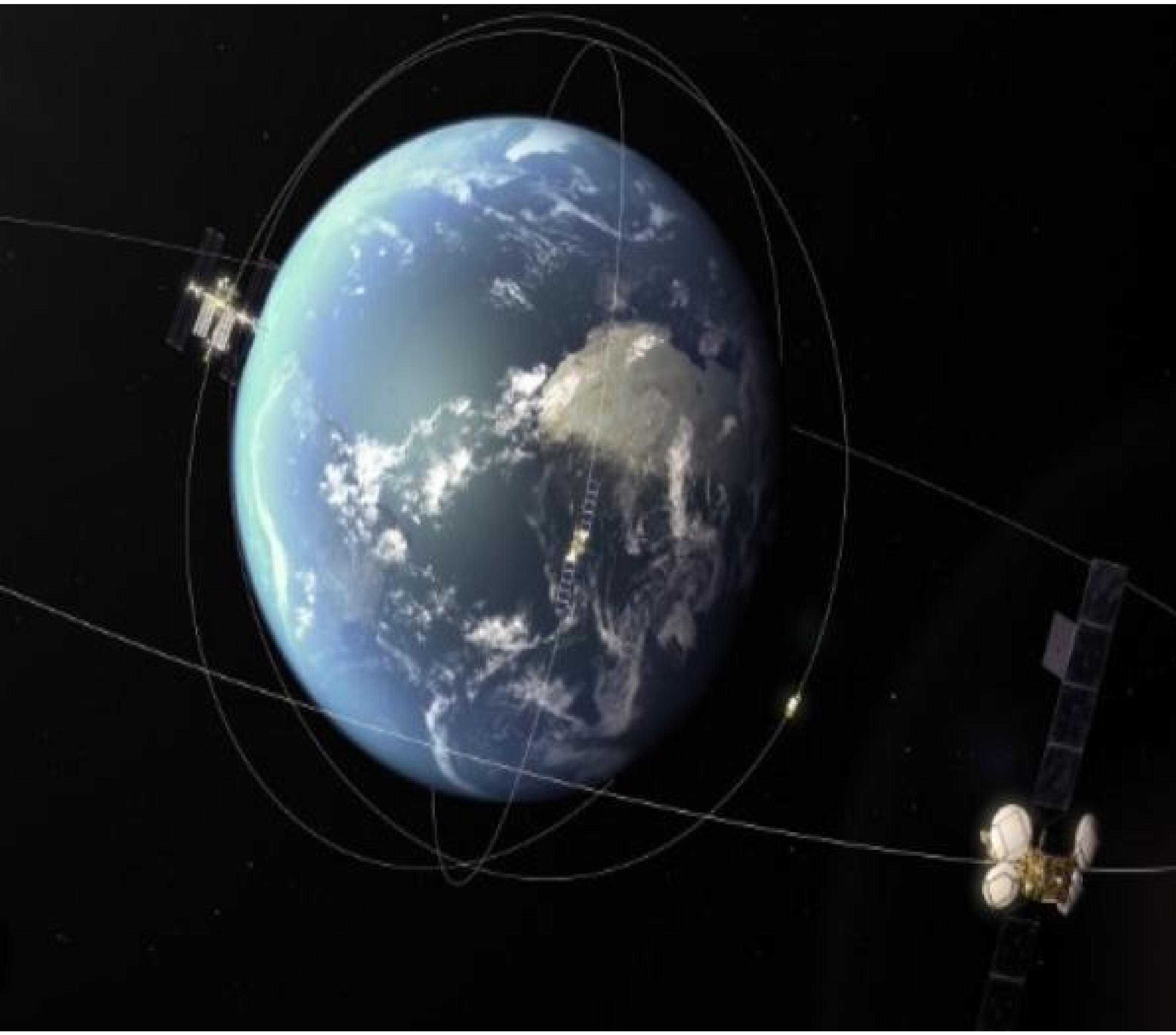
Orbital Parameters Calculator for Satellites



SHE ORBITS
BY: Shireen_ Fathy
25/11/2025

1. Introduction

Earth orbiting satellites play an important role in communication, navigation, scientific observation, and defense. To design, operate, or analyze satellite missions, engineers must calculate key orbital parameters such as orbital speed, period, energy, and orbital size.

This project implements a **Python-based orbital calculator** that computes the essential parameters of a circular Low Earth Orbit (LEO) based on the satellite's altitude above Earth's surface.

The program introduces the student to basic orbital mechanics, Python programming structures, and object-oriented design through:

- Classes & objects
- Mathematical functions
- A dynamic menu using loops
- Lists for storing satellites
- Conditional logic (if/else)

It serves as a simplified but practical model of real orbital analysis.

2. Project Objectives

The goals of this project are:

1. Develop a Python console application capable of storing multiple satellites.
2. Compute orbital parameters automatically using classical orbital mechanics.
3. Implement OOP principles through a '***Satellite***' class.
4. Use a data structure (list) to store, display, and delete satellites.
5. Provide a user-friendly text-based interface for interaction.
6. Apply gravitational constant and Earth radius to compute:
    - Semi-major axis
    - Orbital velocity
    - Orbital period
    - Orbital circumference
    - Orbital energy

The program allows the user to build a small ***"satellite database"*** and compute orbital parameters in real time.

3. Orbital Mechanics Background

A satellite in a circular orbit is governed by Kepler’s laws.
The gravitational force provides the centripetal force:

$$v = \sqrt{\frac{\mu}{a}}$$

Where:

- Earth’s gravitational parameter:

$$\mu = 398600 km3/s2$$

- Semi-major axis:

$$a = R_{Earth} \quad + h$$

- Altitude above Earth : h
- Orbital period is:

$$T = \frac{2\pi a}{v}$$

- Specific orbital energy:

$$E = -\frac{\mu}{2a}$$

These simplified equations allow accurate calculations for circular orbits.

4. Code Structure Overview

The project consists of:

4.1 *‘Satellite Class’*

This class represents each satellite and automatically calculates its orbit at creation.

Attributes stored:

Parameter	**Meaning**
name	Satellite name
altitude	Orbit altitude above Earth (km)
a	Semi-major axis
e	Eccentricity (0 for circular)
v	Orbital velocity
T	Orbital period
C	Orbital circumference
E	Specific orbital energy

The class uses the gravitational constant and Earth radius as fixed class-level constants:

```
import math
class Satellite:
    """
    Represents a satellite with name and altitude.
    Calculates orbital parameters automatically.
    """
    MU = 398600  # km^3/s^2, standard gravitational parameter
    EARTH_RADIUS = 6371  # km
```

4.2 Methods

4.2.1 *__init__()*

Runs automatically when a satellite is created.
Performs all orbital calculations.

```
    def __init__(self, name, altitude):
        self.name = name
        self.altitude = altitude  # km
        self.a = self.EARTH_RADIUS + altitude  # semi-major axis
        self.e = 0.0  # assume circular orbit
        self.v = math.sqrt(self.MU / self.a)  # km/s
        self.T = 2 * math.pi * self.a / self.v  # orbital period in seconds
        self.C = 2 * math.pi * self.a  # orbital circumference in km
        self.E = -self.MU / (2 * self.a)  # orbital energy (km^2/s^2)
```

4.2.2 *info()*

Returns a formatted text summary of all computed orbital parameters.

```
    def info(self):
        return (f"Satellite: {self.name}\n"
                f"Altitude: {self.altitude:.2f} km\n"
                f"Semi-major axis (a): {self.a:.2f} km\n"
                f"Eccentricity (e): {self.e}\n"
                f"Orbital speed (v): {self.v:.2f} km/s\n"
                f"Orbital period (T): {self.T/60:.2f} min\n"
                f"Orbital circumference (C): {self.C:.2f} km\n"
                f"Orbital energy (E): {self.E:.2f} km^2/s^2\n")
```

5. Data Structure

All satellite objects are stored in a simple dynamic list:
This structure allows:

- Easy append
- Index-based access
- Intuitive deletion
- Fast iteration

```
# --------------- Data Structure --------------- #
satellites = []
```

6. Supporting Functions

The program includes several user-interface functions that operate on the ***'satellites'*** list.

6.1 *'add_satellite()'*

- Asks user for name & altitude
- Creates a "***Satellite"*** object
- Appends it to the list
- Prints success message

```
# Function.1: Add Sat
def add_satellite():
    print("\n--- Add New Satellite ---")
    name = input("Enter satellite name: ")
    altitude = float(input("Enter altitude above Earth surface (km): "))
    sat = Satellite(name, altitude)
    satellites.append(sat)
    print(f"Satellite {name} added successfully!\n")
```

6.2 *'show_satellites()'*

- Prints a numbered list of all satellites
- Useful before selecting or deleting

```
# Function.2: Show Sat
def show_satellites():
    print("\n--- Satellite List ---")
    if not satellites:   #### OR if len(satellites)==0:
        print("No satellites saved yet.\n")
        return
    for i, sat in enumerate(satellites, 1):
        print(f"{i}. {sat.name} - Altitude: {sat.altitude} km")
    print()
```

6.3 *'delete_satellite()'*

- Displays list
- Allows user to remove a satellite by number
- Handles invalid input

```
# Function.3: Delete SAT
def delete_satellite():
    print("\n--- Delete a Satellite ---")
    if not satellites:
        print("No satellites to delete.\n")
        return
    show_satellites()
    try:
        choice = int(input("Enter satellite number to delete: "))
        if 1 <= choice <= len(satellites):
            removed = satellites.pop(choice - 1)
            print(f"Satellite {removed.name} deleted successfully!\n")
        else:
            print("Invalid number.\n")
    except ValueError:
        print("Please enter a valid number.\n")
```

6.4 *‘show_orbital_parameters()’*

- Shows detailed orbital data for a selected satellite
- Uses the ***“.info()”*** method of the class

```
# Function.4: Show- orbit- parameter
def show_orbital_parameters():
    print("\n--- Orbital Parameters ---")
    if not satellites:
        print("Add satellites first.\n")
       return
```

7. Program Flow (Menu System)

The program uses a ***“while True”*** loop to keep showing a menu:

1. Add Satellite
2. Show Satellite List
3. Delete Satellite
4. Show Orbital
5. Exit

A matching ***“if/elif”*** block calls the appropriate function.
This design ensures:

- Simple user interaction
- Infinite loop until exit
- Error handling for invalid choices

8. Python Code script

```
# Satellite Orbital Calculator
import math
class Satellite:
    """
    Represents a satellite with name and altitude.
    Calculates orbital parameters automatically.
    """
    MU = 398600  # km^3/s^2, standard gravitational parameter
    EARTH_RADIUS = 6371  # km
    def __init__(self, name, altitude):
        self.name = name
        self.altitude = altitude  # km
        self.a = self.EARTH_RADIUS + altitude  # semi-major axis
```

```python
        self.e = 0.0  #  circular orbit
        self.v = math.sqrt(self.MU / self.a)  # km/s
        self.T = 2 * math.pi * self.a / self.v  # orbital period in seconds
        self.C = 2 * math.pi * self.a  # orbital circumference in km
        self.E = -self.MU / (2 * self.a)  # orbital energy (km^2/s^2)
    def info(self):
        return (f"Satellite: {self.name}\n"
                f"Altitude: {self.altitude:.2f} km\n"
                f"Semi-major axis (a): {self.a:.2f} km\n"
                f"Eccentricity (e): {self.e}\n"
                f"Orbital speed (v): {self.v:.2f} km/s\n"
                f"Orbital period (T): {self.T/60:.2f} min\n"
                f"Orbital circumference (C): {self.C:.2f} km\n"
                f"Orbital energy (E): {self.E:.2f} km^2/s^2\n")
# ---------------- Data Structure ---------------- #
satellites = []
# ---- Functions ------ #
# Function.1: Add Sat
def add_satellite():
    print("\n--- Add New Satellite ---")
    name = input("Enter satellite name: ")
    altitude = float(input("Enter altitude above Earth surface (km): "))
    sat = Satellite(name, altitude)
    satellites.append(sat)
    print(f"Satellite {name} added successfully!\n")
# Function.2: Show Sat
def show_satellites():
    print("\n--- Satellite List ---")
    if not satellites:   # OR if len(satellites)==0:
        print("No satellites saved yet.\n")
        return
    for i, sat in enumerate(satellites, 1):
        print(f"{i}. {sat.name} - Altitude: {sat.altitude} km")
    print()
# Function.3: Delete SAT
def delete_satellite():
    print("\n--- Delete a Satellite ---")
    if not satellites:
        print("No satellites to delete.\n")
        return
    show_satellites()
    try:
        choice = int(input("Enter satellite number to delete: "))
        if 1 <= choice <= len(satellites):
            removed = satellites.pop(choice - 1)
            print(f"Satellite {removed.name} deleted successfully!\n")
        else:
            print("Invalid number.\n")
    except ValueError:
        print("Please enter a valid number.\n")
# Function.4: Show- orbit- parameter
def show_orbital_parameters():
    print("\n--- Orbital Parameters ---")
    if not satellites:
        print("Add satellites first.\n")
        return
    show_satellites()
    try:
        choice = int(input("Select satellite number to calculate parameters: "))
        if 1 <= choice <= len(satellites):
            sat = satellites[choice - 1]
            print("\n" + sat.info())
        else:
            print("Invalid selection.\n")
    except ValueError:
        print("Please enter a valid number.\n")
# ---------------- Main Menu ---------------- #
def main_menu():
    while True:
        print("============================")
        print(" Satellite Orbital Calculator")
        print("============================")
        print("1. Add Satellite")
        print("2. Show All Satellites")
        print("3. Delete Satellite")
        print("4. Show Orbital Parameters")
        print("5. Exit")
      choice = input("Enter choice: ")
        if choice == "1":
            add_satellite()
        elif choice == "2":
            show_satellites()
        elif choice == "3":
            delete_satellite()
        elif choice == "4":
            show_orbital_parameters()
        elif choice == "5":
            print("Exiting program...")
            break
        else:
            print("Invalid choice, try again.\n")
if __name__ == "__main__":
    main_menu()
```

9. Result

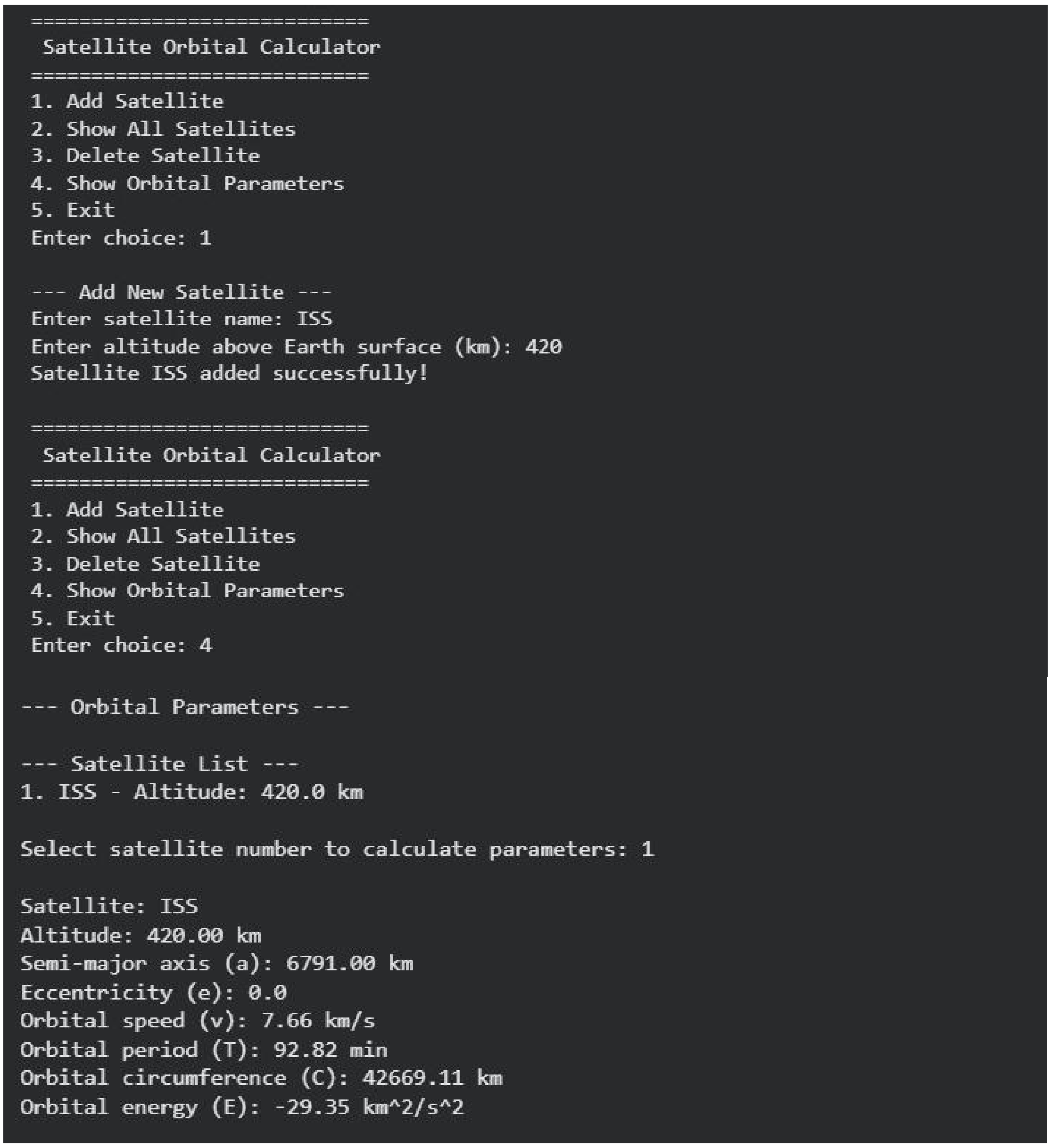
9.1 Adding a satellite

```
Enter satellite name: ISS
Enter altitude above Earth surface (km): 420
                                        Satellite ISS added successfully!
```

9.2 Viewing orbital parameters

```
Satellite: ISS
Altitude: 420.00 km
Semi-major axis (a): 6791.00 km
Eccentricity (e): 0.0
```

Orbital speed (v): 7.67 km/s
Orbital period (T): 92.85 min
Orbital circumference (C): 42658.29 km
Orbital energy (E): -29.35 km^2/s^2



```
============================
 Satellite Orbital Calculator
============================
1. Add Satellite
2. Show All Satellites
3. Delete Satellite
4. Show Orbital Parameters
5. Exit
Enter choice: 1

--- Add New Satellite ---
Enter satellite name: ISS
Enter altitude above Earth surface (km): 420
Satellite ISS added successfully!

============================
 Satellite Orbital Calculator
============================
1. Add Satellite
2. Show All Satellites
3. Delete Satellite
4. Show Orbital Parameters
5. Exit
Enter choice: 4
```

```
--- Orbital Parameters ---

--- Satellite List ---
1. ISS - Altitude: 420.0 km

Select satellite number to calculate parameters: 1

Satellite: ISS
Altitude: 420.00 km
Semi-major axis (a): 6791.00 km
Eccentricity (e): 0.0
Orbital speed (v): 7.66 km/s
Orbital period (T): 92.82 min
Orbital circumference (C): 42669.11 km
Orbital energy (E): -29.35 km^2/s^2
```