

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) A **continue** statement will always exit a loop
 - (g) The body of a **while** loop in Python is always guaranteed to execute at least once.
 - (h) The loop condition in a **while** loop is evaluated before each iteration.
 - (i) The loop control variable (e.g. **x** in **for x in range(2)**) can be accessed outside the loop.
 - (j) Any loop can be replaced by straight line code.

a	b	c	d	e	f	g	h	i	j

Questions 2-8 are multiple choice. Each counts 2 points. **Write the letter of the BEST answer in the box on page 3. Please write your answer in UPPERCASE. Each problem has a single answer.**

2. If L is a list, what is a difference between:

- (i) `for x in L`
- (ii) `for i in range( len(L) )`

This question shouldn't be on here because we haven't introduced lists. So just take a guess or skip it.

- A. there is no difference
 - B. in the loop body you'd refer to elements of L as x in (i) and L[i] in (ii)
 - C. version (i) isn't legal
 - D. version (ii) isn't legal
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. They allow for more efficient memory utilization.
 - B. They are used to handle errors and exceptions that may occur during loop execution.
 - C. They enable complex patterns of iteration by placing loops inside each other.
 - D. They have no purpose. You can use a single loop in any instance you use nested loops.
5. Which of the following commands only make sense inside a loop?
- A. `break`
 - B. `continue`
 - C. `max`
 - D. `if`
 - E. both A and B

6. What does the following code print?

```
x = 1
for i in range(10):
    x *= i
print(x)
```

- A. the product of the integers 1 to 10
 - B. the sum of integers 0 to 10
 - C. 0
 - D. the numbers from 0 to 10
 - E. it raises an error
7. 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.
8. You'll exit a while loop if:
- A. **break** executes
 - B. the condition becomes True
 - C. **continue** executes
 - D. any of the above
 - E. either A or B

2	3	4	5	6	7	8

The following questions (2 points each) ask 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 in the box provided. Don't worry about line breaks. If there's an error, say so, but you don't need to specify the type of error.

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

```
10. for i in range(3):
     for j in range(i):
         print(i + j, end=" ")
```

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

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

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

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

    print(i)
```

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