



Middlebury

CSCI 146: Intensive Introduction to Computing

Fall 2025

Midterm 1 Review

Question 1

Consider the following Python code. What are the values of **s**, **r** and **y** after the code executes?

```
s = 5  
r = 3
```

```
def bar(s):  
    r = len(s)  
    return r
```

```
def foo(s):  
    r = 1  
    for i in range(len(s)):  
        r = s[i:]  
        r = r * (-bar(s[i:]))  
    return r
```

```
y = foo("test")
```

s = 5
r = 3

"test"
 $r = r * (-4)$

"est"
 $r = r * (-3)$

"st"
 $r = r * (-2)$

"t"
 $r = r * (-1)$

$\frac{r}{-4}$

-4

12

-24

24

Question 2

Write a function named `random_test` that takes three integer parameters: `max`, `threshold` and `num`. The function should generate `num` random integers between 0 and `max` (inclusive) and return the number of these values that were less or equal to the threshold

```
import random
def random_test(max, threshold, num):
    # ...
    count = 0
    for i in range(num):
        r = random.randint(0, max)
        if r <= threshold:
            count = count + 1 # count += 1
    return count
```

count
keeps track
of values
≤ threshold

Question 3

What would the following function return if we invoked it with 4 as the argument?

```
def mystery(x):  
    count = 1  
    total = 0  
  
    while count <= x:  
        total += count * count  
  
    return total
```

Question 4

For each of the following tasks, indicate which would be most appropriate, a **for** loop, a **while** loop, or both are equally appropriate:

- Transform each letter in a string.
- Count the number of coin flips rolls to get 10 heads.
- Solicit a valid password from a user.
- Perform an operation on every point in a 3-D grid of known size.
- Perform an operation on every "a" in a string.

Question 5

The following function is supposed to prompt the user for numbers until they enter a blank line, then it tells the user how many of the numbers were less than zero.

Unfortunately, the function has a few problems and doesn't work correctly. Identify and fix the problems.

```
def neg_count():  
    answer = input("Enter a number: ")  
    below = 0  
    while answer != "":  
        if answer < 0:  
            below += 1  
        ...  
    print("Below: " + below)
```

returns a
str

Runtime error
float(answer)

str(below)
Runtime error

syntax
vs
runtime
vs
logical

Question 6

Rewrite each of the boolean expressions below more compactly.

- Rewrite:

~~True~~ and x

- Rewrite:

y > 0 and y < 10

$0 < y < 10$

- Rewrite:

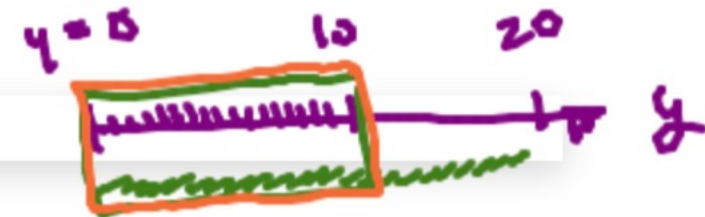
(y > 0 and z == 1) or (y > 0 and z == 2)

$(y > 0) \text{ and } (z == 1 \text{ or } z == 2)$

- Rewrite:

(y > 0 and y < 10) and (y > 0 and y < 20)

$0 < y < 10$



$y > 0 \text{ and } y < 10$

- Rewrite:

(y > 0 and y < 10) and (y <= 0 or y >= 10)

False

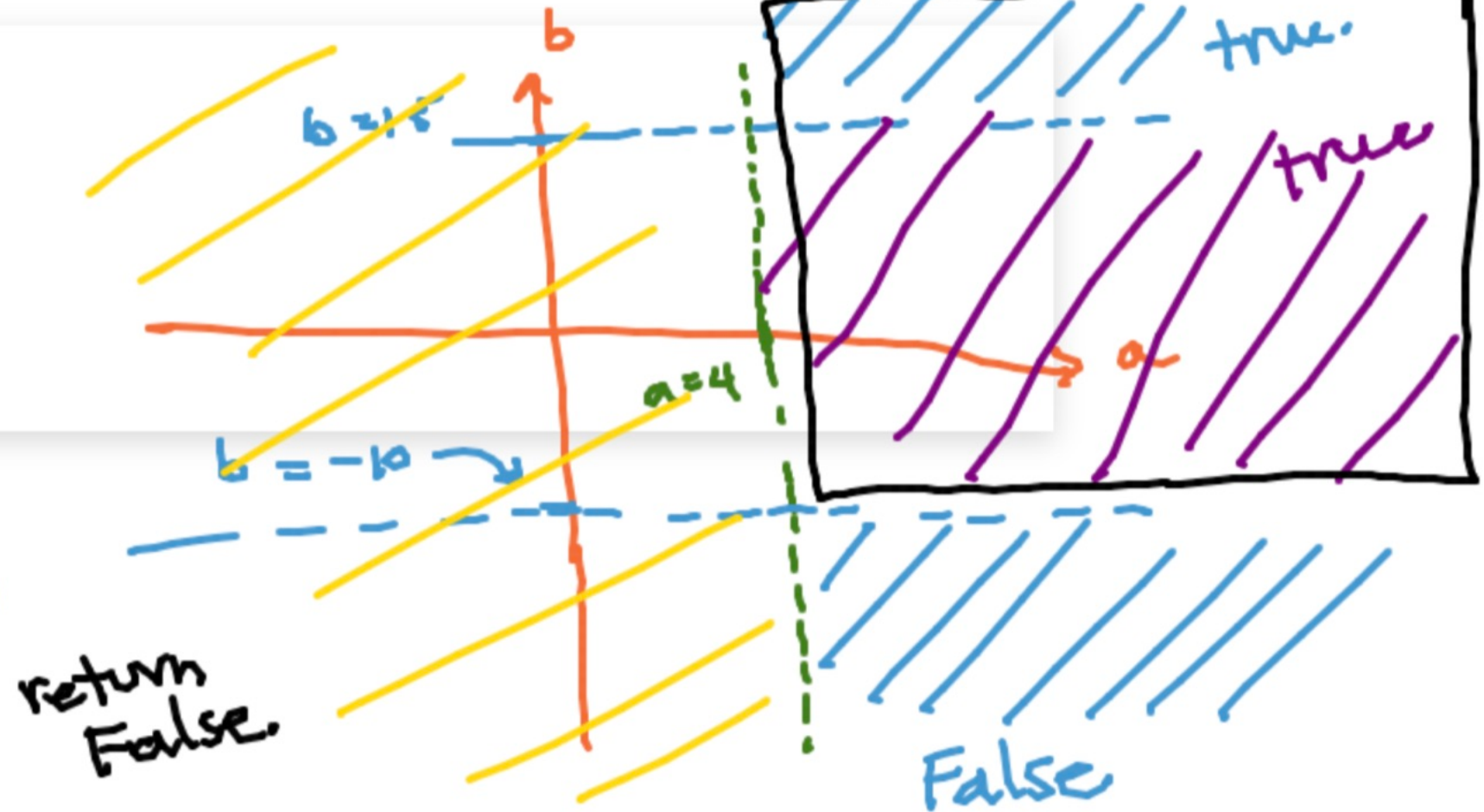


Question 7

The function below has two integer parameters **a** and **b**. The function works as desired, however, it is very verbose. Rewrite the function to have identical behavior (i.e., for all possible values of **a** and **b** return the same value), but to be as concise as possible.

```
def could_be_better(a, b):  
    if a > 4:  
        if b > 15:  
            return True  
        elif b < -10:  
            return False  
        else:  
            return True  
    else:  
        return False
```

`return a > 4 and b >= -10`



not to scale.

Question 8

Write a function named **lessthan** with two arguments, **seq1** and **seq2**, and performs lexicographic less than of those sequences. You should only compare individual elements of the sequences.:

```
>>> lessthan(["a", "b"], ["a", "b", "c"])
True
>>> lessthan(["a", "c"], ["a", "b", "c"])
False
>>> lessthan(["a", "b", "c"], ["a", "b", "c"])
False
```

"ab" < "abc"

1. len of sequences
2. comparing individual items in the sequences
3. use shortest sequence to determine loop bounds.

```
def lessthan(s1, s2):
    if len(s1) < len(s2):
        n = len(s1)
    else:
        n = len(s2)
    for i in range(n):
        if s1[i] < s2[i]:
            return True
        elif s1[i] > s2[i]:
            return False
    return len(s1) < len(s2)
```

```
s1[i] == s2[i] keep going
s1[i] < s2[i] return True
s1[i] > s2[i] return False
```

Question 9

Draw below what the following program would draw on the screen:

```
from turtle import *

def draw_something(x, y, length):
    penup()
    goto(x, y)
    pendown()
    goto(x + length, y)

for i in range(10):
    draw_something(0, i*5, i*5)

done()
```

Question 10

Write a function named `intersection` that has two sequences as parameters and returns a copy of its first argument containing just those items in the first sequence present in the second. It should work for any combination of list and string arguments. As a hint, slicing evaluates to a sequence of the same type and equality operators can be applied to values of different types. For example:

```
>>> intersection(["a", "b"], "aac")  
["a"]
```

Question 11

Imagine you are given a list of column positions in a building (in sorted ascending order). Write a function name `span` that takes the list as argument and returns the longest beam you will need to span between the columns, i.e., the maximum distance between any two adjacent columns. You can assume there are at least two columns.