List of lists:

Some of a list themselves are a list.

If list is like a vector of Math, list of lists is like a matrix.

```
a = [
      [11, 22, 33],
      [44, 55, 66]
]
```

This is an example of a list of lists.

```
print(len(a), len(a[1]) # 2 3
```

a is a list of 2 elements, a[1] is a list of 3 elements.

The list of lists need not be rectangular.

```
b = [
      [ 1, 2, 3 ]
      [ 4, 5, 6, 7],
      [ 8, 9]
]
```

Let us try some simple examples of lists of lists.

Example 1: Generate an identity matrix.

Version 1:

This creates an empty list. Adds n empty rows. Appends an element each time in the innermost loop.

The element is generated by the expression `(i//j) * (j//i).`

This expression will be **1** `if(i = j)` and **0** otherwise.

This is based on the trick of integer division - not a good idea.

```
# file : 5_list_identity.py
"""
# generates identity matrix
# bad program - depending on integer division
# version 1:
"""
```

```
n = 4
a = [] # empty matrix
for i in range(1, n + 1) :
      a.append([]) # add a row each time
      for j in range(1, n + 1) :
            a[i-1].append((i//j) * (j//i))  # trick ?

for x in a :
      for e in x :
            print(e, end = " ")
      print()
```

Version 2:

We create the list of lists the way we did last time. But we append the element based on whether it is an element on the diagonal or otherwise. There are n squared comparisons and n squared appending.

```
# file : 6_list_identity.py
# version 2
n = 4
a = []
for i in range(1, n + 1) :
      a.append([])
      for j in range(1, n + 1) :
            if i == j :
                  a[i-1].append(1)
            else:
                  a[i-1].append(0)
for x in a :
      for e in x :
            print(e, end = " ")
      print()
```

Version 3:

We avoid n * n comparisons. We put 0 every in the matrix and then change the elements on the diagonal to 1. This has n squared appending and n assignments. This is definitely easier to understand and is efficient.

```
# file : 7_list_identity.py
# version 3
```

```
n = 4
a = []
for i in range(1, n + 1) :
      a.append([])
      for j in range(1, n + 1) :
            a[i-1].append(0)
      a[i - 1][i - 1] = 1
for x in a :
      for e in x :
            print(e, end = " ")
      print()
```

Example 2: display a Pascal triangle.

```
# file : 8_disp_Pascal.py
a = [
      [1],
      [1, 1],
      [1, 2, 1],
      [1, 3, 3, 1],
      [1, 4, 6, 4, 1],
      [1, 5, 10, 10, 5, 1]
]
n = 5
# display Pascal triangle
#print(a)

for i in range(n + 1) : # go thro n + 1 rows from 0 to n
      # output # of spaces which decreases as we move to the next row - as i
increases
      print("   " * (n - i), end = "")
      for j in range(i + 1): # display i + 1 elements
            print("{0:6}".format(a[i][j]), end = "")
      print()

$ python 8_disp_Pascal.py
                     1
                  1     1
               1     2     1
            1     3     3     1
```

```
        1     4     6     4     1
    1     5    10    10     5     1
```

This program indicates how we can control the display.

file: 9_format.py

```
# formatting

x = 10; y = 20
print(x, y, x + y)
print("{0:5} and {1:5} is {2:6}".format(x, y, x + y))
# {0:5} output the zeroth argument of format using width of 5 characters
# {1:5} output the first argument of format using width of 5 characters
# {2:6} output the second argument of format using width of 6 characters
```