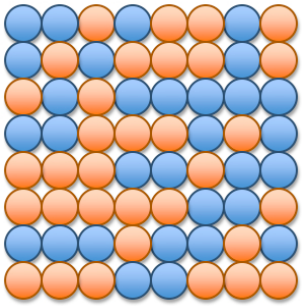


UNIT 3B
Algorithmic Thinking

Finding the maximum

How do we find the maximum in a sequence of integers shown to us one at a time?

```
183
```

What's the maximum?

Finding the maximum

Required: a non-empty *list* of integers.

1. Set *max_so_far* equal to the first number in the *list*.
2. For each number *n* in the *list*:
    a. If *n* is greater than *max_so_far*,
        then set *max_so_far* equal to *n*.

Return: *max_so_far* as the maximum of the *list*.

Representing Lists in Python

We will use a list to represent a collection of data values.

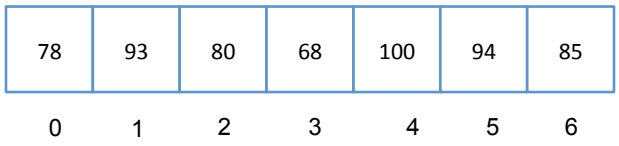
```
scores = [78, 93, 80, 68, 100, 94, 85]

colors = ['red', 'green', 'blue']
```

A list is an *ordered* sequence of values.

Some List Operations

```
>>>scores = [78, 93, 80, 68, 100,94,85]

>>>type(scores)
<class 'list'>

>>> len(scores)
7

>>>80 in scores
True

>>>scores + scores
[78, 93, 80, 68, 100, 94, 85, 78, 93, 80, 68,
  100, 94, 85, 94]
```


List Indices




```
>>>scores[0]
78
>>> scores[6]
85
>>> scores[7]
IndexError: list index out of range
>>>scores[1:3]
[93, 80]
>>>scores[1:7:2]
[93, 68, 94]
>>>scores.index(100)
4
>>>scores[-1]
85
```

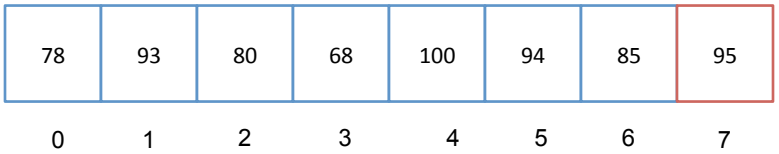
Operation	Result
`x in s`	`True` if an item of *s* is equal to *x*, else `False`
`x not in s`	`False` if an item of *s* is equal to *x*, else `True`
`s + t`	the concatenation of *s* and *t*
`s * n, n * s`	*n* shallow copies of *s* concatenated
`s[i]`	*i*th item of *s*, origin 0
`s[i:j]`	slice of *s* from *i* to *j*
`s[i:j:k]`	slice of *s* from *i* to *j* with step *k*
`len(s)`	length of *s*
`min(s)`	smallest item of *s*
`max(s)`	largest item of *s*
`s.index(i)`	index of the first occurence of *i* in *s*
`s.count(i)`	total number of occurences of *i* in *s*

source: docs.python.org

7

Lists Are Mutable

```
>>>scores.append(95)
>>>scores
[78, 93, 80, 68, 100, 94, 85, 95]
```



8

Operation	Result
`s[i] = x`	item *i* of *s* is replaced by *x*
`s[i:j] = t`	slice of *s* from *i* to *j* is replaced by the contents of the iterable *t*
`del s[i:j]`	same as `s[i:j] = []`
`s[i:j:k] = t`	the elements of `s[i:j:k]` are replaced by those of *t*
`del s[i:j:k]`	removes the elements of `s[i:j:k]` from the list
`s.append(x)`	same as `s[len(s):len(s)] = [x]`
`s.extend(x)`	same as `s[len(s):len(s)] = x`
`s.count(x)`	return number of *i*'s for which `s[i] == x`
`s.index(x[, i[, j]])`	return smallest *k* such that `s[k] == x` and `i <= k < j`
`s.insert(i, x)`	same as `s[i:i] = [x]`
`s.pop([i])`	same as `x = s[i]; del s[i]; return x`
`s.remove(x)`	same as `del s[s.index(x)]`
`s.reverse()`	reverses the items of *s* in place
`s.sort([key[, reverse]])`	sort the items of *s* in place

source: docs.python.org

9

Iterating over Lists

```
def print_colors(colors):
    for c in colors:
        print(c)
def print_colors2(colors):
    for i in range(0,len(colors)):
        print(colors[i])
def print_skip(colors):
    for i in range(0,len(colors),2):
        print(colors[i])
```

Python binds c to the first item in colors, then execute the statement in the loop body, binds c to the next item in the list colors etc.

10

Finding the max using Python

1. Set *max_so_far* equal to the first number in the *list*.
2. For each number *n* in the *list*:
    a. If *n* is greater than *max_so_far*,
        then set *max_so_far* equal to *n*.

Return: *max_so_far* as the maximum of the *list*.

```
def findmax(list):
    max_so_far = list[0]
    for i in range(1,len(list)):
        n = list[i]
        if n > max_so_far:
            max_so_far = n
    return max_so_far
```

11

Alternate Version

```
def findmax2(list):
    max_so_far = list[0]
    for item in list:
      if item > max_so_far:
          max_so_far = item
    return max_so_far
```

12

A 2000 year old algorithm (procedure) for generating a table of prime numbers.

2, 3, 5, 7, 11, 13, 17, 23, 29, 31, …

A positive integer is "prime" if it is not divisible by any smaller positive integers except 1.

Sieve of Eratosthenes

To make a list of every prime number less than n:

1. Create a list *numlist* with every integer from 2 to n, in order. (Assume n > 1.)
2. Create an empty list *primes*.
3. Copy the first number in *numlist* to the end of *primes*. (It must be prime. Why?)
4. Iterate over *numlist* to remove every number that is a multiple of the most recently discovered prime number.
5. Halt if *numlist* is empty. Otherwise, go back to step 3.

Sieve of Eratosthenes - Example

```
primes = []
numlist = [2,3,4,5,6,7,8,9,10,11,12,13,
  14,15,16,17,18,19,20,21,22,23,24,25]

primes = [2]
numlist = [3,5,7,9,11,13,15,17,19,21,23,25]

primes = [2,3]
numlist = [5,7,11,13,17,19,23,25]

primes = [2,3,5]
numlist = [7,11,13,17,19,23]       etc.
```


Lists: Two Special Cases

```
values = []
```

This is the empty list (a list with length 0).

```
values = []
for i in range(1,10):
    values.append(i)
```

This is the list with the first 9 positive integers in order.

Starting the algorithm in Python

To make a list of every prime number less than n:
1. Create a list *numlist* with every integer from 2 to n, in order. (Assume n > 1.)
2. Create an empty list *primes*.

```
def sieve(n):
    numlist = []
    for i in range(2,n+1):
        numlist.append(i)
    primes = []
```


Continuing...

3. Copy the first number in *numlist* to the end of *primes*. (It must be prime. Why?)

```
...
primes.append(numlist[0])
...
```

Does this operation remove the first element from *numlist*?

Removing multiples of a prime

4. Iterate over *numlist* to remove every number that is a multiple of the most recently discovered prime number.

Where is the most recently discovered prime added to the **`primes`** list?

`primes[len(primes)-1]` (i.e. last element)

Removing multiples of a prime

4. Iterate over *numlist* to remove every number that is a multiple of the most recently discovered prime number.

How do we determine whether a number **`x`** is a multiple of the most recent prime?

Use the modulo operator!

```
x % primes[len(primes)-1] == 0
```

Sifting: Removing Multiples of a Number - UPDATED

```
def sift(list,k):
# remove all multiples of k from list
    index = 0
    while index < len(list):
        if list[index] % k == 0:
            list.remove(list[index])
        else:
            index = index + 1
    return list
```


Removing multiples of a prime

4. Iterate over *numlist* to remove every number that is a multiple of the most recently discovered prime number.

```
lastprime = primes[len(primes)-1]
numlist = sift(numlist, lastprime)
```

Removing multiples of a prime

5. Halt if *numlist* is empty. Otherwise, go back to step 3.

We need to repeat steps 3 and 4:

```
primes.append(numlist[0])
lastprime = primes[len(primes)-1]
numlist = sift(numlist, lastprime)
```

until *numlist* is empty. How do we do this?

Repeating a task

Since we want to repeat a task, use a loop!
Since we don't know how many iterations are necessary, we will use a while loop.

```
while len(numlist) > 0 :
    primes.append(numlist[0])
    lastprime = primes[len(primes)-1]
    numlist = sift(numlist, lastprime)
```

Repeating a task

Since we want to repeat a task, use a loop!
Since we don't know how many iterations are necessary, we will use a while loop.

```
while len(numlist) >= 1 :
    primes.append(numlist[0])
    lastprime = primes[len(primes)-1]
    numlist = sift(numlist, lastprime)
```


Repeating a task

Since we want to repeat a task, use a loop!
Since we don't know how many iterations are necessary, we will use a while loop.

```
while len(numlist) != 0 :
    primes.append(numlist[0])
    lastprime = primes[len(primes)-1]
    numlist = sift(numlist, lastprime)
```

Final Algorithm in Python

```
def sift(list,k):
# remove all multiples of k from list
    index = 0
    while index < len(list):
        if list[index] % k == 0:
            list.remove(list[index])
        else:
            index = index + 1
    return list
```


Final Algorithm in Python (cont'd)

```
def sieve(n):
    numlist = []
    for i in range(2,n+1):
        numlist.append(i)
    primes = []
    while len(numlist) > 0:
        primes.append(numlist[0])
        lastprime = primes[len(primes)-1]
        numlist = sift(numlist, lastprime)
    return primes
```