Count Controlled Loops
A count controlled loop is a repetition structure that iterates a specific number of times.

In contrast, a condition controlled loop iterates a variable number of times – we control the # of iterations through our Boolean condition.
Count Controlled Loops

You can write a count controlled loop using a while() loop.

For example:

```python
counter = 0
while counter < 5:
    print ("This will print 5 times")
    counter += 1
```
Python (and all other programming languages) have special structures which can be used to implement a count controlled loop without needing to use a condition controlled loop (though you could always use a condition controlled loop if you wanted to)
For Loops

The “for” loop is Python’s native count controlled loop.

For Example:

```python
for num in [1,2,3,4,5]:
    print ("This will print 5 times")
```
For Loops

```
for variable in [value1, value2, etc]:
    statement
    statement
    statement
```

- The "for" keyword begins a loop.
- "in" keyword.
- Target variable.
- List of items to be iterated over.
- Indentation indicates that the statements under the while should be repeated.
- Statements to be executed.
For Loops

The “for” loop will iterate once for each item defined in the list passed to it when the loop begins.

Lists in Python are defined by the square bracket characters “[“ and “]”. Items in a list are separated by a comma.

The first time a “for” loop iterates the target variable will assume the value of the first item in the list.

The second time a “for” loop iterates the target variable will assume the value of the second item in the list.

This continues until you reach the end of the list.
For Loops

```python
for c in [1,2,3,4]:
    print (c)
```

1
2
3
4
For Loops

We will talk more about lists near the end of the semester. With that said, lists can contain collections of different kinds of data.

For example:

```python
for name in ['Craig', 'John', 'Chris']:
    print ('The current user is:', name)
```
A bug collector collects bugs every day for seven days.

Write a program that keeps a running total of the number of bugs collected during the seven days. The loop you write should ask for the number of bugs collected for each day, and when it finishes it should display the total number of bugs collected.
Write a program that iterates over the following student names:

John
Mary
Michael
Sophie

Ask the user to input a test score for each student

Calculate the average test score for the class
Challenge

Rewrite the following loop as a “for” loop:

```python
x = 0

while x < 5:
    print (“hi”)
    x += 1
```
Challenge

Rewrite the following loop as a “while” loop:

```python
for x in [10,20,30,40]:
    print ("hi")
```
range() function

So far we have been iterating over lists using pre-defined values in our for() loops.

For Example:

```python
for x in [1,2,3,4,5]:
    print ('hi')
```

The `range()` function lets you dynamically generate lists based on criteria that you define.
range () function

for i in range(5):
    print ('iteration #', i)

iteration # 0
iteration # 1
iteration # 2
iteration # 3
iteration # 4
The range() function takes at least one argument. In its simplest form it takes a single integer.

The range() function returns an “iterable”, which is a Python data type similar to a list.

When passed a single integer the range function will generate an iterable that will cause a for() loop from 0 to the number specified minus one.
# range () function

<table>
<thead>
<tr>
<th>range () call</th>
<th>iterable</th>
</tr>
</thead>
<tbody>
<tr>
<td>range(5)</td>
<td>[0,1,2,3,4]</td>
</tr>
<tr>
<td>range(10)</td>
<td>[0,1,2,3,4,5,6,7,8,9]</td>
</tr>
</tbody>
</table>
range () function

You can pass additional parameters to the range() function to cause it to behave differently.

For Examples:

```python
range(1, 5)  # set a start and end value for the range
            # [1, 2, 3, 4]

range(5, 10)  # [5, 6, 7, 8, 9]

range(0, 10, 2)  # set a start, end and step (or increment) value
                 # [0, 2, 4, 6, 8]

range(1, 10, 2)  # [1, 3, 5, 7, 9]
```
Loop Target

In a for loop we generally use the target variable as a reference value for some kind of calculation.

Remember that the value of the target variable changes with each iteration of the loop.
Challenge

Write a program that calculates the square of the numbers between 1 and 10

Print out the number and its square as your loop iterates

<table>
<thead>
<tr>
<th>Number</th>
<th>Square</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
</tr>
</tbody>
</table>
Challenge

Write a program that prints out the following pattern of characters:

**
****
******
********
**********
Challenge

Write a program that asks the user to enter in an integer

Then find all numbers between 1 and 10,000 that are evenly divisible by that number
Challenge

Extensions:

Extend your program to collect two integers when it starts up

Find all numbers in the specified range that are divisible both of the two supplied numbers

Extension: print your results such that you print 10 #'s per line
User Controlled Ranges

In many cases a programmer knows how many iterations he or she needs in order to accomplish a desired task.

However, sometimes we need to ask the user to control the # of iterations within a loop.

You can easily do this by substituting a variable within the range() function to control the start, end and step values of the iterable that will be generated.
Challenges

Write a program that generates random lottery numbers for the user

Ask the user for the number of digits they need as well as the high and low value of each digit (i.e. 6 digit number with digits ranging from 1 to 60)

Generate the desired lottery number
Reverse ranges

The step value passed to the `range()` function does not necessarily have to be positive.

If you pass a negative step value to the `range()` function it will count backwards for you.
Write a countdown program that prompts the user for a max value (i.e. 30)

Print out a countdown from that number down to zero, then print “blast off!”
Nested Loops

A nested loop can be described as a “loop inside of a loop”

It’s the same idea as nested selection statements (“if” statements inside other “if” statements)
Challenge

Write a program that prints out an Addition table for the number 5.

For example:

5 plus 1 is 6
5 plus 2 is 7
5 plus 3 is 8
...
5 plus 10 is 15
Challenge

Write a program that prints out an Addition table for the numbers 5 and 6.

For example:

5 plus 1 is 6
5 plus 2 is 7
5 plus 3 is 8
...
5 plus 10 is 15

6 plus 1 is 7
6 plus 2 is 8
6 plus 3 is 9
...
6 plus 10 is 16
Write a program that prints out the Addition tables for the numbers 1 through 10

Extend your program to allow the user to type in a range of Addition tables they want printed.

For Example:

Addition Table Generator 2000!
Enter the first number in your range: 1
Enter the last number for your table: 20
Challenge

Write a program that prints out a multiplication table for the numbers 1 through 5:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>6</td>
<td>9</td>
<td>12</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>25</td>
</tr>
</tbody>
</table>

Can you modify your program to print out a multiplication table for the numbers 1 through 10?
Write a program that prints out every possible time value for a single day.

Print out the hours, minutes and seconds starting at midnight and continue on to 11:59.59 PM.

Output each value as follows:

0:0:0
0:0:1
0:0:2
...
23:59:59
Nested Loops

Some notes on nested loops:

The innermost loop will iterate through all its iterations for every single iteration of an outer loop.

Inner loops complete their iterations faster than outer loops.

To get the total number of iterations of a nested loop, multiply the number of iterations of all the loops.
Challenge

Write a program that lets a teacher calculate grades for his or her class.

Ask the teacher for the # of students in class as well as the # of assignments.

Allow the teacher to input the desired values and calculate the average score for each student based on the information given.
Challenge

Write a program that prints the pattern at the bottom using nested loops
Challenge

Reproduce the pattern at the bottom using nested loops

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
Challenge

Write a program to test to see if a given number is prime (this does not require a nested loop)

Next, use this program to all prime numbers between 2 and 1,000
next steps:

begin “Self Paced Learning Module # 6”