

for loops and while loops

You must get checked out by your lab CA **prior to leaving early**. If you leave without being checked out, you will receive 0 credits for the lab.

Restrictions

The Python structures that you use in this lab should be restricted to those you have learned in lecture so far. Please check with your teaching assistants in case you are unsure whether something is or is not allowed!

Create a new python file for each of the following problems.

Your files should be named `lab[num]_q[num].py` similar to homework naming conventions.

Problem 1: *Loops galore!*

Solve this problem by hand.

What would be printed for each of the following code snippets? If there is an infinite loop state that.

(a)

```
for i in range(0,10):  
    print(i)
```

(b)

```
for i in range(1, 13, 2):  
    print(i)
```

(c)

```
for i in range(43, 31, -3):  
    print(i)
```

(d)

```
num = 1  
while num < 28:
```

```
print(num)
num *= 3
```

(e)

```
num = 13
while num < 23:
    if num % 2 == 0:
        print(num)
    else:
        print("fizz")
    num += 1
```

Problem 2: *Multiple Use Calculator*

This problem is a brief extension on last weeks single use calculator (**Lab 3 Problem 4**). This extension will let the user use the calculator as many times as they want.

1. Begin by copying your solution from the *Single Use Calculator* into a new file.
2. Now make the additions necessary to have the calculator run until the user inputs Q to quit. Otherwise they should just hit enter to continue making calculations.

The following is the expected output of the program:

```
This is a four operation calculator.
Hit enter to continue and Q to quit calculator:
Enter your first number: 2
Enter the operation (+, -, *, /): +
Enter your second number: 3
2.0 + 3.0 = 5.0
Hit enter to continue and Q to quit calculator:
Enter your first number: 3
Enter the operation (+, -, *, /): *
Enter your second number: 9
3.0 * 9.0 = 27.0
Hit enter to continue and Q to quit calculator:
Enter your first number: 3
Enter the operation (+, -, *, /): 2
Enter your second number: 3
3.0 2 3.0 is an invalid operation.
Hit enter to continue and Q to quit calculator: Q
Goodbye!
```

This should be a relatively small addition to the logic of your code. Think about what type of loop to use to work for any number of calculation requests.

Problem 3: *I've got the power(s)*

This problem will be printing the powers of a given base and every number up to the maximum power.

1. ask the user for two **positive integers**, a **base**, and a **power**, and
2. print, one by one, the result of raising that base by every power from 0 to **power**.

Here's an example of this programs execution:

If, for example, the user enters **2** and **7**:

```
Please enter a positive integer to serve as the base: 2
Please enter a positive integer to serve as the highest power: 7
2 ^ 0 = 1
2 ^ 1 = 2
2 ^ 2 = 4
2 ^ 3 = 8
2 ^ 4 = 16
2 ^ 5 = 32
2 ^ 6 = 64
2 ^ 7 = 128
```

If the user instead enters **10** and **3**:

```
Please enter a positive integer to serve as the base: 10
Please enter a positive integer to serve as the highest power: 3
10 ^ 0 = 1
10 ^ 1 = 10
10 ^ 2 = 100
10 ^ 3 = 1000
```

A few things to keep in mind:

- You may assume the user input will be numerical.
- Since the options for the **base** and the highest **power** are literally infinite, the use of a **loop** will come in very handy.
- If the user attempts to enter a negative number into your program, **print an error message instead of the list of powers.**
- If the user attempts to enter a float into your program, **print an error message instead of the list of powers.** There's a few ways to check for this, but make sure it's something that you have learned in class already!

```
Please enter a positive integer to serve as the base: 3.4
Please enter a positive integer to serve as the highest power: 4
ERROR: Both values must be POSITIVE INTEGERS.
```

```
Please enter a positive integer to serve as the base: -345324325436
Please enter a positive integer to serve as the highest power: 2
ERROR: Both values must be POSITIVE INTEGERS.
```

Problem 4: *Higher and Higher*

Write a program, that will find the largest value out of X-number of user-entered **positive** values. This will be done in three steps:

1. Asking the user to enter the number of values they want to enter.
2. Asking the user to enter the values.
3. Printing the largest of these values (what happens if the user entered 0 in step 1?)

This behavior will look like this:

```
Please enter how many positive values you want to consider: 5
Enter your values:
24
27.5
30.234
100
1
The largest of these values is 100.0.
```

Do NOT use lists for this problem, if you know what lists are. Again, that's no fun.

Problem 5: *Guess the Number*

This is a fun little number guessing game that focuses on both loops and conditional statements.

1. Generate a random target number between 0 and 100 inclusive.
2. Allow the user to guess the number a maximum of 5 times.
3. Each time the user guesses print a message telling them whether they guessed to high or too low
4. At the end either tell the user that they guessed the correct number or that they did not

Below is an example of the programs expected output **when the player loses**:

```
Enter your guess: 50
Your guess is too high
Enter your guess: 25
Your guess is too low
Enter your guess: 35
Your guess is too high
Enter your guess: 27
Your guess is too low
Enter your guess: 29
Your guess is too low
```

```
Enter your guess: 32
Sorry you could not guess the correct number was 31
```

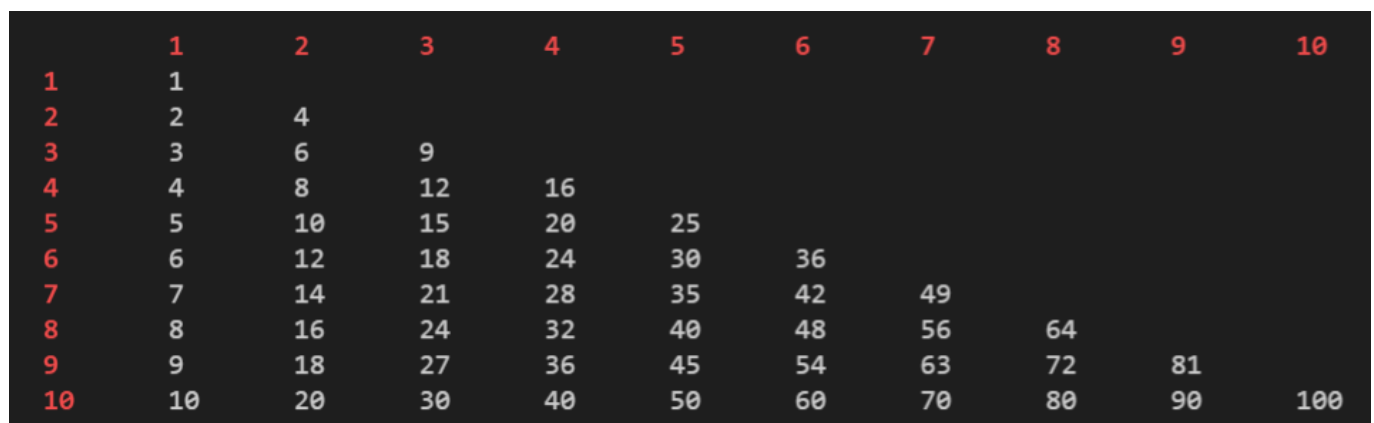
Below is an example of the programs expected output when the player wins:

```
Enter your guess: 50
Your guess is too low
Enter your guess: 75
Your guess is too low
Enter your guess: 85
Your guess is too high
Enter your guess: 80
Your guess is too high
Enter your guess: 76
Your guess is too low
Enter your guess: 78
Your guess is correct the number was 78
```

Think about what type of loop to use for this and how to structure the loop before you start coding

Problem 6: *Half Times Table*

A times table, or multiplication table, is a table of the products of a certain range of numbers (commonly from 1-10). Each cell of the table is the product of its row and column. The following is an image of a half of a times table from 1-10.



	1	2	3	4	5	6	7	8	9	10
1	1									
2	2	4								
3	3	6	9							
4	4	8	12	16						
5	5	10	15	20	25					
6	6	12	18	24	30	36				
7	7	14	21	28	35	42	49			
8	8	16	24	32	40	48	56	64		
9	9	18	27	36	45	54	63	72	81	
10	10	20	30	40	50	60	70	80	90	100

Figure 1: *Half Times Table*

For this problem, you will be displaying the half times table results (the numbers in white of the image above). However, there are a few rules:

- For every row (horizontal line), there is a 50% chance it will be displayed in reverse order. For example, for row 4, it could be displayed as 4 8 12 16, or it could be displayed as 16 12 8 4.
- Use **for loops** instead of while loops for this.
- You may not modify the `end` parameter of the `print` function (if you don't know what this means, don't worry about it.)

Your output will look something like this:

1									
4	2								
9	6	3							
16	12	8	4						
5	10	15	20	25					
6	12	18	24	30	36				
49	42	35	28	21	14	7			
64	56	48	40	32	24	16	8		
9	18	27	36	45	54	63	72	81	
100	90	80	70	60	50	40	30	20	10

Hints:

- The bulk of your program will be in one big for loop, which will go through every row of the table.
- For each row of the table, you will be doing different things depending on a certain condition. What does that sound like out of the structures you have learned?
- For each row, you may also want to have another loop that gives you the values of the products for each cell (row * column)
- Directly printing out a product as soon as you find it is not going to work, since `print()` will add a new line. Instead, you can add each product to a string that stores all the values in a single line, and printing that string after finishing one line. (If you define this string in the right place, it will reset by itself after each line.)
- You can just add a space in between each cell of the table, or if you want it to look nice, add a `"\t"` character in between each cell instead.