Quiz: Memory Management (Practice Problems)

1 Assignment vs Copy vs Deep Copy

Note 1. Assignment makes two variable names refer to the same object. Changing the contents of one variable actually changes the contents of the object, and therefore changes the contents of both variables. In order to create new, distinct, objects, you must copy the variable. When lists contain non-container objects like integers, `copy` and `deepcopy` behave exactly the same way. When lists contain containers, then texttttcopy will not make copies of the inner containers, but `deepcopy` will.

Problem 1. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
import copy
xs = [1, 2, 3]
ys = xs
ys.append('A')
print('xs=', xs)
EOF
$ python3 foo.py
```

Problem 2. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
import copy
xs = [1, 2, 3]
ys = copy.copy(xs)
ys.append('A')
print('xs=', xs)
EOF
$ python3 foo.py
```

Problem 3. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
import copy
xs = [1, 2, 3]
ys = copy.deepcopy(xs)
ys.append('A')
print('xs=', xs)
EOF
$ python3 foo.py
```

Problem 4. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = xs
ys.append('A')
print('xs=', xs)
EOF
$ python3 foo.py
```

Problem 5. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = xs
ys[0].append('A')
print('xs=', xs)
EOF
$ python3 foo.py
```

Problem 6. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = copy.copy(xs)
ys[0].append('A')
print('xs=', xs)
EOF
$ python3 foo.py
```

Problem 7. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = copy.deepcopy(xs)
ys[0].append('A')
print('xs=', xs)
EOF
$ python3 foo.py
```

2 Default Arguments

Note 2. Variables that are parameters to a function are always local variables. If a default value is provided, then the behavior depends on the type of the value. For *primitive* types (int, float, string, bool), the value can never change. For all other types, the object that the variable references is created once when the function is first defined. All subsequent calls to the function will use the same object, and the changes to the object will persist.

Problem 8. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
def foo(x=4):
    x += 1
    return x
y = foo()
y = foo()
y = foo()
y = foo()
print('y=', y)
EOF
$ python3 foo.py
```

Problem 9. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
xs = [1, 2, 3]
def foo(xs=[]):
    xs.append(len(xs) + 1)
    return len(xs)
y = foo()
y = foo()
y = foo()
y = foo()
print('y=', y)
EOF
$ python3 foo.py
```

Problem 10. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
xs = [1, 2, 3]
def foo(xs=[]):
    xs.append(len(xs) + 1)
    return len(xs)
y = foo([1])
y = foo([1, 2, 3])
y = foo()
y = foo([1, 2])
print('y=', y)
EOF
$ python3 foo.py
```

Problem 11. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
xs = [1, 2, 3]
def foo(xs=[]):
    xs.append(len(xs) + 1)
    return len(xs)
y = foo([1])
y = foo([1, 2, 3])
y = foo([1, 2])
y = foo()
print('y=', y)
EOF
$ python3 foo.py
```

Problem 12. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
xs = [1, 2, 3]
def foo(xs=[]):
    xs.append(len(xs) + 1)
    return len(xs)
y = foo([1])
y = foo([1, 2, 3])
y = foo()
y = foo([1, 2])
print('y=', y)
EOF
$ python3 foo.py
```

Problem 13. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
xs = [1, 2, 3]
def foo(xs=[]):
    xs += [4]
    return len(xs)
y = foo()
y = foo()
y = foo()
y = foo()
print('y=', y)
EOF
$ python3 foo.py
```

Problem 14. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
def foo(xs=[1]):
    xs = xs + xs
    return len(xs)
y = foo()
y = foo()
y = foo()
y = foo()
print('y=', y)
EOF
$ python3 foo.py
```

3 Reversing a List

Note 3. The following problems illustrate different ways that you might try to reverse a list in python. At first glance, they all look the same. Due to memory management issues, however, they are not all correct. Understanding memory management is important when tracking down these subtle bugs, and you are guaranteed to run into these situations in later assignments in this course.

Writing a function that reverses a list is one of the classic interview questions for python programming jobs. Interviewers frequently say that they are shocked by the number of interviewees who fail these questions because they don't understand memory management. Pay special attention to these problems so that you do not fail future interview questions and can get a great job.

Problem 15. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
def reverse_list(xs):
    ys = xs
    for i in range(len(ys)):
        ys[i] = xs[-i-1]
    return ys
xs = [1, 2, 3]
ys = reverse_list(xs)
print('xs=', xs)
print('ys=', ys)
EOF
$ python3 foo.py
```

Problem 16. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
import copy
def reverse_list(xs):
    ys = copy.copy(xs)
    for i in range(len(ys)):
        xs[i] = ys[-i-1]
    return ys
xs = [1, 2, 3]
ys = reverse_list(xs)
print('xs=', xs)
print('ys=', ys)
EOF
$ python3 foo.py
```

Problem 17. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
import copy
def reverse_list(xs):
    ys = copy.copy(xs)
    for i in range(len(ys)):
        ys[i] = xs[-i-1]
    return ys
xs = [1, 2, 3]
ys = reverse_list(xs)
print('xs=', xs)
print('ys=', ys)
EOF
$ python3 foo.py
```

Problem 18. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
def reverse_list():
    ys = xs
    for i in range(len(ys)):
        ys[i] = xs[-i-1]
    return ys
xs = [1, 2, 3]
ys = reverse_list()
print('xs=', xs)
print('ys=', ys)
EOF
$ python3 foo.py
```

Note 4. Because reversing a list is a common task, python provides two inbuilt methods to accomplish it. The `reverse` function does not create a copy of a list, but instead changes the list in place. The `reversed` function makes a copy of the list, reverses the copy, and leaves the original list unmodified. Both functions are widely used in python code.

Problem 19. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
xs = [1, 2, 3]
ys = xs.reverse()
print('xs=', xs)
print('ys=', ys)
EOF
$ python3 foo.py
```

Problem 20. Write the output of the final command in the following terminal session. If the command has no output, then leave the problem blank.

```
$ cd; rm -rf quiz; mkdir quiz; cd quiz
$ cat > foo.py <<EOF
xs = [1, 2, 3]
ys = list(reversed(xs))
print('xs=', xs)
print('ys=', ys)
EOF
$ python3 foo.py
```