# ENGR 21:
# Computer Engineering Fundamentals

Lecture 8
Thursday, September 25, 2025

# Introducing `numpy`

# What is numpy?

https://numpy.org/
https://github.com/numpy/

- Numerical Computing in Python
  - Gives Python the capabilities of MATLAB
  - But less $$
- Free & Open Source
- Lives on github
- A package that you must import
  - Does not come pre-installed with Python installation
- Much faster than regular Python
- Optimized for scientific + engineering calculations

# Features of the numpy package

- Data types for numerical computing
  - `float16`
  - `float32`
  - `float64`
  - `int8, int16, int32, int64`
  - `uint8, uint16, uint32, uint64`
- Vast library of functions useful for engineering
- Everything is 'array-native' –
  - works seamlessly on large sets of numbers

```
>>> import numpy

>>> import math

>>> numpy.linspace(1,5,9)

array([1. , 1.5, 2. , 2.5, 3. , 3.5, 4. ,
4.5, 5. ])

>>> numpy.logspace(1,3,3)

array([  10.,  100., 1000.])

>>> math.sqrt(range(1,10))

TypeError: must be real number, not range



>>> numpy.sqrt(range(1,10))

array([1.        , 1.41421356, 1.73205081,
2.        , 2.23606798,

       2.44948974, 2.64575131, 2.82842712, 3.
       ])
```

# Introducing `pip`, Python's native package manager

# Installing Packages for Python

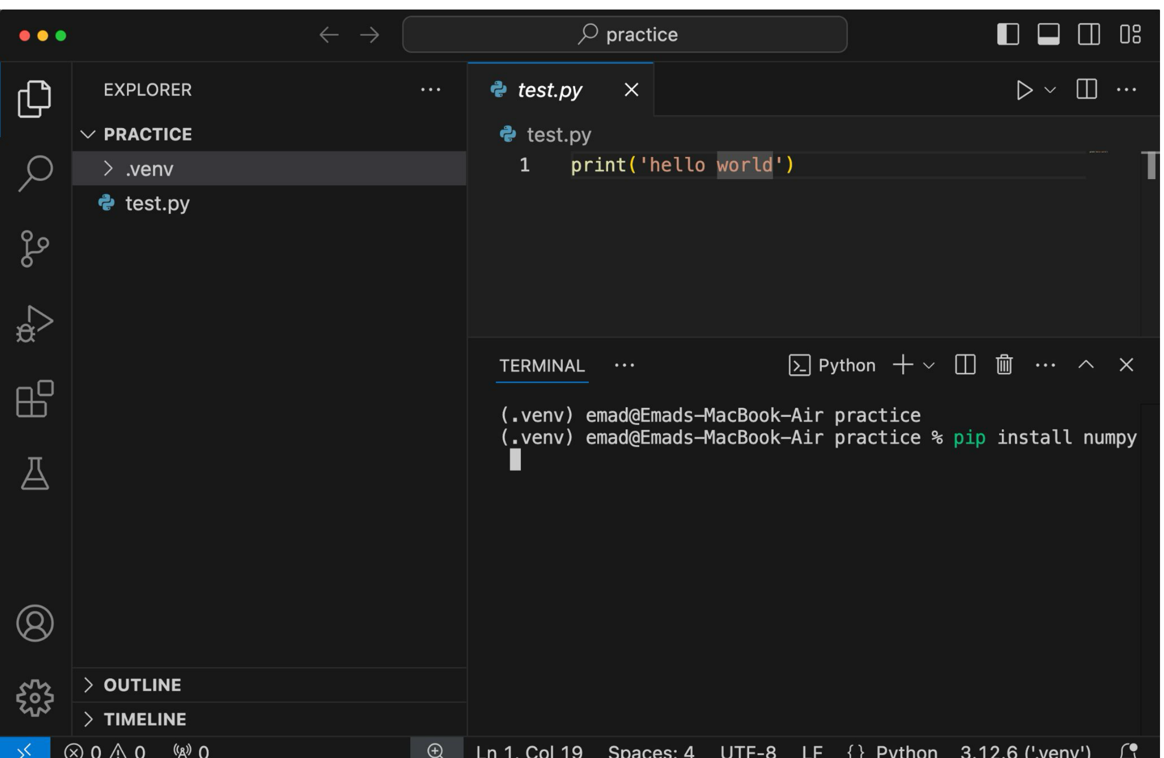
Inside a Terminal (NOT Python REPL):

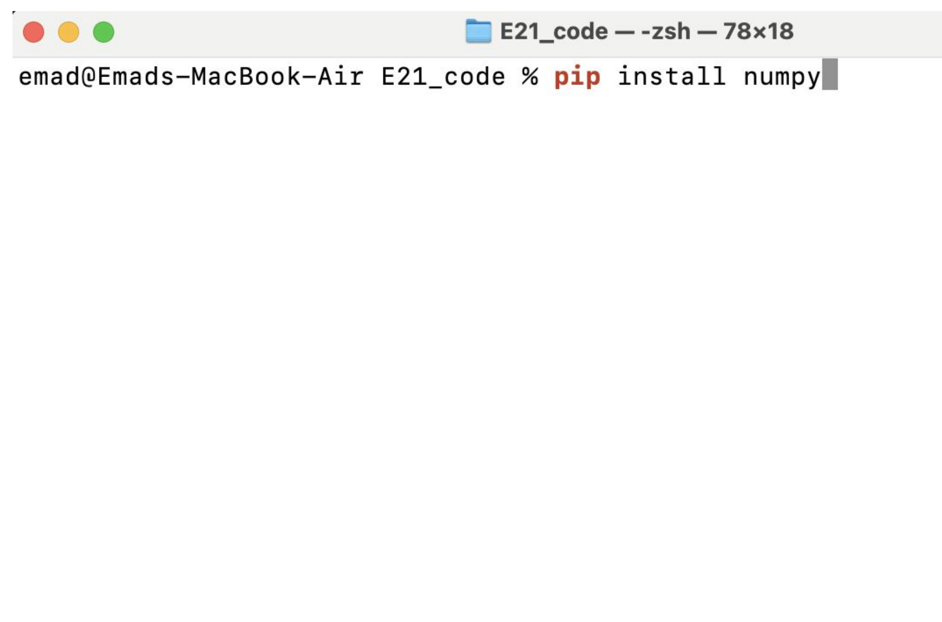
numpy

```
pip install <package name>
```

pip3

Options for Terminals:

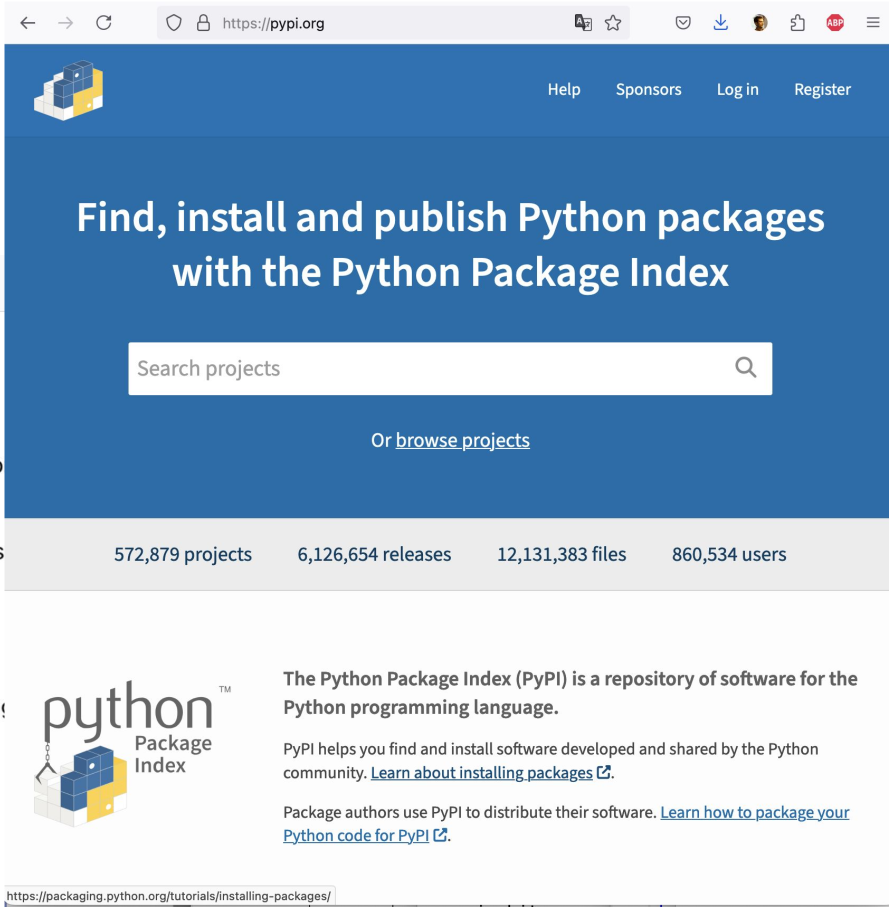


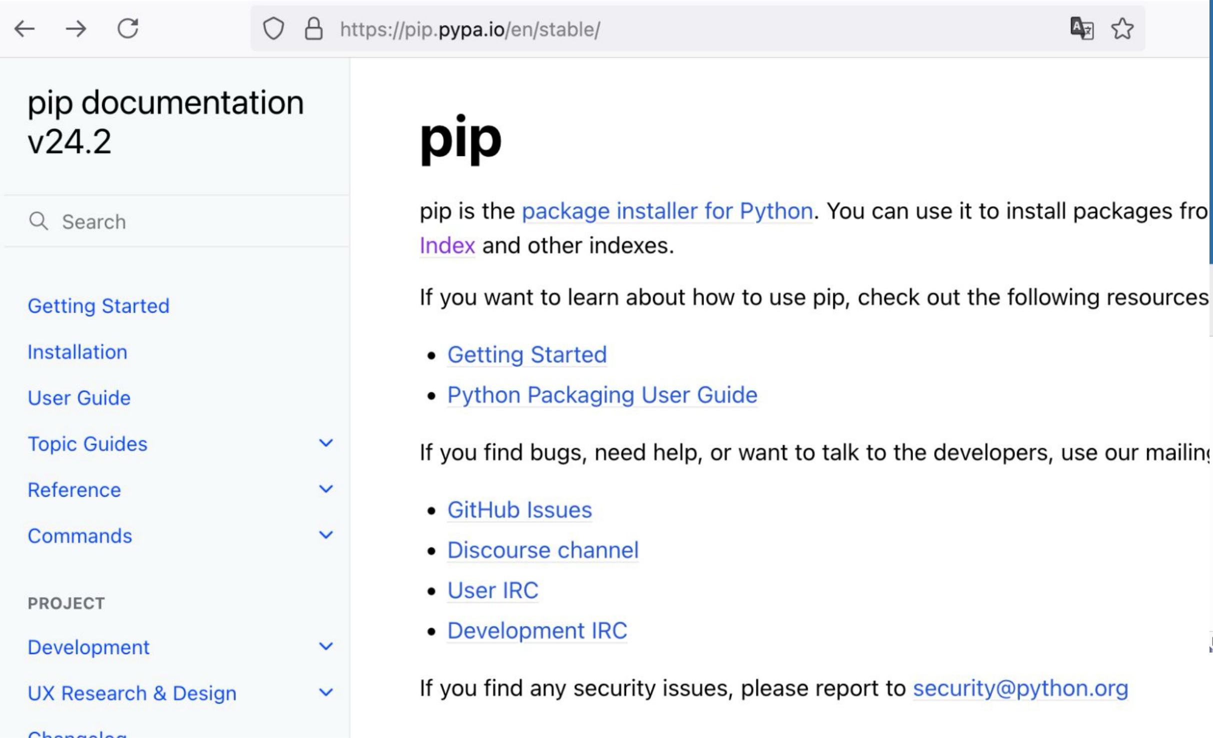



# What does `pip` do, exactly?

A ‘package manager’ for Python

Looks up packages on the official Python Package Index https://pypi.org/



# Some useful `pip` commands

```
pip3 install <package name>
pip3 list
pip3 uninstall <package name>
```

# Install `numpy` on your computer

1. Run `pip3 install numpy` from a terminal
   a. Either inside VS Code or in your operating system's terminal.
2. Check that numpy works
   a. Enter Python REPL (enter `python3` into a terminal window)
   b. Enter `import numpy` at the REPL

# Back to `numpy`

# Conventions for numpy

Three approaches to importing the package

Write at the top of any Python file that will use numpy one of the following

each time code runs

```
import numpy
```

Must use numpy.<function>

imports all functions

```
from numpy import *
```

Can simply use <function>

```
import numpy as np
```

Must use np.<function>

Only a convention!
Can choose any name

# Why does it matter how packages are imported?

Three choices:

```
import numpy
```

```
from numpy import *
```

```
import numpy as np
```

from 'standard library'

```
>>> import math
>>> import numpy

>>> sqrt(4)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqrt' is not defined

>>> math.sqrt
<built-in function sqrt>

>>> numpy.sqrt
<ufunc 'sqrt'>

>>> math.sqrt(4)
2.0

>>> numpy.sqrt(4)
np.float64(2.0)
```

type : 'float'

type 'np.float64'

Calculations in numpy are:

- More efficient
- More scalable
- Faster

Compared to regular Python

# Arrays in numpy

- Like a python list, but better
- Can be multidimensional

| | | | | | |
|---|---|---|---|---|---|
| ■ | □ | □ | ■ | □ | □ |
| □ | ■ | □ | □ | ■ | ■ |
| ■ | ■ | □ | ■ | □ | □ |
| □ | □ | ■ | □ | ■ | ■ |
| ■ | □ | ■ | □ | □ | ■ |
| □ | ■ | □ | ■ | ■ | □ |

## Native Python approach:

```
[True, False, False, True, False, False,
 False, True, False, False, True, True,
 True, True, False, True, False, False,
 False, False, True, False, True, True,
 True, False, True, False, False, True,
 False, True, False, True, True, False]
```

Or, we could have made a list of lists ...

## Numpy approach

```
array([[ True, False, False,  True, False, False],
       [False,  True, False, False,  True,  True],
       [ True,  True, False,  True, False, False],
       [False, False,  True, False,  True,  True],
       [ True, False,  True, False, False,  True],
       [False,  True, False,  True,  True, False]])
```

# Floating-point numbers

## Recall: How do we write non-integers?

| | |
|---|---|
| Decimal Numbers | 21.5679<br>$\cdots \; 10^1 \; 10^0 \; 10^{-1} \; 10^{-2} \; 10^{-3} \cdots$ |
| Binary Numbers | 101.10101<br>$\cdots \; 2^1 \; 2^0 \; 2^{-1} \; 2^{-2} \cdots$ |

## Task: Interpret the following binary numbers:

| Binary | Decimal /Fraction |
|---|---|
| 1.1101101001<br>$2^{-1} + 2^{-2} + 2^{-4} + \cdots$ | $\frac{1897}{1024} \approx 1.85$ |
| 1.0010011000 | |

$$1 + \frac{1}{2} + \frac{1}{4} + 0 + \frac{1}{16} + \frac{1}{32} + 0 + \frac{1}{128} + 0 + 0 + \frac{1}{1024}$$

# By how much do binary fractions increment?

Preliminary exercise:

1.796
1.797 bigger by $\frac{1}{1000}$

| Binary | Decimal |
|---|---|
| 1.0010011000 | $1\frac{19}{128}$ |
| 1.0010011001 | bigger by $\frac{1}{1024}$ |

$\frac{1}{8} + \frac{1}{64} + \frac{1}{128} = \frac{16}{128} + \frac{2}{128} + \frac{1}{128} = \frac{19}{128}$

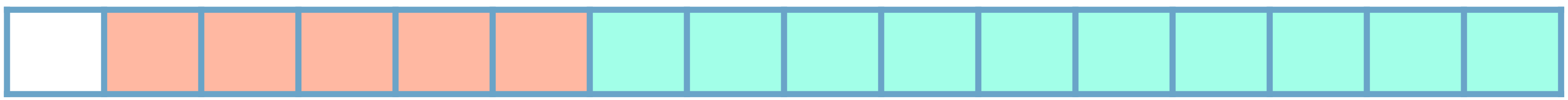
$\frac{1}{8} + \frac{1}{64} + \frac{1}{128} + \frac{1}{1024}$

# The IEEE Standard for Floating-Point Binary Numbers

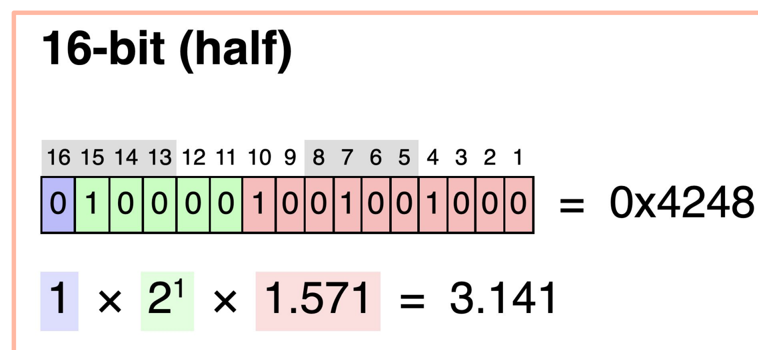
$\underbrace{1.23}\times\underbrace{10}^{79}$

$$\boxed{+/-}\ 1.\ \boxed{\text{Significand}} \times 2^{\boxed{\text{Exponent}}}$$

Bias. Subtract bias from exponent to get true value.



| Size | Sign | Significand | Exponent | Bias | Colloquial Name |
|---|---|---|---|---|---|
| 16-bit | 1 | 10 | 5 | 15 | Half precision |
| 32-bit | 1 | 23 | 8 | 127 | Single precision |
| 64-bit | 1 | 52 | 11 | 1023 | Double precision |




https://evanw.github.io/float-toy/

# The Exponent Bias

The exponent bias is introduced to allow floating-point numbers to have both positive and negative exponents on the “2”.

### 16-bit (half)

| 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

= 0x494B

$$1 \times 2^{3} \times 1.324 = 10.59$$

10010 :- 18

### 16-bit (half)

| 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

= 0x294B

$$1 \times 2^{-5} \times 1.324 = 0.04136$$

$2^0 \times 0 + 2^1 \times 1 + 2^2 \times 0 + 2^3 \times 1$

01010 :- 10 $= +2 + 8$

## Practice reading the IEEE Standard for Floating-Point Binary Numbers

$$\boxed{\phantom{0}}\ 1.\boxed{\phantom{000000000}} \times 2^{\boxed{\phantom{0000}}}$$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

1) Interpret exponent bits as an integer.
2) Subtract bias from it to get exponent
3) Interpret significand as the binary digits in the number 1._ _ _ _ _
4) Multiply significand with $2^{\text{exponent}}$
5) Interpret sign bit as: $\begin{cases} \text{negative if } 1 \\ \text{positive if } 0 \end{cases}$

**Exponent Bias:**
Subtract **15** from exponent

**16-bit (half)**

| 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | = 0x4248 |

$1 \times 2^1 \times 1.571 = 3.141$

# How much do 16-bit binary floats increment? v.2

Calculate the value of this number

$1\,\frac{381}{1024}$

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |

$2^4 + 2^1$

$= 16 + 2 = 18$

subtract bias

Exponent 3

$2^{-2} + 2^{-4} + 2^{-7} + 2^{-9} + 2^{-10}$

$\frac{1}{4} + \frac{1}{16} + \frac{1}{128} + \frac{1}{512} + \frac{1}{1024}$

$= \frac{256}{1024} + \frac{64}{1024} + \frac{8}{1024} + \frac{2}{1024} + \frac{1}{1024}$

$= 331/1024.$

Combine : $\boxed{1\,\frac{331}{1024} \times 2^3} = \frac{1024+331}{1024} \times 2^3 = \frac{1355}{128} \approx 10.5859$

# How much do 16-bit binary floats increment? v.2

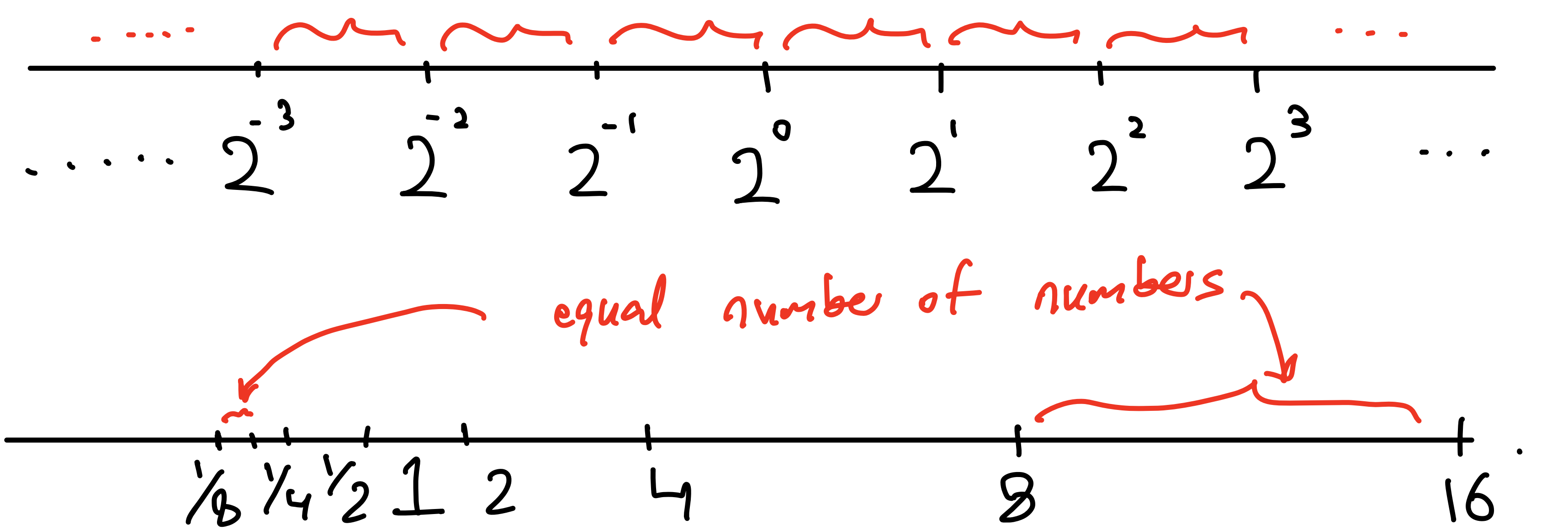
| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 10.5859 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 10.5938 |

Calculate the value of the next number

Difference ≈ 0.01 ?

# Gaps on the number line

Floating-point system assigns an equal number of numbers to each interval shown below



# Machine Epsilon for a floating-point number system

The difference between 1 and the next higher floating-point number