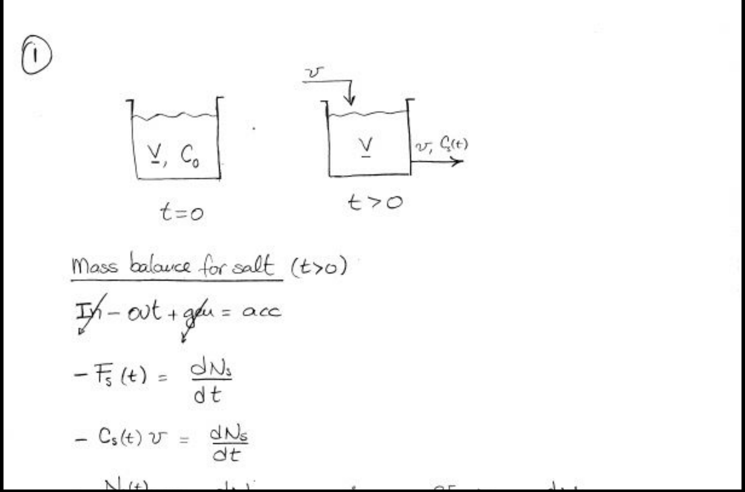
The Cooper Union for the Advancement of Science and Art

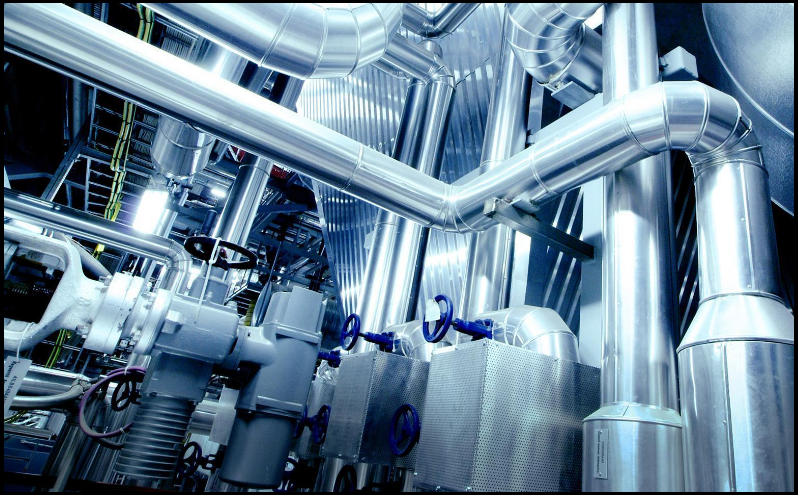

①
V, C_0
t=0
v
V
v, $C_s(t)$
t>0

Mass balance for salt (t>0)

$\cancel{In} - out + \cancel{gen} = acc$

$-F_s(t) = \frac{dN_s}{dt}$

$-C_s(t)\,v = \frac{dN_s}{dt}$



ChE352
Numerical Techniques for Chemical Engineers

Professor Stevenson

Lecture 2

Engineering topics of interest

Computer programming ✅
Biochemistry ⏳
Food science 🗳️
Linguistics ⏳
Machine learning ⏳
Power plants 🗳️
Research skills ⏳
Sustainability 🗳️
Optimization algorithms ⏳
Materials science 🗳️
Renewable energy 🗳️

Nuclear engineering 🗳️
Geochemistry 🗳️
Electrochemistry 🗳️
Cosmetic sciences ⏳
Data analysis ⏳
Engineering management ⏳
Environmental chemistry 🗳️
Electrical & mechanical eng. 🗳️
Process Simulation ⏳
How many licks does it take to get to the center of a tootsie pop? 🍭

Numerical methods apply to all parts of engineering

Important data types for this class

```
1  # integer
3.14  # floating point (key to this class)
'3.14'  # string
[1, 2.0, '3']  # list (any types)
np.array([1, 2, 3])  # array (one type)
{1, 2, '3'}  # set
{"H": 1, "He": 2, "Li": '3'}  # dictionary
```

What is each type good for?

Test-driven design

- Testing is easier than writing correct code
- Automated testing is the best kind

Example: find Z (compressibility) in the Soave-Redlich-Kwong equation of state:

$$Z^3 - Z^2 + \left(A - B - B^2\right)Z - AB = 0$$

Plugging in values of Z, A, B as a test is very simple. You can write a test function right now, without knowing anything about solver code.

Testing can be all you need

```python
def is_srk_solution(Z, A, B):
  srk = Z**3 - Z**2 + (A - B - B**2)*Z - A*B
  return abs(srk) < 1e-3
```

$$Z^3 - Z^2 + \left(A - B - B^2\right)Z - AB = 0$$

Testing can be all you need

```
def is_srk_solution(Z, A, B):
  srk = Z**3 - Z**2 + (A - B - B**2)*Z - A*B
  return abs(srk) < 1e-3
```

Run the test on a range of possible solutions:

```
A, B = 2.0, 3.0  # inputs
low, high, step = 0.0, 10.0, 1e-4
for Z in np.arange(low, high, step):
  if is_srk_solution(Z, A, B):
      print('Found solution:', Z)
```

Python error messages

```
for Z in np.arange(low, high, step):
  if is_sk_solution(Z, A, B):
      print('Found solution:', Z)
```

Running this code gives:

```
NameError: name 'is_sk_solution' is not defined
```

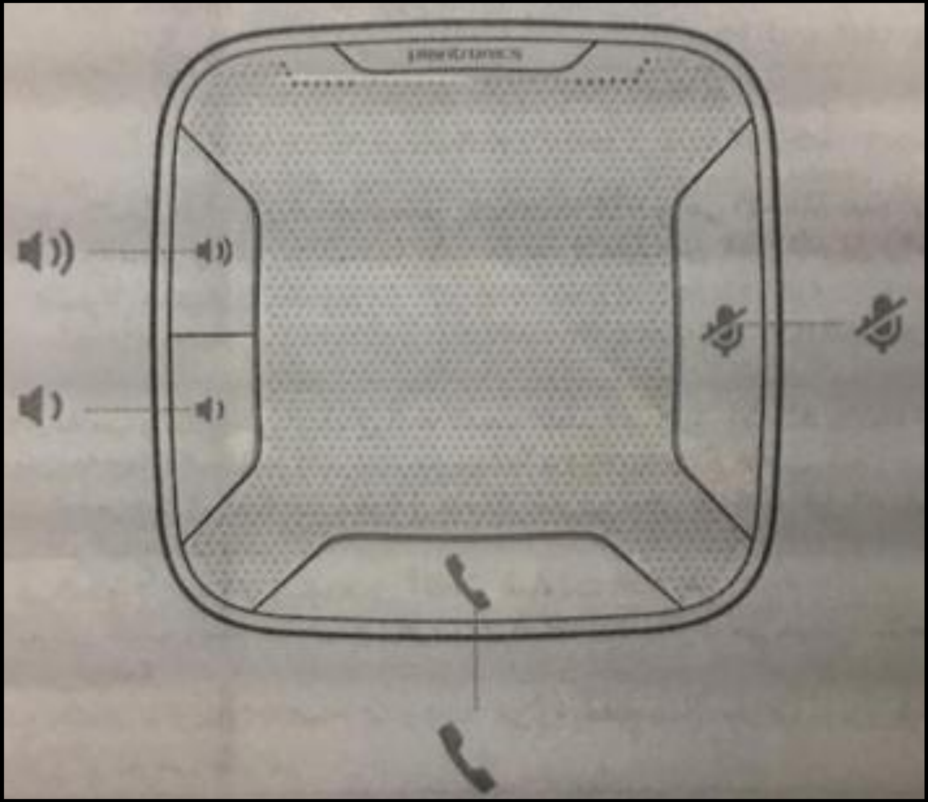
What went wrong?

Python error message tips

- Google your error messages
  - Especially the class of exception
- Run your code often, so you know which change caused any new errors
  - This is much easier if your code is fast
- print() key variables to show more details
  - If there's no output at all, add print statements early in the code, see which outputs appear

Program for Readability

- Readable code is about ***empathy***
- Put yourself in a reader's place
- Describe your assumptions clearly
- The code says ***what*** you're doing - the comments say ***why***



```
# Bad programmers
# comment their code
# like this diagram
```

Readability example

What do you think this function does?

```
def relerr(p, r, eps=1e-6):
    return (p - r) / (abs(r) + eps)
```

How can we make it more readable?

Readability example

```
# Relative error of prediction vs reference
def relative_error(prediction,
                   reference,
                   epsilon=1e-6):
   error = prediction - reference
   # epsilon prevents divide-by-zero issues
   return error / (abs(reference) + epsilon)
```

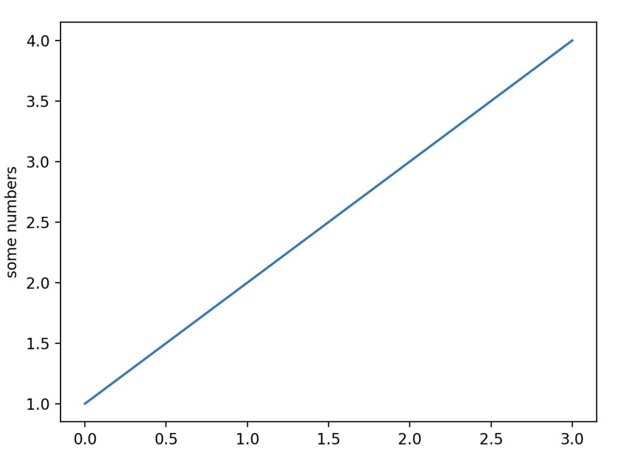
You don’t have to code from scratch

- Start with a basic example
  - Cite your code sources!
- Slowly change it into what you want
- Run it every few minutes

```
# https://matplotlib.org/stable/tutorials/introductory/pyplot.html

import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.show()

# Not the code I wanted, but enough to help
# So I will copy it and cite it at the top
```



You don't have to code from scratch

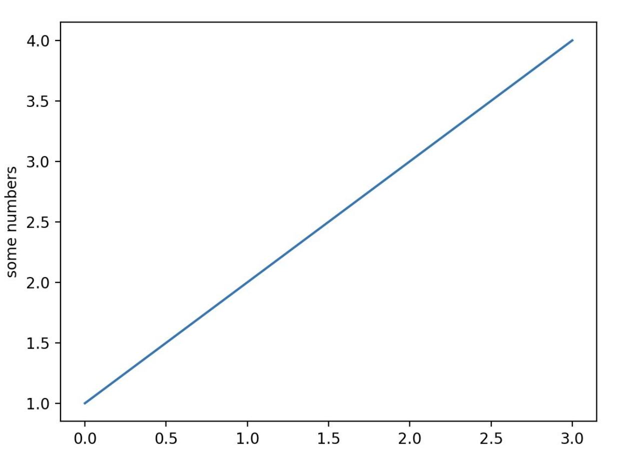
Open up Google Colab, make a new notebook, and try the example below.

Then, modify it to plot something else

```
# https://matplotlib.org/stable/tutorials/introductory/pyplot.html

import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.ylabel('some numbers')
plt.show()

# Not the code I wanted, but enough to help
# So I will copy it and cite it at the top
```



```
import time

print('Lecture paused')
time.sleep(600)  # seconds
print('More Python')
```

NumPy = Numerical Python

- Python is designed to be *flexible* to use
  - `mixed_type_list = [1, 2.0, '3']`

- Numpy is a package for doing math (especially linear algebra) *fast*
  - `eigenvalues, eigenvectors = eig(A)`
  - Can calculate eigenvectors of a million-entry matrix in seconds
  - Great tool for turning science & math into code

Numpy arrays

np.array of ints

```
import numpy as np

x = np.array([1, 4, 3])

y = np.array([[1, 4, 3], [9, 2, 7]])

print('x.shape:', x.shape, ' x.size:', x.size)

print('y.shape:', y.shape, ' y.size:', y.size)
```

```
x.shape: (3,)  x.size: 3
y.shape: (2, 3)  y.size: 6
```

tuple of ints

int

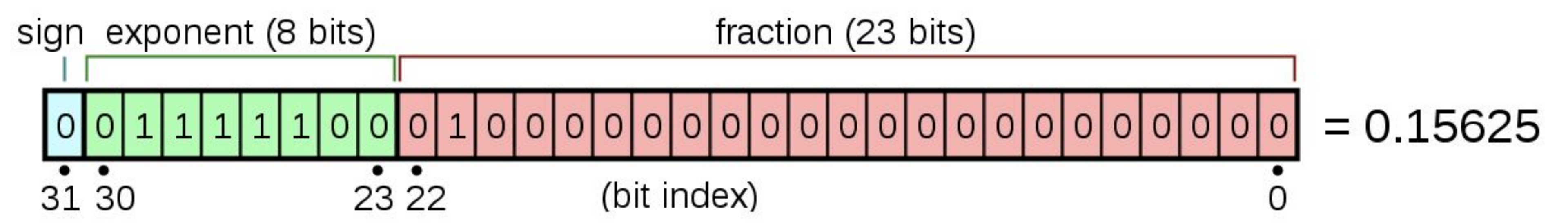
Loss of precision

How many real numbers are there?

How much information can be stored in a single real number?

What is floating point?

- Computer math is almost always floating point
- Like scientific notation on binary numbers



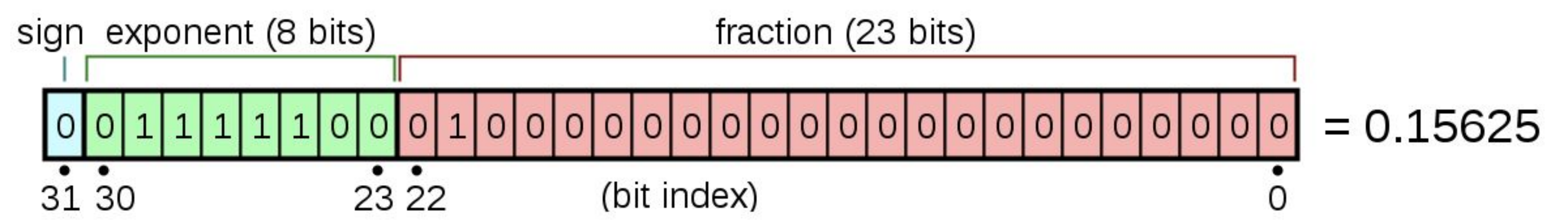
- Not every real number can be represented

How many decimal digits can we store in 23 bits?

What numbers can't be represented?

What is floating point?

- Computer math is almost always floating point
- Like scientific notation on binary numbers



- np.float32 holds ~7 decimal digits
- np.float64 holds ~16 decimal digits
- Not every real number can be represented
- Too big = overflow, too small = underflow
- Only binary fractions (no exact 1/3, 1/5, etc)

Python warmup continued

- I will provide class time and help to work on kaggle.com/learn/python - today and at office hours
- Graded by automated tests: you need all right answers, but the only penalty is to keep trying
- Any parts you don't finish in class will become HW #1