Object Oriented Python — Practice

CSCI 40

Fall 2026

Total Score: _____/4

LLM evaluation *2026-08-26*

Re-evaluation pending after quiz-bank corrections.

Object Oriented Programming

1 Memory Management

Note. 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 1. 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.

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

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.

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

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

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

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

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

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

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

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

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

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

1.1 Default Arguments

Note. Variables that are parameters to a function are always local variables. A default expression is evaluated once, when the function is defined, and later calls that omit the argument reuse the resulting object. Mutating a mutable default therefore affects later calls. Rebinding the local parameter does not change the stored default, and immutable objects cannot be mutated in place.

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.

```
1  def foo(x=4):
2      x += 1
3      return x
4  y = foo()
5  y = foo()
6  y = foo()
7  y = foo()
8  print('y=', y)
```

Problem 13. 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.

```
1  xs = [1, 2, 3]
2  def foo(xs=[]):
3      xs.append(len(xs) + 1)
4      return len(xs)
5  y = foo()
6  y = foo()
7  y = foo()
8  y = foo()
9  print('y=', y)
```

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.

```
1  xs = [1, 2, 3]
2  def foo(xs=[]):
3      xs.append(len(xs) + 1)
4      return len(xs)
5  y = foo([1])
6  y = foo([1, 2, 3])
7  y = foo()
8  y = foo([1, 2])
9  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.

```
1  xs = [1, 2, 3]
2  def foo(xs=[]):
3      xs.append(len(xs) + 1)
4      return len(xs)
5  y = foo([1])
6  y = foo([1, 2, 3])
7  y = foo([1, 2])
8  y = foo()
9  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.

```
1  xs = [1, 2, 3]
2  def foo(xs=[]):
3      xs.append(len(xs) + 1)
4      return len(xs)
5  y = foo()
6  y = foo([1, 2, 3])
7  y = foo()
8  y = foo()
9  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.

```
1  xs = [1, 2, 3]
2  def foo(xs=[]):
3      xs += [4]
4      return len(xs)
5  y = foo()
6  y = foo()
7  y = foo()
8  y = foo()
9  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.

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

1.2 Reversing a List

Note. 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 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.

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

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.

```
1  import copy
2  def reverse_list(xs):
3      ys = []
4      xs = copy.copy(xs)
5      for i in range(len(xs)):
6          ys.append(xs.pop())
7      return ys
8  xs = [1, 2, 3]
9  ys = reverse_list(xs)
10 print('xs=', xs)
11 print('ys=', ys)
```

Problem 21. 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.

```
1  def reverse_list(xs):
2      ys = []
3      xs = xs[:]
4      for i in range(len(xs)):
5          ys.append(xs.pop())
6      return ys
7  xs = [1, 2, 3]
8  ys = reverse_list(xs)
9  print('xs=', xs)
10 print('ys=', ys)
```

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.

```
1  def reverse_list(xs):
2      ys = xs
3      for i in range(len(ys)):
4          ys[i] = xs[-i-1]
5      return ys
6  xs = [1, 2, 3]
7  ys = reverse_list(xs)
8  print('xs=', xs)
9  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.

```
1  import copy
2  def reverse_list(xs):
3      ys = copy.copy(xs)
4      for i in range(len(ys)):
5          xs[i] = ys[-i-1]
6      return ys
7  xs = [1, 2, 3]
8  ys = reverse_list(xs)
9  print('xs=', xs)
10 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.

```
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 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():
    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. You must know which particular built-ins mutate an object and which return a new value. For lists, `xs.sort()` and `xs.reverse()` mutate `xs` and return `None`, while `sorted(xs)` and `reversed(xs)` leave `xs` unchanged and return new results. The spelling of a function or the fact that it is a method is not a general rule: many methods, such as `str.lower()`, return a new value without mutating the original.

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.

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

```
1  xs = [1, 2, 3]
2  ys = list(reversed(xs))
3  print('xs=', xs)
4  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.

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

Problem 29. 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.

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

1.4 Circular References

Note. Things get really weird when lists contain themselves.

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.

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

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.

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

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.

```
1  xs = []
2  xs.append(xs)
3  xs.append(xs)
4  xs.append(xs)
5  xs.append(xs)
6  y = len(xs[0][1][2])
7  print('y=', y)
```

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.

```
1  xs = []
2  xs.append(xs)
3  xs.append(xs)
4  xs.append(xs)
5  xs.append(xs)
6  y = len(xs[:0][1][2])
7  print('y=', y)
```

2 The class keyword

Problem 34. 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 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.

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

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.

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

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.

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

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.

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

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

```
1  class Foo:
2      def __init__(self, message):
3          Foo.message = message
4  a = Foo('hello world')
5  b = Foo('hola mundo')
6  Foo.message = 'salve mundo'
7  print('a.message=', a.message)
```

Problem 44. 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.

```
1  class Foo:
2      def __init__(self, message):
3          Foo.message = message
4  a = Foo('hello world')
5  b = Foo('hola mundo')
6  b.message = 'salve mundo'
7  print('a.message=', a.message)
```

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.

```
1  class Foo:
2      def __init__(self, message=None):
3          self.message = message
4  a = Foo()
5  b = Foo('hola mundo')
6  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=None):
        if message:
            self.message = message
a = Foo()
b = Foo('hola mundo')
print('a.message=', a.message)
```

3.1 Tricky Problems

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:
    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 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, 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 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, 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 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, 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 Instance methods

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):
        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 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):
        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 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):
        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 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):
        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)
```