**ODE Solver in Python**

Implementation of object-oriented approach and differential algorithms to solve ordinary differential equations in Python

**Introduction**

ODEs have remarkable applications and it has the ability to predict the world around us. They are used in a variety of disciplines like engineering, economics, physics, chemistry and biology. ODEs help to predict the exponential growth and decay, population and species growth. Some of the uses of ODEs are:

* Modelling the growth of diseases
* Describes the movement of electricity
* Describes the motion of the pendulum, waves
* Used in Newton’s second law of motion and Law of cooling

These equations are often solved analytically using the rules of differentiation and integration. But there are no any general-purpose computer algorithms to find the analytical solution of ODEs. Computers solve the ODEs algebraically to find their approximate solutions. I have used several commercial software that solve ODEs and I was familiar with how they work. It was a great experience to implement different techniques and concepts taught in the bootcamp to actually try and build my own ODE solving application.

**Design and Implementation**

In this project, I have tried to build an application that can solve the ODEs. User can type the input parameters and differential function, set the time steps for which the solution is required and run the program to solve that equation.

ODElib library/package has been created, which contains the main algorithm to solve any given function. In the package, there is a module called ODESolver, which has the superclass ODESolver, that takes the user entered function as input parameter, sets up the initial conditions and creates environment necessary to march the solution.

In the same module, the subclass ForwardEuler of the superclass ODESolver is defined, that has a method called advance(), which tells the solve() method in superclass how to proceed the solution. This basically means, the Forward Euler solution algorithm has been implemented in the ForwardEuler subclass, and it will tell the solve() function to solve an ODE by Forward Euler method. More subclasses can be defined to implement different numerical methods.

ODESolver module is imported in the solver file and when input parameters are passed and the program is run, the program will output how the function changes as per the time.

**Conclusion**

ODESolver application was designed using the concepts taught in the Nucamp’s Python Fundamental course. Some of the main concepts or paradigms used in this project are as follows:

* Loops and conditional statements
* Data structures
* Building packages, libraries and modules
* Object oriented programming
* Algorithms
* Efficient programming practices

The application works satisfactorily and has good scope to continue working on it.

**Demonstration**

Step 1: Input the an ODE function

ODE functions are of the type u’ = f(u, t)

u’ – derivative of u

f(u, t) – function of u and t

For example, if u = cos(t), differentiating u we get u’ = -sin(t)

Input function = -sin(t)

Step 2: Input initial condition

Initial condition is the value of function at certain initial point

If you set it as 1, you will observe that the graph will start from 1

Step 3: Input time for which you want to solve the equation (T) and time step size for which you want to calculate the result (dt)

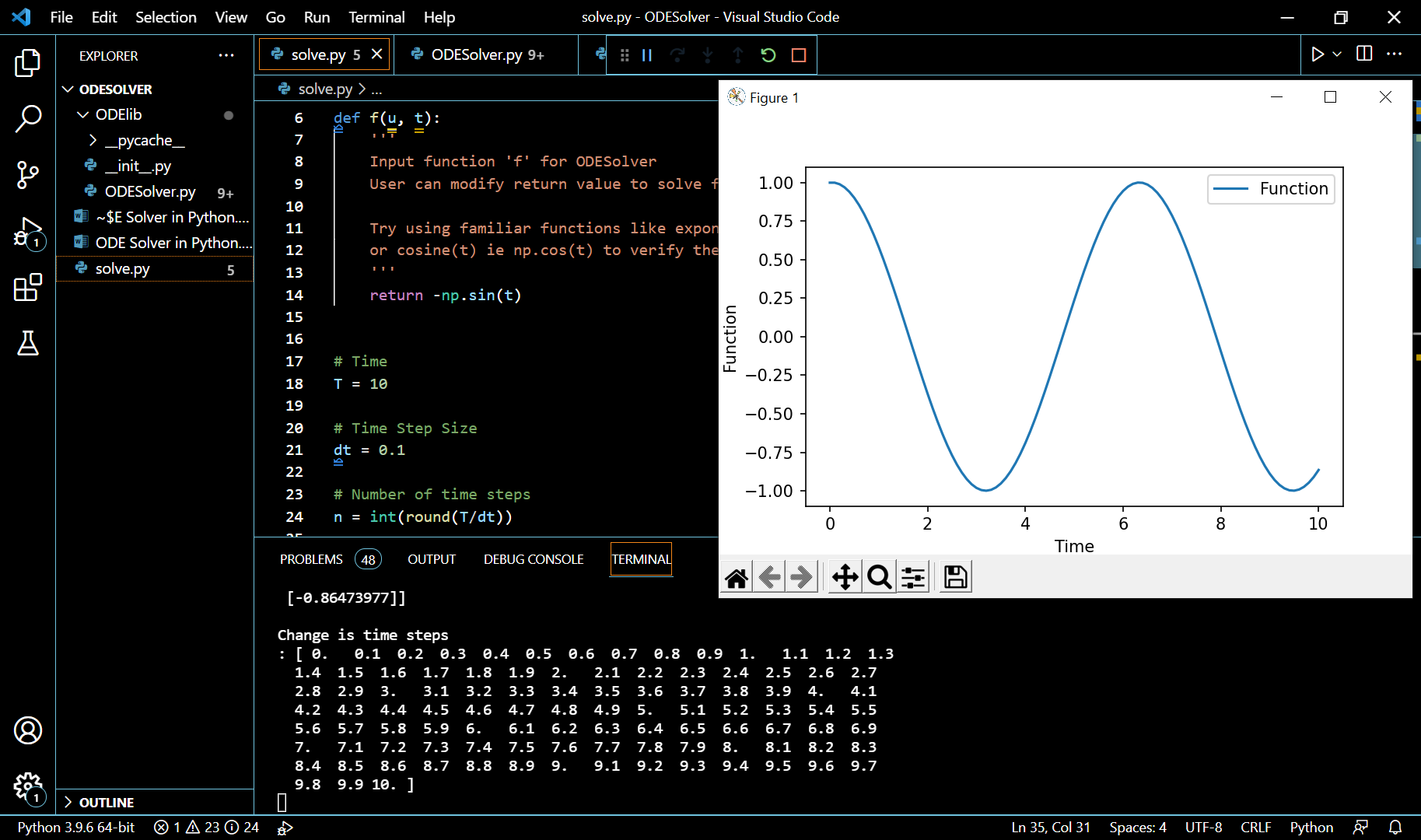
Here the rate of change of function u, over the time period T will be calculated after the interval of time steps.

For example, if the T = 10 and dt = 2, the program will calculate how the value of given function changes from 0 to 2, 2 to 4, 4 to 6, 6 to 8 and 8 to 10.

We know that the differentiation of cos(t) would give us -sin(t). So, when -sin(t) is passed as input function, our solution would be cos(t).

The application produces a graph of cos(t) as shown in Figure 1.

Hence it can be seen that the application works successfully.



d

c

b

a

Figure 1: File Structure (a), Input Parameters (b), Terminal Output (c) and Graph of Solution (d)