§5.1-5.9: Arrays
Py 10.1-10.7: Lists

1 Oct 2007
CMPT14x
Dr. Sean Ho
Trinity Western University

Reminders on labs/homeworks

- Please upload your .py file separately from your lab write-up
  - So the TA can run your program
- Please put a raw_input() or something at the end of your program to pause before quitting
  - Otherwise the window will close and we can't see the output
- If the HW problem doesn't specify Python or M2, you may use either, but specify which language you're using

What's on today (§5.1-5.5, Py 10.1-10.7)

- Python lists vs. M2/C arrays
- Lists as function parameters
- Multidimensional arrays/lists
- Python-specific list operations
  - Membership (in)
  - Concatenate (+), repeat (*)
  - Delete (del), slice ([s:e])
  - Aliasing vs. copying lists

Array parameters in M2/C/etc.

- In statically-typed languages like M2, C, etc., the procedure declaration needs to specify that the parameter is an array, and the type of its elements:
  - M2:

    ```
    PROCEDURE Average(myList: ARRAY of REAL) : REAL;
    ```

  - C:

    ```
    float average(float* myList, unsigned int len) {
    ```

- In M2, HIGH(myList) gets the length
- In C, length is unknown (pass in separately)

Multidimensional arrays

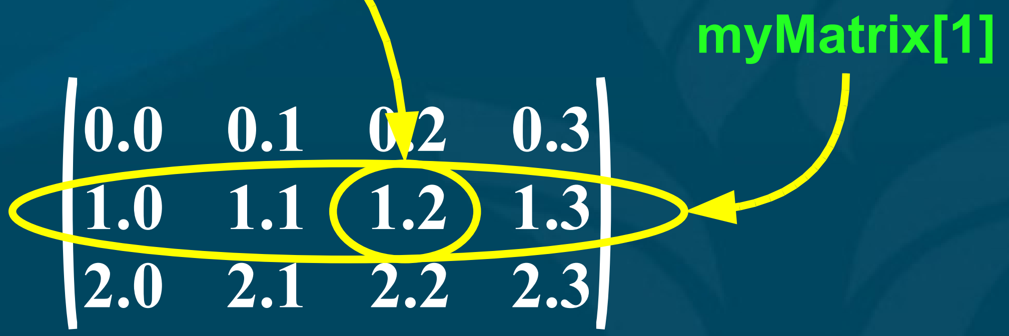
- Multidimensional arrays are simply arrays of arrays:

```
myMatrix = [ [0.0, 0.1, 0.2, 0.3],
             [1.0, 1.1, 1.2, 1.3],
             [2,0, 2.1, 2.2, 2.3] ]
```

- Accessing:

```
myMatrix[1][2] = 1.2
```

- Row-major convention:



myMatrix[1]

$$\begin{vmatrix} 0.0 & 0.1 & 0.2 & 0.3 \\ 1.0 & 1.1 & 1.2 & 1.3 \\ 2.0 & 2.1 & 2.2 & 2.3 \end{vmatrix}$$

Iterating through multidim arrays

```
def matrix_average(matrix):
    """Return the average value from the 2D matrix.
    Pre: matrix must be a non-empty 2D array of scalar values."""
    sum = 0
    num_entries = 0
    for row in range(len(matrix)):
        for col in range(len(matrix[row])):
            sum += matrix[row][col]
        num_entries += len(matrix[row])
    return sum / num_entries
```

- What if rows are not all equal length?

List operations (Python specific)

```
myApples = [ "Fuji", "Gala", "Golden Delicious" ]
```

- Test for list membership:

```
if "Fuji" in myApples:                    # True
```

- Concatenate:

```
[ 'a', 'b', 'c' ] + [ 'd', 'e' ]
```

- Repeat:

```
[ 'a', 'b', 'c' ] * 2
```

- Modify list entries (mutable):

```
myApples[1] = "Braeburn"
```

- Convert a string to a list of characters:

```
list("Hello World!")              # ['H', 'e', 'l', 'l', 'o', ...]
```

More list operations

- Delete an element of the list:

```
del myApples[1]        # [ "Fuji", "Golden Delicious" ]
```

- List slice (start:end):

```
myApples[0:1]      # [ "Fuji", "Gala" ]
```

- Assignment is aliasing:

```
yourApples = myApples        # points to same array
```

- Use a whole-list slice to copy a list:

```
yourApples = myApples[:]
    # [:] is shorthand for [0:-1] or [0:len(myApples)-1]
```

Sieve of Eratosthenes

- Problem: list all the prime numbers between 2 and some given big number.
  - You had a homework that was similar: test if a given number is prime, and list its factors
  - How did you solve that?
    - Procedure is_prime() (pseudocode):

```
Iterate for factor in 2 .. sqrt(n):
    If (n % factor == 0), then
        We've found a factor!
```

- But this is wasteful: really only need to test prime numbers for potential factors

Listing all primes

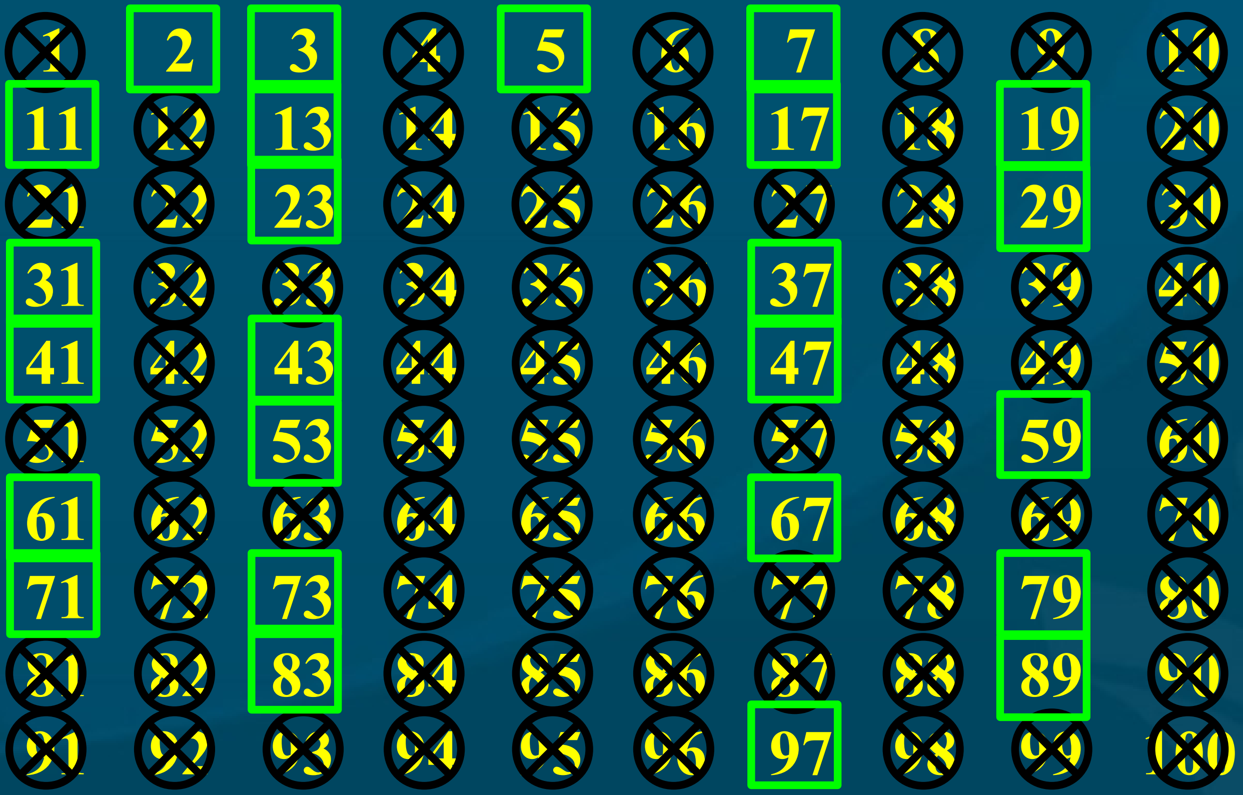
- We could tackle this problem by repeatedly calling is_prime() on every number in turn:

```
for num in range(2, max):
    if is_prime(num) ...
```

- But this could be really slow if max is big
- Is there a smarter way to eliminate non-prime (composite) numbers?

Sieve of Eratosthenes

- The sieve works by a process of elimination: we eliminate all the non-primes by turn:



~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	~~49~~	~~50~~
~~51~~	~~52~~	53	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	~~65~~	~~66~~	67	~~68~~	~~69~~	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	~~77~~	~~78~~	79	~~80~~
~~81~~	~~82~~	83	~~84~~	~~85~~	~~86~~	~~87~~	~~88~~	89	~~90~~
~~91~~	~~92~~	~~93~~	~~94~~	~~95~~	~~96~~	97	~~98~~	~~99~~	~~100~~

Prime sieve: pseudocode

1) Create an array of booleans and set them all to true at first. (true = prime)
2) Set array element 1 to false. Now 2 is prime.
3) Set the values whose index in the array is a multiple of the last prime found to false.
4) The next index where the array holds the value true is the next prime.
5) Repeat steps 3 and 4 until the last prime found is greater than the square root of the largest number in the array.

Prime sieve: Python code

```
"""Find all primes up to a given number, using Eratosthenes'
   prime sieve."""
import math                               # sqrt
size = input("Find all primes up to: ")

# Initialize: all numbers except 0, 1 are prime
primeFlags = range(size+1)                    # so pF[size] exists
for num in range(size+1):
    primeFlags[num] = True

primeFlags[0] = False
primeFlags[1] = False
```

Prime sieve: Python code (p.2)

```
# Computation: eliminate all non-primes
for num in range(2, int(math.sqrt(size))+1):
    if primeFlags[num]:             # got a prime
        # Eliminate its multiples
        for multiple in range(num**2, size+1, num):
            primeFlags[multiple] = False

# Output
print "Your primes, sir/madam:",
for num in range(2, size+1):
    if primeFlags[num]:
        print num,
```

http://twu.seanho.com/python/primesieve.py

Summary of today (§5.1-5.5, Py 10.1-10.7)

- Python lists vs. M2/C arrays
- Lists as function parameters
- Multidimensional arrays/lists
- Python-specific list operations
  - Membership (in)
  - Concatenate (+), repeat (*)
  - Delete (del), slice ([s:e])
  - Aliasing vs. copying lists

TODO

- Lab 03 due Wed:
  - M2 ch4 # (23 / 27 / 36)
- Read through M2 ch5 and Py ch7, plus Py ch10
- Midterm ch1-5 this Fri 5Oct