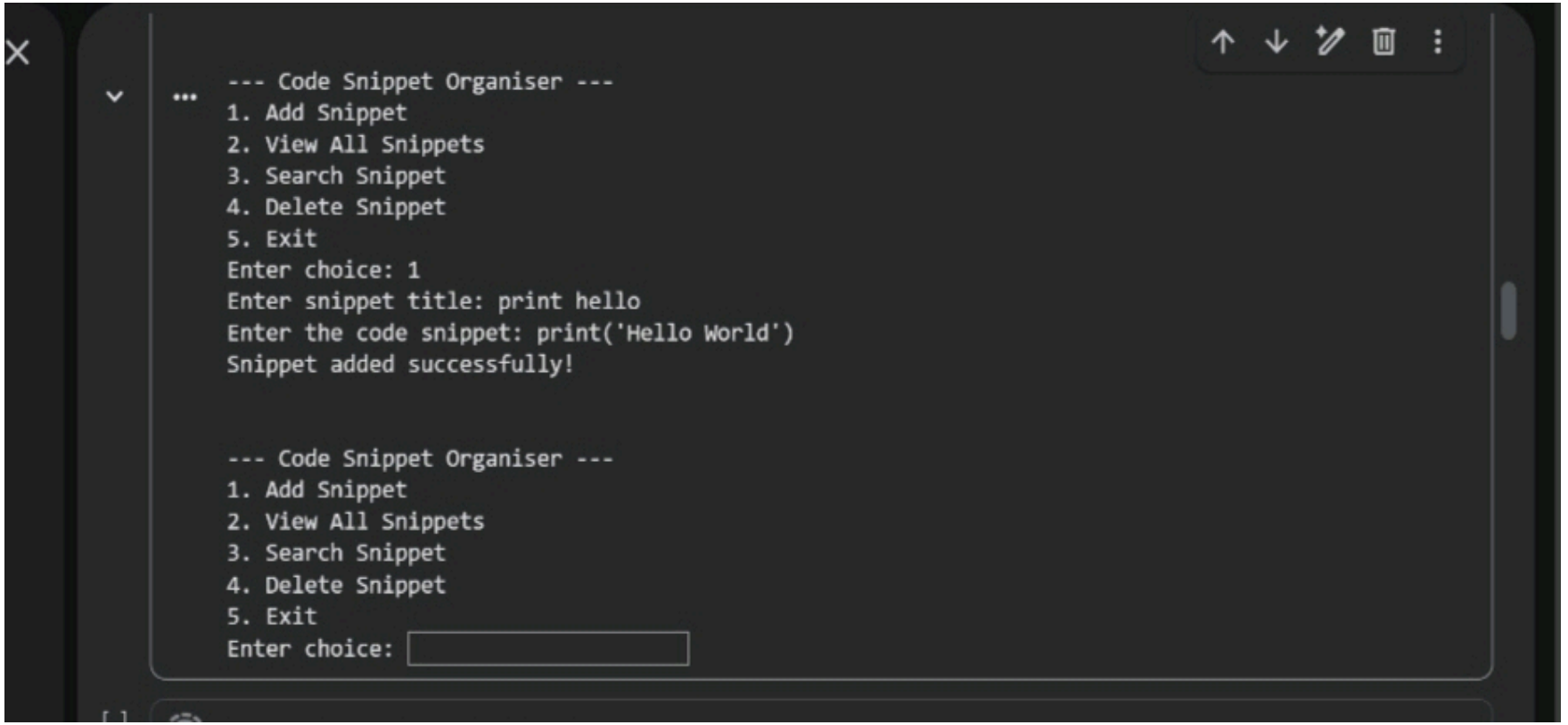
```
choice = input("Enter choice: ")

if choice == '1':
    add_snippet()
elif choice == '2':
    view_snippets()
elif choice == '3':
    search_snippet()
elif choice == '4':
    delete_snippet()
elif choice == '5':
    print("Exiting... Goodbye!")
    break
else:
    print("Invalid choice! Try again.\n")
```

```
        elif choice == '3':
            search_snippet()
        elif choice == '4':
            delete_snippet()
        elif choice == '5':
            print("Exiting... Goodbye!")
            break
        else:
            print("Invalid choice! Try again.\n")


main()
```

Screenshots / Results



APPLICATIONS

Useful for programmers to store frequently used code.

Helps students organise practice programs.

Ideal for small development teams to share common snippets.

Can be used as a base for building larger code-management tools.

Testing Approach

Test Cases:

- 1. Add snippet → Check if stored
- 2. Search snippet → Verify correct output
- 3. Delete snippet → Confirm removal
- 4. Invalid inputs → Handle errors

Challenges Faced

- Handling duplicate snippet names
- Ensuring user-friendly menu
- Maintaining readability in console output

Learnings & Key Takeaways

- Improved understanding of Python data structures
- Learned how to apply modular programming
- Understood project documentation and structure

Future Enhancements

- Add file-based storage
- Add GUI using Tkinter
- Add categories and tags
- Add snippet export/import feature

References

- 1. Python Official Documentation
- 2. Course Materials – Python Essentials
- 3. TutorialsPoint, GeeksforGeeks