CS 61A Midterm 2 Review

Dan Wang, Chris Giola, and Jon Kotker

Eta Kappa Nu, Mu Chapter
University of California, Berkeley

3 March 2012

Scoping

What is printed after the code is executed in Python 3?

```python
x = 3
def f():
    x = 4
print(x)
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```
x = 3
def f():
    x = 4
print(x)
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```
x = 3
def f():
    x = x + 1
print(x)
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```
x = 3
def f():
    x = x + 1
print(x)
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```python
x = 3
def f():
    global x
    x = 4
print(x)
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```
x = 3
def f():
    global x
    x = 4
print(x)
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```
def f():
    x = 3
    def g():
        x = 4
    g()
    print(x)
f()
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```
def f():
    x = 3
    def g():
        x = 4
    g()
    print(x)
f()
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```python
def f():
    nonlocal x
    x = 3
    def g():
        x = 4
    g()
    print(x)
f()
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```
def f():
    nonlocal x
    x = 3
    def g():
        x = 4
    g()
    print(x)
f()
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```
def f():
    x = 3
    def g():
        nonlocal x
        x = 4
    g()
    print(x)
f()
```

1. 3
2. 4
3. x
4. Error

Scoping

What is printed after the code is executed in Python 3?

```python
def f():
    x = 3
    def g():
        nonlocal x
        x = 4
    g()
    print(x)
f()
```

1. 3
2. 4
3. x
4. Error

Mutable Types

What is printed after the code is executed in Python 3?

```python
x = [1, 2]
y = x
y[0] = 3
print(x[0])
```

1. 1
2. 2
3. 3
4. Error

Mutable Types

What is printed after the code is executed in Python 3?

```
x = [1, 2]
y = x
y[0] = 3
print(x[0])
```

1. 1
2. 2
3. 3
4. Error

Mutable Types

What is printed after the code is executed in Python 3?

```
x = [1, 2]
y = [x, 3]
y[0] = [4, 5]
print(x)
```

1. `[4, 5]`
2. `[1, 2]`
3. `[[4, 5], 2]`
4. Error

Mutable Types

What is printed after the code is executed in Python 3?

```python
x = [1, 2]
y = [x, 3]
y[0] = [4, 5]
print(x)
```

1. `[4, 5]`
2. `[1, 2]`
3. `[[4, 5], 2]`
4. Error

Mutable Types

What is printed after the code is executed in Python 3?

```python
x = [1, 2]
y = [x, 3]
y[0][0] = [4, 5]
print(x)
```

1. `[4, 5]`
2. `[1, 2]`
3. `[[4, 5], 2]`
4. Error

Mutable Types

What is printed after the code is executed in Python 3?

```
x = [1, 2]
y = [x, 3]
y[0][0] = [4, 5]
print(x)
```

1. `[4, 5]`
2. `[1, 2]`
3. `[[4, 5], 2]`
4. Error

Classes

Convert the following below-the-line implementation of a class representing a point on the cartesian plane to a Python 3 class:

```
import math
def make_point(x, y):
    def point(op, *opnds):
        nonlocal x, y
        if op == 'distance_from_origin' and len(opnds) == 0:
            return math.sqrt(math.pow(x, 2) + math.pow(y, 2))
        elif op = 'distance_from_point' and len(opnds) == 1:
            return math.sqrt(math.pow(x - opnds[0]('x'), 2)
                + math.pow(y - opnds[0]('y'), 2))
        elif op = 'x' and len(opnds) == 0:
            return x
        elif op = 'y' and len(opnds) == 0:
            return y
        else:
            raise ValueError()
    return point
```

Classes

Solution

```python
import math
class Point:
    def __init__(self, x, y):
        self.x, self.y = x, y

    def distance_from_origin(self):
        return math.sqrt(math.pow(self.x, 2)
            + math.pow(self.y, 2))

    def distance_from_point(self, p):
        return math.sqrt(math.pow(x - p.x, 2)
            + math.pow(y - p.y, 2))
```