(1) CALCULATOR:-

Design a simple calculator with basic arithmetic operations. Prompt the user to input two numbers and an operation choice. Perform the calculation and display.

```
def add(x, y):
    return x + y

def subtract(x, y):
    return x - y

def multiply(x, y):
    return x * y

def divide(x, y):
    if y != 0:
        return x / y
    else:
        return "Cannot divide by zero"

def calculator():
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))

    print("Choose operation:")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")

    choice = input("Enter choice (1/2/3/4): ")

    if choice in ['1', '2', '3', '4']:
        if choice == '1':
            result = add(num1, num2)
        elif choice == '2':
            result = subtract(num1, num2)
        elif choice == '3':
            result = multiply(num1, num2)
        elif choice == '4':
            result = divide(num1, num2)

        print(f"Result: {result}")
    else:
        print("Invalid input. Please choose a valid operation.")

# Run the calculator
calculator()
```

```
Enter the first number: 4
Enter the second number: 5
Choose operation:
1. Addition
2. Subtraction
3. Multiplication
```

```
4. Division
Enter choice (1/2/3/4): 1
Result: 9.0
```

(2) PASSWORD GENERATOR:-

A password generator is a useful tool that generates strong and random passwords for users. This project aims to create a password generator application using Python, allowing users to specify the length and complexity of the password. User Input: Prompt the user to specify the desired length of the password. Generate Password: Use a combination of random characters to generate a password of the specified length. Display the Password: Print the generated password on the screen.

```
import random
import string

def generate_password(length):
    characters = string.ascii_letters + string.digits + string.punctuation
    password = ''.join(random.choice(characters) for _ in range(length))
    return password

def password_generator():
    try:
        length = int(input("Enter the desired length of the password: "))
        if length <= 0:
            print("Invalid length. Please enter a positive number.")
            return

        generated_password = generate_password(length)
        print(f"Generated Password: {generated_password}")
    except ValueError:
        print("Invalid input. Please enter a valid number for the password length.")

# Run the password generator
password_generator()
```

```
Enter the desired length of the password: 3
Generated Password: o;5
```

(3) TO-DO LIST

A To-Do List application is a useful project that helps users manage and organize their tasks efficiently. This project aims to create a command-line or GUI-based application using Python, allowing users to create, update, and track their to-do lists

```
import json
import os
from datetime import datetime

class Task:
```

```
    def __init__(self, name, due_date=None, priority=None):
        self.name = name
        self.due_date = due_date
        self.priority = priority
        self.completed = False

class ToDoList:
    def __init__(self, filename='todolist.json'):
        self.filename = filename
        self.tasks = []
        self.load_tasks()

    def load_tasks(self):
        if os.path.exists(self.filename):
            with open(self.filename, 'r') as file:
                data = json.load(file)
                self.tasks = [Task(**task_data) for task_data in data]

    def save_tasks(self):
        with open(self.filename, 'w') as file:
            data = [task.__dict__ for task in self.tasks]
            json.dump(data, file, indent=2)

    def add_task(self, name, due_date=None, priority=None):
        new_task = Task(name, due_date, priority)
        self.tasks.append(new_task)
        self.save_tasks()
        print(f'Task "{name}" added.')

    def view_tasks(self):
        for i, task in enumerate(self.tasks, start=1):
            status = 'Completed' if task.completed else 'Pending'
            print(f'{i}. {task.name} - Due: {task.due_date} - Priority: {task.priority} -

    def complete_task(self, task_index):
        if 1 <= task_index <= len(self.tasks):
            self.tasks[task_index - 1].completed = True
            self.save_tasks()
            print(f'Task "{self.tasks[task_index - 1].name}" marked as completed.')
        else:
            print('Invalid task index.')

# Example Usage
if __name__ == "__main__":
    todo_list = ToDoList()

    while True:
        print("\n1. Add Task\n2. View Tasks\n3. Complete Task\n4. Exit")
        choice = input("Enter your choice (1/2/3/4): ")

        if choice == '1':
            name = input("Enter task name: ")
            due_date = input("Enter due date (optional, format: YYYY-MM-DD): ")
            priority = input("Enter priority (optional): ")
            todo_list.add_task(name, due_date, priority)

        elif choice == '2':
            todo_list.view_tasks()

        elif choice == '3':
```

```
        task_index = int(input("Enter the index of the task to complete: "))
        todo_list.complete_task(task_index)

    elif choice == '4':
        break

    else:
        print("Invalid choice. Please enter a valid option.")


1. Add Task
2. View Tasks
3. Complete Task
4. Exit
Enter your choice (1/2/3/4): 2

1. Add Task
2. View Tasks
3. Complete Task
4. Exit
Enter your choice (1/2/3/4): 1
Enter task name: Manish
Enter due date (optional, format: YYYY-MM-DD): 2024-01-20
Enter priority (optional): 1
Task "Manish" added.

1. Add Task
2. View Tasks
3. Complete Task
4. Exit
Enter your choice (1/2/3/4): 4
```