Branching and Repetition (Looping)

Discussion 1

What is the output of the following Python program?

```
value = 6
if value % 2 == 0:
   print("first", value)
elif value % 3 == 0:
   print("second", value)

while value <= 9:
   value = value + 1
   if value == 8:
      continue
   else:
      pass
   print ("third", value)
else:
   print ("fourth", value)

print("fifth", value)
```

first 6
third 7
third 9
third 10
fourth 10
fifth 10

Discussion 2

The following program calculates the number of input strings with letter ‘a’, and end the program when the input string is “####”. Here is an expected sample run:

Sample :
enter a string (enter #### to stop): apple
enter a string (enter #### to stop): banana
enter a string (enter #### to stop): strawberry
enter a string (enter #### to stop): book
enter a string (enter #### to stop): ####
3 strings with letter 'a'

```
while True:
    str = input("enter a string: ")
    for letter in str:
      if letter == 'a':
          break
      count +=1

    print(count , "strings with letter 'a'")
```

There are some errors in the above program. Please indicate where the errors are and how to correct them.

Discussion 3

Write a simple Python program to implement the Pseudocode of FizzBuzz problem in discussion #1.

Discussion 4

Write a Python program that reads an integer from the user, which is the width of the pattern below, and then prints out the pattern. Suggestion: use nested **for** loops. Hint: **print("*",end="")**.

```
Please enter pattern width: 5

*
**
***
****
*****
****
***
**
*
```

Further discussion: Is it possible to use **for** only twice? Or even once? (of course no **while**)

Branching and Repetition (Looping)

Guideline for instructor

Discussion 1

Answer	Discussion
first 6	`value % 2 == 0 is True, enter True, although` `value % 3 == 0 is True, it is skipped`
third 7	6 + 1 =7, display “third 7” **pass** has no effect (does nothing) but helps in **indicating** an empty statement/ suite/ block. When 7 + 1, value becomes 8, • The **continue** statement continues with the next iteration of the loop. • Skip some portion of the **while** suite we are executing and have control flow back to the beginning of the **while** loop. • Exit early from this iteration of the loop (not the loop itself), and keep executing the **while** loop.
third 9	
third 10	
fourth 10	While-else • It is entered after the **while** loop’s Boolean expression becomes False. • This entry occurs even when the expression is initially False and the **while** loop has never run. • A handy way to perform some final tasks when the loop ends normally.
fifth 10	Statement after while loop

Branching and Repetition (Looping)

Discussion 2

errors	Code with errors:

• variable count has not been initialized
• Wrong use of break, results in all letter not equal to 'a' contribute to count
• Condition update is missing, looping never end. Sentinel condition is missing.
• Wrong Indentation of (print ()) causes output of every input, instead of being summary of result analysis
• Not error, but need avoid: str is a built in function.
	`while True:`
	`  str = input("enter a string: ")`
	`  for letter in str:`
	`  if letter == 'a':`
	`  break`
	`  count +=1`
	
	`  print(count , "strings with letter 'a'")`

Suggested code (two versions)

```
count=0
while True:
    str_sentinal  = input("enter a string (enter #### to stop): ")
    if str_sentinal =="####":
        break
    for letter in str_sentinal :
      if letter == 'a':
          count +=1
          break


print(count , "strings with letter 'a'")
```

```
count = 0
str_sentinal = input("enter a string (enter #### to stop): ")
while str_sentinal != "####":
    for letter in str_sentinal:
      if letter == 'a':
          count +=1
          break
    str_sentinal = input("enter a string (enter #### to stop): ")
print(count , "strings with letter 'a'")
```

Discussion 3

Suggested Solution:

```
for num in range(1, 21):
    if num % 15 ==0:
        print("FizzBuzz")
    elif num %3 ==0:
        print("Fizz")
    elif num %5==0:
        print("Buzz")
    else:
        print(num)
```

Discussion 4

Suggested Solution:

Using two nested for loops: one for upper half and the other for lower half

```
width = int(input("Please enter pattern width: "))


for i in range(1, width+1):
    for j in range(i):
        print("*",end="")
    print()
for i in range(width-1,0, -1):
    for j in range(i):
        print("*",end="")
    print()
```

max = input ("Please enter an integer for the maximum width of the pyramid: ")

```
max_int = int(max)

for count in range(1, max_int+1):
    print(count * "*")

for count in range(max_int, 0, -1):
    print(count * "*")
```

Revision:

print() # print a line break

print("*",**end**="")

Python's print() function comes with a parameter called 'end'. By default, the value of this parameter is '\n', i.e. start a new line. You can end a print statement with any character/string using this parameter to replace '\n'. end="" indicates continue display in the same line.

Counter controlled loop: The number of repetitions can be known before the loop body starts; just repeat the loop on each element in a preset sequence.

Program design:

for loop

Guidance: This question aims to make sure the students understand nested loops. For the discussion part on using fewer "for" loops, it depends on the available time… may just give them some hints and leave this part as a take-home exercise.

Using just two for loops: outer for goes through each row

```
width = int(input("Please enter pattern width: "))


for i in range (1,width * 2):
    if i < width:
        count = i
    else:
        count = width * 2 - i
    for j in range (count):
```

```
        print("*", end ="")
    print()
```

Using only one for loop

```
width=int(input('Please enter pattern width: '))
for i in range (1,width * 2):
    if i < width:
        count = i
    else:
        count = width * 2 - i
    print("*"*count)
```