

Föreläsning 5

DD1314

Programmering / OPEN

6 hp

Innehåll

- Klass & objekt
- Instantiering
- Instansvariabler & instansmetoder
- Konstruktör
- self
- Inkapsling (i mån av tid)

Exempel

```
# Dice1.py  
import random
```

```
utfall = 0
```

```
def kasta():  
    global utfall  
    utfall = random.randint(1,6)
```

```
kasta()
```

```
print (utfall)
```

Klass & objekt

- Tärningsexempel; vi kan bara beskriva en (1) tärning med första exemplet.
- Ur en klass kan man skapa godtyckligt antal objekt, t ex 2, 19 tärningar
- En klass innehåller:
 - instansvariabler
 - instansmetoder
 - Konstruktör
- Metoder är funktioner som är deklarerade inuti en klass.
- Att skapa ett objekt kallas instantiering
- objekt = instans, orden används tyvärr omväxlande.
- Instansmetoder och instansvariabler är unika för respektive objekt,

Exempel

```
# Dice2.py  
import random
```

```
class Dice:
```

```
    def kasta(self):  
        self.utfall = random.randint(1,6)
```

```
d1 = Dice()  
d2 = Dice()  
d1.kasta()  
d2.kasta()  
print (d1.utfall)  
print (d2.utfall)  
d2.kasta()  
print (d1.utfall)  
print (d2.utfall)
```

Instantiering

- När man skapar ett nytt objekt måste man ange klassen:
 - $d1 = Dice()$
- Man kan även skapa variabler av de fördefinierade variablerna på samma sätt
 - $s = str('Hej')$
 - $i = int(5)$
 - $pi = float(3.14)$
- Man får en variabel som refererar till objektet, dvs håller reda på var i minnet objektet lagras.
- Därför skapas **inte** ett nytt objekt av $d2 = d1$ utan en referens till samma objekt som $d1$ refererar till

Instansvariabler & instansmetoder

- De **variabler** som är unika för varje tilltänkt objekt (t ex tärningens utfall för en tärning) blir en **instansvariabel**. Dessa når man genom referensen (t ex `d1.utfall`). En instansvariabel existerar under hela objektets livscykel och är tillgänglig från objektets samtliga metoder.
- De **metoder** som är unika för varje tilltänkt objekt (t ex att kasta tärningen) blir en **instansmetod**. Dessa når man genom referensen (t ex `d1.kasta()`). En instansmetod och dess eventuella lokala variabler existerar (likt andra metoder/funktioner) endast då den exekveras

Konstruktor

- Om objektet man skapar behöver några initiala egenskaper (t ex att en tärning ska vara av en specifik färg för att skilja den från andra tärningar) kan man skriva en konstruktor. En konstruktor är en metod som anropas vid instantiering.
- Konstruktorns namn är alltid `__init__`
- Via parametrar kan man ge indata till konstruktorn
- Normalt består en konstruktor av tilldelningssatser som ger instansvariablerna värden.
d1 = Dice('röd')

self

- I en instansmetod kan man använda referensen *self*.
- *self* refererar till det aktuella objektet:

```
def __init__(self, färg):  
    self.färg = färg
```

- I exemplet ovan refererar *self* till det objekt som konstruktorn håller på att skapa.

Exempel 1 / 2

```
# Dice3.py
import random

class Dice:

    def __init__(self, färg):
        self.utfall = 0
        self.färg = färg

    def kasta(self):
        self.utfall = random.randint(1,6)
```

Exempel 2 / 2

```
d1 = Dice('röd')
d2 = Dice('blå')
d1.kasta()
d2.kasta()
print (d1.färg) + ': ', d1.utfall)
print (d2.färg + ': ', d2.utfall)
d2.kasta()
print (d1.färg + ': ', d1.utfall)
print (d2.färg + ': ', d2.utfall)
```

Inkapsling

- *publikt* - variabler och metoder som man kommer åt även från andra klasser eller huvudprogrammet.

self.utfall = 0

- *privat* - variabler och metoder som man bara kommer åt inifrån samma klass, börjar med två underscore (__)

self.__utfall = 0

- Man bör deklarera alla sina variabler som *privata* för att minimera manipulation utifrån. Detta kallas inkapsling.
- Vill man förändra variabelns värden får man göra det genom en metod

def kasta(self):

Exempel 1 / 2

```
# Dice4.py  
import random
```

```
class Dice:
```

```
    def __init__(self, färg):  
        self.__utfall = 0  
        self.__färg = färg
```

```
    def kasta(self):  
        self.__utfall = random.randint(1,6)
```

```
    def färg(self):  
        return self.__färg
```

```
    def utfall(self):  
        return self.__utfall
```

Exempel 2 / 2

```
d1 = Dice('röd')
d2 = Dice('blå')
d1.kasta()
d2.kasta()
print (d1.färg() + ': ', d1.utfall())
print (d2.färg() + ': ', d2.utfall())
d2.kasta()
print (d1.färg() + ': ', d1.utfall())
print (d2.färg() + ': ', d2.utfall())
```