## CSCI 204: Data Structures & Algorithms

**Algorithm Analysis** 

Pretend we had a list with x different names. We build the following method to see if a name is in our list:

def is\_name\_in\_list(my\_list, search\_name):
 for item in my\_list:
 if item == search\_name: - search\_t return True return False

How do we measure the "speed" of a program? What do we need to know to determine how fast this will run?

### Why does this matter?

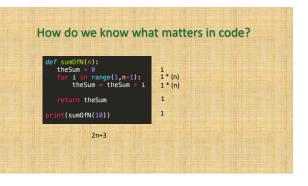
- Computers are so fast! But...
- Large Scale Data
- Google, Twitter, Facebook.. Big Data
- Limited Resources - phones, watches, wearable computing
- High Performance Environments - milliseconds matter

### Measure the work instead of timing

- If we actually measure time, e.g., using the Linux time command, we can't account for the speed differences among different computers.
- · Rather, we'd measure the steps an algorithm or a program will take when comparing them.
- Try a few examples with the time command ...

### **Big-O Notation**

- No need to count precise number of steps
- · Classify algorithms by order of magnitude
  - Execution time
  - Space requirements
- Big O gives us a rough upper bound
- · Goal is to give you intuition

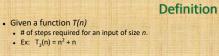


How do we know wha	Williams Brits for the Williams Brits for the
<pre>def sumOfN(n): theSum = 0 for i in range(1,n+1): theSum = theSum + i return theSum print(sumOfN(10))</pre>	<pre>def sumORN(n):     theSum = 0     for i in range(1,n=1):         theSum = theSum + i     doubleIt = theSum + 2     halveIt = theSum/2     theSum = halveIt     return theSum</pre>
2n+3	<pre>print(sumOfN(10))</pre>

	scribing Growth
f(n)	Name
1	Constant
log n	Logarithmic
n	Linear
n log n	Log Linear
n <sup>2</sup>	Quadratic
n <sup>3</sup>	Cubic
2"	Exponential
Let's Visualize It See for example	Health Health H

### Does it REALLY matter?

- Try out two examples
  - -time python bubblesort.py
  - -time python quicksort.py
- Try out a few more examples from mainRun.py which calls various operations in bigO.py



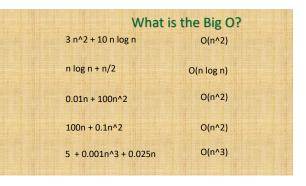
• Suppose there exist a function f(n) for all integers  $n \ge 0$  such that

 $T(n) \leq c f(n)$ 

for some constant c and for all large values of  $n \ge m$  (a constant).

We say function T(n) is on the order of f(n). In our above example, T(n) is on the order of  $n^2$ .





# Code Evaluation #1

def ex1( n ): count = 0 for i in range( n ): count += i return count

# Code Evaluation #2 def ex2(n): count = 0 for i in range(n): count += 1 for j in range(n): count += 1 return count

