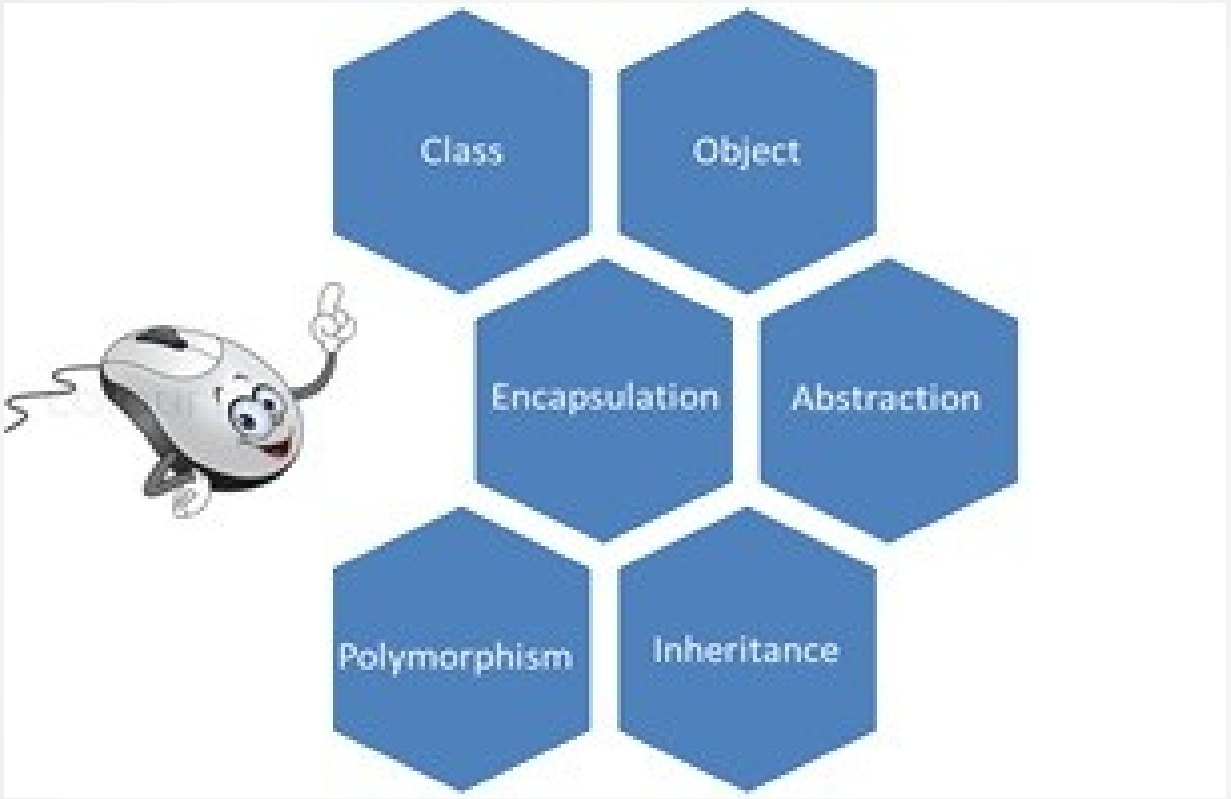
OOP

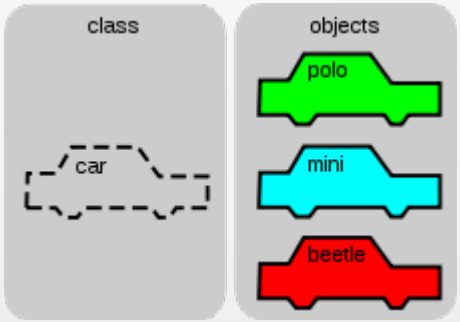

Class
Object
Encapsulation
Abstraction
Polymorphism
Inheritance

OOP

- Objects
- Objects communicate by sending messages
- Class oriented vs Nonclass oriented
- Pure oop vs mixed
- Python are class oriended mixed language

Class

- Class is template of objects, but its object itself too (class of class is metaclass)
- Class is abstraction of object it's like object Type
- When you crate object you use constructor method __init__()
- When you destroy object you use destructor method __del__()
- Properties and methods



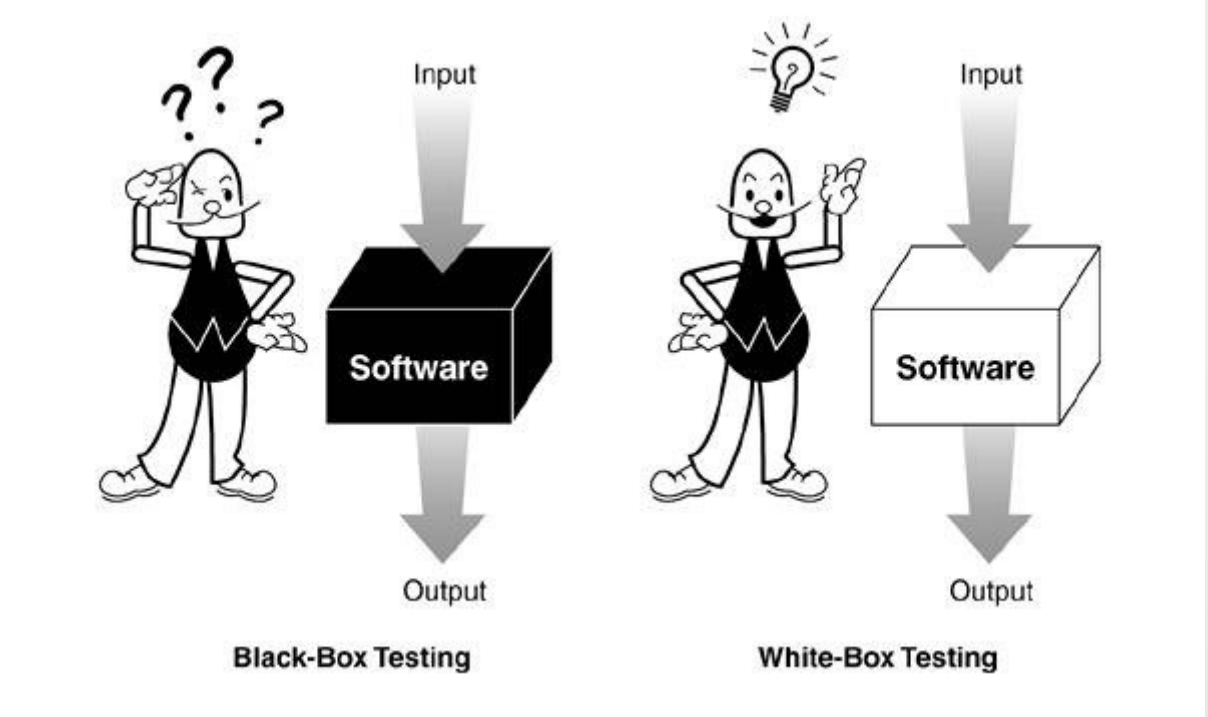
Objects

- Base of OOP
- Class is only template of Objects
- Abstraction of reality

t_1

Encapsulation

- Black box and White box
- Security
- Testing
- Conventions



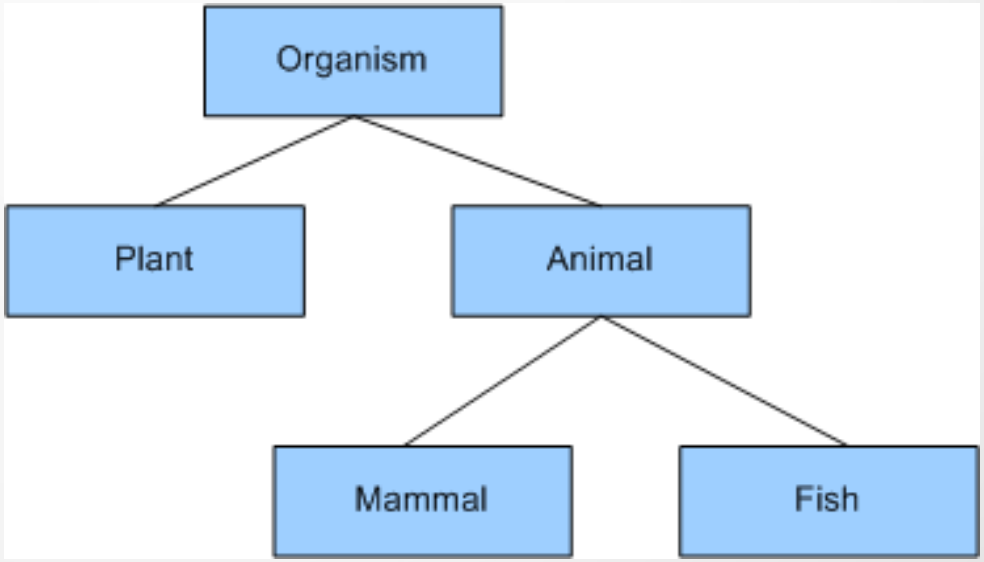
Encapsulation

- Private – it's visible only inside the class you can't access this property/method from outside
- Protected – it's visible inside the class and inside inhrited class
- Public – it's visible outside the class this is API of class, this is what you have to use

t_2

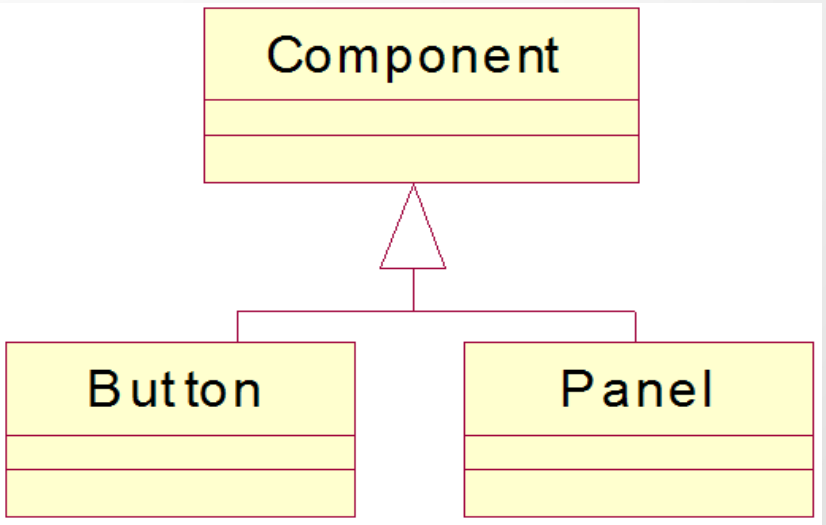
Abstraction

- Object represent real object in world
- Class representet abstractions of objects
- Decomposition of problem into smaller easy to solve abstractions
- Practical use: inheritance
  - reusable code
  - easy to testing
  - easy to change



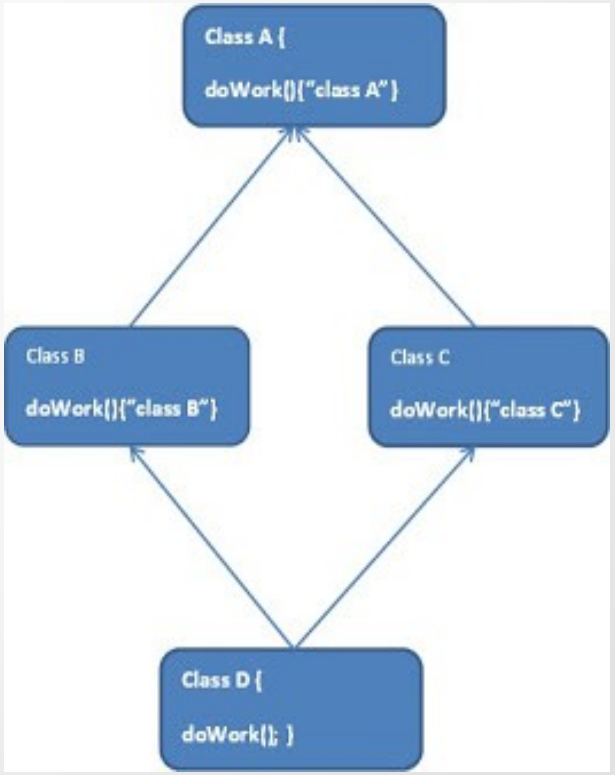
Inheritance

- Allow us to write simple reusable code
- Easy to rewrite and repair code
- Abstraction
- We inherit from more abstract class to more special!!!
- UML:
  - White arrow from child to parrent
  - Buttnon and Panel are Components inherit properties and methods from it

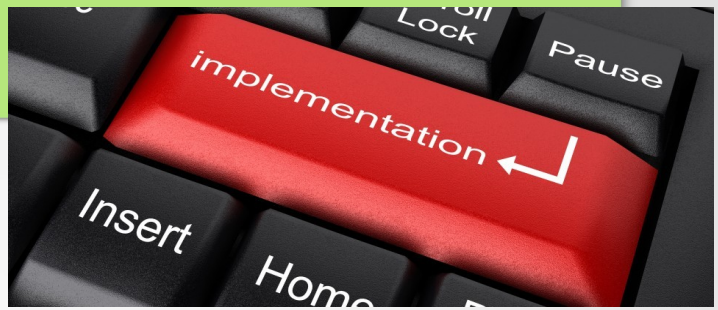


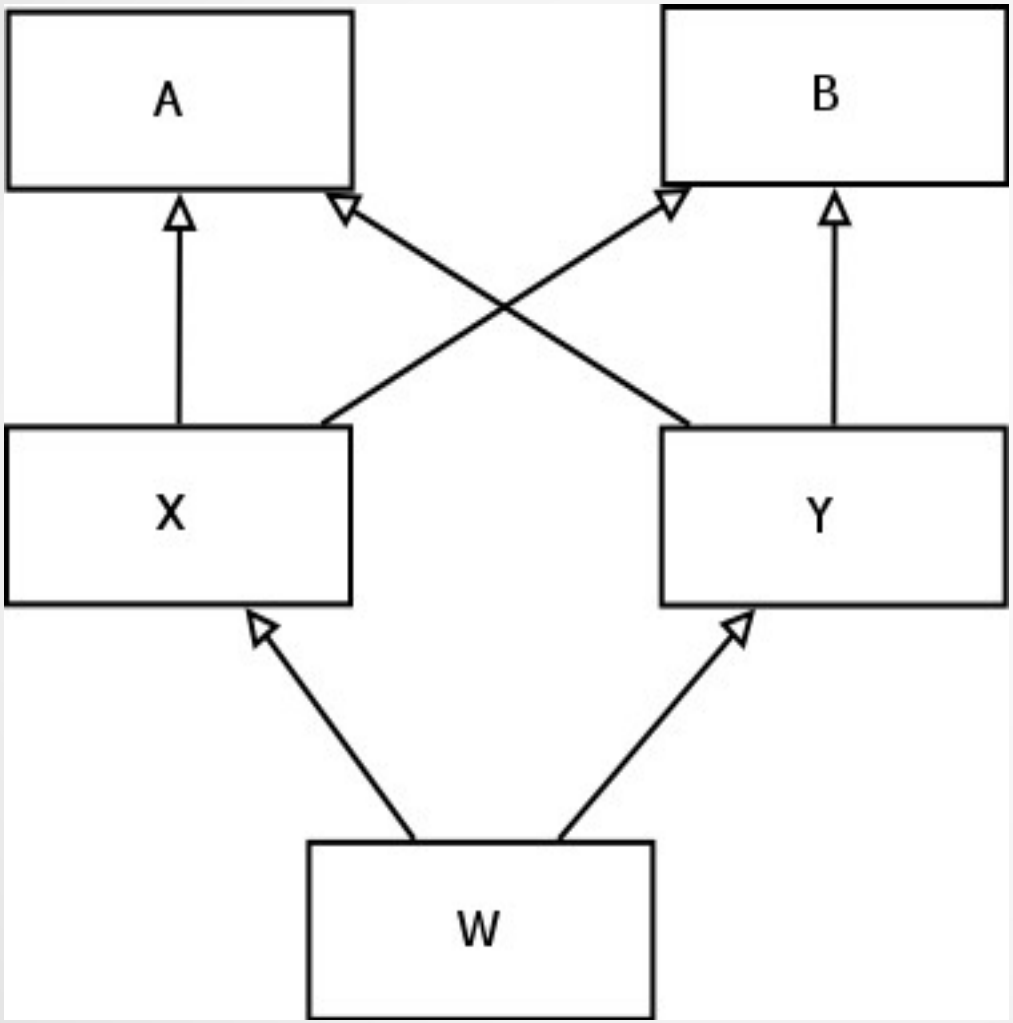
Multiple inheritance

- In python we can define more than one parrent class
- Problems
  - Order of calling __init__()
  - Multiple definition of some method/property
  - Diamond problem



t_3





Polymorphism

- Definition: each object have different answer on the same message
- In python is polymorphism everywhere
  - Dynamic language
  - Track variable type