Solar System

USING

PYTHON WITH GUI

LIVE Wire

**FOR LIVE CAREERS**

**J.JAYABHARATHI**

**Python programming**

**Guided by**

**Syed khaja**

**ABSTRACT**

In this Python Project, we are going to visualize a Solar System. We will see how the planets revolve around the sun. Also, we will be using different colors to show different planets and assign them with speed and paths to make our project look real. We are going to use the Math Module and Turtle Module to create objects in the solar system and depict their movement.

**Project Prerequisites**

In this project, we will require basic knowledge of python and its concepts. In this project we will be using two modules:

* Math Module
* Turtle Module

**SYSTEM CONFIQURATION**

**HARDWARE REQUIRMENTS:**

Processor : AMD Ryzen 3

RAM : 8 GB

Hard Disk : 256GB

**SOFTWARE REQUIRMENTS:**

OS : WINDOWS 11

Front-End : Python tkinter, collections

IDE : PYTHON ROLE

**OVERVIEW OF THE SOFTWARE**

**PYTHON IDLE** - IDLE stands for Intergrated Development and Learning Environment The story behind the name IDLE is similar to Python. Guido Van Rossum named Python after the British comedy group Monty Python while the name IDLE was chosen to pay tribute to Eric Idle, who was one of the Monty Python's founding members. IDLE comes bundled with the default implementation of the Python language since the 01.5.2bl release. It is packaged as an optional part of the Python packaging with many Linux, Windows, and Mac distributions.

**PYTHON IDLE:**

IDLE, as shown above, is a very simple and sophisticated IDLE developed primarily for beginners, and because of its simplicity, it is highly considered and recommended for educational purposes. It offers a variety of features that you will look in detail along with examples later in this tutorial.

**Collection - The collection Module in Python provides different types of containers.**

A Container is object that is used to store different objects and provide a way to access the contained and iterate over them. Some of the built-in containers are Tuple. List, Dictionary, etc.

**Tkinter- Python offers multiple options for developing GUI(Graphical User Interface)**

Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the TK GUI toolkit.

As with most other modern Tk bindings, Tkinter is implemented as a Python wrapper around a commands, which are fed to this embedded interpreter, thus making it possible to mix Python complete Tel interpreter embedded in the Python interpreter. Tkinter calls are translated into Tel and Tel in a single application.

**Source Code:**

import turtle

import time

from math import \*

screen = turtle.Screen()

screen.tracer(50)

sun = turtle.Turtle()

sun.shape('circle')

sun.color('yellow')

class Planet(turtle.Turtle):

def \_\_init\_\_(self,name,radius, color):

super().\_\_init\_\_(shape='circle')

self.name = name

self.radius = radius

self.c = color

self.color(self.c)

self.up()

self.pd()

self.angle = 0

def move(self):

x = self.radius\*cos(self.angle)

y = self.radius\*sin(self.angle)

self.goto(sun.xcor()+x,sun.ycor()+y)

mercury = Planet("Mercury",40, 'grey')

venus = Planet("Venus",80, 'orange')

earth=Planet("Earth",100,'blue')

mars = Planet("Mars",150, 'red')

jupiter=Planet("Jupiter",180, 'brown')

saturn=Planet("Saturn",230, 'pink')

uranus=Planet("Uranus",250, 'light blue')

neptune=Planet("Neptune",280, 'black')

myList = [ mercury, venus,earth, mars,jupiter,saturn,uranus,neptune]

while True:

screen.update()

for i in myList:

i.move()

mercury.angle += 0.010

venus.angle += 0.009

earth.angle += 0.008

mars.angle += 0.007

jupiter.angle += 0.006

saturn.angle += 0.005

uranus.angle += 0.004

neptune.angle += 0.003

**Output :**

