Modeling and Simulation in Python (Downey)

September 19, 2023

1 Exercise 1

Our first batch of exercises comes (mostly) from the textbook by Downey. I have marked the difficulty:

- 'hard' (***) = 30 points
- 'medium' (***) = 20 points
- 'easy' (***) = 10 points

We'll briefly discuss the solutions in class. Enjoy.

Mathematical functions (*)

1. Import π, sine, cosine and square root from NumPy on one line.
2. Compute $sin^2(\pi/4)+cos^2(\pi/4)$. The result is approximately 1, because $\forall\theta \in (0, 360] : sin^2(\theta) + cos^2(\theta) = 1$.
3. Mathematically, both sine and cosine of $\pi/4$ are $1/sqrt2$. Compute this expression, and print out the difference between the two results, too.

Solution:

```
from numpy import pi, sin, cos, sqrt
x = sin(pi/4)**2+cos(pi/4)**2
print(x)
y = (1/sqrt(2))**2 + (1/sqrt(2))**2
print(y)
print(x-y)
```

Python errors (**)

Use the falling penny myth Python example to generate four different Python errors: `SyntaxError`, `NameError`, `TypeError` and `ValueError`.

To show all errors, use the equations for velocity, $v = at$, and for time, $t = \sqrt{2h/a}$, where a denotes the acceleration.

You have to use a different solution for each code cell. You can also check if the AI can debug your deliberate errors.

```
# SyntaxError
print(v = a t)

# NameError
v = a * t

# TypeError
a = 9.8
t = '3.4'
v = a * t

# ValueError
a = 9.8
t = 3.4
v = a * t
print(float('v'))
```

Falling penny revisited (**)

Suppose you bring a 10-foot pole to the top of the Empire State Building and use it to drop the penny from `h` plus 10 feet.

1. Define a variable named `foot` that contains the unit `foot` provided by the `UnitRegistry()` named `units`.
2. Define a variable named `pole_height` and give it the value 10 feet.
3. Add `h = 381 * meter` to `pole_height`, which is in feet.
4. Write the addition the other way around.
5. Print the results using f-strings.

Solution:

```
# import pint.UnitRegistry
from pint import UnitRegistry
# create instance of UnitRegistry
units = UnitRegistry()
# define meter and foot
meter = units.meter
foot = units.foot
# define pole_height
pole_height = 10 * foot
# define height h
h = 381 * meter
# add h to pole_height
print(f'h [m] plus pole_height [ft]: {h + pole_height}')
# add pole_height to h
print(f'pole_height [ft] plus h [m]: {pole_height + h}')
```

Combining models (***)

So far, we have considered two models of a falling penny:

1. If we ignore air resistance, the penny falls with constant acceleration, and we can compute the time `t` to reach the sidewalk and the velocity `v` of the penny when it gets there.

2. If we take air resistance into account, and drop the penny at its terminal velocity, it falls with constant velocity `v_t`.

Now, let's consider a third model that includes elements of the first two: assume that the acceleration of the penny is `a` until the penny reaches $29 m/s$, and then $0 m/s^2$ afterwards. What is the total time for the penny to fall $381 m$?

Tip: you can break this question into three parts:

1. How long would the penny take to reach terminal velocity with constant acceleration `a`?

2. How far would it fall during that time?

3. How long would it take to fall the remaining distance with terminal velocity?

Assign each intermediate result to a variable with a meaningful name. And assign units to all quantities!

Solution

Establish units:

```
from pint import UnitRegistry
units = UnitRegistry()
meter = units.meter
second = units.second
```

Establish constants:

```
g = 9.8 * meter / second**2
h = 381 * meter
v_t = 29 * meter / second
```

1. How long would the penny take to reach terminal velocity with constant acceleration `a = g`?

   ```
   t_1 = v_t / g
   print(f'Time to reach terminal velocity: {t_1:.2f}s')
   ```

2. How far would it fall during that time?

   ```
   # distance h_1 traveled with constant acceleration g during t_1
   h_1 = g * t_1**2 / 2
   print(f'Height fallen in that time: {h_1:.2f}s')
   ```

3. How long would it take to fall the remaining distance with terminal velocity (and zero acceleration)?

   ```
   t_2 = (h - h_1) / v_t
   print(f'Time to fall the remaining distance: {t_2:.2f}s')
   ```

Therefore:

```
print(f'Total time to fall from Empire State building: {t_1+t_2:.2f}s')
```

Alternatively, you can also solve the problem mathematically first and compute:

$$t_1 = \frac{v_t}{g} \tag{1}$$

$$h_1 = \frac{gt_1^2}{2} = \frac{gv_t^2}{2g} \tag{2}$$

$$t_2 = \frac{(h - h_1)}{v_t} = \frac{(h - \frac{gv_t^2}{2g})}{v_t} = \frac{1}{v_t}(h - \frac{v_t^2}{2g}) \tag{3}$$

$$t = t_1 + t_2 = \frac{v_t}{g} + \frac{1}{v_t}(h - \frac{v_t^2}{2g}) \tag{4}$$

Or in code:

```
t = v_t/g + (h - v_t**2/(2*g))/v_t
print(f'Total time to fall from Empire State building: {t:.2f}s')
```

Bonus error: when you violate unit rules, you get a `DimensionalityError` from `pint`.

```
print(a + t)
```

Error message (excerpt):

```
raise DimensionalityError(pint.errors.DimensionalityError:
Cannot convert from 'meter/second**2' ([length] / [time] ** 2)
to 'second' ([time])
```