

PROGRAMMERINGSTEKNIK

Föreläsning 7

Repetition + 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.



SPECIELLA METODER

- `__init__`
Anropas automatiskt när nya objekt skapas.
Använd den för initiering av attributen!
- `__str__`
Låt den returnera en strängrepresentation av objektet, så vet `print()` hur det ska skrivas ut.
- `__lt__`
Skriv en sådan metod om du vill kunna jämföra två objekt (It står för "less than", operatorn `<`)

__STR__ I KLASSEN HUSDJUR

- Ska returnera en sträng som representerar objektet
- Anropas automatiskt om man skriver ut objektet.

```
h = Husdjur("Hamish")  
print(h)
```
- Bra att ha när man testar programmet!

```
def __str__(self):  
    """Version 1: bara husdjurets namn"""  
    beskrivning = "Jag heter " + self.namn  
    return beskrivning
```

```
def __str__(self):
    """Version 2: alla attribut"""
    beskrivning = self.namn + ", "
    beskrivning += str(self.glad) + ", "
    beskrivning += str(self.hunger)
    return beskrivning
```

```
def __str__(self):
    """Version 3: begripligare utskrift"""
    beskrivning = self.namn + " är "
    if self.glad > 5:
        beskrivning += "glad: (^_^)"
    else:
        beskrivning += "ledsen: (T_T)"
    if self.hunger > 3:
        beskrivning += " och hungrig!"
    else:
        beskrivning += " och mätt."
    return beskrivning
```

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.



PUBLIKT OCH PRIVAT

- Om ett attribut eller en metod definieras med ett namn som börjar med två understräck (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.

Kan ett attribut och en metod ha samma namn?

- ☒ Ja
- ☐ Nej
- ☐ Vet inte

```
def __init__(self):
    """ Ger attributen slumpade värden """
    self.__namn = \
        choice("BCFKR") + choice("iouy") + \
        2 * choice("nst") + choice("aey")
    self.__glad = randrange(10)
    self.__hunger = randrange(3)
    self.__kon = choice(("hona", "hane"))
    self.__preferens = choice(("samma", "annat"))
```

INKAPSLING

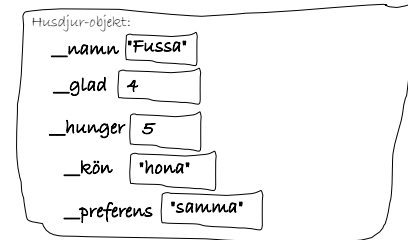
- En väluppfostrad programmerare använder bara attributen *inuti* klassdefinitionen.
- I huvudprogrammet anropar man en åtkomstmetod eller ändringsmetod istället!
- Mer att skriva i början men enklare när man vill ändra i klassen senare.
- Knepig? Använd då privata attribut!



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

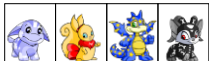
```
def bytNamn(self, nyttNamn):  
    """Ändringsmetod för namnet"""  
    self.__namn = nyttNamn
```

RITA UPP ETT OBJEKT



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()
```