Skolan för Datavetenskap och kommunikation

2D1311 Programmeringsteknik med PBL

Föreläsning 3

Kapitel 5 & 6

- Listor
- Uppslagslistor
- Funktioner
- Parametrar
- Namnrymden

Listor

- Listor liknar tupler, men är flottare:
  - Skrivs med hakparenteser:
    `blommor=["ros","fiol","pion"]`
  - Går att ändra (muterbara):
    `blommor[1]="viol"`
  - Har metoder:
    `blommor.sort()`

Listmetoder

Metod	Beskrivning
`append(x)`	Lägger till x sist i listan.
`sort()`	Sorterar i stigande ordning.
`reverse()`	Vänder listan.
`count(x)`	Räknar antalet x i listan.
`index(x)`	Första förekomsten av x i listan.
`insert(i,x)`	Stoppar in x på plats `i`.
`pop([i])`	Plockar ut elementet på plats `i`.
`remove(x)`	Tar bort första förekomsten av x.

Läsa in till en lista

```
# Frågar efter ord som rimmar och läser
# in i en lista, som sedan skrivs ut sorterad.
lista=[]
ord=raw_input("Vad rimmar på hus? ")
while ord!="":
    lista.append(ord)
    ord=raw_input("Kan du ett till? ")
print "Du hittade", len(lista),"ord!"
lista.sort()
print lista
```

Uppgift: Rita steg för steg vad som händer i rimordsprogrammet!

Matriser

- Genom att nästla listor kan man skapa en matris:

```
>>> matris = [[1,2,3],[4,5,6],[7,8,9]]
>>> matris[0]
[1, 2, 3]
>>> matris[0][2]
3
```

Uppslagslistor

- En uppslagslista lagrar datapar.
- Varje datapar har en nyckel och ett värde.
- Nycklarna måste vara unika och omuterbara.
- Runt uppslagslistan används klamrar `{}`.
- En uppslagslista har inte ordning!

uppslagslista=dictionary

Använda en uppslagslista

```
# Kollar saldo på bankkontot
konton={"Mart":25045,"Fuad":20000,"Hanna":38000}
namn=raw_input("Namn: ")
if namn in konton:
    print "Ditt saldo:",konton[namn]
else:
    print "Tyvärr, du har inget konto."
```

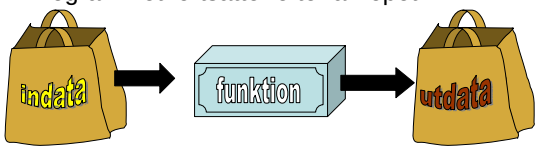
Uppgift: Föreslå något som man kan använda en uppslagslista till!

Några kända funktioner

Funktion	Indata (parametrar)	Utdata (returvärde)
`ut=input("Gissa:")`	strängen "Gissa"	den inlästa gissningen
`ut=str(bredd)`	talet i variabeln bredd	talet som sträng
`ut=len("kålrot")`	strängen "kålrot"	antal tecken i strängen
`ut=range(0,100,2)`	startvärde 0, övre gräns 100, steg 2	en lista med tal

Hur funktioner fungerar

- Funktionen anropas.
- Indata skickas in via parametrar, utdata returneras.
- Programmet fortsätter efter anropet.



Uppgift: Du vill skriva en funktion som avgör om en låneansökan ska beviljas.
- Vad är indata (parametrar)?
- Vad är utdata (returvärden)?

Hur man definierar en funktion

- Skriv först
  `def` *funktionsnamn*`(`*parametrar*`):`
- Sen, indenterat:
  - En kommentarrad som beskriver vad funktionen gör, inom tredubbla citationstecken, tex `"""Beräknar arean"""`.
  - Satserna som funktionen ska utföra.
  - Allra sist `return` *returvärde/returvärden*
  - Anger man inget returvärde blir det `None`

Exempel från Tic-tac-toe i kap 6

Funktionen definieras så här:

```
def ask_number(question, low, high):
    """Ask for a number within a range."""
    response = None
    while response not in range(low, high):
        response = input(question)
    return response
```

returvärde

Funktionen anropas så här:

question low high

```
move = ask_number("Your move? (0 - 8):", 0, 8)
```

Namnrymder

- Varje gång en funktion anropas skapas en namnrymd för just det anropet.
- Parametrar och variabler som skapas inuti funktionen blir *lokala* (finns bara i funktionen).
- Man kan också komma åt *globala* variabler, som skapats utanför funktionen, men inte ändra på dom.

Namnrymdsexempel

```
DIFF = 32
FAKTOR = 5.0/9.0

def celsius(f):
    """Konverterar fahrenheit till celsius"""
    c = (f-DIFF)*FAKTOR
    return c

print "13 Fahrenheit blir ",celsius(13),"C."
```

- Konstanterna `DIFF` och `FAKTOR` är globala och finns i hela programmet.
- Parametern `f` och variabeln `c` är lokala och finns bara inuti funktionen celsius.

Rekursion

- Fakultet kan definieras rekursivt:
  n! = (n-1)! *n
- En rekursiv funktion i Python har:
  - Ett anrop av funktionen själv inuti funktionsdefinitionen.
  - Ett basfall som inte är rekursivt.
  - En if-sats som väljer mellan basfallet och det rekursiva anropet.

Rekursivt exempel

```
def f(n):
    """Rekursiv fakultetsfunktion"""
    if n>1:
        return f(n-1)*n
    else:
        return 1
```

Uppgift: Vad skriver följande rekursiva funktion ut, om den anropas med `rek(1)` ?

```
def rek(x):
  if x<4:
     print x
     rek(x+1)
     print x, "igen!"
```