Python Lists & Loops

Functions and Parameter Passing

```
def foo(x,L):
        x=20
        L[2]=12
a=9
B=[1,2,3]
foo(a,B)
print(a)
9
print(B)
[1,2,12]
```

Parameter Passing: Pass means to provide an argument to a function

- Function arguments and variables created inside a function belongs to the *local scope* of that function, and can only be used inside that function

- **Python passes arguments by assignment:**
  - **so when we call foo(a,B) it has the effect**
    - **x=a, L=B — local variables get assigned the passed objects**

Python Assignment Statement

- How does assignment work?
  - L=[1,2,3] — a list object is created and the name L is bound to it.
  - B=L — no copy of L is made. The name B is also associated with same object.
  - B.pop() will cause the underlying (common) object to change and reduce in length by 1.
  - same with B.append(x) or B[1]=12
  - what if we do B=[4,3,7]
    - a new list object consisting of 4,3,7 is created and
    - B is disassociated from the old object and associated with this new object.
    - L still refers to old object. This assignment does not change the old object.

```
>>> L=[1,2,3]
>>> B=L
>>> B.pop()
3
>>> L
[1, 2]
>>> L.append(6)
>>> B
[1, 2, 6]
>>> B=[3,4,7]
>>> L
[1, 2, 6]
>>>
```

Functions and Parameter Passing

```
def foo(x,L):
        x=20
        L[2]=12
a=9
B=[1,2,3]
foo(a,B)
print(a)
9
print(B)
[1,2,12]
```

Selection Sort - correctness

```
def SelSort(A,left,right):
    for i in range(left, right):
        min=i
        for k in range(i+1,right):
            #INV that A[min] is the minimum of A[i:k]
            if A[k] < A[min]:
                min=k
        # Finally A[min] is the minimum of A[i:right]
        A[i],A[min]=A[min],A[i]
```

Selection Sort - correctness

```
def SelSort(A,left,right):

    for i in range(left, right):

        # INV  left<=i<right  A[left:i]  is a sorted list in ascending order and

        #           each element in A[left:1] is less (or =) any element in remaining list A[i:right])

        min=i

        for k in range(i+1,right):

                        if A[k] < A[min]:

                min=k

         A[i],A[min]=A[min],A[i]
```

Slection Sort : Complexity

- Space?
  - O(1) constant space beyond the space for original list
- Time?
  - finding minimum is linear so n + (n-1) + (n-2) …+1 = $O(n^2)$

Insertion Sort

— Like Selection Sort reuse original list divided into a sorted part and a part that is unsorted
— At each step you take one item from unsorted part and insert it into the sorted part
— This works best if you keep the sorted part on the right of the list as below (sorted part in red)

Initial List [5,2,1,6,4]

[5,2,1,6,**4**] Initially i=1 (A[len-i:] is a singleton sorted list) item to be inserted next 6

[5,2,1,**4,6**] i=2 (A[len-i:] sorted) item to be inserted next A[len-i-1] i.e 4

[5,2,**1,4,6**] i=3 item inserted 1

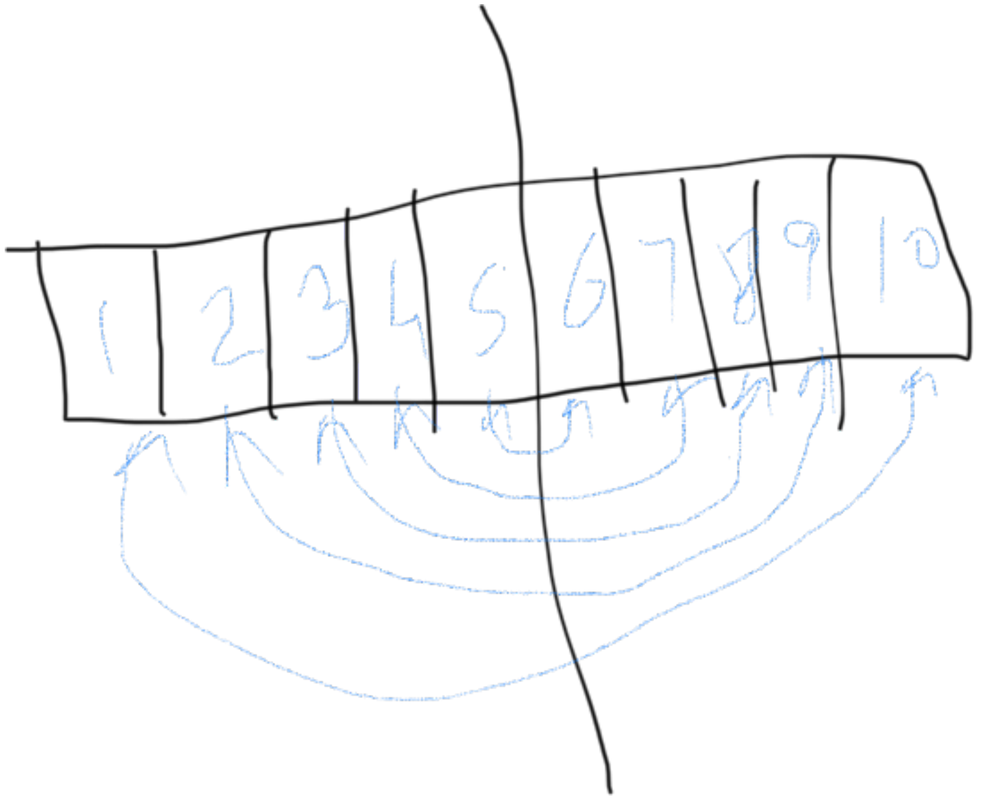
[5,**1**,**2,4,6**] i=4 item inserted 2

[**1**,**2,4,5,6**] i=5 item inserted 5 Finally A[len-len:]=A[0:]=A is sorted

Reverse contents of a list - in-place

- simple reverse

```
def rev(L):
    A=[]
    for i in range(len(L)-1,-1,-1):
        A.append(L[i])
    return A
def new_rev(A):
    n=len(A)
    for i in range(n//2):
            A[i],A[n-i-1]=A[n-i-1],A[i]
```



Remove Duplicates in a sorted array

- A=[1,2,2,3,4,5,5,6,6,6,7,22,22,59]
- return [1,2,3,4,5,6,7,22,59]

```
def clean(A):
    B=[0:1]
    for i in range(1,len(A)):
        if A[i] != A[i-1]:
            B.append(A[i])
    return B
```

Will give Error for A=[]

- Correctness?
- Complexity?
-