This document is available at http://stanford.edu/~danfrank/cme193/exercises/exercises-1.pdf.

1. True/False

State whether the following statements are True or False as it would be evaluated in Python (i.e., how it was described in lecture). Assume that the variable x has a boolean value of False and that the variable y has the value 10.

- (a) x and (8 < y < 12)
- (b) 'CME ' + '193' == 'cme193'
- (c) (y != 12 2) or x
- (d) 'py' * 2 + 'thonic' == 'pypythonic'

2. Arithmetic

State what x is after each of the following scripts is executed.

- (a) x = 2 y = 3 x *= y x /= y * 2
- (b) x = 'py' x += 'thon' y = z = 'py' x += y + z x *= 2
- (c) x = 1 y = 2 if x and y: x = 3
- (d) x = 'hello' y = 2 x += y
- (e) x = 'hello' y = 2 x += str(y)

3. Functions and Flow

For each of the following Python scripts, state what gets printed.

(a) def func_a():
 a = 2
 b = a + 3
 c = b * b
 return b + c

```
(b) def func_b(x):
        i = 0
        while x > 1:
            x = x / 2
            i = i + 1
       return i
   print func_b(10.0)
(c) def func_c():
       i = 0
        j = 0
        while (i < 16):
            if i > 3:
                i += 2
            if i < 10:</pre>
                j += i
            else:
                j -= 1
            i += 1
       return j
   print func_c()
```

(d) The elif statement combines the concepts of an else and an if statement. It follows an if statement. If the if statement is false, then the elif statement is evaluated. If the elif statement is true, that code block executes.

```
def func_d(x=0):
       if x < 0:
           return 'hello'
       elif x > 0:
           return 'world!'
       else:
           return ''
   print func_d(104) + func_d() + func_d(-11)
(e) def func_e(a, b):
       if a == b:
           return func_e(a - 1, b + 1)
       if a > b:
           def inner_func_e(x):
               if x < 0:
                   return 10
               else:
                   return 7
           return inner_func_e(a) + inner_func_e(b)
       return max(a, b)
   print func_e(7, 7) + func_e(7, -7) + func_e(-7, 7)
```

4. Applications

Consider the following snippet of Python code:

```
def func():
    step = 1
    point1 = 2
    point2 = point1 + step
    fp1 = point1 ** 3 + 3 * point1 + 3
    fp2 = point2 ** 3 + 3 * point2 + 3
    return (fp2 - fp1) / step

print func()
```

- (a) What gets printed?
- (b) What is this function doing?
- (c) Describe some abstractions for this function. What can be provided as parameters?

Using concepts from the next lecture, here is a much more powerful function:

```
def derivs(f, points, step=1):
    return [(f(p + step) - f(p)) / step for p in points]
print derivs(lambda(x): x ** 3 + 3 * x + 3, [2, 3, 4, 5], 0.1)
```