Vectors, Lists, Arrays, and all That

Some Inconsistent Terminology

- Vector
  - Mathematically
    - A geometrical object with a magnitude and a length
  - Computer Science and some programming languages (C++, Matlab)
    - A homogenous (all same type) ordered data structure with a variable length
  - Other languages (Fortran, Python)
    - A homogenous (all same type) ordered data structure with a fixed length; a one-dimensional array.

More Terminology

- List
  - Computer Science
    - "List" usually means "linked list." An ordered but *not* indexed list of arbitrary types
  - Some computer languages (e.g. Python)
    - A list is an ordered (indexed), inhomogeneous data structure. Like a "vector" in Matlab but elements need not be the same type.
  - Henceforth we will use the Python terminology for lists/arrays.

Python Lists

- Lists are ordered collections of objects. Each element of the list can be of any type. Elements can be referenced by an *index*.
- Lists are dynamically sized.
- Items can be appended with the append *method*.

```
myL=[]
myL.append("First")
```

Sublists

- subL=L[1:3]
- This is elements 1 and 2. Don't forget how it works in Python!
  - Numbering starts at 0 so these are the second and third elements
- Lists are *mutable* so you can change elements

```
myL=[1,2,3]
myL[1]=4
print myL
```

List Operations

- Slice
  - L2=L1[0]; L3=L1[1:4]
- Concatenate
  - L4=[1, 2, 3]+[4,5,6]
- Append
  - L1.append(“Graham”)
- Extend
  - L1.extend([“Graham”,”Michael”])
- Shorten
  - del L2[3]
- Length
  - LofL1=len(L1)

Some Useful Built-in Functions

- `reduce(func, S)`
  Successively applies a function of two variables to sequence S and produces a single result. E.g.
  - `L=reduce(sum, a)`

  sums all the elements of a, when sum is defined as x+y.
- `map(func, S)`

Applies the function to each element of S and returns a new list.

  - `L=map(square,S)`
  - Or
  - `L=map(lambda:x=x**2, S)`
- `filter(func, S)`

Applies Boolean function to each element of S, returning True or False, returns a new sequence consisting of all elements of S that are True.

  - `L=filter(lambda x: x>0, S)`

List Comprehension
New Lists from Old

- A list comprehension is a concise way to create a new list. It is powerful but can be confusing.
- Syntax

  expression for *var* in *list* if *condition*

  The *if* is optional.
- Examples

  ```
  x**2 for x in vector
  sqrt(x) for x in vector if x > 0
  ```
- Usually we enclose the comprehension in square brackets

  ```
  v=[sqrt(x) for x in vector if x>0]
  ```

Arrays

- Arrays are ordered structures of fixed size. Each element can be referenced by its index.
- Arrays are available in Python via the NumPy package.

Python Arrays

- Python arrays are an add-on via NumPy.

```
import numpy
A=numpy.array ([1, 0, 0, 0])
l=len(A)
```

Array Elements

- Each element can be addressed by its index
- Python

  `A[3]`

  – Starts at 0 by default

Subarrays

- Python

```
import numpy
A=numpy.zeros(100)
B=A[0:11]
```

Array Operations

- In Fortran and NumPy the mathematical functions are *overloaded* to accept array arguments. They operate on the array(s) *elementwise*. Examples:

```
T=numpy.ones(4)
A=3.0*T+numpy.ones_like(T)
I=numpy.array([1,0,0,0])
A=math.pi*I
B=sin(A)
C=A/B  (remember: elementwise)
```

Important Fact to Remember

- You can optimize your programs if you can use NumPy or otherwise avoid loops
  - In general, list comprehensions are faster than loops, as are built-in list functions like map and reduce
  - NumPy arrays are *much* faster than lists
    - But the size is fixed once the array is initialized
    - All operations are elementwise. There is a type *matrix* that is like a two-dimensional array but has some operations defined differently