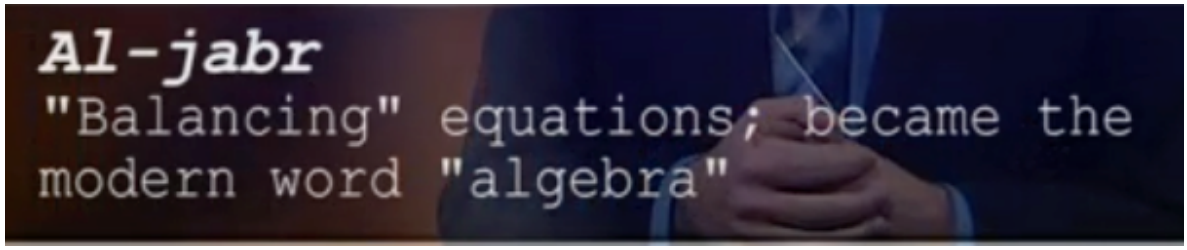
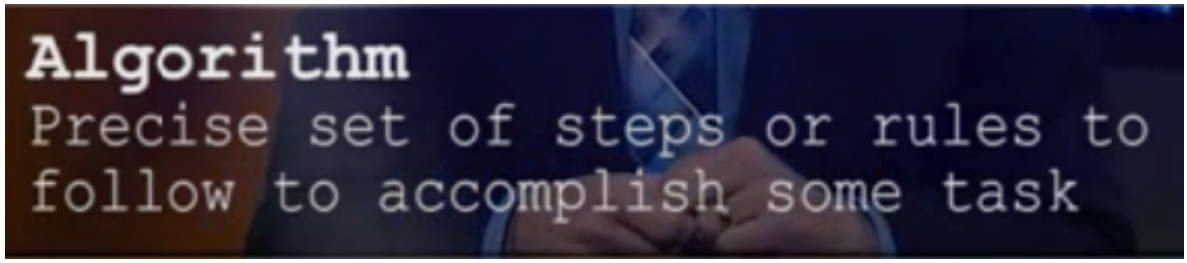
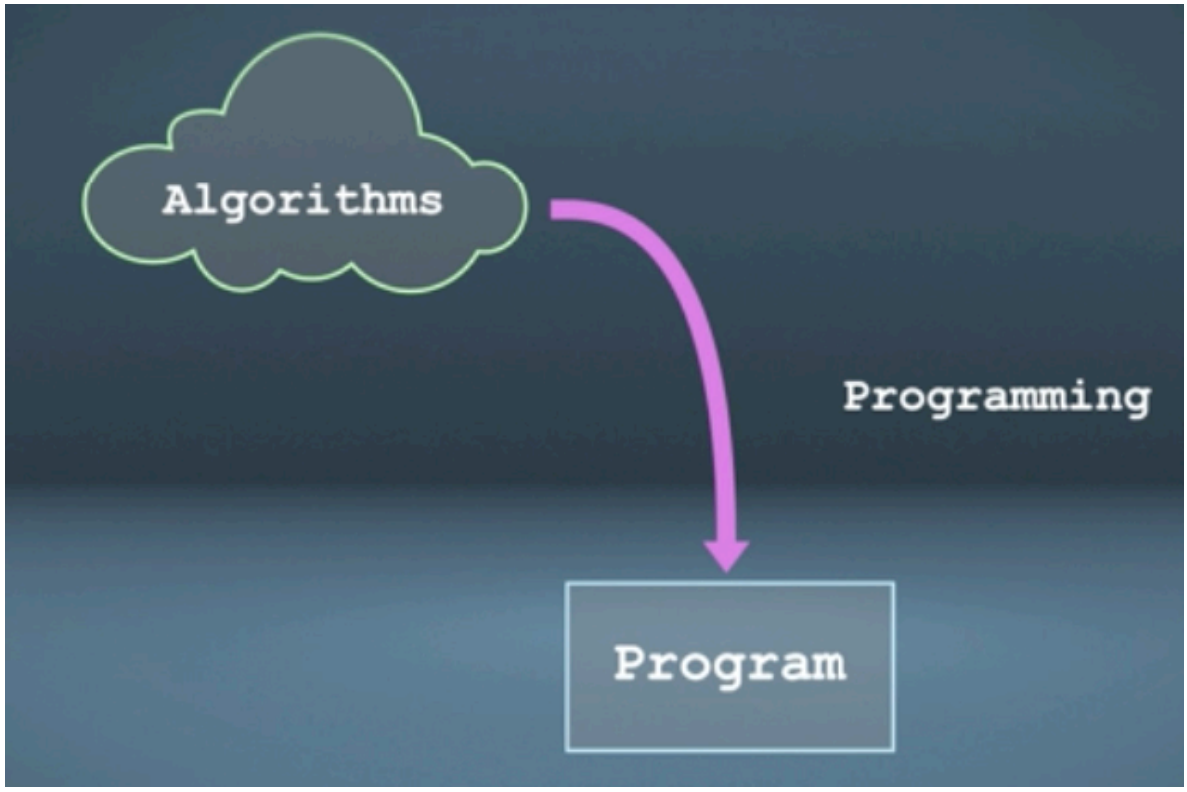
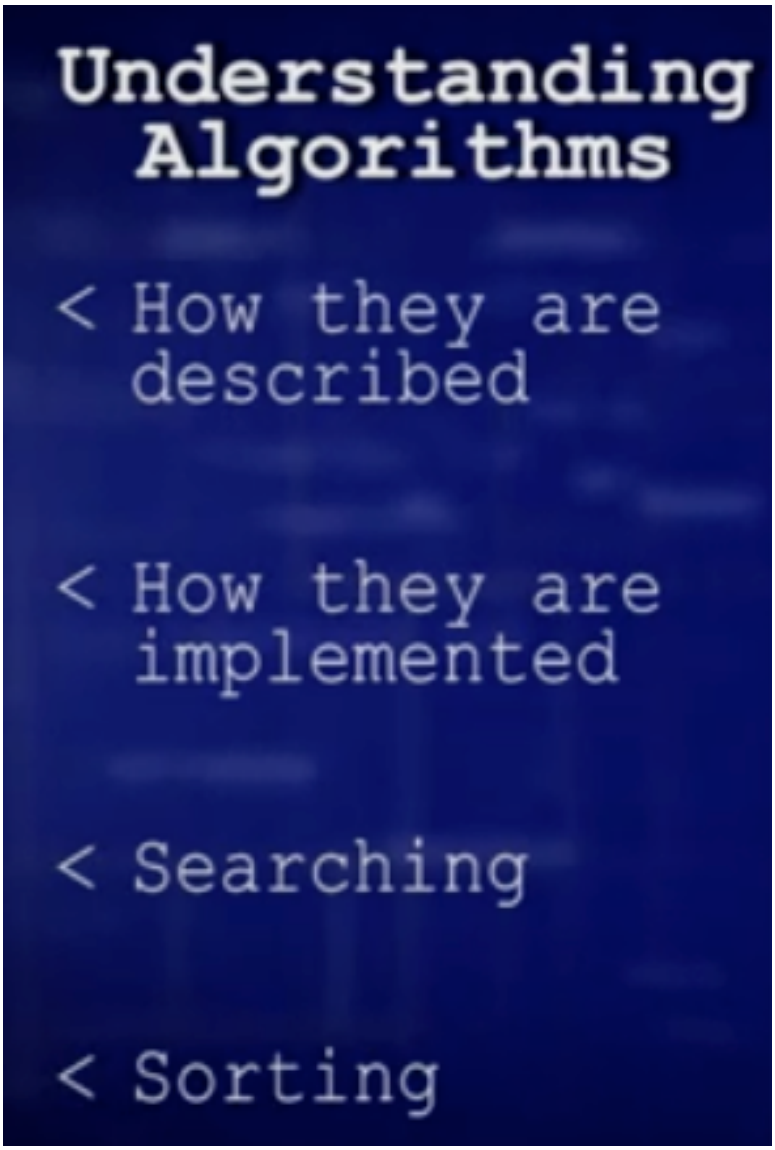
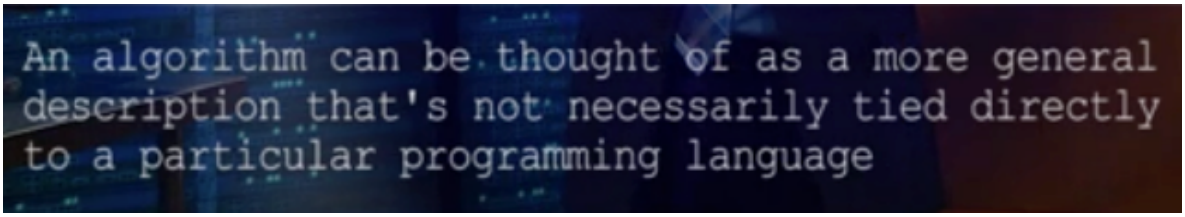
CSIes20
Algorithms: Searching and Sorting

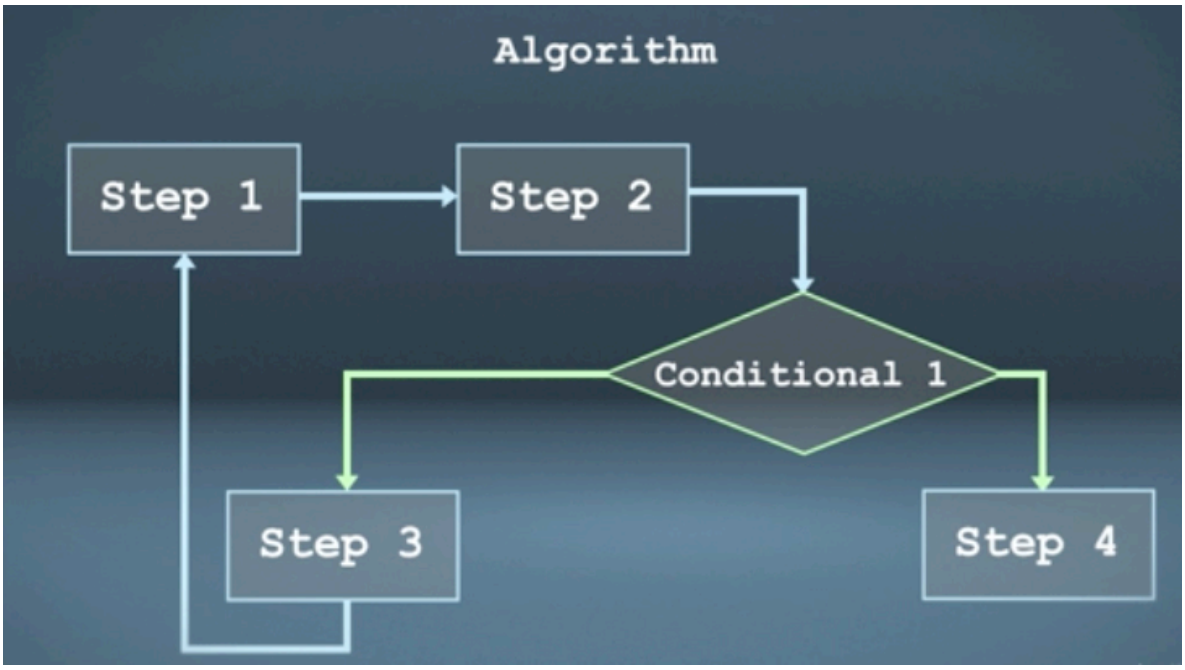


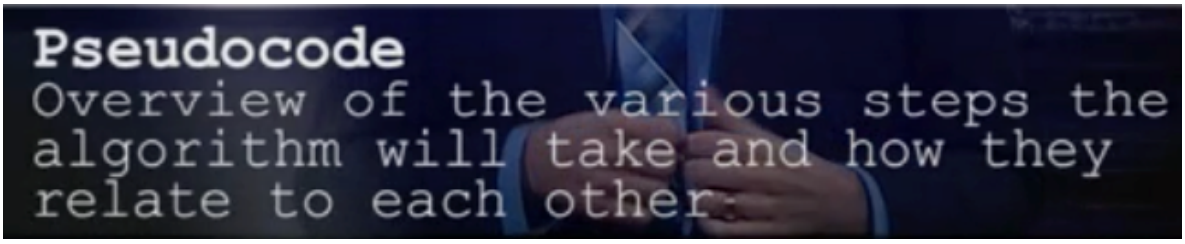




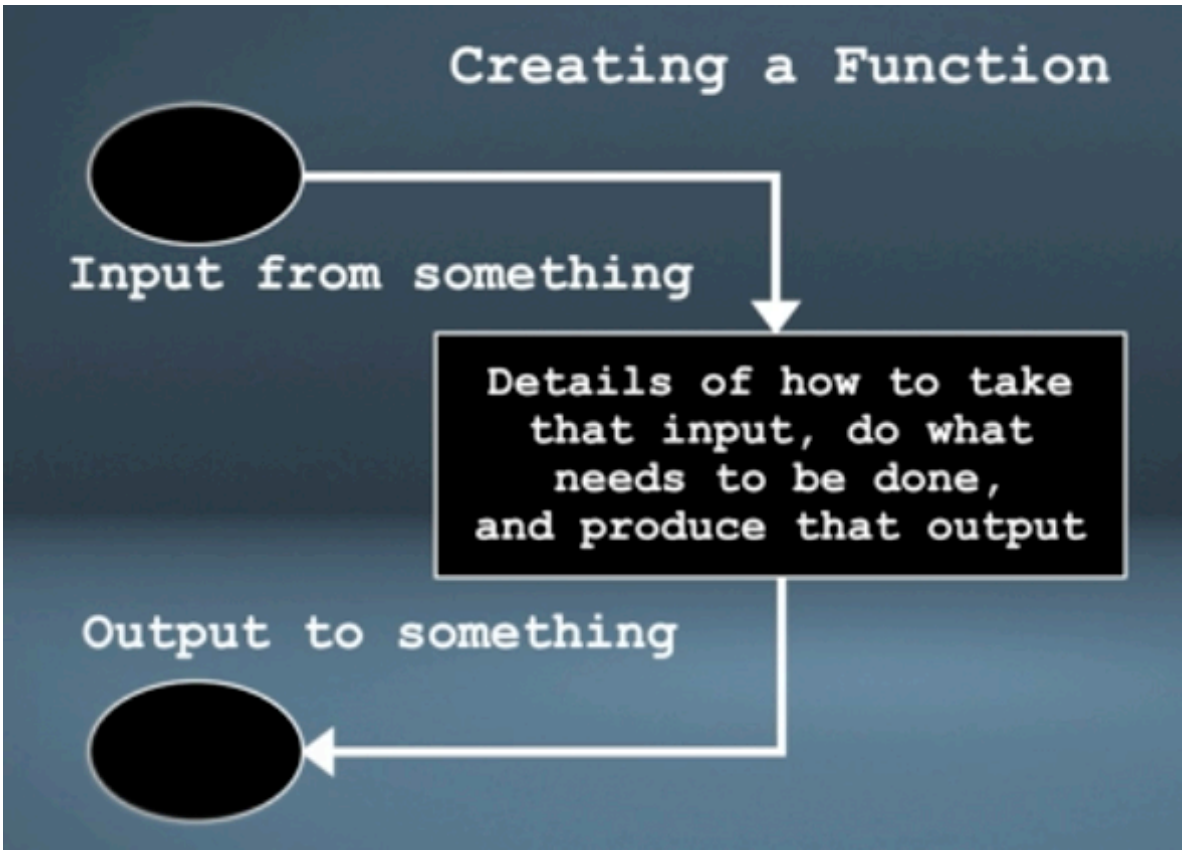

Understanding Algorithms
< How they are described
< How they are implemented
< Searching
< Sorting


An algorithm can be thought of as a more general description that's not necessarily tied directly to a particular programming language



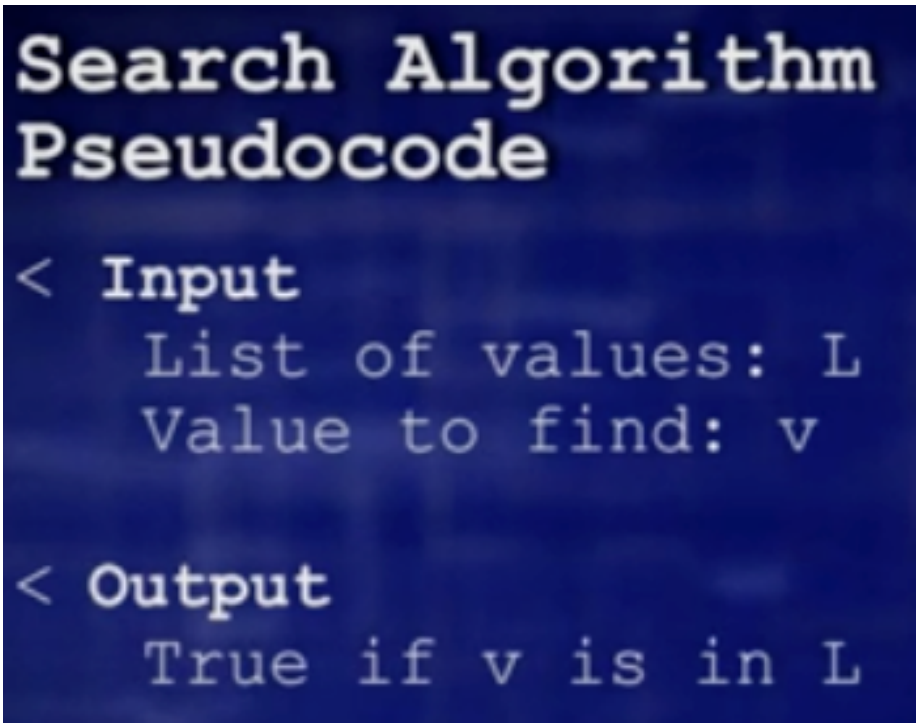


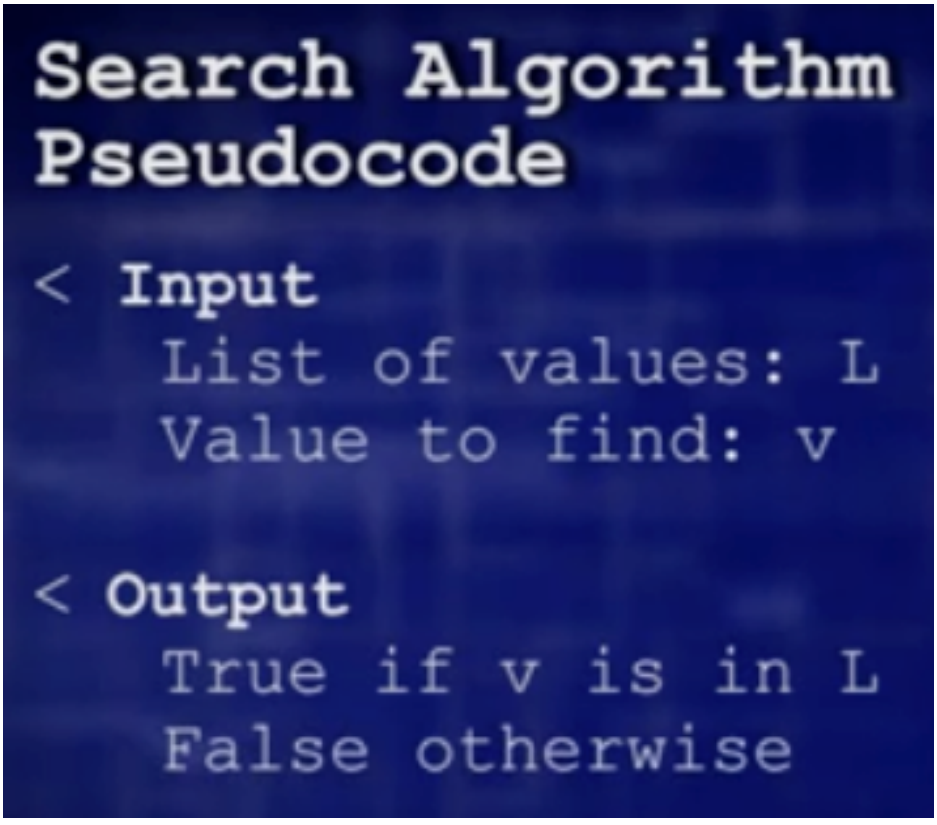

```
1  Step 1
2  Step 2
3  If (condition)
4    Step 3
5    Go to Step 1
6  Else
7    Step 4
```



Search Algorithm

- Try to find out whether a particular value is in a list or not
- The list is just a collection of values
- The search will return either True or False


Search Algorithm
Pseudocode
< Input
List of values: L
Value to find: v
< Output
True if v is in L


Search Algorithm
Pseudocode
< Input
List of values: L
Value to find: v
< Output
True if v is in L
False otherwise

Search Algorithm Pseudocode

< Let i be the index of the first element in the list

< While i is less than the size of the list:
- if element i of list L matches v, return True
- otherwise, increment i

< Return False

OUTPUT

```
1  def isIn(L, v):
2      i = 0
3      while (i<len(L)):
4          if L[i] == v:
5              return True
6          else:
7              i += 1
8      return False
9
10 favorite_foods = ['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
11
```

OUTPUT
True
False

```
1  def isIn(L, v):
2      i = 0
3      while (i<len(L)):
4          if L[i] == v:
5              return True
6          else:
7              i += 1
8      return False
9
10 favorite_foods = ['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
11 print(isIn(favorite_foods, 'gumbo'))
12 print(isIn(favorite_foods, 'coconut'))
```

OUTPUT
True
False
True
False

```
1  def isIn(L, v):
2      i = 0
3      while (i<len(L)):
4          if L[i] == v:
5              return True
6          else:
7              i += 1
8      return False
9
10 favorite_foods = ['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
11 print(isIn(favorite_foods, 'gumbo'))
12 print(isIn(favorite_foods, 'coconut'))
13 print ('gumbo' in favorite_foods)
14 print ('coconut' in favorite_foods)
```

< Input:
List of values IN SORTED ORDER: L
Value to find: v

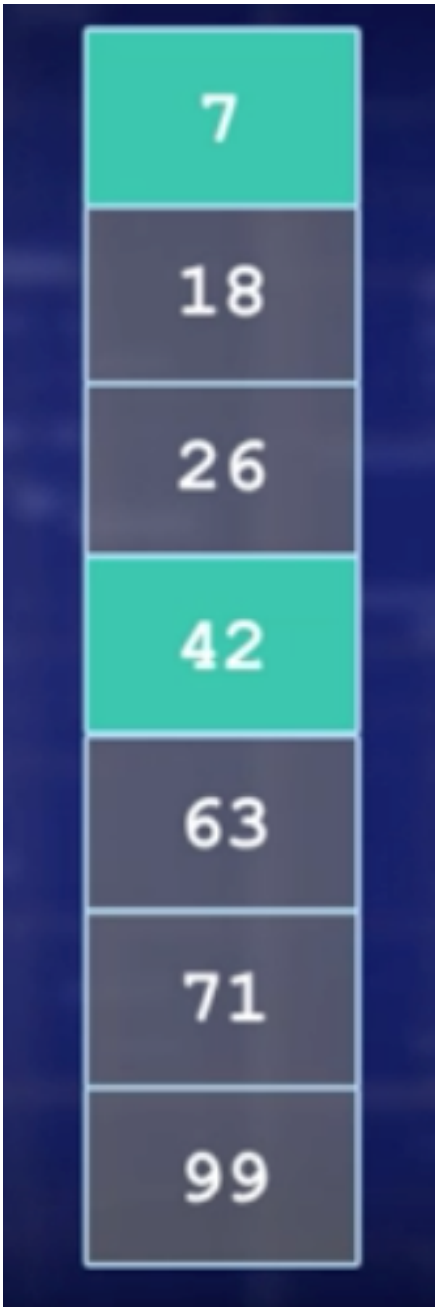
< Output:
True if v is in L, False otherwise

1. Set low = 0 and high = length of list - 1

2. If L[low] == v or L[high] == v, return True

3. While low < high-1 #Value is between L[low] and L[high]

```
< Input:
    List of values IN SORTED ORDER: L
    Value to find: v

< Output:
    True if v is in L, False otherwise


1. Set low = 0 and high = length of list - 1

2. If L[low] == v or L[high] == v, return True

3. While low < high-1                    #Value is between L[low] and L[high]
    A. midpoint = low+(high-low)/2       #Integer division
    B. If L[midpoint] == v, return True
```



```
< Input:
    List of values IN SORTED ORDER: L
    Value to find: v

< Output:
    True if v is in L, False otherwise


1. Set low = 0 and high = length of list - 1

2. If L[low] == v or L[high] == v, return True

3. While low < high-1                    #Value is between L[low] and L[high]
    A. midpoint = low+(high-low)/2       #Integer division
    B. If L[midpoint] == v, return True
    C. If L[midpoint] < v, set low = midpoint
    D. else set high = midpoint

4. return False
```

Binary search
Reducing the search range by a factor of two (at each iteration)

Linear search
Looking at each item, one after the other

```
OUTPUT
True
False
```

```
1  def binaryIn(L, v):
2      low = 0
3      high = len(L)-1
4      if L[low] == v or L[high] == v:
5          return True
6      while low < (high-1):
7          midpoint = low + (high-low) // 2
8          if L[midpoint] == v:
9              return True
10         elif L[midpoint] < v:
11             low = midpoint
12         else:
13             high = midpoint
14     return False
15 favorite_foods = ['barbeque', 'chicken and dumplings', 'gumbo', 'ice cream', 'pecan pie', 'pizza']
16 print(binaryIn(favorite_foods, 'gumbo'))
17 print(binaryIn(favorite_foods, 'coconut'))
18
```

OUTPUT
True
False

```
1  def binaryIn(L, v):
2      if len(L) < 1:
3          return False
4      low = 0
5      high = len(L)-1
6      if L[low] == v or L[high] == v:
7          return True
8      while low < (high-1):
9          midpoint = low + (high-low) // 2
10         if L[midpoint] == v:
11             return True
12         elif L[midpoint] < v:
13             low = midpoint
14         else:
15             high = midpoint
16     return False
17 favorite_foods = ['barbeque', 'chicken and dumplings', 'gumbo', 'ice cream', 'pecan pie', 'pizza']
18 print(binaryIn(favorite_foods, 'gumbo'))
19 print(binaryIn(favorite_foods, 'coconut'))
```

Testing Process

< What if we had just one element or just two?

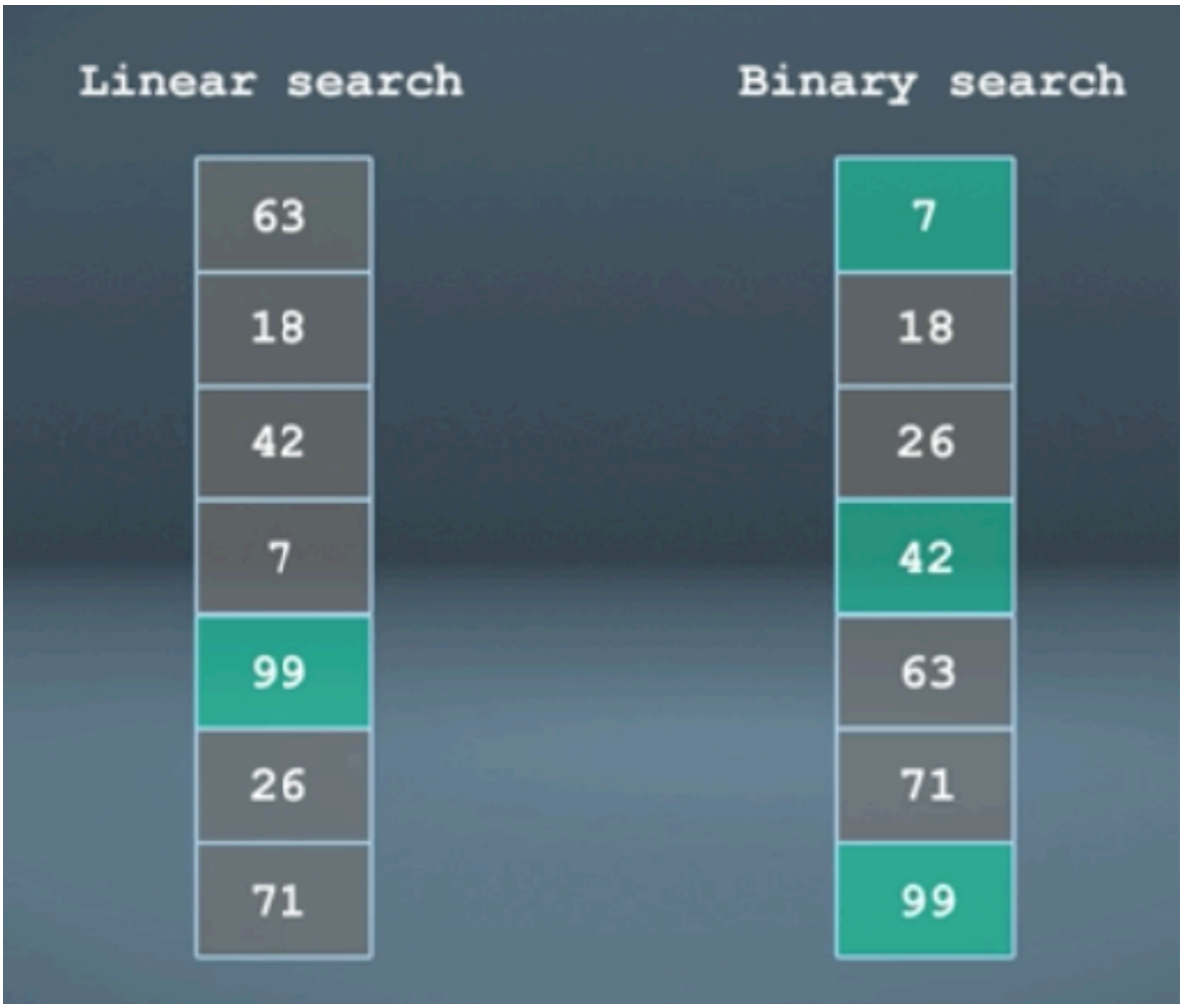
< What if the thing we were looking for was the very first one, or the very last one?

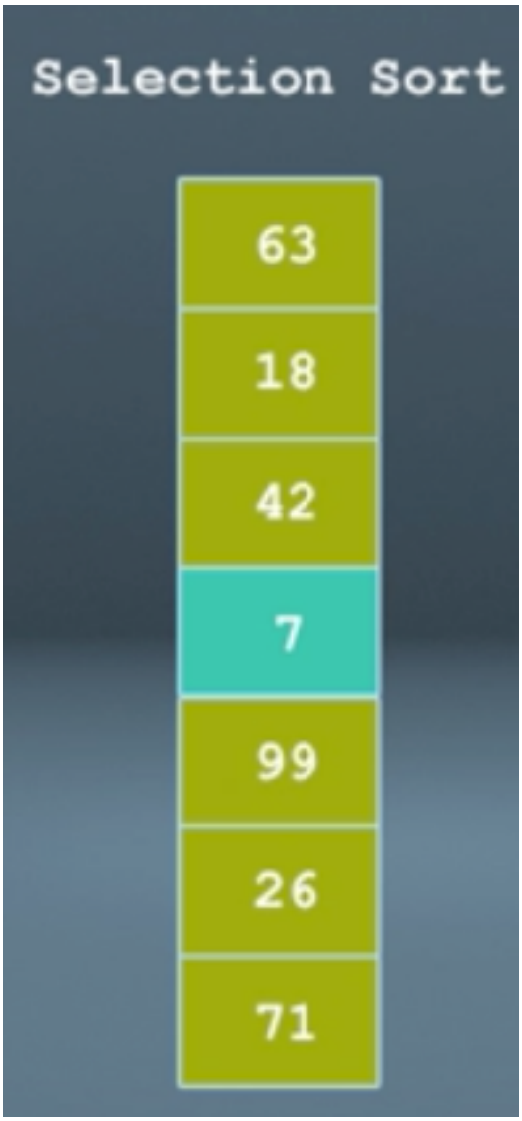
< What if it was a giant list?

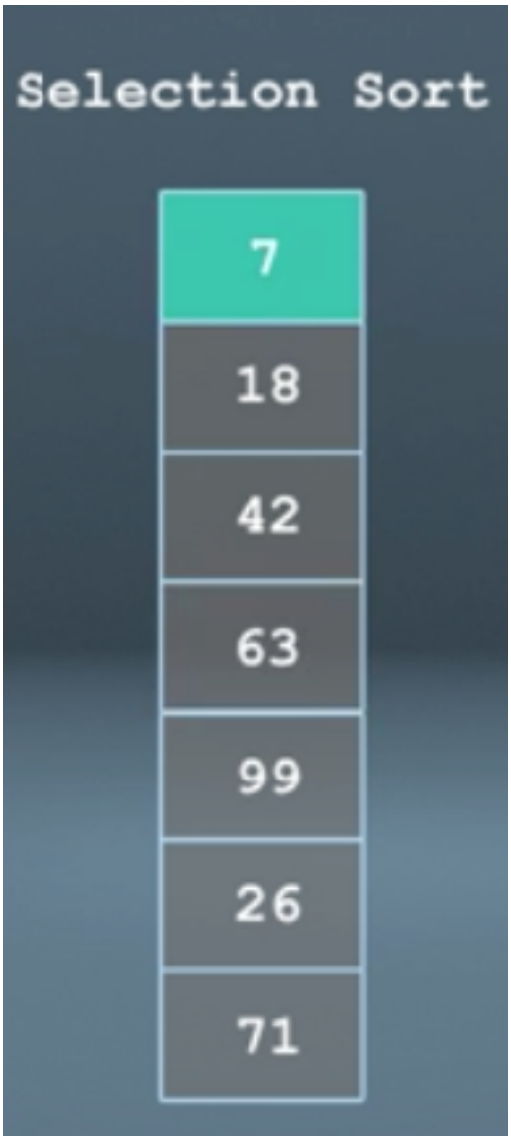
```
< Input:
    List of values IN SORTED ORDER: L
    Value to find: v

< Output:
    Index of entry that's equal to v, if v is in L
    -1 if v is not in L

1. Set low = 0 and high = length of list - 1

2. If L[low] == v, return low

3. if L[high] == v, return high

4. While low < high-1                    #Value is between L[low] and L[high]
    A. midpoint = low+(high-low)/2       #Integer division
    B. If L[midpoint] == v, return midpoint
    C. If L[midpoint] < v, set low = midpoint
    D. else set high = midpoint

5. return -1
```

```
OUTPUT
2
-1
```

```
1  def binaryIn(L, v):
2      if len(L) < 1:
3          return False
4      low = 0
5      high = len(L)-1
6      if L[low] == v:
7          return low
8      elif L[high] == v:
9          return high
10     while low < (high-1):
11         midpoint = low + (high-low) // 2
12         if L[midpoint] == v:
13             return midpoint
14         elif L[midpoint] < v:
15             low = midpoint
16         else:
17             high = midpoint
18     return -1
19 favorite_foods = ['barbeque', 'chicken and dumplings', 'gumbo', 'ice cream', 'pecan pie', 'pizza']
20 print(binaryIn(favorite_foods, 'gumbo'))
21 print(binaryIn(favorite_foods, 'coconut'))
```


Linear search
63
18
42
7
99
26
71
Binary search
7
18
26
42
63
71
99


Selection Sort
63
18
42
7
99
26
71



In-place sort

Directly sorting the elements of a given list

Out-of-place sort

Making a copy of a given list and then sorting the copy, leaving the original list unchanged

< Input: unsorted list L

< Output: L, with elements sorted smallest to largest

1. maxindex = length of L - 1

2. For i = 0 to maxindex
   a. Find the index with the smallest value between i and maxindex
   b. Swap that element with the element in index i

OUTPUT

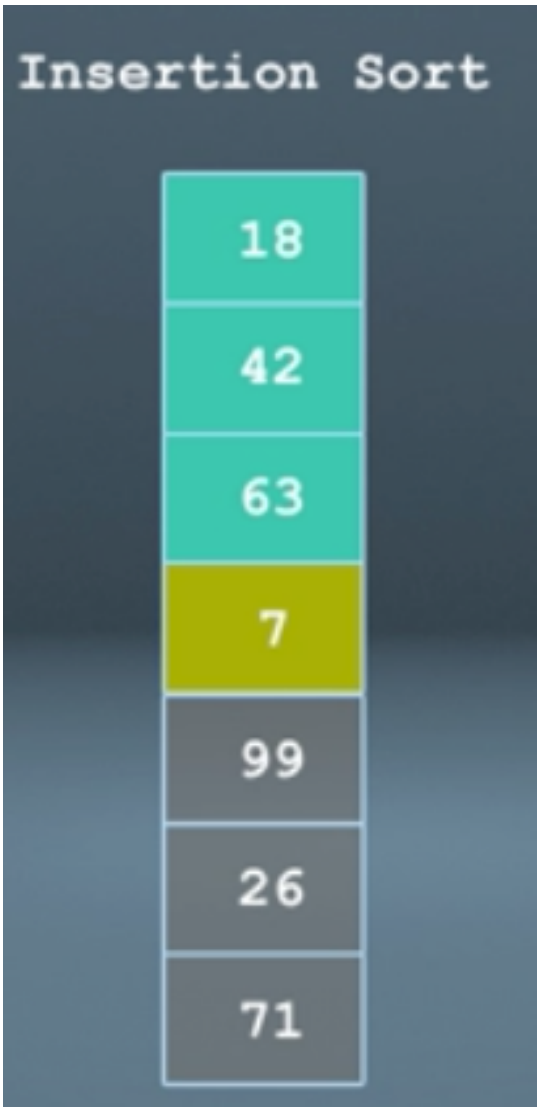
```
['barbeque', 'chicken and dumplings', 'gumbo', 'ice cream', 'pecan pie', 'pizza']
```

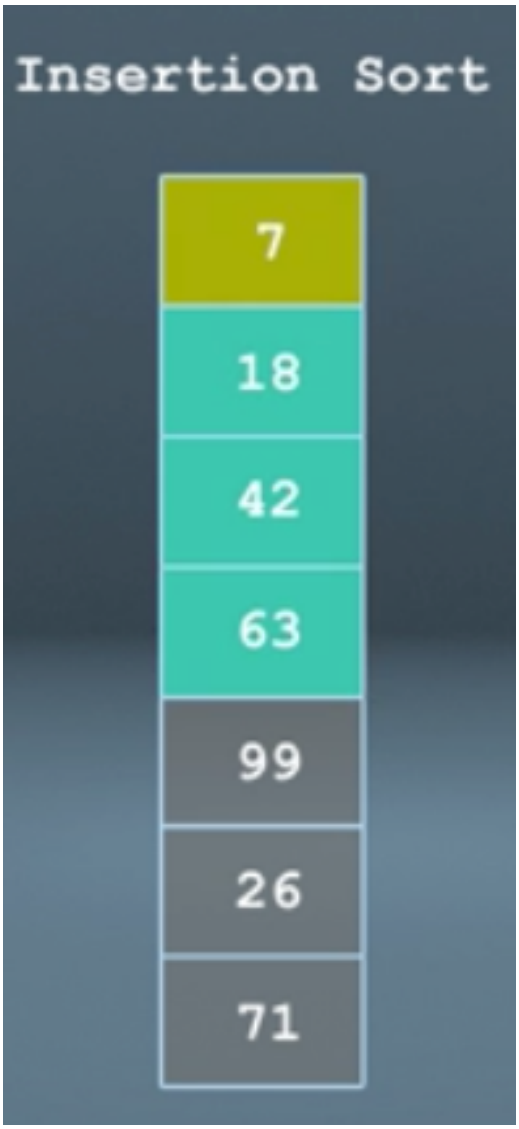
```
1  def selectionSort(L):
2      maxindex = len(L) - 1
3      for i in range(0,maxindex+1):
4          #find the smallest remaining
5          min_index = i
6          for j in range(i+1,maxindex+1):
7              if L[j] < L[min_index]:
8                  min_index = j
9          #swap that item
10         temp = L[i]
11         L[i] = L[min_index]
12         L[min_index] = temp
13 favorite_foods = ['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
14 selectionSort(favorite_foods)
15 print(favorite_foods)
```

Insertion sort

< For each iteration "n", sort the first "n" elements

< Each iteration add 1 to "n"


Insertion Sort
18
42
63
7
99
26
71



OUTPUT

```
['barbeque', 'chicken and dumplings', 'gumbo', 'ice cream', 'pecan pie', 'pizza']
```

```
def insertionSort(L):
    for i in range(0,len(L)):
        temp = L[i]
        j = i-1
        while (j >= 0) and (L[j] > temp):
            L[j+1] = L[j]
            j -= 1
        L[j+1] = temp
favorite_foods = ['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
insertionSort(favorite_foods)
print(favorite_foods)
```

OUTPUT

```
['barbeque', 'chicken and dumplings', 'gumbo', 'ice cream', 'pecan pie', 'pizza']
```

```
favorite_foods = ['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
favorite_foods.sort()      #this is an in-place sort
print(favorite_foods)
```

OUTPUT

```
favorite_foods = ['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
sorted_favorites = sorted(favorite_foods)     #this is an out-of-place sort
```

OUTPUT

```
['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
['barbeque', 'chicken and dumplings', 'gumbo', 'ice cream', 'pecan pie', 'pizza']
```

```
favorite_foods = ['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
sorted_favorites = sorted(favorite_foods)     #this is an out-of-place sort
print(favorite_foods)
print(sorted_favorites)
```

OUTPUT

```
['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
['pizza', 'pecan pie', 'ice cream', 'gumbo', 'chicken and dumplings', 'barbeque']
```

```
1  favorite_foods = ['pizza', 'barbeque', 'gumbo', 'chicken and dumplings', 'pecan pie', 'ice cream']
2  sorted_favorites = sorted(favorite_foods, reverse = True)
3  print(favorite_foods)
4  print(sorted_favorites)
```