StefinRacho_Homework_1

September 4, 2024

CS 4010 - Homework 1

Due Friday, 9/13/2024

For each homework set: Create a New iPython (Jupyter) notebook. Name the notebook FirstAndLastName_Homework1 and save it. To submit, export or print your notebook as a pdf, with all outputs visible. Upload both the pdf and a copy of your notebook (.ipynb) in Canvas.

1. For a computer with 4 GB of RAM, how many int values can the RAM hold? Note that your computer might have a different size of int. Make sure your code checks this and compute it properly.
   - Repeat for Complex values.

[2]:
```
# The following code was made with knowledge from
# the following link: https://www.geeksforgeeks.org/
 ↪how-to-find-size-of-an-object-in-python/
import sys

int_size = sys.getsizeof(0)
print(f"The size of an int in python (on this computer) is: {int_size} bytes.")

# To calculate how many int values the
# RAM can hold we need to divide 4 billion bytes by
# the size of an int.
ram_size = 4 * (10**9) # 4 GB in bytes
num_ints = ram_size // int_size

print(f"This computer with 4 GB of RAM can hold approximately {num_ints} int␣
 ↪values.")
```

```
The size of an int in python (on this computer) is: 28 bytes.
This computer with 4 GB of RAM can hold approximately 142857142 int values.
```

2. Search for Mark Newman's website on Computational Physics textbook.
   - Go to the programs and data page.
   - Open dropped.py, copy and paste the code into your notebook
   - Run the code in your notebook. Try at least 3 value of height and time. Copy the codes to 3 boxes at least.
     - 1: The object is still in flight
     - 2: The object passed through the ground
     - 3: The object landed on the ground. Output ~ 0 (doesn't have to be exact)

[4]:
```
h = float(input("Enter the height of the tower: "))
t = float(input("Enter the time interval: "))

s = 9.81*t**2/2

print("The height of the ball is",h-s,"meters")
```

```
Enter the height of the tower:  30
Enter the time interval:  2
```

```
The height of the ball is 10.379999999999999 meters
```

[5]:
```
h = float(input("Enter the height of the tower: "))
t = float(input("Enter the time interval: "))

s = 9.81*t**2/2

print("The height of the ball is",h-s,"meters")
```

```
Enter the height of the tower:  40
Enter the time interval:  1
```

```
The height of the ball is 35.095 meters
```

[6]:
```
h = float(input("Enter the height of the tower: "))
t = float(input("Enter the time interval: "))

s = 9.81*t**2/2

# h = 50
# t = 3.1919
print("The height of the ball is",h-s,"meters")
```

```
Enter the height of the tower:  50
Enter the time interval:  3.1919
```

```
The height of the ball is 0.026753382949998183 meters
```

3. Exercise 2.1

[8]:
```
import numpy as np

height = float(input("Enter the height of the tower in meters: "))

# We can use the following formula to calculate the
# height of the ball with variables gravity and time:
# h = 1/2gt^2
# Solving for time, we get:
# t = sqrt(2h/g)
gravity = 9.8
time = np.sqrt((2* height) / gravity)
```

```
print(f"The ball takes approximately {time} seconds to hit the ground.")
```

```
Enter the height of the tower in meters:  100

The ball takes approximately 4.5175395145262565 seconds to hit the ground.
```

4. Exercise 2.2

[10]:
```
# PART B
# The formula they gave us is:
# h = ((G * M * T**2) / (4 * pi**2))**(1/3) - R
G = 6.67 * 10**-11
M = 5.97 * 10**27 # Convert kg to g by added 3 to the exponent
R = 6371 * 1000 # Convert km to m by multiplying by 1000

T = float(input("Enter your desired value of T: "))
height = ((G * M * T**2) / (4 * np.pi**2))**(1/3) - R
print(f"For a satellite to orbit the Earth once every {T} \
seconds it needs to be launched to a height of \
approximately {height} meters above the Earth's surface.")
```

```
Enter your desired value of T:  555555555

For a satellite to orbit the Earth once every 555555555.0 seconds it needs to be
launched to a height of approximately 146008965087.80573 meters above the
Earth's surface.
```

[11]:
```
# PART C
T = 86400 # Seconds per day
height_day = ((G * M * T**2) / (4 * np.pi**2))**(1/3) - R
print(f"For a satellite to orbit the Earth once per day it \
needs to be launched to a height of \
approximately {height_day} meters above the Earth's surface.")

T = 5400 # Seconds per 90 minutes
height = ((G * M * T**2) / (4 * np.pi**2))**(1/3) - R
print(f"For a satellite to orbit the Earth once per 90 minutes \
needs to be launched to a height of \
approximately {height} meters above the Earth's surface.")

T = 2700 # Seconds per 45 minutes
height = ((G * M * T**2) / (4 * np.pi**2))**(1/3) - R
print(f"For a satellite to orbit the Earth once per 45 minutes \
needs to be launched to a height of \
approximately {height} meters above the Earth's surface.")
```

```
For a satellite to orbit the Earth once per day it needs to be launched to a
height of approximately 415898101.7617496 meters above the Earth's surface.
For a satellite to orbit the Earth once per 90 minutes needs to be launched to a
```

```
height of approximately 60132216.25372859 meters above the Earth's surface.
For a satellite to orbit the Earth once per 45 minutes needs to be launched to a
height of approximately 35523401.02189176 meters above the Earth's surface.
```

The conclusion that I can draw from the last calculation is that as the orbital period decreases so does the altitude at which the satellite must orbit. In other words, in order for a satellite to orbit the Earth faster it must be closer to Earth's surface.

[13]:
```
# PART D
T = 86148 # Seconds per sidereal day
height_sidereal_day = ((G * M * T**2) / (4 * np.pi**2))**(1/3) - R
difference = height_day - height_sidereal_day
print(f"The difference in altitude between a day orbit vs a sideral day orbit␣
 ↪is {difference} meters.")
```

```
The difference in altitude between a day orbit vs a sideral day orbit is
821478.4627931714 meters.
```

5. Exercise 2.3

[15]:
```
x = float(input("Enter a value for the x-coordinate: "))
y = float(input("Enter a value for the y-coordinate: "))

# To calculate r given x and y we can use the following formula:
r = np.sqrt(x**2 + y**2)

# To calculate theta given x and y we can use the following formula:
theta = np.degrees(np.arctan2(y, x))

print(f"The corresponding polar coordinates are:\nr = {r}\ntheta = {theta}")
```

```
Enter a value for the x-coordinate:  5
Enter a value for the y-coordinate:  4

The corresponding polar coordinates are:
r = 6.4031242374328485
theta = 38.65980825409009
```

6. Exercise 2.6

[17]:
```
l1 = float(input("Enter the distance to the Sun at perihelion (l1): "))
v1 = float(input("Enter the velocity at perhelion (v1): "))
print(f"l1 = {l1}\nv1 = {v1}")

# Given l1 and v1 we can solve for v2 by using the following formula:
# v2^(2)-((2GM))/((v1)l1)v2-[v1^(2)-((2GM))/(l1)]=0
# Simplifying we get:
# v2 = v1
v2 = v1
print(f"v2 = {v2}")
```

```
# Now that we have v2 we can calculate l2 by the following formula:
# l2 = l1(v1)/v2
l2 = l1*v1/v2
print(f"l2 = {l2}")

# Now that we have l1 and l2 we can calculate the Semi-major axis:
semi_major_axis = (1/2)*(l1+l2)
print(f"semi major axis = {semi_major_axis}")

semi_minor_axis = np.sqrt((l1*l2))
print(f"semi minor axis = {semi_minor_axis}")
print(f"pi = {np.pi}")

orbital_period = (2 * np.pi * semi_major_axis * semi_minor_axis)/(l1*v1)
print(f"orbital period = {orbital_period} seconds, or {orbital_period/(3.
  ↪154*10**7)} years.")

orbital_eccentricity = (l2 - l1)/(l2 + l1)
print(f"orbital eccentricity = {orbital_eccentricity}")
```

```
Enter the distance to the Sun at perihelion (l1):  147100000000
Enter the velocity at perhelion (v1):  30287

l1 = 147100000000.0
v1 = 30287.0
v2 = 30287.0
l2 = 147100000000.0
semi major axis = 147100000000.0
semi minor axis = 147100000000.0
pi = 3.141592653589793
orbital period = 30516609.72318543 seconds, or 0.9675526228023282 years.
orbital eccentricity = 0.0
```

7. Exercise 2.7

[19]:
```
cat_nums = [1]
cat_num = 1
limit = 1000000000

while (cat_num <= limit):
    n = len(cat_nums) - 1
    cat_num = ((4 * n + 2)/(n + 2))*cat_nums[n]
    if (cat_num > limit):
        break
    cat_nums.append(cat_num)
    print(cat_nums)
```

```
[1, 1.0]
```

```
[1, 1.0, 2.0]
[1, 1.0, 2.0, 5.0]
[1, 1.0, 2.0, 5.0, 14.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0, 16796.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0, 16796.0, 58786.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0, 16796.0, 58786.0,
208012.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0, 16796.0, 58786.0,
208012.0, 742900.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0, 16796.0, 58786.0,
208012.0, 742900.0, 2674440.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0, 16796.0, 58786.0,
208012.0, 742900.0, 2674440.0, 9694845.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0, 16796.0, 58786.0,
208012.0, 742900.0, 2674440.0, 9694845.0, 35357670.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0, 16796.0, 58786.0,
208012.0, 742900.0, 2674440.0, 9694845.0, 35357670.0, 129644790.0]
[1, 1.0, 2.0, 5.0, 14.0, 42.0, 132.0, 429.0, 1430.0, 4862.0, 16796.0, 58786.0,
208012.0, 742900.0, 2674440.0, 9694845.0, 35357670.0, 129644790.0, 477638700.0]
```

[ ]: