1 Memory Management

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 `copy` will not make copies of the inner containers, but `deepcopy` will. Slices always make "shallow" copies.

Problem 2. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [1, 2, 3]
ys = xs
ys.append('A')
print('xs=', xs)
```

Problem 3. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [1, 2, 3]
ys = copy.copy(xs)
ys.append('A')
print('xs=', xs)
```

Problem 4. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [1, 2, 3]
ys = xs[1:]
ys.append('A')
print('xs=', xs)
```

Problem 5. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [1, 2, 3]
ys = xs[:]
ys.append('A')
print('xs=', xs)
```

Problem 6. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [1, 2, 3]
ys = copy.deepcopy(xs)
ys.append('A')
print('xs=', xs)
```

Problem 7. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = xs
ys.append('A')
print('xs=', xs)
```

Problem 8. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = xs[:]
ys.append('A')
print('xs=', xs)
```

Problem 9. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = xs
ys[0].append('A')
print('xs=', xs)
```

Problem 10. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = copy.copy(xs)
ys[0].append('A')
print('xs=', xs)
```

Problem 11. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = copy.deepcopy(xs)
ys[0].append('A')
print('xs=', xs)
```

Problem 12. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
xs = [[1, 2, 3], [4, 5, 6]]
ys = xs[0][:]
ys.append('A')
print('xs=', xs)
```

1.1 Default Arguments

Note 13. 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 14. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
def foo(x=4):
    x += 1
    return x
y = foo()
y = foo()
y = foo()
y = foo()
print('y=', y)
```

Problem 15. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
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)
```

Problem 16. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
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)
```

Problem 17. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
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)
```

Problem 18. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
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)
```

Problem 19. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
xs = [1, 2, 3]
def foo(xs=[]):
    xs += [4]
    return len(xs)
y = foo()
y = foo()
y = foo()
y = foo()
print('y=', y)
```

Problem 20. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
def foo(xs=[1]):
    xs = xs + xs
    return len(xs)
y = foo()
y = foo()
y = foo()
y = foo()
print('y=', y)
```

1.2 Reversing a List

Note 21. 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 22. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
def reverse_list(xs):
    ys = []
    for i in range(len(xs)):
        ys.append(xs.pop())
    return ys
xs = [1, 2, 3]
ys = reverse_list(xs)
print('xs=', xs)
print('ys=', ys)
```

Problem 23. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
import copy
def reverse_list(xs):
    ys = []
    xs = copy.copy(xs)
    for i in range(len(xs)):
        ys.append(xs.pop())
    return ys
xs = [1, 2, 3]
ys = reverse_list(xs)
print('xs=', xs)
print('ys=', ys)
```

Problem 24. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
def reverse_list(xs):
    ys = []
    xs = xs[:]
    for i in range(len(xs)):
        ys.append(xs.pop())
    return ys
xs = [1, 2, 3]
ys = reverse_list(xs)
print('xs=', xs)
print('ys=', ys)
```

Problem 25. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
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)
```

Problem 26. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
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)
```

Problem 27. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
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)
```

Problem 28. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
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)
```

1.3 Built-in methods

Note 29. You absolutely must know which built-in functions make copies and which do not. The way to remember is that any function of the form *XXXed* (i.e. past tense of verb XXX) returns a copy of the result that has been XXXed. Any function that is a method (i.e. appears to the right of a dot) modifies the object (the variable to the left of the dot) in place.

Problem 30. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
xs = [1, 2, 3]
ys = xs.reverse()
print('xs=', xs)
print('ys=', ys)
```

Problem 31. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
xs = [1, 2, 3]
ys = list(reversed(xs))
print('xs=', xs)
print('ys=', ys)
```

Problem 32. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
xs = [1, 2, 3]
ys = xs.sort()
print('xs=', xs)
print('ys=', ys)
```

Problem 33. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
xs = [1, 2, 3]
ys = list(sorted(xs))
print('xs=', xs)
print('ys=', ys)
```

1.4 Circular References

Note 34. Things get really weird when lists contain themselves.

Problem 35. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
xs = [1, 2, 3]
xs.append(xs)
y = xs[3][3][0]
print('y=', y)
```

Problem 36. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
xs = [1, 2, 3]
xs.append(xs)
y = xs[0][3][3]
print('y=', y)
```

Problem 37. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
xs = []
xs.append(xs)
xs.append(xs)
xs.append(xs)
xs.append(xs)
y = len(xs[0][1][2])
print('y=', y)
```

Problem 38. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
xs = []
xs.append(xs)
xs.append(xs)
xs.append(xs)
xs.append(xs)
y = len(xs[:0][1][2])
print('y=', y)
```

2 The class keyword

Problem 39. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    pass
a = Foo()
a.message = 'hello world'
b = Foo()
b.message = 'hola mundo'
print('a.message=', a.message)
```

Problem 40. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    pass
a = Foo()
b = Foo()
b.message = 'hello world'
print('a.message=', a.message)
```

Problem 41. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    message = 'salve munde'
a = Foo()
b = Foo()
b.message = 'hola mundo'
print('a.message=', a.message)
```

Problem 42. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    pass
a = Foo()
b = Foo()
b.message = 'hola mundo'
Foo.message = 'salve munde'
print('a.message=', a.message)
```

Problem 43. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    pass
a = Foo()
a.message = 'hello world'
b = Foo()
b.message = 'hola mundo'
Foo.message = 'salve munde'
print('a.message=', a.message)
```

3 Constructors

Note 44. The `__init__` function is called the *constructor* for the class. It is called automatically for us when an object is created.

Problem 45. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, message):
        self.message = message
a = Foo('hello world')
b = Foo('hola mundo')
print('a.message=', a.message)
```

Problem 46. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, message):
        message = message
a = Foo('hello world')
b = Foo('hola mundo')
print('a.message=', a.message)
```

Problem 47. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    message = 'salve munde'
    def __init__(self, message):
        message = message
a = Foo('hello world')
b = Foo('hola mundo')
print('a.message=', a.message)
```

Problem 48. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, message):
        Foo.message = message
a = Foo('hello world')
b = Foo('hola mundo')
print('a.message=', a.message)
```

Problem 49. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, message):
        Foo.message = message
a = Foo('hello world')
b = Foo('hola mundo')
Foo.message = 'salve mundo'
print('a.message=', a.message)
```

Problem 50. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, message):
        Foo.message = message
a = Foo('hello world')
b = Foo('hola mundo')
b.message = 'salve mundo'
print('a.message=', a.message)
```

Problem 51. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, message=None):
        self.message = message
a = Foo()
b = Foo('hola mundo')
print('a.message=', a.message)
```

Problem 52. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, message=None):
        if message:
            self.message = message
a = Foo()
b = Foo('hola mundo')
print('a.message=', a.message)
```

3.1 Tricky Problems

Problem 53. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, xs=[]):
        self.xs = xs
        self.xs.append('init')
a = Foo(['hello'])
b = Foo()
c = Foo(['hola'])
d = Foo()
x = len(d.xs)
print('x=', x)
```

Problem 54. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, xs=None):
        if not xs:
            xs = []
        self.xs = xs
        self.xs.append('init')
a = Foo(['hello'])
b = Foo()
c = Foo(['hola'])
d = Foo()
x = len(d.xs)
print('x=', x)
```

Problem 55. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, xs=[]):
        self.xs = xs
        xs.append('init')
a = Foo(['hello'])
b = Foo()
c = Foo(['hola'])
d = Foo()
x = len(d.xs)
print('x=', x)
```

Problem 56. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self, xs=None):
        if not xs:
            xs = []
        self.xs = xs
        xs.append('init')
a = Foo(['hello'])
b = Foo()
c = Foo(b.xs)
d = Foo()
x = len(d.xs)
print('x=', x)
```

3.2 class methods

Problem 57. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self):
        self.xs = []
    def bar(self, x):
        self.xs.append(x)
a = Foo()
b = Foo()
a.bar('hola')
b.bar('mundo')
x = len(a.xs)
print('x=', x)
```

Problem 58. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self):
        self.xs = []
    def bar(self, x):
        self.xs.append(x)
a = Foo()
b = Foo()
b.xs = a.xs
a.bar('hola')
b.bar('mundo')
x = len(a.xs)
print('x=', x)
```

Problem 59. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self):
        self.xs = []
    def bar(self, x):
        self.xs.append(x)
a = Foo()
b = Foo()
a.bar(b.xs)
b.bar('mundo')
x = len(a.xs)
print('x=', x)
```

Problem 60. The following code (circle one)

terminates without error throws an exception runs forever

If the code terminates without error, write the output. If the code throws an exception, state the exception.

```
class Foo:
    def __init__(self):
        self.xs = []
    def bar(self, x):
        self.xs.append(x)
a = Foo()
b = Foo()
a.xs = b.xs
a.bar(b.xs)
b.bar('mundo')
x = len(a.xs)
print('x=', x)
```