Looping and Matrix

1. Write a function f1(n) that prints the following triangle.

```
>>> f1(5)
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

>>> f1(0)

>>> f1(1)
1
```

Show Solution

2. Write a function f2(n) that prints the following triangle.

```
>>> f2(3)
1
2 3
4 5 6

>>> f2(0)

>>> f2(1)
1

>>> f2(5)
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

3. Write a function f3(n) that prints the following triangle.

```
>>> f3(3)
1
2 3
4 5 6
2 3
1

>>> f3(4)
1
2 3
4 5 6
7 8 9 10
4 5 6
2 3
1

>>> f3(0)
>>> f3(1)
1
```

4. Write a function f4(n) that prints the following triangle.

```
>>> f4(3)
1
2 3
4 5 6
7 8
9

>>> f4(0)
>>> f4(1)
1
```

5. Write a function f5(matrix) that prints the sum of every row in the matrix.

```
>>> f5([[1,0],[0,1]])
1
1

>>> f5([[1,2,3],[4,5,6]])
6
15

>>> f5([[1],[2],[3],[4]])
1
2
3
4
```

6. Write a function f6(matrix) that returns the sum of all the elements in the matrix.

```
>>> f6([[1,0],[0,1]])
2

>>> f6([[1,2,3],[4,5,6]])
21

>>> f6([[1],[2],[3],[4]])
10
```

7. Write a function f7(matrix) that returns the product of all the elements in the matrix.

```
>>> f7([[1,0],[0,1]])
0

>>> f7([[1,2,3],[4,5,6]])
720

>>> f7([[1],[2],[3],[4]])
24
```

8. Write a function f8(matrix) that will print the odd numbers in the matrix with each row on one line.

```
>>> f8([[1,0],[0,1]])
1
1

>>> f8([[1,2,3],[4,5,6]])
1 3
5

>>> f8([[1],[2],[3],[4]])
1

3
```

9. Write a function f9(matrix1, matrix2) that will return the sum of matrix1 and matrix2. Assume matrix1 and matrix2 have the same dimensions.

```
>>> f9([[1,0],[0,1]],[[1,0],[0,1]])
[[2, 0], [0, 2]]

>>> f8([[1,2,3],[4,5,6]],[[-1,-1,-1],[-1,-1,-1]])
[[0, 1, 2], [3, 4, 5]]

>>> f15([[1],[2],[3],[4]],[[4],[3],[2],[1]])
[[5], [5], [5], [5]]
```

10. Write a function f10(matrix1, matrix2) that will return the product of matrix1 and matrix2. Assume len(matrix1) == len(matrix2[0]).

```
>>> f10([[1,0],[0,1]],[[1,0],[0,1]],)
[[1, 0], [0, 1]]

>>> f10([[1,2,3],[4,5,6]],[[-1,-1],[-1,-1],[-1,-1]])
[[-6, -6], [-15, -15]]

>>> f10([[4,3,2,1]],[[1],[2],[3],[4]])
[20]
```

11. Write a function f11(matrix) that returns True if the matrix is the identity matrix, and False otherwise. Assume len(matrix) == len(matrix[0]).

```
>>> f11([[1]])
True

>>> f11([[1,0,0],[0,1,0],[0,0,1]])
True

>>> f11([[1,0,0],[0,1,5],[0,0,1]])
False
```

12. Write a function f12(rows, cols) that returns a two dimensional list where each element corresponds to how many adjacent neighbors it has. Neighbors are defined as spaces above, below, to the left, and to the right.

```
>>> f12(3,3)
[[2, 3, 2], [3, 4, 3], [2, 3, 2]]

>>> f12(5,1)
[[1], [2], [2], [2], [1]]

>>> f12(5,0)
[[], [], [], [], []]

>>> f12(0,5)
[]

>>> f12(2,2)
[[2, 2], [2, 2]]
```