Python

Patterns, Functions for Arrays and Strings, 4 ways to use Arrays, 2D/ND arrays, Modules, Advanced I/O

Patterns (using loops and functions)

Square, triangles of stars

Number pyramids

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

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

```
1
12
123
1234
12345
```

```
    1
   121
  12321
 1234321
123454321
```

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

Making functions for String/Arrays

String manipulation functions

toupper
tolower
toproper
reverse
substr
concatenate
remove extra blanks (WS)
pad required blanks
search, at
toint, tobool, tofloat, toVector
fromInt, fromVector
____toString

Array manipulation functions

accumulation
max, min
location of min, max
search, location of
joining arrays
reverse
subarray
len of data in array
move subarray to other location

sorting
indexing

Arrays (in little depth)

An array is collection of values, with every value is accessed by an index, i.e.,

```
pfmarks = [0]*50; // makes an array of 50 ints
fnn = [i for i in range(1,51)]; // makes an array of 50 ints
cityname = [""]*12; // makes an array of 12 strings
```

6th element of each array is accessed as pfmarks[5] and cityname[5] respectively. There is no 20th element in cityname (as its size is 12) and the same of pfmarks is accessed as pfmarks[19].

This type of arrays are one dimensional arrays or simply arrays.

we are still using List as arrays

Arrays (in little depth)

Arrays may be used in following 4 scenarios

- Data in array completely filled it
- Data in array is less than its size
    - Data in array is at its lower indices with an additional variable for its ***data size***
    - Data in array is at its lower indices with an ***end of data*** marker place after the last data value
    - Data in array at anywhere but empty locations are marked with a ***special/sentinel value***

2 dimensional arrays

```
marks = [[0 for c in range(cols)] for r in range(rows)]
# makes a 2D array of rows x cols ints
```

This time, marks can be considered as subject wise marks of students.

If there are 7 subject and 40 students, then

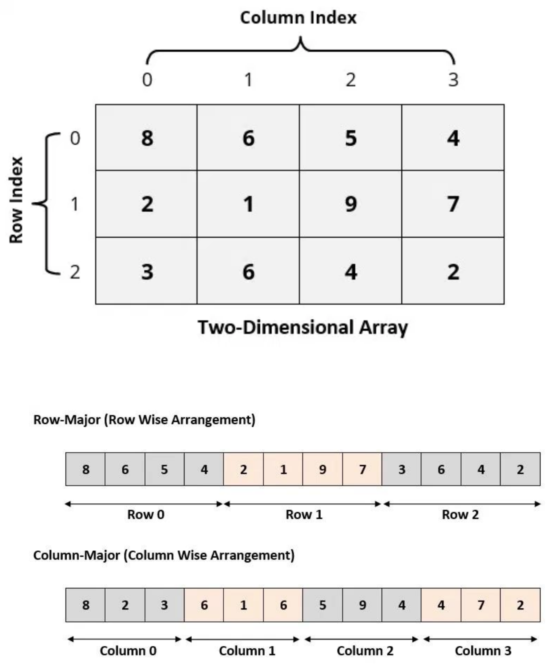
```
marks1 = [[0 for c in range(7)] for r in range(40)]
marks2 = [[0 for c in range(40)] for r in range(7)]
```

- ***marks1*** above is an array of 40 rows and 7 columns, with first dimension as student and second dimension as subject, while
- ***marks2*** above is an array of 7 rows and 40 columns, with first dimension as subject and second dimension as student.

Rectangular data

A two dimensional array is an array of arrays, i.e., every row or column is itself an array.

Generally processed with nested loops.



	0	1	2	3
0	8	6	5	4
1	2	1	9	7
2	3	6	4	2

Rectangular data

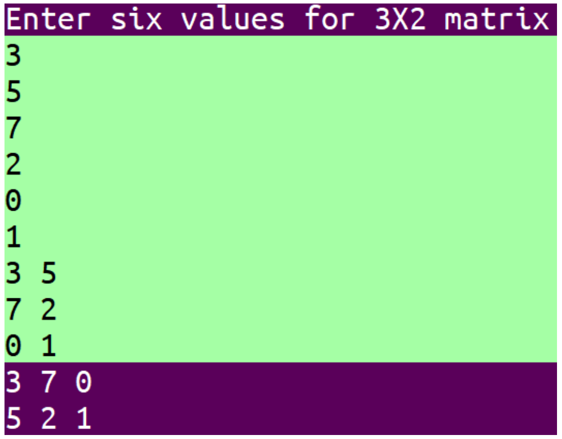
```
ROWS = 3
COLS = 2

rda = [[0]* COLS for r in range(ROWS)];

print("Enter six values for 3X2 matrix")

for r in range(ROWS):
    for c in range(COLS):
        rda[r][c] = int(input())

for r in range(ROWS):
    for c in range(COLS):
        print(rda[r][c], end=" “)
    print()
```



```
Enter six values for 3X2 matrix
3
5
7
2
0
1
3 5
7 2
0 1
3 7 0
5 2 1
```

```
for c in range(COLS):
    for r in range(ROWS):
        print(rda[r][c], end=" “)
    print()
```

Multidimensional data and Triangular data

ia = [0 for i in range(S)]
# [i]th element is at ith location in linear array

fa = [[0.0 for c in range(S2)] for r in range (S1)]
# [i1][i2]th element is at (i1*S2+i2)th location in linear array

ba = [[[False for c in range(S3)] for r in range (S2)] for p in range (S1)]
# [i1][i2][i3]th element is at (i1*S2*S3+i2*S3+i3)th location in linear array

????? # [i1][i2][i3][i4]th element is at (i1*S2*S3*S4+i2*S3*S4+i3*S4+i4)th location in linear array

Generalize it

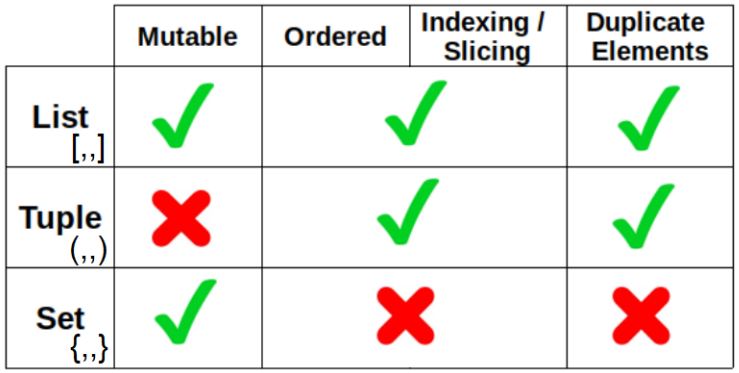
What about triangular data

Objects (attributes, methods and mutability)

Everything in Python is an **object**. Each object has its own data **attributes** and **methods** associated with it. In order to use an object efficiently and appropriately, we should know how to interact with them.

An object whose internal state can be changed is **mutable**.
On the other hand, **immutable** object doesn't allow any change in it once it has been created.

int, float, bool, … are immutable, while list is mutable.



	Mutable	Ordered	Indexing / Slicing	Duplicate Elements
List [,,]	✓	✓		✓
Tuple (,,)	✗	✓		✓
Set {,,}	✓	✗		✗

https://towardsdatascience.com/15-examples-to-master-python-lists-vs-sets-vs-tuples-d4ffb291cf07

modules

from module_name import function_list
from module_name *
import module_name
import module_name as alias

math, statistics, random, datetime , time
cmath, fractions, strings, copy, array

turtle, tkinter, email, sys

and a lot more

Google python ________ module functions
Or https://docs.python.org/3/library/

Little advance I/O

f-strings

```
print(f"{{a={a}, b={b}}}")
```

Format method

```
print("The value of x is {} and y is {} ".format(x,y)) # :5d
```

%operator

```
print("x = %d and y = %f"%(x,y))
```

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```

sys.stdout.write(str), sys.stdin.read(EOF), and sys.stdin.readline()