PROGRAMMERINGSTEKNIK

Föreläsning 7

Mer om klasser och objekt:

- Klass, instans och self
- Speciella metoder
- Polymorfism
- Publikt och privat
- Lista av objekt

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

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.

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.
- Metoden `__str__` som automatiskt anropas av print är ett exempel.

SPECIELLA METODER

- `__init__`

  Anropas automatiskt när nya objekt skapas.

- `__str__`

  Anropas automatiskt av *print()*

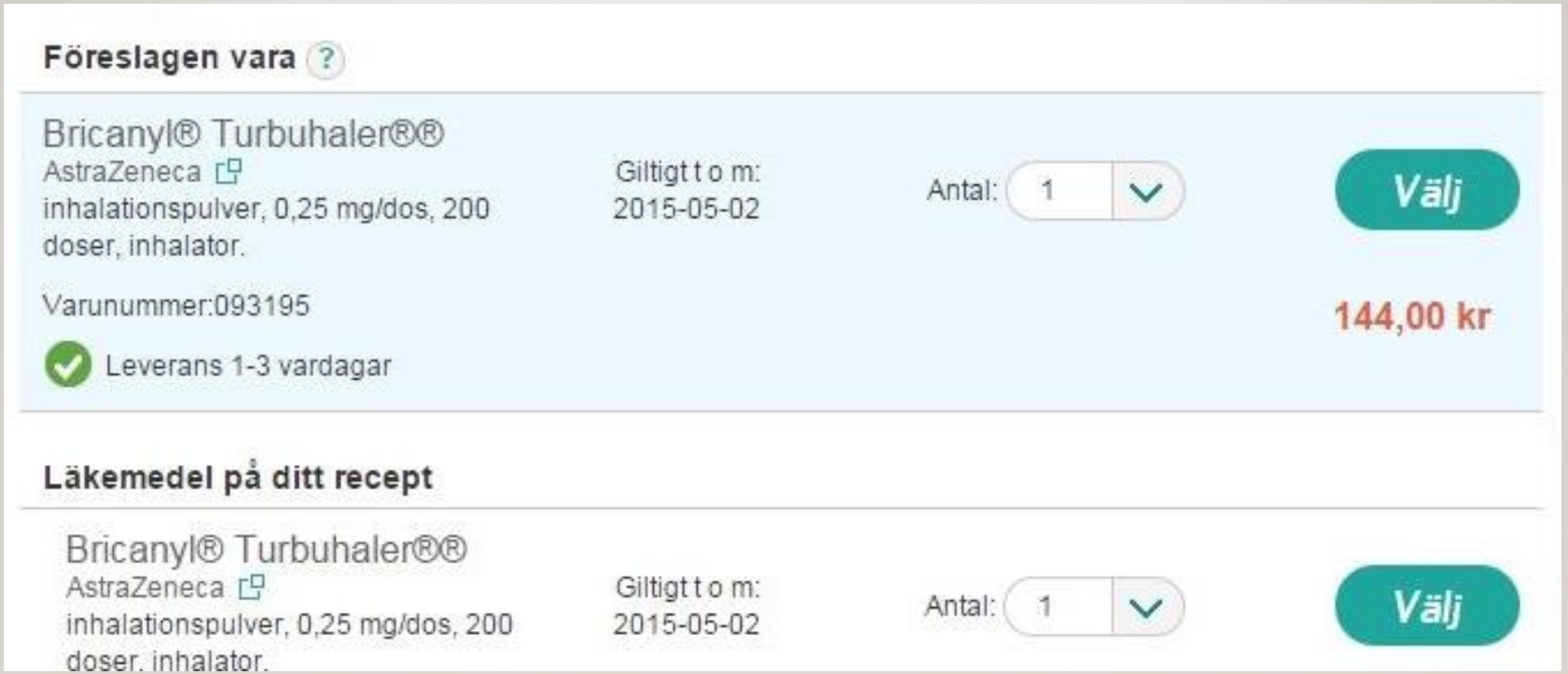
- `__lt__`

  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.
- Knepigt? 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.

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


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

LISTA AV OBJEKT

- Flera objekt i samma program?

```
djur1 = Husdjur()
djur2 = Husdjur()
```

...

- Enklare att lägga husdjuren i en lista!

[0] [1] [2] [3]

SKAPA LISTAN

```
lista = []
for i in range(n):
    nytt = Husdjur()
    lista.append(nytt)
```

ANROPA METOD FÖR VARJE DJUR

```
for djur in lista:
    djur.banna()
```