CS 521 PYTHON PROJECT DOCUMENTATION

AKASH VIGNYAN KENDRA

-Shruti Krishna Pagadala

The python Project I have worked on is called “Akash Vignyan Kendra” which means Space Knowledge Centre. It has knowledge about space and rockets and when the user gives the required data of the rocket, it gives the output of whether the rocket with its dimensions escapes the planets gravitational force or not. The Python program defines a **Rocket** class with methods to calculate escape velocity and determine whether a rocket can escape a given planet. It also defines a set of rockets and prints their details, as well as a list of planets in our solar system.

**Escape velocity** is the minimum velocity required for an object to break free from the gravitational pull of a planet and enter space. It is an important concept in space exploration, as rockets must achieve escape velocity to leave the Earth's atmosphere and enter orbit or travel to other celestial bodies.

The exact amount of energy required to escape the gravitational pull of a mass-bearing object is known as the escape velocity. Since everything has mass, everything has a gravitational pull that can be measured. Consider a deep well (referred to as an energy well by physicists) as a good analogy for escape velocity. You need enough energy to climb out of the well if you are at the bottom and wish to get out (escape). You will need to use more energy to reach the top of a well that is deeper than you think. If you just have the strength to ascend halfway, you will inevitably descend to the bottom. The escape velocity is a method of calculating the exact amount of energy required to reach the well's lip while having no energy left over to walk away.

The formula for escape velocity depends on the mass and radius of the planet, as well as the distance from the center of the planet at which the object starts.

A picture containing font, white, text, graphics

Description automatically generated

The calculation of escape velocity is based on the principle of conservation of energy, which states that the total energy of an isolated system remains constant. In the case of an Earth-rocket system, we assume that the system is isolated, and that the sum of the initial kinetic and potential energy of the rocket is equal to the final kinetic and potential energy when it reaches escape velocity.

A picture containing font, typography, calligraphy, line

Description automatically generated

To derive the formula for escape velocity, we equate the initial and final energies of the system and solve for the velocity needed for the rocket to escape the gravitational pull of the Earth. This formula shows that the escape velocity is directly proportional to the mass and radius of the planet and inversely proportional to the distance from the center of the planet at which the object starts.

For example- the escape velocity calculation for earth would be

A picture containing text, font, white, line

Description automatically generated

i.e., a rocket will for sure need to have a speed greater than 11.2 km/s in-order to escape the earths gravitational force.

Therefore, calculating escape velocity requires an understanding of the principles of conservation of energy, mass and radius of the planet, and the starting distance from the center of the planet. Once escape velocity is achieved, the rocket or object will eventually stop at an infinite distance from the planet.

My program has used the following functions, classes and methods to achieve the required output.

* **Math:**

The Math module is used to access the mathematical functions defined in the module, such as square root.

* **Rocket:**

This is a class named Rocket which represents a rocket. The Rocket class has three attributes: height, weight, and width, which are initialized in its constructor \_\_init\_\_(). The \_\_repr\_\_() method provides a string representation of a Rocket object.

It has the following methods:

* **Constructor Method**

The \_\_init\_\_ method is a constructor method that initializes the Rocket class with three parameters: height, weight, and width. These parameters are assigned to instance variables with the same names.

* **Representation Method**

The \_\_repr\_\_ method is used to return a string representation of the object. In this case, it returns a formatted string containing the height, weight, and width values of the Rocket instance.

* **Calculate\_escape\_velocity Method-**

The calculate\_escape\_velocity method takes in two parameters, planet\_mass and planet\_radius, and returns the escape velocity of the rocket from the planet. It calculates the escape velocity using the gravitational constant, G, and the formula for escape velocity.

* The calculate\_escape\_velocity() method takes the planet's mass and radius as input parameters, and returns the escape velocity for a rocket escaping from the planet using the formula derived from the law of universal gravitation. The method can\_escape\_planet() takes a planet name as input parameter, and determines whether the given rocket can escape the planet based on the escape velocity required for the given planet, the rocket's volume, density, and mass, and the escape velocity derived from the formula.
* **Can\_escape\_planet**

The can\_escape\_planet method takes in a planet\_name parameter and determines if the rocket can escape the planet. It gets the planet information from a dictionary named planets and calculates the rocket's volume, density, and mass. It then calculates the planet's escape velocity and uses it to calculate the rocket's escape velocity. If the rocket's escape velocity is greater than the planet's escape velocity, the method returns True. If not, it calculates the required mass and thrust needed for the rocket to escape and prints a message indicating.

Basically, The can\_escape\_planet() method determines whether the rocket can escape from a given planet. The method takes a planet name as input, and it retrieves the planet's mass and circumference from the planets dictionary. The method then calculates the rocket's density, mass, and escape velocity using the rocket's height, weight, and width. Finally, it compares the rocket's escape velocity to the planet's escape velocity to determine whether the rocket can escape.

* The program defines a set of rockets and prints their details using a for loop, and then defines a list of planets and prints their names.
* Planets Dictionary: The planets dictionary contains the mass and circumference of the eight planets in our solar system. The keys of the dictionary are the planet names in lowercase, and the values are tuples containing the planet's circumference in meters and mass in kilograms.
* Rockets Set: The rockets\_set dictionary contains a set of rockets. Each key is a rocket name, and the value is a Rocket object with the rocket's height, weight, and width as arguments.
* Main Function: The main program prompts the user to enter the rocket's height, weight, and width, and the planet's name. It creates a new Rocket object with these values and calls the can\_escape\_planet() method to determine whether the rocket can escape from the planet. If the rocket can escape, the program prints a message indicating that the rocket can escape. If the rocket cannot escape, the program prints a message indicating that the rocket cannot escape and exits the loop.

In Conclusion, the user inputs the mass, height and width of the rocket and after the required computations the program returns whether it will overcome the gravitational force and gives the output of required mass and thrust by that rocket.