```python
def deleteDuplication(l):
    ret = []
    memo = set()
    for i in l:
        if i in memo:
            continue
        else:
            ret.append(i)
            memo.add(i)
    return ret

print(deleteDuplication([1,2,3,1,5,5,4]))

class Solution:
    def __init__(self):
        self.nums = []

    def insert(self, num, idx = -1):
        self.nums.insert(idx, num)

    def get_mean(self):
        return sum(self.nums) / len(self.nums)

    def get_mode(self):
        return max(self.nums, key=self.nums.count)

    def get_max(self):
        return max(self.nums)

s = Solution()
for i in range(10):
    s.insert(i)
s.insert(9)

print(s.get_mean())
print(s.get_mode())
print(s.get_max())
```

```python
def rotate_matrix(matrix):
    rows = len(matrix)
    length = rows - 1
    start = 0
    while length > 0:
        for d_i in range(length):
            cur = start + d_i
            matrix[start][cur], matrix[cur][rows - start - 1], matrix[rows - start - 1][rows - cur - 1], matrix[rows - 1 - cur][start]  = matrix[rows - 1 - cur][start], matrix[start][cur], matrix[cur][rows - start - 1], matrix[rows - start - 1][rows - cur - 1]
        length -= 2
        start += 1
matrix = [
  [1,2,3],
  [4,5,6],
  [7,8,9]
]
rotate_matrix(matrix)
for i in matrix:
    print(i)

class Ball:
    def __init__(self):
        self.scores = []

    def on_ball_rolled(self, number_pinned_down):
        if self.scores and len(self.scores[-1]) == 1 and self.scores[-1][0] < 10:
            self.scores[-1].append(number_pinned_down)
            if len(self.scores) >= 2 and self.scores[-2][0] == 10:
                self.scores[-2].append(number_pinned_down)
        else:
            self.scores.append([number_pinned_down])
            if len(self.scores) >= 2 and sum(self.scores[-2]) == 10:
                self.scores[-2].append(number_pinned_down)
    def score(self, idx):
        cur_sum = 0
        for i in self.scores:
            cur_sum += sum(i)
        return (sum(self.scores[idx]), cur_sum)

ball_score = Ball()
a = [3, 3, 1, 0, 10, 7, 3, 5, 2]
for i in a:
    ball_score.on_ball_rolled(i)

print(ball_score.scores)
for i in range(5):
    print(ball_score.score(i))

# Dropout Layers can be an easy and effective way to prevent overfitting
in your models. A dropout layer randomly drops some of the connections
between layers. This helps to prevent overfitting, because if a connection
is dropped, the network is forced to Luckily, with keras it's really easy
to add a dropout layer.

# https://rushter.com/blog/python-garbage-collector/

# with statement in Python is used in exception handling to make the code
cleaner and much more readable. It simplifies the management of common
resources like file streams. Observe the following code example on how the
use of with statement makes code cleaner.
# with open('file_path', 'w') as file:
#     file.write('hello world !')

# In Python, functions are the first class objects, which means that –

# Functions are objects; they can be referenced to, passed to a variable
and returned from other functions as well.
# Functions can be defined inside another function and can also be passed
as argument to another function.
```

```
# Decorators are very powerful and useful tool in Python since it allows
programmers to modify the behavior of function or class. Decorators allow
us to wrap another function in order to extend the behavior of wrapped
function, without permanently modifying it.

# In Decorators, functions are taken as the argument into another function
and then called inside the wrapper function.
```