CS357

Objectives

1. Design a numerical experiment in Python+numpy+scipy

2. Write efficient Python+numpy by avoiding loops

3. Study how arrays of data scale in Python

4. Observe the limits of precision

1. An experiment

- Brownian motion

  Brownian motion is the random motion of
  particles suspended in a fluid (a liquid or
  a gas) resulting from their collision with
  the quick atoms or molecules in the gas or
  liquid. The term "Brownian motion" can also
  refer to the mathematical model used to
  describe such random movements, which is
  often called a particle theory. -Wikipedia

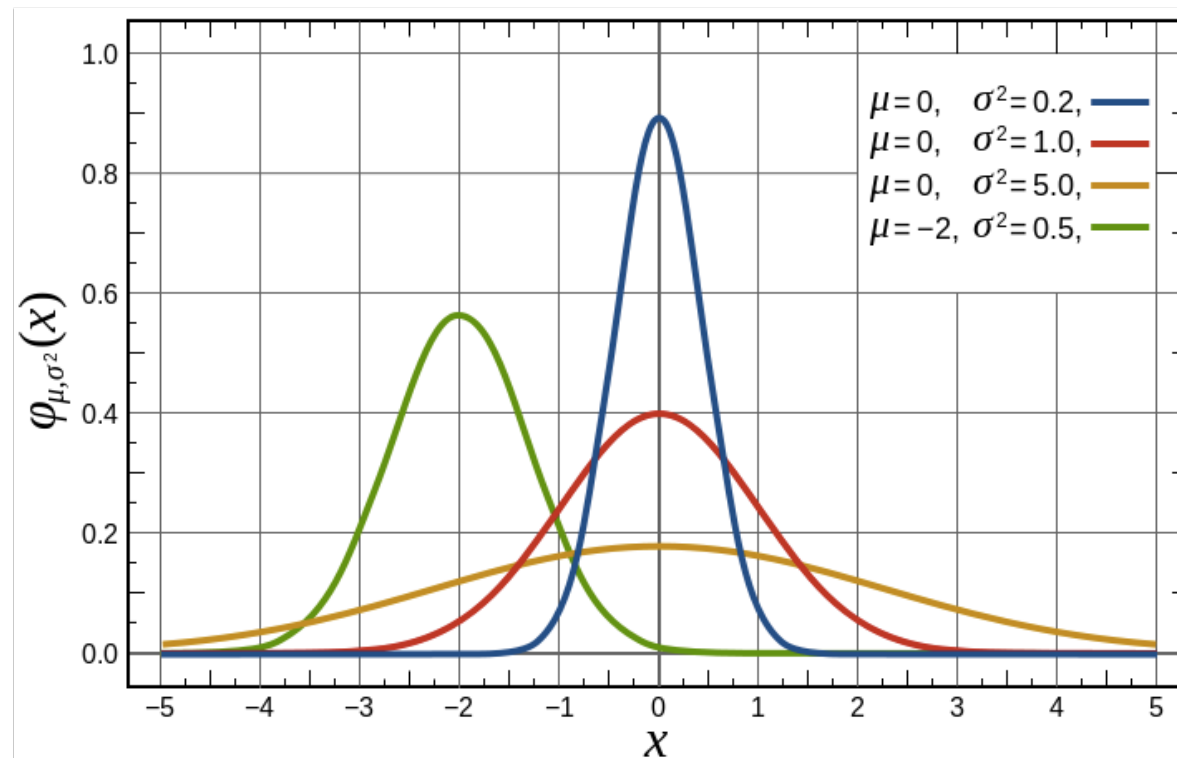$$x(t + \Delta t) = x(t) + \mathcal{N}(0, \sigma^2 \Delta t; t, t + \Delta t)$$

update
particle position

random
amount

1. An experiment

- Brownian motion

$$x(t + \Delta t) = x(t) + \mathcal{N}(0, \sigma^2 \Delta t; t, t + \Delta t)$$

mean

variance   independence
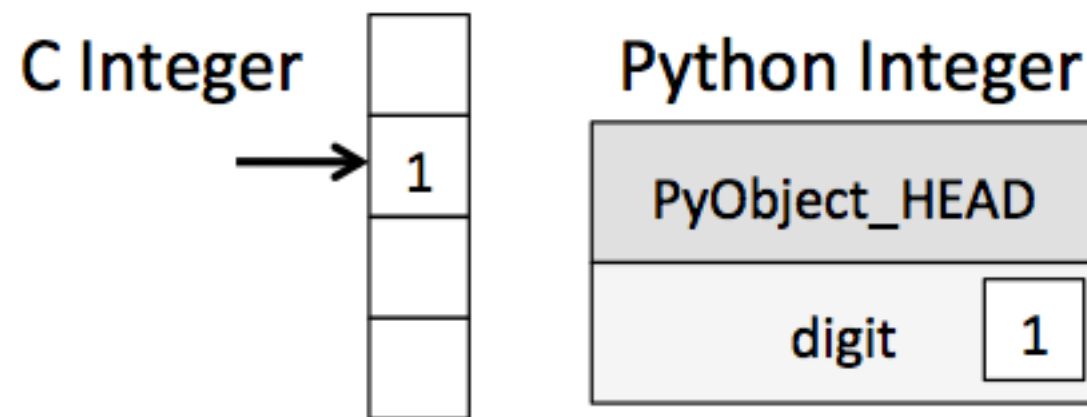
1. An experiment

- Write an algorithm!

- How does the distance from the origin depend on *n,* the number of steps?

- Is our algorithm efficient?

- Can we repeat the algorithm?

2. Loops

- Is Python slow?

  - yes: interpreted, dynamically typed

  - no: can call libraries and JIT

2. Loops

- https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/

- Dynamic typing

2. Loops

- https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/

```
/* C code */
int a = 1;
int b = 2;
int c = a + b;
```

1. Assign <int> 1 to a
2. Assign <int> 2 to b
3. call binary_add<int, int>(a, b)
4. Assign the result to c

2. Loops

-

```
# python code
a = 1
b = 2
c = a + b
```

1. Assign 1 to a

- **1a.** Set a->PyObject_HEAD->typecode to integer
- **1b.** Set a->val = 1

2. Assign 2 to b

- **2a.** Set b->PyObject_HEAD->typecode to integer
- **2b.** Set b->val = 2

3. call binary_add(a, b)

- **3a.** find typecode in a->PyObject_HEAD
- **3b.** a is an integer; value is a->val
- **3c.** find typecode in b->PyObject_HEAD
- **3d.** b is an integer; value is b->val
- **3e.** call binary_add<int, int>(a->val, b->val)
- **3f.** result of this is result, and is an integer.

4. Create a Python object c

- **4a.** set c->PyObject_HEAD->typecode to integer
- **4b.** set c->val to result

2. Loops

- https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/

- Compilers optimize during compilation process

  - repeated operations

  - caching

  - etc

- Interpreted codes need JIT or C extensions. This is possible, just an extra step in Python

2. Loops

- https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/

- Lists have similar advantages/disadvantages.

  - In C, you would use a buffered array
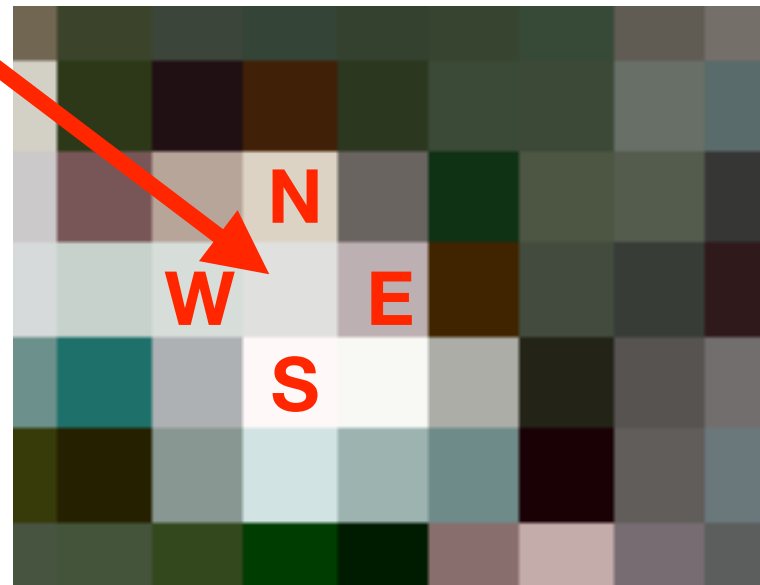
  - With bumpy you get this
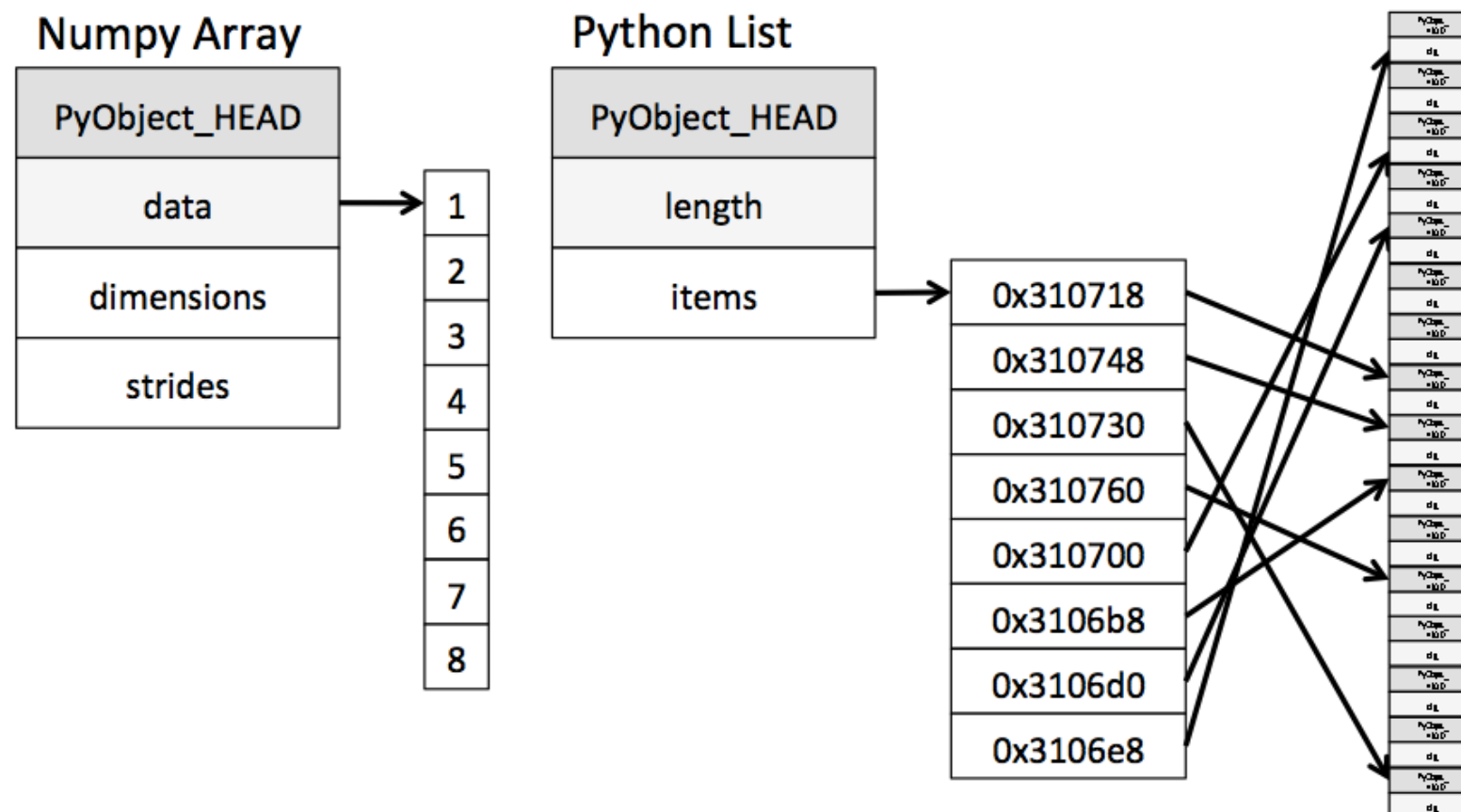
2. Loops

- Let's look at an example: average the pixels

2. Loops

- Let's look at an example: average the pixels

$$pixel \leftarrow \frac{1}{8} \left(4pixel + pixel_{north} + pixel_{south} + pixel_{west} + pixel_{east}\right)$$

3. How "fast" are arrays?

1D

$$v = \begin{bmatrix} v_0, v_1, \ldots, v_{n-1} \end{bmatrix}$$

$$w = \begin{bmatrix} w_0, w_1, \ldots, w_{n-1} \end{bmatrix}$$

2D

$$A = \begin{bmatrix} a_{00} & a_{01} & \ldots \\ \vdots & \ddots & \vdots \\ & & a_{n-1,n-1} \end{bmatrix}$$

$$B = \begin{bmatrix} b_{00} & b_{01} & \ldots \\ \vdots & \ddots & \vdots \\ & & b_{n-1,n-1} \end{bmatrix}$$

4. Limits of Precision

- What if we have a function and want to estimate it's derivative?

  - A simple test, but useful! We wouldn't need to know the explicit derivative.

$$f(x) \qquad\qquad f'(x) \ \textcolor{red}{?}$$