Lab11

February 23, 2024

Exercise 1: create a class

Create a class called Star to store data and methods associated with individual stars. It should contain the following attributes: name as a string right ascension as a tuple (hours, minutes, seconds) declination as a tuple (degrees, arcminutes, arcseconds) distance as a float (pc) absolute magnitude as a float The constructor method should take these as arguments and use them to set the values of the attributes associated with the object. Give the class two methods: appmag returns the apparent magnitude using **str** returns the string representation of the star, like the following (use embedded newlines) Star: Betelgeuse RA: 05h 55m 10.3s Dec: +07d 24' 25.4" Mag: -5.85 Dist: 220.0 pc $m - M = 5(\log d - 1)$

[6]:
```
import math
from riseset import riseandset
class Star:
    def __init__(self, name, right_ascension, declination, distance,␣
 ↪absolute_magnitude):
        self.name = name
        self.right_ascension = right_ascension
        self.declination = declination
        self.distance = distance
        self.absolute_magnitude = absolute_magnitude

    def appmag(self):
        return self.absolute_magnitude + 5 * (math.log10(self.distance) - 1)

    def __str__(self):
        ra = f"{self.right_ascension[0]}h {self.right_ascension[1]}m {self.
 ↪right_ascension[2]}s"
        dec = f"{self.declination[0]}d {self.declination[1]}' {self.
 ↪declination[2]}\""
        return (f"Star: {self.name}\n"
                f"RA: {ra}\n"
                f"Dec: {dec}\n"
                f"Mag: {self.appmag():.2f}\n"
                f"Dist: {self.distance} pc")
    def riseandset(self, lam, phi, y, m, d):
      return riseandset(self.right_ascension, self.declination, lam, phi, y, m,␣
 ↪d)
```

```
betelgeuse = Star("Betelgeuse", (5, 55, 10.3), (7, 24, 25.4), 220.0, -5.85)
print(betelgeuse)
Betelgeuse = Star("Betelgeuse", (5, 55, 10.3), (7, 24, 25.4), \
220., -5.85)
print(Betelgeuse.riseandset(40.112, -88.221, 2020, 3, 2))
```

```
((12, 37, 56), (1, 24, 2))
```

[8]:
```
class Particle:
    def __init__(self, m, x, y, z, vx, vy, vz):
        self.m = m
        self.x = x
        self.y = y
        self.z = z
        self.vx = vx
        self.vy = vy
        self.vz = vz
        self.ax = 0.0
        self.ay = 0.0
        self.az = 0.0

    def move(self, dt):
        self.x += self.vx * dt + 0.5 * self.ax * dt ** 2
        self.y += self.vy * dt + 0.5 * self.ay * dt ** 2
        self.z += self.vz * dt + 0.5 * self.az * dt ** 2

        self.vx += self.ax * dt
        self.vy += self.ay * dt
        self.vz += self.az * dt

    def compute_accels(self, particles):
        G = 6.67430e-11
        for p in particles:
            if p != self:
                dx = p.x - self.x
                dy = p.y - self.y
                dz = p.z - self.z
                r_squared = dx**2 + dy**2 + dz**2
                r_cubed = math.sqrt(r_squared) * r_squared
                accel = G * p.m / r_cubed
                self.ax += accel * dx
                self.ay += accel * dy
                self.az += accel * dz

class StarParticle(Particle):
    def __init__(self, m, x, y, z, vx, vy, vz, t):
```

```
        super().__init__(m, x, y, z, vx, vy, vz)
        self.t = t

    def age_stars(self, dt):
        self.t += dt


p1 = Particle(10, 0, 0, 0, 1, 0, 0)
p2 = Particle(5, 0, 5, 0, 0, 1, 0)
p3 = Particle(8, 0, 0, 5, 0, 0, 1)

particles = [p1, p2, p3]

for p in particles:
    p.compute_accels(particles)
    p.move(0.1)
    print("Position:", p.x, p.y, p.z)
    print("Velocity:", p.vx, p.vy, p.vz)

star = StarParticle(20, 0, 0, 0, 1, 1, 1, 0)
star.age_stars(0.1)
print("Star Age:", star.t)
```

```
Position: 0.1 6.6743e-14 1.0678880000000001e-13
Velocity: 1.0 1.3348599999999998e-12 2.1357759999999997e-12
Position: 2.6681189685425146e-15 5.099999999999829 3.775554231738985e-14
Velocity: 5.3362379370850283e-14 0.9999999999965767 7.551108463477969e-13
Position: 2.6681189685425904e-15 2.335785311852168e-14 5.099999999999843
Velocity: 5.33623793708518e-14 4.671570623704336e-13 0.9999999999968738
Star Age: 0.1
```

[ ]:
```
%%capture
# Here we use a script to generate pdf and save it to google drive.

# After executing this cell, you will be asked to link to your GoogleDrive␣
 ↪account.
# Then, the pdf will be generated and saved to your GoogleDrive account and you␣
 ↪need to go there to download;

from google.colab import drive
drive.mount('/content/drive')
# install tex; first run may take several minutes
! apt-get install texlive-xetex
# file path and save location below are default; please change if they do not␣
 ↪match yours
! jupyter nbconvert --output-dir='/content/drive/MyDrive/' '/content/drive/
 ↪MyDrive/Colab Notebooks/Lab11.ipynb' --to pdf
```