Skolan för Datavetenskap och kommunikation

DD1311 Programmeringsteknik med PBL

Föreläsning 13-14

Bra att ha till P-uppgiften:

- Åtkomstmetoder
- Kopiera en lista
- Sortera en lista
- Formatterad utskrift (snygga tabeller)
- Avlusning

Nästa tisdag:

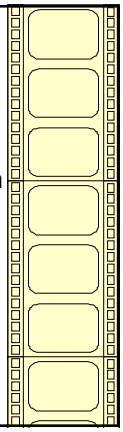
- Felhantering
- Grafiska gränssnitt

Nyheter

- Idag kl 16-17 ska Grupp 10 vara i sal E51
- Omprov kl 13.00-14.00 i CSC-biblioteket.
- Specredovisningar kl 13.00 i sofforna utanför CSC-biblioteket.
- Lab 5 kl 13.00 i seminarierum 1625.
- Lab 4 kl 13.30 i seminarierum 1625.
- Lab 3 kl 13.50 i seminarierum 1625.
- Lab 2 kl 14.00 i seminarierum 1625.

Film-programmet

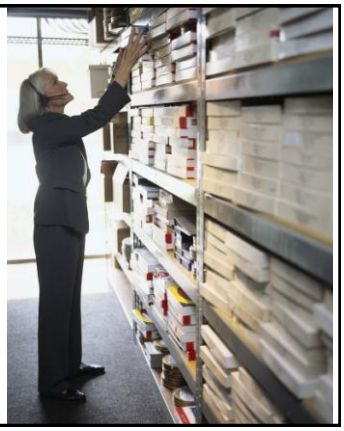
- Programmet läser in information om filmerna från fil.
- Varje film lagras i ett Video-objekt och läggs i en lista.
- Den som kör programmet kan betygssätta filmerna.
- Listan kan sorteras efter medelbetyg eller efter antal visningar.



Sortering

Man kan sortera böcker efter t ex

- Författare
- Titel
- Ämne



Sortering

- Python-listor har ju en inbyggd sort-metod:

```
lista.sort()
```

- En lista med tal sorteras i stigande ordning, en lista med strängar sorteras i bokstavsordning.
- Men om man har en lista av objekt - vilket attribut sorterar `sort()` på då?

Åtkomstmetoder

- En åtkomstmetod returnerar värdet av ett attribut.
- Den får inte ha samma namn som attributet!

```
def seVisningar(self):
  """Returnerar antal visningar"""
  return self.visningar
```

Parametrar till sort

- Nu kan `sort` anropas med parametern *key* satt till åtkomstmetoden för det vi vill sortera på!

```
listan.sort(key=Video.seVisningar)
```

- För att vända på ordningen kan vi använda parametern reverse.

```
listan.sort(key=Video.seVisningar,
 reverse=True)
```

Kopiera en lista

- Tilldelningssatser med enkla värden, t ex heltal, ger en ny variabel med samma innehåll. När det gäller datastrukturer, t ex listor får vi istället en ny *referens* till listan.

- Vill man ha en *kopia* av listan kan man använda `deepcopy` i modulen `copy`:

```
import copy
kopia = copy.deepcopy(listan)
```

Snygg utskrift

- Man kan ange *formattering* för print-satser
- Skriv `print a % b` där *a* är en sträng med formatteringskoder och *b* är en tuppel med värden.
- Exempel från filmprogrammet:

```
print "%-55s %5.1f %10d" %
 (self.namn, self.medelBetyg(),
 self.visningar)
```

Exempel på formatteringskoder

%7s	Sträng i 7 positioner	`_ _ _ k a t t`
%-7s	Sträng i 7 positioner, vänsterjusterat	`k a t t _ _ _`
%4d	Heltal i 4 positioner	`_ _ 1 7`
%5.2f	Flyttal i 5 positioner med 2 decimaler	`_ 3 . 1 4`

Avlusning

- Lär dig tolka felutskrifter!

```
Traceback (most recent call last):
  File "filmer.py", line 124, in <module>
    tittaromrostning(listan)
  File "filmer.py", line 102, in titta
    film.ny_visning(1)
TypeError: ny_visning() takes exactly 1
 argument (2 given)
```

Kontrollutskrifter

- Använd kontrollutskrifter för att hittar var i programmet felet uppstår.
- En kontrullutskrift är en vanlig print-sats, till exempel:

```
print "Klar med inläsningen"
```

- Du kan också skriva ut variabelvärden för att se hur dom ändras under körning.

Testning

- Skriv upp (i en textfil) vilka indata du vill testa programmet med. Missa inte gränsfallen.
- Efter att du har fixat ett fel – testa att programmet fortfarande fungerar för *alla* indata.