

**CSE 111 Bio:
Program Design I
Lecture 19:
genetics**

Tanya Berger-Wolf (CS) & Boris Igić (Bio)
University of Illinois, Chicago
November 2, 2016

append, extend etc.

```
>>> L = [6, 3]
>>> L.append([9,11])
>>> L
[6, 3, [9, 11]]

>>> L = [6,3]
>>> L = L + [9,11])
>>> L
[6, 3, 9, 11]

>>> L = [6,3]
>>> L.extend([9,11])
>>> L
[6, 3, 9, 11]
```

```
>>> uniquify([9,9,7,5,3,5,7])
[9, 3, 5, 7]

The in syntax might be useful:
>>> 9 in [4,2,7]
False

def uniquify(L):
    '''Returns a list of the unique elements from L.'''
```

Worksheet

```
>>> uniquify([9,9,7,5,3,5,7])
[9, 3, 5, 7]

def uniquify(L):
    '''Returns a list of the unique elements from L.'''
    uniqL=[]
    for i in range(len(L)):
        if L[i] not in L[(i+1):]:
            uniqL.append(L[i])
    return uniqL
```

Worksheet

If you have time after uniquify...

Write a function to sort a list of numbers in ascending order using the selection sort algorithm. Selection sort works like this. First it scans over the list and finds the smallest element in it. Then it puts this in the zero position of the list by swapping it with whatever was already there. Next, it scans the list again, but this time it starts in the 1 position (skipping the 0 position). Again it finds the smallest element and swaps it into the 1 position. It proceeds like this until the whole list has been sorted. Try to do this without using any python built-ins such as min!

```
[9, 6, 8, 5, 7, 4]
[4, 6, 8, 5, 7, 9]
[4, 5, 8, 6, 7, 9]
[4, 5, 6, 8, 7, 9]
[4, 5, 6, 7, 8, 9]
[4, 5, 6, 7, 8, 9]
[4, 5, 6, 7, 8, 9]
```


```
def selectionSort(L):
    '''Sort list L in ascending order.'''
```

Variables inside and outside of functions

```
def func1():
    '''A snazzy demo function.'''
    i=42
    func2()
    return i

def func2():
    '''An even snazzier demo function.'''
    i=7
    print "I own", i , "Ladas."
```

>>> func1()
???

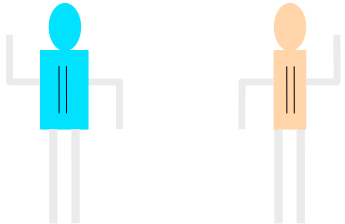


- A. I own 42 Ladas
- B. I own 7 Ladas
- C. I own 42 Ladas
I own 7 Ladas
- D. I own 7 Ladas
42
- E. I own 42 Ladas
7

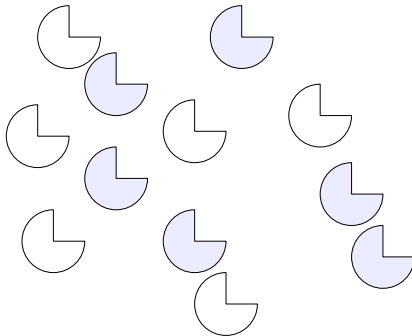
Demo

http://upload.wikimedia.org/wikipedia/commons/thumb/c/cb/Lada_1200.jpg/420px-Lada_1200.jpg

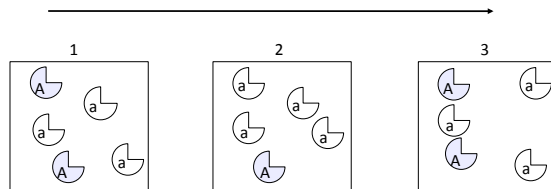
A simulation of variation in populations



Populations and alleles



A simple model of allele frequency changes over time



- Two alleles
- Fixed population size
- Haploid: have one copy of each sequence
- Asexual reproduction
- Make next generation by sampling with replacement

Representing a population on a computer

```
>>>popList=["A","A","a","A","a"]
```

We can sample using the random module

```
>>> import random
>>> random.choice(popList)
"A"
>>> popList
["A","A","a","A","a"] # Didn't change the original list
```

```
import random

def createInitialPop(popSize, pA):
    """Create starting pop given popSize and proportion of
    A's."""
    num0=int(round(pA*popSize))
    num1=popSize-num0
    popL=["A"]*num0+["a"]*num1
    return popL

def nextGen(popL):
    """Given the population of the current generation, obtains
    next generation by sampling with replacement."""
```