Seven Test-Taking Tips

Avoid losing points for no reason whatsoever….

1. Read the Problem

```
Write some procedure modifies x and does
not return a value.
```

```
def MySolution(x):
    blah
    blah
    return x
```

Just ask yourself: Does my solution live up to the specification?

2. Use Small Examples

```
s = 'abcdefghijklmnopqrstuvwxyz'
for i in range(26):
    for j in range(i+1,26):
        for k in range(j+1,26):
            print s[i]+s[j]+s[k]
```

```
s = 'abcde'
for i in range(5):
    for j in range(i+1,5):
        for k in range(j+1,5):
            print s[i]+s[j]+s[k]
```

3. Hand-Execute your Solution on a Small Example

```
m = len(x)/2
for k in m:
    x[k] = x[2*k]
    x[k+m] = x[2*k+1]
```

A good way to catch overwriting mistakes

4. Properly Recall What You've Done

No

"This question is just like Assignment X so I will repeat that solution without thinking."

Yes

" This question reminds me of Assignment X and so some of the ideas I used there may be applicable."

5. Watch for Subscript Out Of Bounds

```
P some list of points
n = len(L)
sigma = 0
for k in range(n):
    sigma += P[k].Dist(P[k+1])
```

When you are using a formula for a subscript, check "end conditions" like k = 0 and k = n-1

6. Make Sure the "dot" notation is Being Used Correctly

No

```
P some list of points
for k in range(len(P):
      print P.x
```

Yes

```
P some list of points
for k in range(len(P):
      print P[k].x
```

7. Ask: "What Values is the Loop Variable Taking on?"

```
for x in L:
```

```
Things L can be:
      some range
      list of ints or floats
      a string
      list of objects
      a dictionary
      an open file
```

7. Cont'd

Yes

```
for S in L:
    print S.nwords
```

No

```
for S in L:
    print L[S].nwords
```

Yes

```
for k in len(L):
    print L[k].nwords
```