Review for finals #3
Due Tuesday 5/24/22 (7th period), Wednesday 5/25/22 (6th period)

As you know, we are reviewing Python. Your final exam will be on Python. Today we're going to review for loops and the modulo command.

for loops

A for loop iterates a list of items ("iterates" means "goes through a list"). For example, this code:

```
numList = [1,2,3,4]
for x in numList:
    print(x)
```

generates this output:

```
1
2
3
4
```

Here is another for loop, this time with a non-numerical list:

```
farmList = ["Goat","Pig","Chicken"]
for animal in farmList:
    print(animal)
```

The above code generates the following output:

```
Goat
Pig
Chicken
```

Finally, if you want to make a for loop and just have a number that goes up, one at a time, you can do the following. This code:

```
for x in range(0,5):
    print(x)
```

generates the following:

```
0
1
2
3
4
```

The following code prints out the farm animals but in a more complicated way than previous code I used above. See if you can understand it:

```
farmList = ["Goat","Pig","Chicken"]
for x in range(len(farmList)):
    print(farmList[x])
```

The above for loop uses the range call which puts the numbers from zero to 1 less than the length of the farmList variable into x, one at a time. It's definitely easier to just say "for x in farmList:" if you just want the items one at a time, but I wanted you to see all the options. Sometimes you need to know the index number of each item in a list.

That's it for **for loops**.

Now let's review the modulo operator

The modulo operator or % symbol divides one number by another and gives you the remainder. For example, 10%2 gives you 0 (10 divided by 2 gives you 5 evenly, that is, there is no remainder). 10%3 gives you 1, because when you divide 10 by 3 you end up with a remainder of 1.

Here we take user input (a number) and find all the factors of that number using the modulo operator (%). (Factors are numbers that divide into a number evenly. For example, 1, 2, 3, 4, 6, and 12 are all factors of 12.)

This code:

```
number = int(input("Please enter a number: "))
print("You entered",number)
print("We will now find all factors of",number)
for divisor in range(1,number+1):
    if number%divisor==0:
        print(divisor,"is a factor of",number)
```

generates:

```
Please enter a number: 48
You entered 48
We will now find all factors of 48
1 is a factor of 48
2 is a factor of 48
3 is a factor of 48
4 is a factor of 48
6 is a factor of 48
8 is a factor of 48
12 is a factor of 48
16 is a factor of 48
24 is a factor of 48
48 is a factor of 48
```

(continued on next page)

Today I would like you to do something similar to the program I just showed you:

- Create a program on repl called Review 3.
- Ask the user for a number (use the variable name "number")
- Ask the user for a second number (use the variable name "divisor") that is smaller than the first number
- If it is not smaller, start over
- Use one or more try/except structures to make sure that the user is actually entering numbers for number and divisor above.
- Once you have the two numbers, make a for loop go from 1 to number
- Report all numbers less than number that are evenly divisible by divisor
- Report the count of how many numbers were found like this: "There are 6 numbers smaller than 43 divisible by 7"
- Finally, use the following list variable (copy it out of this PDF) to create output that looks like the output shown below. Hint: you will use a for loop, and a print call.

```
nums = [2, 4, 6, 7, 7, 8, 8, 8, 8, 9, 10, 10, 7, 8, 7, 6, 4, 3, 3, 3, 2, 2]
```

Sample output:

```
Please enter a number: 43
Please enter a smaller number: 7
7 is divisible by 7
14 is divisible by 7
21 is divisible by 7
28 is divisible by 7
35 is divisible by 7
42 is divisible by 7
There were 6 numbers less than 43 divisible by 7

And now a silhouette of a person:

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