Midterm Review

- logistics
  - [ed link]
- content
  - functions
  - control (while, if)
  - higher-order functions
  - environment diagrams
  - functional abstraction (lambda expressions)
  - book: sections 1.1 - 1.6
- how to study
  - read, watch, and code (by hand)
  - review assignments
  - be able to: write code, read code, execute code

Consider the following function and function call. What is the output generated by python?

```
def mystery(a,b):
    return a + b

print(mystery("one plus two ","equals ") + str(mystery(1,2)))
```

Consider the following function and function call. What is the output generated by python?

```python
def mystery(a,b):
    return a + b

print(mystery("one plus two ","equals ") + str(mystery(1,2)))
```

```
one plus two equals 3
```

Consider the following function and function call. What is the output generated by the last print statement?

```
x = 3
def test(x):
    x = x + 1
    return x

test(1)
print(x)
```

Consider the following function and function call. What is the output generated by the last print statement?

```
x = 3
def test(x):
    x = x + 1
    return x

test(1)
print(x)
3
```

Consider the following function and function call. What is the output generated by python?

```python
def mystery(f,x,y):
    if x < y:
        return f(x)
    return f(y)

pow( mystery(abs,10,-3), 3 )
```

Consider the following function and function call. What is the output generated by python?

```
def mystery(f,x,y):
    if x < y:
        return f(x)
    return f(y)

pow( mystery(abs,10,-3), 3 )
27
```

Consider the following function and function call. What is the output generated by python?

```python
i = 0
while i <= 4:
    if i % 2 == 0:
        j = 0
        while j < 2:
            print( i, j )
            j = j + 1
    i = i + 1
```

Consider the following function and function call. What is the output generated by python?

```python
i = 0
while i <= 4:
    if i % 2 == 0:
        j = 0
        while j < 2:
            print( i, j )
            j = j + 1
    i = i + 1
```

```
0 0
0 1
```

Consider the following function and function call. What is the output generated by python?

```
i = 0
while i <= 4:
    if i % 2 == 0:
        j = 0
        while j < 2:
            print( i, j )
            j = j + 1
    i = i + 1
```

```
0 0
0 1
2 0
2 1
```

Consider the following function and function call. What is the output generated by python?

```
i = 0
while i <= 4:
    if i % 2 == 0:
        j = 0
        while j < 2:
            print( i, j )
            j = j + 1
    i = i + 1
```

```
0 0
0 1
2 0
2 1
4 0
4 1
```

Write a Python function count_down that takes as input two integers x and y and prints the values y, y-1, ..., x. For example calling count_down(3,7) will yield the following output:

```
7
6
5
4
3
```

You can assume that x <= y.

Write a Python function count_down that takes as input two integers x and y and prints the values y, y-1, ..., x. For example calling count_down(3,7) will yield the following output:

```
7
6
5
4
3
```

You can assume that x <= y.

```python
def count_down(x,y):
    i = y
    while i >= x:
        print(i)
        i = i - 1
```

A digit is a non-negative integer less than 10. Integers contain digits. For example:

- the integer 21 contains the digits 1 and 2
- the integer 474 contains the digit 4 twice and the digit 7 once
- the integer 400 contains the digit 4 once and the digit 0 twice
- the integer -77 contains the digit 7 twice.
- the integer 0 is a 0-digit number that contains no digits.

Implement *count*, which takes a digit element and an integer as input and returns the number of times the digit appears in the integer. You may assume that digit > 0 and digit < 10.

You may call built-in functions that do not require import, such as min, max, abs, and pow.

Warning: n % d and n // d may not behave as you expect for negative n. You should not evaluate % or // for negative values of n.

```
def count(element, box):
    """Count how many times digit element appears in integer box
       >>> count(2, 222122)
       5
       >>> count(0, -2020)
       2
       >>> count(0, 0)
       0
    """
    box = _________
             (a)
    total = 0
    while box > 0:
        if _________:
              (b)
             total = _________
                        (c)
        box = box // 10
     return total
```

```
def count(element, box):
    """Count how many times digit element appears in integer box
       >>> count(2, 222122)
       5
       >>> count(0, -2020)
       2
       >>> count(0, 0)
       0
    """
    box = _________
             (a)
    total = 0
    while box > 0:
        if box % 10 == element:
             total = _________
                        (c)
        box = box // 10
     return total
```

```
def count(element, box):
    """Count how many times digit element appears in integer box
       >>> count(2, 222122)
       5
       >>> count(0, -2020)
       2
       >>> count(0, 0)
       0
    """
    box = _________
             (a)
    total = 0
    while box > 0:
        if box % 10 == element:
             total = total + 1
        box = box // 10
     return total
```

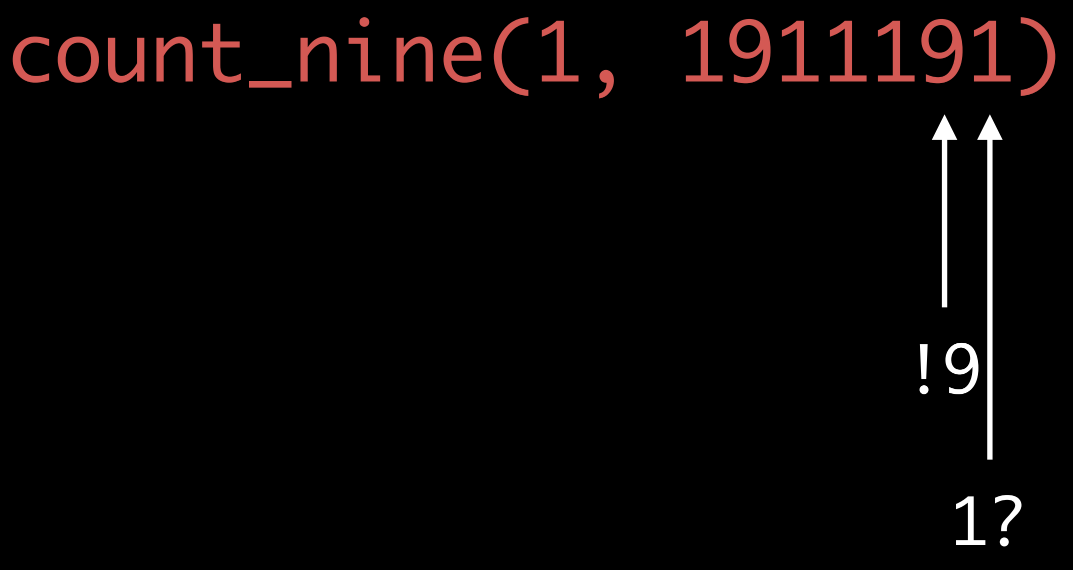
```python
def count(element, box):
    """Count how many times digit element appears in integer box
       >>> count(2, 222122)
       5
       >>> count(0, -2020)
       2
       >>> count(0, 0)
       0
    """
    box = abs(box)
    total = 0
    while box > 0:
        if box % 10 == element:
             total = total + 1
        box = box // 10
     return total
```

Implement count_nine, which takes a digit and a non-negative integer and returns the number of times the digit appears in the integer and is not adjacent to a 9.

```
>>> count_nine(2, 222122)
5
>>> count_nine(1, 1911191)
1
>>> count_nine(9, 9)
1
>>> count_nine(9, 99)
0
```

```
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if _________ and not(nine or _________):
              (a)                       (b)
            total = _________
                       (c)
        nine = _________ == 9
                  (d)
        box = box // 10
    return total
```

```
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if _________ and not(nine or _________):
              (a)                      (b)
            total = _________
                       (c)
        nine = _________ == 9
                  (d)
        box = box // 10
    return total
```



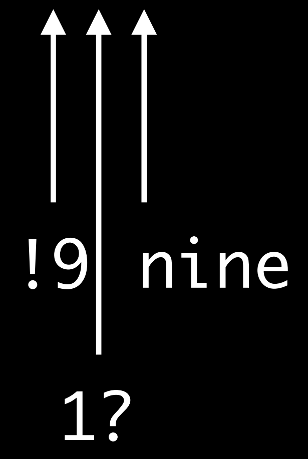
count_nine(1, 1911191)

!9

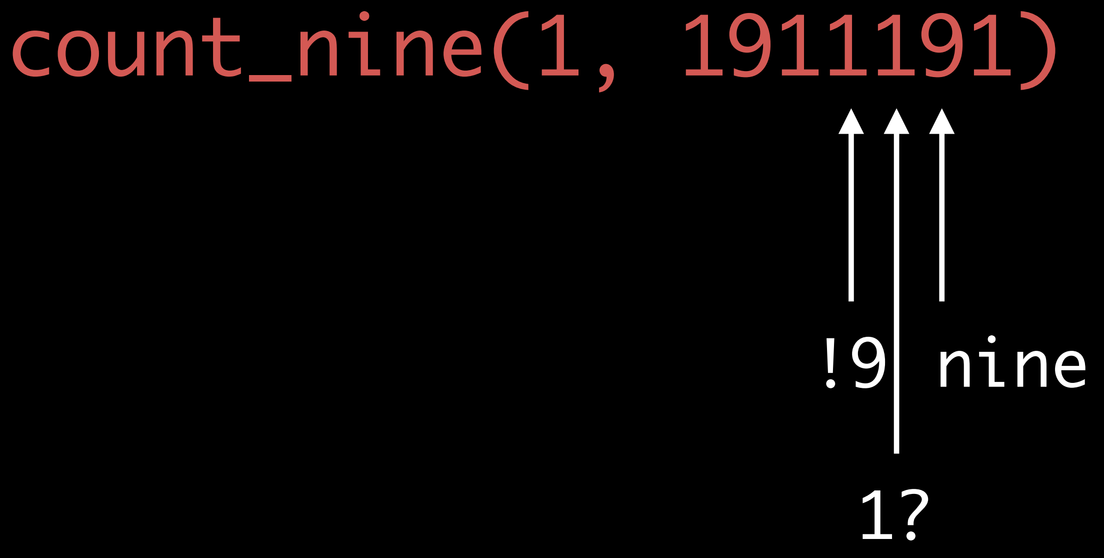
1?

```
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if _________ and not(nine or _________):
              (a)                        (b)
             total = _________
                        (c)
        nine = _________ == 9
                   (d)
        box = box // 10
    return total
```

count_nine(1, 1911191)



```
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if _________ and not(nine or _________):
              (a)                          (b)
            total = _________
                       (c)
        nine = _________ == 9
                  (d)
        box = box // 10
    return total
```



count_nine(1, 1911191)

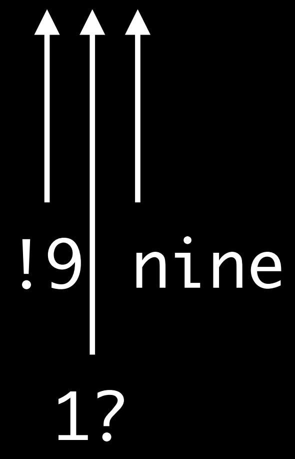
!9

1?

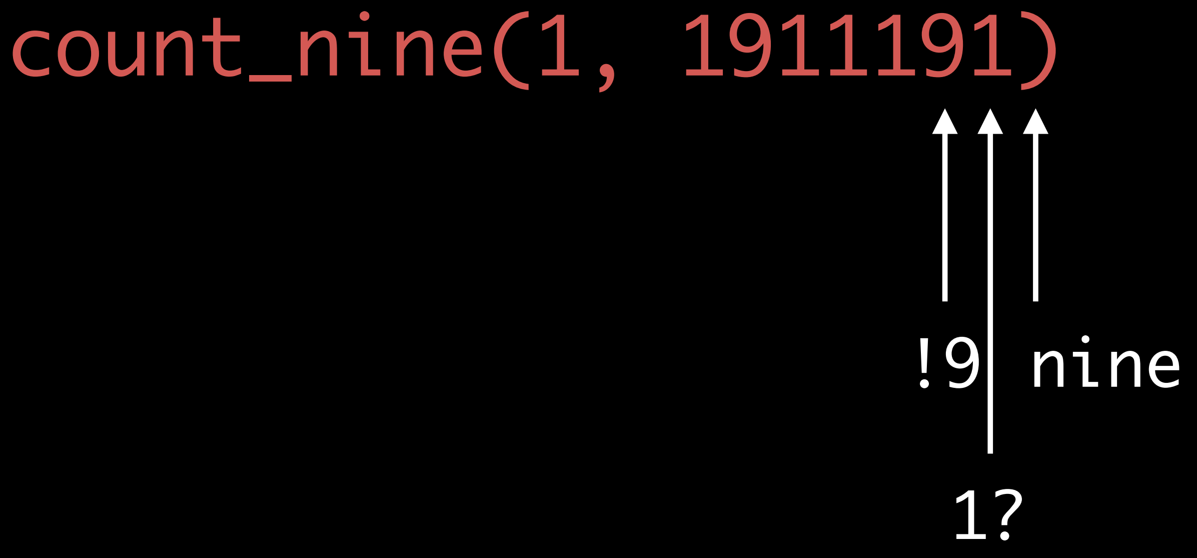
nine

```
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if _________ and not(nine or _________):
              (a)                       (b)
            total = _________
                       (c)
        nine = _________ == 9
                  (d)
        box = box // 10
    return total
```

count_nine(1, 1911191)



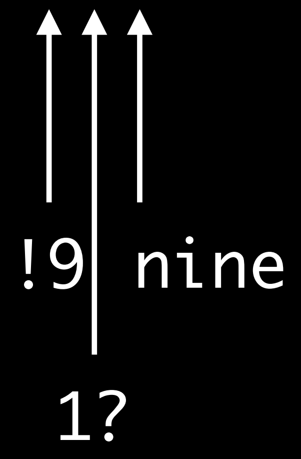
```
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if _________ and not(nine or _________):
              (a)                        (b)
            total = _________
                       (c)
        nine = _________ == 9
                  (d)
        box = box // 10
    return total
```



count_nine(1, 1911191)

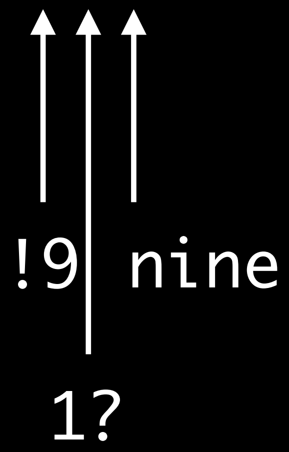
```
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if box % 10 == element and not(nine or _________):
                                                  (b)

            total = _________
                       (c)

        nine = _________ == 9
                  (d)

        box = box // 10
    return total
```

count_nine(1, 1911191)



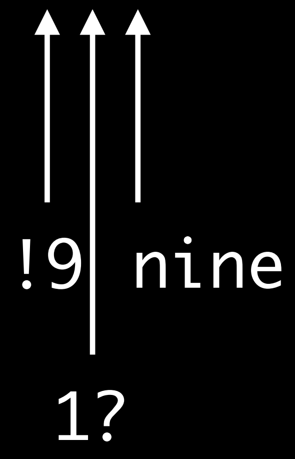
```
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if box % 10 == element and not(nine or (box // 10) % 10 == 9):
            total = _________
                       (c)
        nine = _________ == 9
                  (d)
        box = box // 10
    return total
```

count_nine(1, 1911191)



```
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if box % 10 == element and not(nine or (box // 10) % 10 == 9):
            total = total + 1
        nine = _________ == 9
                  (d)
        box = box // 10
    return total
```

count_nine(1, 1911191)



!9

nine

1?

```python
def count_nine(element, box):
    nine, total = False, 0
    while box > 0:
        if box % 10 == element and not(nine or (box // 10) % 10 == 9):
            total = total + 1
        nine = box % 10 == 9
        box = box // 10
    return total
```

Using a lambda expression, write a function mul_by_num that takes one argument and returns a one argument function that multiplies any value passed to it by the original number. The function’s body must be only one line.

```
>>> f = mul_by_num(5)
>>> g = mul_by_num(2)
>>> f(3)
15
>>> g(-4)
-8
```

Using a lambda expression, write a function mul_by_num that takes one argument and returns a one argument function that multiplies any value passed to it by the original number. The function’s body must be only one line.

```
>>> f = mul_by_num(5)
>>> g = mul_by_num(2)
>>> f(3)
15
>>> g(-4)
-8
```

```
def mul_by_num(num1):
    return lambda
```

Using a lambda expression, write a function mul_by_num that takes one argument and returns a one argument function that multiplies any value passed to it by the original number. The function’s body must be only one line.

```
>>> f = mul_by_num(5)
>>> g = mul_by_num(2)
>>> f(3)
15
>>> g(-4)
-8
```

```
def mul_by_num(num1):
    return lambda num2: num1 * num2
```

Consider the following function and function call. What is the output generated by python?

```python
def mystery(y):
    x = 0
    while x < 5:
        f = lambda z: x + y + z
        x = x + 1
    return f

g = mystery(10)
print(g(20))
```

Consider the following function and function call. What is the output generated by python?

```python
def mystery(y):
    x = 0
    while x < 5:
        f = lambda z: x + y + z
        x = x + 1
    return f

g = mystery(10) # lambda z: 5 + 10 + z
print(g(20))
```

Consider the following function and function call. What is the output generated by python?

```python
def mystery(y):
    x = 0
    while x < 5:
        f = lambda z: x + y + z
        x = x + 1
    return f

g = mystery(10) # lambda z: 5 + 10 + z
print(g(20))
35
```

questions?