CS 171 - Lab 5

Professor Mark W. Boady and Professor Adelaida Medlock

Content by Professor Mark Boady

Detailed instructions to the lab assignment are found in the following pages.

- Complete all the exercises and type your answers in the space provided.

What to submit:

- Lab sheet in PDF
- A screenshot of code and execution for questions 23, 24, 25

Submission must be done via Gradescope

- Please make sure you have tagged all questions when you upload
- We only accept submissions via Gradescope.

Student Name: Tony Kabilan Okeke

User ID (abc123): tko35

Possible Points: 88

Your score out of 88:

Lab Grade on 100% scale:

Graded By (TA Signature):

1. Critical Thinking Questions

Question 1: 7 points

Closely examine the Python program below.

FYI: A **looping structure** allows a block of code to be repeated one or more times. A **while** loop is one of the two looping structures available in Python.

```
1 # Description: This program prints
2 # a person's name 20 times
3
4 name = input ("Enter your name: ")
5 x = 0
6 while (x < 20):
7     print(name)
8     x = x + 1
9 print ("The End")
```

(a) (2 points) In the Python code, circle / highlight all the lines executed during each iteration of the loop.

(b) (1 point) In the IDE, enter and test the code. What does the line of code: **`x = x + 1`** do?
This line increments the value of x by 1 with each iteration of the loop.

(c) (1 point) How does the Python interpreter know what lines of code belong to the loop body?
Indentation. Indented lines of code below the loop header are part of the loop body.

(d) **Every loop structure requires three actions.** Identify the line of code in the Python program that corresponds to each of the three actions.

(a) (1 point) Initialize a variable used in the test condition:
Line 5 `x = 0`

(b) (1 point) Include a test condition that causes the loop to end when the condition is false:
Line 6 `while (x < 20):`

(c) (1 point) Within the loop body, update the variable used in the test condition:
Line 8 `x = x + 1`

Question 2: 3 points

Enter and execute the following code. code does.

```
1   # Description: This program prints
2   # number from 1 to the  value entered by the user
3
4   number = int (input ("Enter a number: "))
5   x = 1
6   while (x <= number ):
7       if (x % 10 == 0):
8           print(x)
9       else:
10          print(x , end="")
11      x = x + 1
```

Identify the line of code in the Python program that corresponds to each of the three actions.

(a) (1 point) Initialize a variable used in the test condition:
`Line 5      x = 1`

(b) (1 point) Include a test condition that causes the loop to end when the condition is false:
`Line 6      while (x <= number ):`

(c) (1 point) Within the loop body, update the variable used in the test condition:
`Line 11      x = x + 1`

Question 3: 1 point

The following code should print the numbers from 1 to 10, but it does not print anything.

```
~~number = 12~~
number = 1
while number <= 10:
      print(number)
      number = number + 1
```

Fix the logic error in this code. Edit the above code. Edits shown in blue.

Question 4: 3 points

Enter and execute the following code:

```
1   number = 1
2   while number <= 10:
3         print(number)
4   number -= 1
```

(a) (1 point) Does the program end?
No, it produces an infinite loop.

(b) (2 points) How could you fix this problem?
- Indent Line 4 so that it is within the body of the loop
- Replace Line 4 with `number += 1`

Question 5: 4 points

Enter and execute the following code:

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

(a) (1 point) State the output

The program prints even numbers between 1 and 10: `2 4 6 8 10`

(b) (1 point) What caused the output to display on one line?

The end argument of the **`print`** function was specified as a space.

(c) (2 points) What control structures are used in this code?

The code includes a repetition structure (the `while` loop), and a decision structure (the `if` statement).

Question 6: 7 points

The following directions will create a program that prompts the user to enter a number between 1 and 10. As long as the number is out of range the program re-prompts the user for a valid number. Complete the following steps to write this code.

(a) (1 point) Write a line of code that prompts the user for a number between 1 and 10.

```
number = int( input(‘Enter a number between 1 and 10: ’) )
```

(b) (1 point) Write a Boolean expression that tests the number the user entered by the code in step (a) to determine if it is not in range.

```
(1 <= number <= 10)
```

(c) (2 points) Use the Boolean expression created in step (b) to write a while loop that executes when the user input is out of range. The body of the loop should tell the user that they entered an invalid number and prompt them for a valid number again.

```
while not (1 <= number <= 10):
      print( “The number you entered is invalid.” )
      number = int( input(‘Enter a number between 1 and 10: ’) )
```

(d) (1 point) Write the code that prints a message telling the user that they entered a valid number.

```
print( “The number you entered is valid.” )
```

(e) (1 point) Put the segments of code from steps (a) – (d) together. Enter and execute the code. Does it work properly? If not, correct it and test it again.

```
number = int( input('Enter a number between 1 and 10: ') )
while not (1 <= number <= 10):
      print( "The number you entered is invalid." )
      number = int( input('Enter a number between 1 and 10: ') )
print( "The number you entered is valid." )
```

The code above works as expected.

(f) (1 point) How many times does the loop execute?

The loop executes as long as the value provided by the user is between 1 and 10. So, it depends on the values provided.

FYI: A looping structure for which you know the number of times it will execute is known as a count-controlled loop.

Question 7: 5 points

Sometimes a programmer does not know how many times data is to be entered. For example, suppose you want to create a program that adds an unknown amount of positive numbers that will be entered by the user. The program stops adding numbers when the user enters a zero or a negative number. Then the program prints the total. Before creating this program, review the three actions required for all loops:

(a) (1 point) Initialize a variable that will be used in the test condition: What will be tested to determine if the loop is executed? Write a line of code that initializes a variable to be used in the test condition of the loop for this program. The variable should contain a value entered by the user.

```
number = int(input('Enter a number: '))
```

(b) (1 point) Include a test condition that causes the loop to end when the condition is false: What is the test condition for the while loop used in this program?

```
number > 0
```

(c) (1 point) Within the loop body, update the variable used in the test condition: Write the code for the loop body.

```
while number > 0:
  number = int(input('Enter a number: '))
  total += number
```

(d) (1 point) Is this a count-controlled loop?

No.

(e) (1 point) Complete the program. Enter your fully functioning code below.

number = int(input('Enter a number: '))

```
total = number

while number > 0:
  number = int(input('Enter a number:'))
  total += number

print('Total =', total)
```

FYI: Short-cut operators provide a concise way of creating assignment statements when the variable on the left-hand side of the assignment statement is also on the right-hand side. The addition short-cut operator (+=) is usually used for incrementing a variable. These are also called compound operators.

Question 8: 5 points

Enter and execute the following code:

```
number = 1
number += 3
print(number)
```

(a) (1 point) What does the **+=** shortcut operator do?
It increments the value of `number` by 3 (Adds 3 to the value of `number`).

(b) (1 point) The code: **`x += 5`** is equivalent to which of the following lines of code?

```
x = 5
x = y + 5
x = x + 5
y = x + 5
```

(c) Replace the operator **+=** with the following shortcut operators and execute the code. Explain what each operator does.

(a) (1 point) **-=**
Decrements `number` by 3; the program prints -2 in the console.

(b) (1 point) ***=**
Multiplies number by 3; the program prints 3 in the console.

(d) (1 point) Is the following line of code valid: **`23 += total`**? Why or why not?
No, it isn't. Compound operators perform an arithmetic operation and assign the value to a variable. In this case, 23 is a literal and therefore can not be on the left hand side of an assignment operator.

Question 9: 2 points

Enter and execute the following code:

```
bonus = 25
salary += bonus
print ("Total salary:", salary )
```

(a) (1 point) What is the output of the preceding code?
The code produces a `NameError` since the `salary` variable is not defined.

(b) (1 point) Rewrite the code so that it produces valid output.

```
salary = int(input('Enter your salary: '))
bonus = 25
salary += bonus
print ("Total salary:", salary)
```

Question 10: 1 point

The following code should print the numbers beginning with 100 and ending with 0. However, it is missing a line of code. Add the missing code, using the shortcut operator. Draw an arrow to indicate where the code belongs.

```
countdown = 100
while countdown >= 0:
    print(countdown)
    countdown -= 1
print("Done!")
```

Question 11: 7 points

Enter and execute the following code:

```
1  doAgain = "y"
2  while doAgain == "y" :
3      word = input("Enter a word:")
4      print("First letter of " + word + " is " + word [0])
5      doAgain = input("Type 'y' to continue and anything else to quit.")
6  print("Done!")
```

(a) (1 point) What does the program do?
The program asks the user to enter a word, then asks if the user would like to continue providing words. If the user enters 'y', they are prompted to enter another word, if not, the program ends.

(b) (1 point) What is the variable name used to store the user's input?
`word`

(c) (1 point) In the print statement, what does **`word[0]`** represent?
It represents the first character (letter) in the provided word (string).

(d) (1 point) Change **`word[0]`** to **`word[1]`** in the print statement above. What is printed?
The second character (letter) in the provided word is printed.

(e) (1 point) When does the program end?
The program ends when the user enters 'y' in response to the prompt in line 5.

FYI: A **sentinel-controlled while loop** is a loop that repeats the loop body until the user enters a pre-specified value.

(f) (1 point) Why is the loop in this program an example of a sentinel control loop?
Because the loop repeats the loop body until the user enters 'y' in response to the prompt in line 5.

(g) (1 point) Examine the print statement in this program:

```
print ("First letter of " + word + " is " + word[0])
```

What happens if you replace the **+** with a **,**?
The strings would be printed together separated by spaces.

Question 12: 3 points
Examine the code below.

```
name = "Simone"
cost = 3.56
numApples = 89
```

What type of data is stored in each variable? (integer, floating point, or string)

(a) (1 point) **name**: string

(b) (1 point) **cost**: floating point

(c) (1 point) **numApples**: integer

FYI: A variable that can store only the values **True** and **False** is called a **Boolean** variable.

Question 13: 1 point
Given the assignment statement: **foundCost = False**
What type of data is stored in **foundCost**? Boolean

Question 14: 3 points
Enter and execute the following two Python programs.

WHILE LOOP

```
name = input ("Enter your name: ")
x = 0
while x < 20:
     print(name)
     x = x + 1
```

FOR LOOP

```
name = input("Enter your name: ")
for x in range (20):
     print(name)
```

(a) (2 points) What is the output for each program?
The while loop prints out the value of name 20 times.
The for loop also prints out the value of name 20 times.

(b) (1 point) Both programs produce the same output. Which code fragment is more concise?
The for loop is more concise.

FYI: The Python predefined function **range()** is used to define a series of numbers and can be used in a FOR loop to determine the number of times the loop is executed.

Question 15: 5 points

Enter and execute the following code fragments and state the output:

(a) (1 point)

```
for x in range (5):
    print (x , end=" ")
```

0 1 2 3 4

(b) (1 point)

```
for x in range (1 ,5):
    print (x , end=" ")
```

1 2 3 4

(c) (1 point)

```
for x in range (3 ,20 ,2):
    print (x , end=" ")
```

3 5 7 9 11 13 15 17 19

(d) (1 point)

```
 numIterations = 6
 for x in range( numIterations ):
    print (x , end=" ")
```

0 1 2 3 4 5

(e) (1 point)

```
 numIterations = 6
 for x in range(1 , numIterations +1):
     print (x , end=" ")
```

1 2 3 4 5 6

Question 16: 3 points

After examining the five code fragments in question 15, explain how the **range()** function works. Include an explanation of the arguments.

The range(start, stop) function returns a sequence of numbers from start to stop, not including end.
If only one argument is provided, range(num) the range from 0 to num-1 is returned.
If a stride argument is provided, the values in the range are incremented by the stride each time.

FYI: In a **for** loop you can include a list of values in place of the **range()** function.

Question 17: 4 points

Enter and execute the following code.

```
for x in [3 ,6 ,9 ,12 ,15 ,18]:
    print(x , end=" ")
```

(a) (2 points) Rewrite this code using the **range()** function.

```
for x in range(3, 18, 3):
   print(x, end=” “)
```

(b) (2 points) Why would you use the **range()** function when you could just list the numbers?

The range function allows the user to programmatically generate any sequence of numbers regardless of how long it is. This is a more concise option when compared to explicitly hardcoding a list which might be very long.

FYI: The **`str()`** function converts its argument into a string.

Question 18: 2 points

Read through the code and determine what it does.

```
favorite = input("Enter your favorite ice cream flavor: ")
for x in range (1 ,5):
      print( str (x) + "." ,     favorite , end="\t")
```

(a) (1 point) What does the program do?

The program concatenates a number (1 – 4) with a period (.) and the value of favorite and prints that in the command line separated by tabs. For example, if `favorite = 'vanilla'`, the program prints:

```
 1. vanilla    2. vanilla    3. vanilla    4. vanilla
```

(b) (1 point) Why is the **`str()`** function needed in the **`print`** statement?

Integers can not be concatenated to strings. So the value of x must be converted to a string.

Question 19: 3 points

Complete the arguments in the following **`range()`** function so that the code prints the **even** numbers between 100 and 200, inclusive.

```
for x in range(  100, 201, 2   ):
      print(x)
```

Question 20: 3 points

Complete the arguments in the following **`range()`** function so that it prints: **`5 4 3 2 1 0`**.

```
for x in range(  5, -1, -1   ):
      print(x)
```

FYI: An accumulator is a variable that stores the sum of a group of values.

Question 21: 5 points

Examine the following code segment.

```
1   total = 0
2   for x in range (5):
3       number = int (input("Enter a number: "))
4       total += number
5   print("The total is:" , total )
```

(a) (1 point) Why is the variable **total** initialized to **0** in the first line of code?
total is set to zero because there are no values to keep track of prior to the loop. Only values provided in response to the prompt in the loop body are added to total.

(b) (1 point) Explain what line 3 of the code does.
It collects user input and converts it to an integer.

(c) (1 point) Explain what line 4 of the code does.
It increments total with the user input.

(d) (1 point) How many numbers does the program prompt for?
It prompts for 5 numbers.

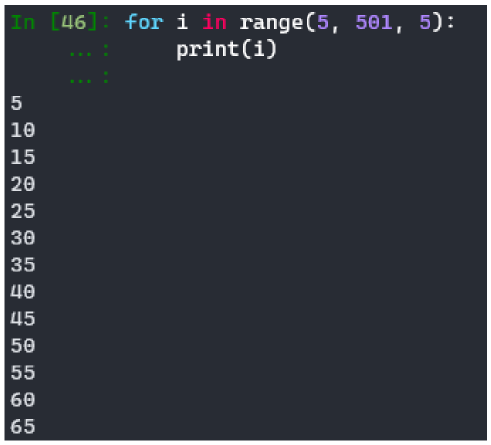
(e) (1 point) What is the accumulator in the code segment?
`total`

Question 22: 1 point

Is it better to use a **for** loop when you know the number of times the loop should be executed or when you do not know?
For loops should be used when the number of iterations is known.

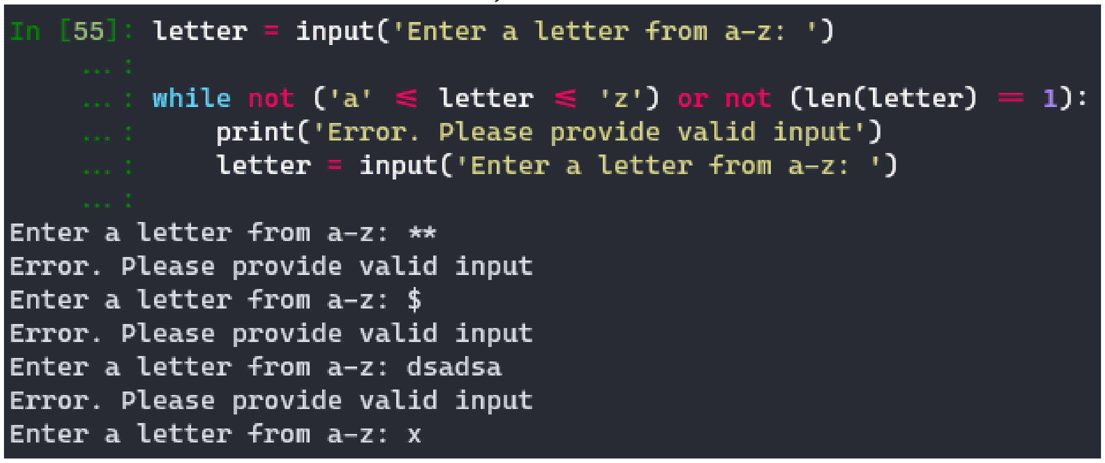
Use the Python Interpreter for these questions. Submit one or more screenshots for each. Show your full source code and an example of the code executing.

Question 23: 3 points


```
In [46]: for i in range(5, 501, 5):
    ...:     print(i)
    ...:
5
10
15
20
25
30
35
40
45
50
55
60
65
```

Question 24: 4 points

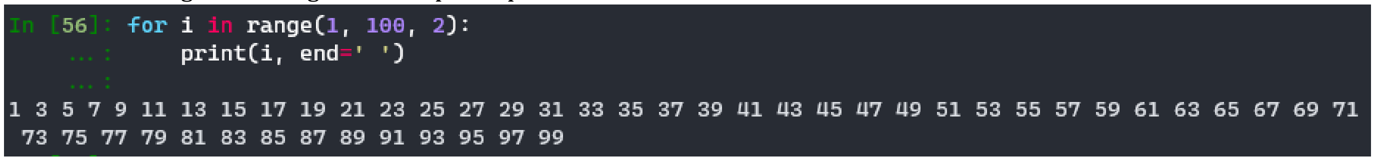
Write code segment that prompts the user for a letter from ’a-z’. As long as the character is not between ’a-z’, the user should be given a message and prompted again for a letter between ’a-z’. Make sure not to accept words that start with letters. You should reject a word like robot.



```
In [55]: letter = input('Enter a letter from a-z: ')
    ...:
    ...: while not ('a' <= letter <= 'z') or not (len(letter) == 1):
    ...:     print('Error. Please provide valid input')
    ...:     letter = input('Enter a letter from a-z: ')
    ...:
Enter a letter from a-z: **
Error. Please provide valid input
Enter a letter from a-z: $
Error. Please provide valid input
Enter a letter from a-z: dsadsa
Error. Please provide valid input
Enter a letter from a-z: x
```

Question 25: 3 points

Write a code segment using a FOR loop that prints all odd numbers between 1 and 100.



```
In [56]: for i in range(1, 100, 2):
    ...:     print(i, end=' ')
    ...:
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67 69 71
 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```