UNIT 4A
Iteration: Searching

Goals of this Unit

• Study an iterative algorithm called linear search that finds the first occurrence of a target in a collection of data.
• Study an iterative algorithm called insertion sort that sorts a collection of data into non-decreasing order.
• Learn how these algorithm scale as the size of the collection grows.
• Express the amount of work each algorithm performs as a function of the amount of data being processed.
Searching

Built-in Search in Ruby

movies = ["up", "wall-e", "toy story", "monsters inc", "cars", "bugs life", "finding nemo", "the incredibles", "ratatouille"]
movies.index("cars") => 4
movies.index("shrek") => nil
movies.index("Up") => nil
movies.include?("wall-e") => true
movies.include?("toy") => false
A Little More about Strings

You can use relational operators to compare strings.

Comparisons are done character by character using ASCII codes.

"smithers" > "burns"  => true
"homer" < "marge"    => true
"homer" < "Marge"    => false
"clancy" > "clayton" => false
"bart" < "bartholomew"  => true
Containment

Design an algorithm that returns true if a list contains a desired “key”, or false otherwise.

A contains? method

```ruby
def contains?(list, key)
  index = 0
  while index < list.length do
    if list[index] == key then
      return true
    end
    index = index + 1
  end
  return false
end
```

What happens if we execute `return` before we reach the end of the method?
A contains? method – version 2

```ruby
def contains?(list, key)
  for item in list do
    if item == key then
      return true
    end
  end
  return false
end
```

A contains? method – version 3

```ruby
def contains?(list, key)
  list.each { |item|
    if item == key then
      return true
    end
  }
  return false
end
```
A contains? method – version 4

```ruby
def contains?(list, key)
    list.each { |x| return true if x == key }
    return false
end
```

Important note: You can use this method on keys of any type, as long as the key's type matches the type of the elements in the array.

---

Search

Design an algorithm that returns the index of the first occurrence of a key in a list if the key is present, or `nil` otherwise.
A search method

def search(list, key)
    index = 0
    while index < list.length do
        if list[index] == key then
            return index
        end
        index = index + 1
    end
    return nil
end

Sorry...

def search(list, key)
    for item in list do
        if item == key then
            return index
        end
    end
    return nil
end
def search(list, key)
    for item in list do
        if item == key then
            return list.index(key)
        end
    end
    return nil
end

What's undesirable about this?