Intro to Coding with Python– Algorithms

Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (https://jcrouser.github.io/)

Plan for Today

- Sorting algorithms:
  - Insertion Sort
  - Selection Sort
  - Bubble Sort
- Comparing algorithms
  - on specific examples
  - more generally

5 volunteers, please!

Debrief

What **similarities / differences** did you notice between the three approaches?

Discussion

How do you know which algorithm is **better**?

1. Consider a specific example

[3, 0, 1, 8, 7, 2, 5, 4, 9, 6]

How many steps will **InsertionSort** take?

How many steps will **SelectionSort** take?

How many steps will **BubbleSort** take?

InsertionSort: visually

https://youtu.be/Q1JdRUh1_98?si=9bXFX0PwLqY_kxm5

SelectionSort: visually

https://youtu.be/Iccmrk2ZWoc?si=5d_bhl1055EOMU26

BubbleSort: visually

https://youtu.be/hahrx5WUeNI?si=qavZQ6vC_3Fewh8g

2. Now consider a different example

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

How many steps will **InsertionSort** take?

How many steps will **SelectionSort** take?

How many steps will **BubbleSort** take?

3. What about in **general**?

[???]

How many steps will **InsertionSort** take?

How many steps will **SelectionSort** take?

How many steps will **BubbleSort** take?

Algorithmic analysis

- **What we want**: a way to get a rough bound (limit) on how many steps an algorithm could take, *given input of a particular size*
  - If the list has 5 items?
  - What about 10?
  - What about `n`?
- **Often we care about**: "what's the **worst** that could happen?"
- **Mathematics** to the rescue!

Asymptotic ("big-O") notation

- We can **avoid details** when they don't matter, and they don't matter when input size (**n**) is big enough
- For **polynomials**: only the leading term matters
- Ignore **coefficients**, talk about **degree**: linear, quadratic

$y = 3x$	$y = 6x-2$	$y = 15x + 44$
$y = x^2$	$y = x^2-6x+9$	$y = 3x^2+4x$

3. What about in **general**?

[???]

How many steps will **InsertionSort** take?

How many steps will **SelectionSort** take?

How many steps will **BubbleSort** take?

InsertionSort

Given some list of comparable items:

For p in [0, ..., n-1]:

    Take the item in position p

    Insert it in the correct spot in positions 0 to p-1

SelectionSort

Given some list of comparable items:

- Find the smallest item.
- Swap it with the item in position 0.
- Repeat finding the next-smallest item, and swapping it into the correct position until the list is sorted.

BubbleSort

While the list is still unsorted:

For p in [0, ..., n-2]:

If item in position p is greater than item in position p+1:

Swap them

Takeaways

- Big-O notation gives us a language to talk about and compare algorithms **before** we implement them (smart!)
- InsertionSort, SelectionSort, and BubbleSort:
  - all the same in the **worst** case
  - different in the **best** case
  - what about the **average** case?
  - is there any algorithm that could do **better** in the worst case? (yes!)
- As your programs get more complex, it's helpful to think about the **algorithm** before you start coding