

# Python Programming

1. SQL (more sophisticated language)

## • Object oriented programming (More abstract)

1. more methods on lists

(a) list.append()

(b) list.extend()

(c) list.insert(c) Insert an item at a given position

(d) list.remove(c) remove the first item from the list

(e) list.pop(c) remove the item at the given position in the lists

## 2. Object Oriented

(1) Definition : A program is made up of many cooperating objects

- Instead of being the "whole program", each object is a little "island" within the program and cooperatively working with other objects.

- A program is made up of one or more objects working together - objects make use of each other's capabilities.

## (2) Object

★ Def : An object is a bit of self-contained Code and Data

Understanding : Objects are used to handle the problems that are separated in different parts. Objects designed by inventors aim to solve some parts of problems.

Class - A template, like the blueprint or factory

Method or Message - A defined capability of a class

Field or attribute - A bit of data in a class

Object or Instance - A particular instance of a class

(3) The new function "dir()" command lists capabilities

: methods that are available in parenthesis .

## • Object Lifestyle

(1) objects are created, used and discarded.

(2) - At the moment of creation (constructor) → often used  
- At the moment of destruction (destructor) → seldom used

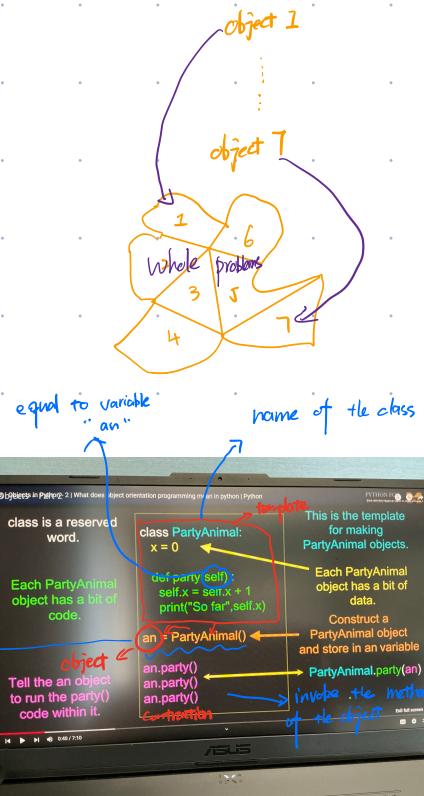
Code in Python: constructor

Constructor: `_init_`

Destructor: `_del_`

running code  
destructor

```
class PartyAnimal:  
    x = 0  
  
    def __init__(self):  
        print('I am constructed')  
  
    def party(self):  
        self.x = self.x + 1  
        print('So far', self.x)  
  
    def __del__(self):  
        print('I am destructed', self.x)  
  
an = PartyAnimal()  
an.party()  
an.party()  
an.party()  
print('an contains', an)
```



Steps:

- 1) First create a class named "Party Animal"

- (2) defined a code use for object

- (3) created a object

- (4) The `an.party()` is like the loops, and run the step (3) triple times.

## • Inheritance

Parent class  $\xrightarrow{\text{inherit}}$  Subclass

- subclasses can retain the functions and templates of parent classes.

```
class PartyAnimal:  
    x = 0  
    name = ""  
    def __init__(self, name):  
        self.name = name  
        print(self.name, "constructed")  
  
    def party(self):  
        self.x = self.x + 1  
        print(self.name, "party count", self.x)  
  
class FootballFan(PartyAnimal):  
    points = 0  
    def touchdown(self):  
        self.points = self.points + 7  
        self.party()  
        print(self.name, "points", self.points)
```

FootballFan is a class which extends PartyAnimal. It has all the capabilities of PartyAnimal and more.