*In this workshop, you'll gain practice with while loops and other aspects of Python that we have seen so far. Your workshop leader will guide you through the process. Please **do not** work ahead.*

1. Consider the following code:

```
number = int(input("Enter a number: "))
x = 1
while x <= number:
    if x % 10 == 0:
        print(x)
    else:
        print(x, end=" ")
    x += 1
```

   (a) Given an input of 11, what does the program above print?

   (b) The following program is similar to the one above, but does not print anything. What would it do if it was run in the Python interpreter?

```
number = 12
while number <= 10:
    print(number, end=" ")
    number += 1
```

   (c) Rewrite the code above to fix the problem. Note: there are a number of ways that the code could be revised to fix the problem; you may choose any reasonable way.

2. The following steps will create a program that prompts the user to enter a number between 1 and 10. As long as the number is outside this range the program should re-prompt the user for a valid number. Complete the steps that follow:

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

   (b) Write a *Boolean expression* that tests whether the number entered by the user is **NOT** valid (not between 1 and 10, inclusive):

   (c) Use the Boolean expression you wrote above to write a `while` loop that executes when the user input is invalid. The body of the loop should tell the user that they typed an invalid value and then reprompt them for new input.

   (d) Say that the "whole" program is run (i.e., the initial prompt plus the while loop you just wrote), and a user types the value 1. How many times does the `while` loop execute?

   (e) Say that the "whole" program is run (i.e., the initial prompt plus the while loop you just wrote), and a user types the value -100, 100, 4. How many times does the `while` loop execute?

3. Consider the following program:

```
doAgain = 'y'
while doAgain == 'y':
    word = input("What's the word? ")
    print("Your word reversed is " + word)
    doAgain = input("Type 'y' to enter another word or anything else to quit.")
print("Ok!")
```

(a) Given inputs of "word" followed by "nope", what would be printed by the above program?

(b) Notice that the program states that the user-input word should be printed in reverse, so "word" should appear as "drow". Fix the program so that the correct output appears. (Hint: use string slicing.)

(c) Say that you want to change the way the loop is controlled and the first two lines are modified as:

```
done = False
while not done:
```

How should the rest of the program be rewritten to make it behave in the same way as originally written?

4. Consider the code below:

```
userInput = input("Enter a string that contains only letters: ")
if userInput.isalpha():
    print("Your string is valid")
else:
    print("Your string is invalid")
```

(a) Identify one string that would cause "Your string is valid" to be printed.

(b) Identify one string that would cause "Your string is invalid" to be printed.

(c) If you wanted to modify the code to declare that a string composed only of digits is valid, what would need to change?

(d) There is no built-in method to check that a string "looks like" a valid floating point number. Say that we wrote a function to try to do this as follows:

```
def isfloatstr(s):
    decimal = s.find('.')
    return decimal >= 0 and s[:decimal].isdigit() and s[(decimal+1):].isdigit()
```

Briefly explain what this code is attempting to do. You may wish to consult the Python documentation for the `find` method.

(e) The function `isfloatstr` has some problems. Identify a string that, if passed to `isfloatstr`, would be identified as *invalid* when it is actually valid.

(f) Revise the `isfloatstr` function to correctly identify strings with floating point numbers.

5. Write a function named `twoChars` (similar to the same function from workshop 5) that accepts two parameters: a string, which can be of any length, and a string composed of a single character. The function should determine whether there are two distinct occurrences of the given character in the string and return a boolean. It should perform its task in a case-insensitive manner. *Use string methods* in your answer; in particular, the `find` or `index` methods may be helpful, as well as other string methods.