Föreläsning 6

DD1310
Programmering / OPEN
6 hp

Exempel 1 / 2

```
# Slask.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

    def __str__(self):
        return self.__färg + ': ' + str(self.__utfall)
```

Innehåll

- Utskriftmetod för objekt (__str__)
- Sortering
- (Arv)
- (Iteration & rekursion) (Ej relaterat till OO)

Exempel 2 / 2

```
d1 = Dice("röd")
d2 = Dice("blå")
d1.kasta()
d2.kasta()
print (d1)
print (d2)
```

__str__

- En användbar metod för att förenkla utskriften / presentationen av ett objekt
- Returnerar en sträng som skrivs ut vid anrop

Sortering

- Vill man sortera en lista kan man t e x skriva
  - tal = [5, 2, 3, 1, 4]
  - tal.sort()
  - print tal
- Fungerar även för strängar
  - ord = [’liten’, ’tuva’, ’välter’, ’ofta’, stort’, ’lass’]
  - ord.sort()
  - print ord

__cmp__

- Vill man sortera en lista av objekt måste man ange VAD i objekten man vill sortera på, det görs med hjälp av en metod __cmp__ som används för jämförelse av två objekt. Denna metod ska returnera
  - +1 om self objektet kommer först
  - -1 om self-objektet kommer sist
  - 0 om de är lika
- (Rita upp hur sorteringen går till)

Arv

- Ibland är de klasser man skriver besläktade med varandra, då kan man lägga det gemensamma i en basklass och det specifika för varje klass i subklasser, t e x kan superklassen ”geometrisk figur” sägas ha subklasserna ”cirkel”, ”rektangel” etc.

Exempel 1 / 2

```
# Sort.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 1 / 3

```
# Arv.py

# basklass
class Fordon:

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

Exempel 2 / 2

```
li = list()
for i in range(10):
    li.append(Dice('röd'))

for i in range(len(li)):
    li[i].kasta()

li.sort(key = lambda: Dice.utfall)
li.reverse()
```

Exempel 2 / 3

```
# subklasser
class Bil(Fordon):

    def __init__(self, märke, motorstyrka):
        Fordon.__init__(self, märke)
        self.motorstyrka = motorstyrka

    def typ(self):
        return 'Bil'

class Cykel(Fordon):

    def __init__(self, märke):
        Fordon.__init__(self, märke)

    def typ(self):
        return 'Cykel'
```

Exempel 3 / 3

```
# huvudprogram

fordon1 = Bil('Volvo', 200)
fordon2 = Cykel('Crescent')

print (fordon1.märke)
print (fordon2.märke)
print (fordon1.typ())
print (fordon2.typ())
```

Exempel

```
# fakultet_rek.py
# n förutsätts vara heltal

def fakultet(n):
    if n == 0:
        return 1
    else:
        return n*fakultet(n-1)

print (fakultet(1))
print (fakultet(2))
print (fakultet(3))
print (fakultet(4))
```

Iteration & rekursion

- Iteration
  - Iteration betyder upprepning.
  - Det innebär att utföra något med hjälp av en slinga.
  - Exempel: 5! = 1*2*3*4*5
- Rekursion
  - Att lösa ett problem genom att först lösa en mindre variant av samma problem.
  - Exempel: 5! = 5*4!
  - Man får en metod som anropar sig själv.
  - Metoden måste veta när den ska sluta, dvs det måste finnas ett basfall, t ex 0! = 1

Exempel

```
# fakultet_iter.py
# n förutsätts vara heltal

def fakultet(n):
    if n == 0:
        return 1
    else:
        f = 1
        while n > 1:
            f = f*n
            n = n - 1
        return f

print (fakultet(1))
print (fakultet(2))
print (fakultet(3))
print (fakultet(4))
```