

## CS303E Week 5 Worksheet: Loops

Name: \_\_\_\_\_ EID: \_\_\_\_\_

*Read the questions carefully, and answer each question in the space provided. Use scratch paper to do your work and then copy your answers neatly and legibly onto the test paper. Only answers recorded on the test paper will be graded.*

1. (10 points: 1 point each) The following are true/false questions. **Write either T or F in the boxes at the bottom of page 1.** If there's any counterexample, it's false.
- (a) The break statement can be used to exit a loop prematurely.
  - (b) It is possible to use the break statement to exit multiple nested loops at once.
  - (c) The continue statement is used to skip the remaining code within the current iteration of a loop and move to the next iteration.
  - (d) The sequence generated from range(n), where n is an integer, starts at 1 and goes up to (but does not include) n.
  - (e) Nested loops require using both for and while loops.
  - (f) Infinite loops occur when the condition controlling the loop never becomes false or when there is no exit statement within the loop body.
  - (g) The while loop in Python is always guaranteed to execute at least once.
  - (h) A while True loop will run indefinitely until a break statement is encountered.
  - (i) The loop condition in a while loop is evaluated before each iteration.
  - (j) The loop control variable in loops like `for x in range(2)` can be accessed outside the loop.

a	b	c	d	e	f	g	h	i	j
T	F	T	F	F	T	F	T	T	T

Questions 2-6 are multiple choice. Each counts 2 points. **Write the letter of the BEST answer in the box on the next page. Please write your answer in UPPERCASE. Each problem has a single answer.**

2. Which of the following statements about the `range()` function in Python is true?
  - A. The `range()` function can generate sequences of both numbers and strings.
  - B. The `range()` function provides an option for randomization in for loops.
  - C. The `range()` function can generate both continuous and non-continuous sequences of numbers in ascending or descending order.
  - D. The `range()` function can only be used with `for` loops and not with `while` loops.
3. When would you typically use a `while` loop instead of a `for` loop?
  - A. When you need to iterate over a known sequence of elements.
  - B. When the number of iterations is fixed and predetermined.
  - C. When the loop requires an exit condition that is not based on the number of iterations.
  - D. When you want to execute a block of code at least once.
4. What is the purpose of nested loops?
  - A. Nested loops allow for more efficient memory utilization.
  - B. Nested loops are used to handle errors and exceptions that may occur during loop execution.
  - C. Nested loops enable complex patterns of iteration by placing loops inside each other.
  - D. Nested loops have no purpose. You can use a single loop in any instance you use nested loops.

5. In a `for` loop statement like `for x in range(3)`, can you modify the value of the loop variable `x` within the loop body to change the number of iterations?
- A. Yes, the loop variable `x` can be modified within the loop body to change the number of iterations.
  - B. No, modifying the loop variable `x` directly within a `for` loop will not affect the number of iterations.
  - C. Modifying the loop variable `x` will cause an error and terminate the loop.
  - D. The loop variable `x` can only be modified indirectly using additional variables.
6. What is the main difference between loops and conditionals (`if/elif/else`) in programming?
- A. Loops are used to execute a block of code repeatedly, while conditionals are used to make decisions based on different conditions.
  - B. Loops and conditionals are interchangeable and can be used interchangeably in any programming scenario.
  - C. Conditionals cannot be nested inside other conditionals, but loops can be nested inside other loops.
  - D. Conditionals are used to iterate over a sequence of values, while loops are used to check for specific conditions and execute code accordingly.

2	3	4	5	6
C	C	C	B	A

The following 7 questions require you to trace the behavior of some Python code and identify the output of that code. For each question, write the output for the code segment on the provided line.

7. (3 points)

```
count = 1
while count <= 10:
    if count % 3 == 0:
        count += 2
        continue
    print(count, end=" ")
    count += 1
```

What's going on here: because we're testing  $\text{count} \% 3$ , the if statement only triggers when count is == a factor of 3. So, it triggers when count is 3, 6, and 9. And when the if statement triggers, count is increased by 2, and we skip the print statement (because we hit the continue).

1 2 5 8

8. (3 points)

```
range(4) = [0, 1, 2, 3]
for i in range(4):
    for j in range(i):
        print(i + j, end=" ")
```

This one is tricky! But essentially, for every value of  $i$  (which starts at 0), we're summing that number with every number less than it. At first  $i == 0$ , so for  $j$  in  $\text{range}(0)$  won't do anything, because  $\text{range}(0)$  is empty. But when  $i == 1$ , the  $j$  in  $\text{range}(1)$  loop will print  $1 + 0 = 0$ . Then when  $i == 2$ , the  $j$  in  $\text{range}(2)$  loop will print  $0 + 2 = 2$ , and  $1 + 2 = 3$ . etc.

1 2 3 3 4 5

9. (3 points)

```
count = 0
while count < 5:
    print(count, end=" ")
    if count == 2:
        break
    count += 1
```

This one simply tests your knowledge of what break does =)

0 1 2

10. (3 points)

```
for i in range(1, 10, 2):
    print(i, end=" ")
```

This one tests your knowledge of the step parameter for range. range(1, 10, 2) tells Python to create a sequence of numbers starting at 1, going up to 10 (exclusive), but because our step parameter is 2, that means after we include one number, we 'jump' two more. Hence, we include 1, then 3, then 5, then 7, then 9.

1 3 5 7 9

11. (3 points)

```
num = 10
while num > 0:
    if num % 2 == 0:
        num -= 1
    else:
        num += 1
    print(num, end=" ")
```

This results in an infinite loop! Notice that num % 2 == 0 tests if a number is even. num starts at 10, so this if statement will trigger — now num is decremented to 9, and 9 is printed. The next iteration, the 'else' will trigger, incrementing num to 10 and printing 10. Since num is 10, the if will trigger the next iteration, then the else on the next one, etc.

Infinite loop

12. (3 points)

```
my_sum = 0
for i in range(1, 5):
    my_sum += i

print(i)
```

This one tests reading comprehension! Yes, we sum 1, 2, 3, 4 all up into my\_sum. But then we print the loop variable i, NOT my\_sum. i was last set to 4, so that is our result.

4

13. (3 points)

```
x = 10
while x in range(10, 5, -2):
    print("Kookie", end = " ")
    x -= 4
```

x starts at 10, which is in our range. So, our loop starts, prints kookie, and decrements x to 6 (which is still in our range). So, our loop runs a second time, printing Kookie again, and decrementing x to 2. Because x is no longer included in our range, our loop ends.

Kookie Kookie