

PROGRAMMERINGSTEKNIK

Föreläsning 7

Mer om klasser och objekt:

- Klass, instans och self
- Speciella metoder
- Polymorfism
- Publikt och privat

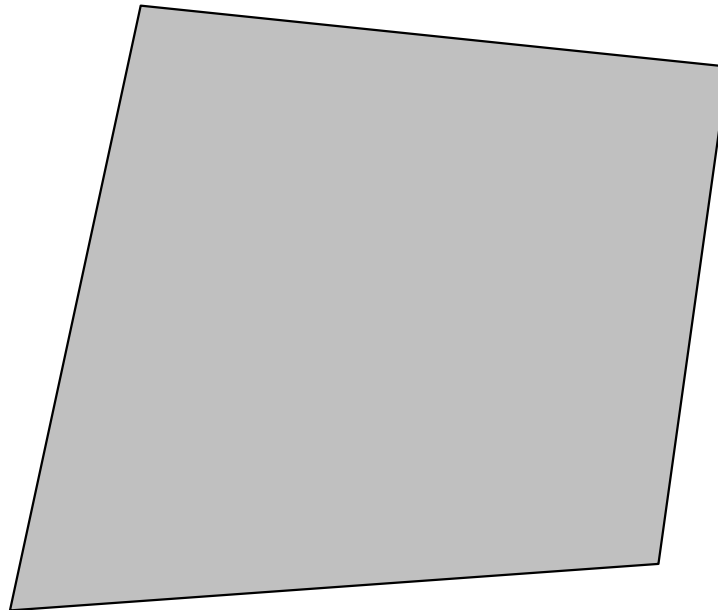
Klass, instanser

- Om du definierar en klass i början av programmet...
- ...så kan du skapa så många objekt (instanser av klassen) du vill i huvudprogrammet.

self

self är en referens till det aktuella objektet.

self



Hur används self?

`self` ska stå överallt i klassdefinitionen:

- först bland parametrarna: `def metod(self, ...)`
- framför varje användning av ett attribut: `self.a`

...men *aldrig* i huvudprogrammet

Vad händer vid metodanrop?

`objekt.metod()`

`self`

metoden init

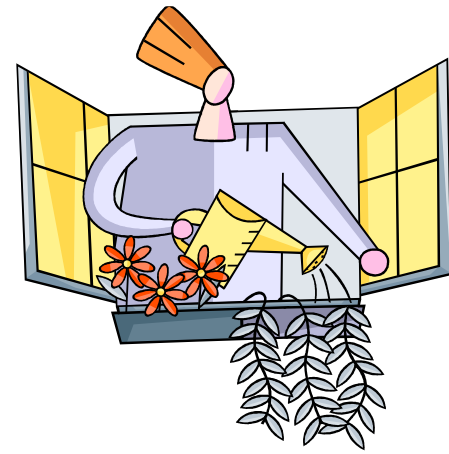
Anropas automatiskt när nya objekt skapas.

Använd den för att ge attributen värden.

```
def __init__(self, a):  
    self.x = a
```



Polymorfism



- Kommer av grekiskans πολλοι (*många*) och μορφη (*form*)
- Med *polymorfism* menas här möjligheten att ha en metod med samma namn i olika klasser och få olika resultat.
- Exempel: `__init__` och `_str__`

Speciella metoder

- `__init__`

Anropas automatiskt när nya objekt skapas.

- `__str__`

Anropas automatiskt av *print()*

- `__lt__`

Anropas automatiskt av <

jämförelse = comparison

vilka attribut ska klassen ha?

- Objekt i programmet motsvarar något objekt i verkligheten.
- Vilka *substantiv* associerar du med det verkliga objektet?
- Välj ut de substantiv som behövs för det program du skriver.

Föreslagen vara

Bricanyl® Turbuhaler®®

AstraZeneca 

inhalationspulver, 0,25 mg/dos, 200
doser, inhalator.

Giltigt t o m:
2015-05-02

Antal:

1



Välj

Varunummer:093195

144,00 kr



Leverans 1-3 vardagar

Läkemedel på ditt recept

Bricanyl® Turbuhaler®®

AstraZeneca 

inhalationspulver, 0,25 mg/dos, 200
doser. inhalator.

Giltigt t o m:
2015-05-02

Antal:

1



Välj

```
class Läkemedel:

    def __init__(self, substans, beredning,
namn, styrka):
        self.substans = substans
        self.beredning = beredning
        self.namn = namn
        self.styrka = styrka
        self.pris = random.randrange(25,
2201)

    def __str__(self):
        return self.namn + " " +
str(self.pris) + " kr"
```

Interaktion mellan objekt

Hur kan man skriva en metod som använder två objekt?

Ha två parametrar: *self* och *other*

METOD-DEFINITION	ANROP
def metod(self, other)	objekt1.metod(objekt2)

jämföra två objekt

```
def __lt__(self, other):  
    if self.attribut < other.attribut:  
        return True  
    else:  
        return False
```

läkemedelsexemplet

```
def __lt__(self, other):  
    if self.pris < other.pris:  
        return True  
    else:  
        return False
```

Inkapsling

- I större program vill man se till att attributen bara kan ändras *inuti* klassdefinitionen.
- I huvudprogrammet anropar man en åtkomstmetod eller ändringsmetod istället!
- Mer att skriva i början men enklare när man vill använda klassen senare.
- Knepig? Använd då privata attribut!



Publikt och privat

- Om ett attribut eller en metod definieras med ett namn som börjar med två understreck (t ex `__namn`) så är den *privat*.
- Det innebär att den endast kan användas inom klassen (man kommer inte åt den från main).
- Annars är den *publik*, och kan användas i vilken del av programmet som helst.

```
class Demo:

    def __init__(self):
        self.offentlig = 1
        self.__privat = 2      #Privat attribut

    def visa_offentlig(self):
        print(self.offentlig)

    def visa_privat(self):
        print(self.__privat)
```

Vilken sats ger felmeddelande?

```
x = Inkapsling()  
x.visa_offentlig()  
x.visa_privat()
```

```
print(x.offentlig)  
print(x.__privat)
```

```
def  namn(self):  
    """Åtkomstmetod för namnet"""  
    return self.__namn
```

```
def  bytNamn(self, nyttNamn):  
    """Ändringsmetod för namnet"""  
    self.__namn = nyttNamn
```