# Minor Project: Family Tree Management System

In this project, students will develop a simple Family Tree Management System using Python. The project will involve the use of functions, classes, and handling data for multiple generations. This exercise will help students practice Object-Oriented Programming (OOP) concepts and data handling.

## Objectives

1. To practice Python programming using functions and classes.

2. To understand and apply Object-Oriented Programming concepts in real-life data.

3. To develop a simple project within 1 hour using basic Python constructs.

## Dataset

The dataset consists of information about a family tree for 5 generations. Each person will have attributes like name, birth year, and relationships (parent, children). Below is the sample dataset for implementation:

Sample Dataset:  
 Generation 1: John (born 1940), Mary (born 1945)  
 Generation 2: Robert (born 1965), Linda (born 1970)  
 Generation 3: David (born 1990), Sarah (born 1992)  
 Generation 4: Kevin (born 2015), Emma (born 2018)  
 Generation 5: Liam (born 2040)

## Project Requirements

1. Create a Python class `Person` with attributes: name, birth\_year, parents, and children.

2. Implement functions to add family members and display relationships.

3. Display the family tree (text-based) up to 5 generations.

4. Show parent-child relationships clearly.

5. Make the program interactive by allowing users to add new members.

## Example Code Structure

class Person:  
 def \_\_init\_\_(self, name, birth\_year):  
 self.name = name  
 self.birth\_year = birth\_year  
 self.parents = []  
 self.children = []  
  
 def add\_child(self, child):  
 self.children.append(child)  
 child.parents.append(self)  
  
def display\_family\_tree(person, generation=0):  
 print(" " \* generation \* 4 + f"{person.name} ({person.birth\_year})")  
 for child in person.children:  
 display\_family\_tree(child, generation + 1)  
  
# Sample usage  
john = Person("John", 1940)  
mary = Person("Mary", 1945)  
robert = Person("Robert", 1965)  
linda = Person("Linda", 1970)  
  
john.add\_child(robert)  
mary.add\_child(robert)  
robert.add\_child(Person("David", 1990))  
robert.add\_child(Person("Sarah", 1992))  
  
display\_family\_tree(john)

## Expected Output

Family Tree (Example Output):  
John (1940)  
 Robert (1965)  
 David (1990)  
 Sarah (1992)  
Mary (1945)  
 Robert (1965)  
 David (1990)  
 Sarah (1992)