Skolan för Datavetenskap och kommunikation

2D1311 Programmeringsteknik med PBL

Föreläsning 2

Kapitel 3 & 4

- if-satsen
- while-slingan
- slumptal
- algoritmer
- for-slingan
- strängar
- tupler

Exempel

```
talet = 17
gissning = input("Gissa
  talet: ")
if gissning == talet:
    print "Hurra, du vann!"
else:
    print "Fel svar."
print "---------------"
```

if-satsen

- *if-satsen* används för val mellan två alternativ.
- De satser som ska utföras efter if måste indenteras (med tab).
- Villkoret har värdet `True` eller `False`.
- Datatypen kallas *boolean*.

if-sats=if structure
villkor=condition
jämföra=compare

Operatorer i villkor

Operator	**Betyder**	**Om vi satt** `dag=9` **blir**
`==`	lika med	`dag==9` True
`!=`	skilt från	`dag!=9` False
`>`	större än	`dag>5` True
`<`	mindre än	`dag<5` False
`>=`	större än eller lika med	`dag>=5` True
`<=`	mindre än eller lika med	`dag<=9` True

Kombinera villkor

- Villkor kan kombineras med operatorerna and,or,not
- Exempel:

```
if pris<1000 and taltid>=100:
  print ”Telefonfynd!”
```

`and`	och (True endast om bägge villkoren är uppfyllda)
`or`	eller (True om något av villkoren är uppfyllt)
`not`	icke (True om villkoret inte är uppfyllt)

Uppgift: Skriv ett program som läser in ålder och kollar om man får se barnförbjuden film!

Slumptal

- Oförutsägbara program är roligare!
- Slumptal får man genom att
  1. Importera random-modulen med satsen `import random`
  2. Anropa randrange()-funktionen med gränser `random.randrange(start,stopp)`

     vilket ger ett slumptal som är större än eller lika med start, men mindre än stopp.

slumptal=random number

Talgissning 1.0

```
import random
talet = random.randrange(1,21)
gissning = input("Gissa ett tal
  mellan 1 och 20: ")
if gissning == talet:
    print "Hurra, du vann!"
else:
    print "Fel svar."
print "---------------"
```

Uppgift: Skriv ett program som kastar en tärning och skriver ut resultatet!

while-slingan

- En while-slinga upprepar ett antal satser så länge som ett villkor är uppfyllt.
- Så länge som kannan inte rinner över:
  - Fyll på mer vatten!
- Så länge som du inte har somnat:
  - Räkna ett får till!
- Så länge som du inte gissat rätt tal:
  - Gissa en gång till!

Algoritm för talgissning

En algoritm är en stegvis beskrivning av vad programmet ska göra. Exempel:

1. Slumpa ett tal.
2. Låt användaren göra en gissning
3. Så länge som gissningen är fel:
   - Om gissningen är för hög : uppmana användaren att ge ett lägre tal och läs in ny gissning.
   - Om gissningen är för låg : uppmana användaren att ge ett högre tal och läs in en ny gissning.
4. När gissningen är rätt – skriv ut beröm.

Talgissning 2.0

```
# Talgissning, version 2.0
import random
tal = random.randrange(1,101)
gissning = input("Gissa mitt tal: ")
while gissning != tal:
    if gissning > tal:
        gissning = input("Lägre:")
    elif gissning < tal:
        gissning = input("Högre:")
print "Bravo, du gissade rätt!"
```

for-slingan

En for-slinga upprepar ett antal satser för varje element i en sekvens.

- För varje telefonnummer på listan:
  - Ring upp personen!
- För varje chokladbit i asken:
  - Ät upp den!
- För varje bokstav i en sträng:
  - Skriv ut bokstaven!

sekvens=sequence

Exempel 1

```
for tkn in "Hej":
    print tkn
```

```
H
e
j
```

Räkna upp tal

Funktionen range() ger en sekvens av tal.

`range(10)`

ger sekvensen

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

Programmet nedan skriver ut talen 0..9

```
for i in range(10):
    print i
```

Uppgift: Vad händer här?

```
sum = 0
for i in range(5):
    sum = sum + i
print sum
```

Strängar: index och skivning

`ord="pizza"`	p i z z a 0 1 2 3 4
`ord[0]➔"p"`	Varje bokstav i strängen är numrerad med ett *index*.
`ord[1:3]➔"iz"`	En delsträng – från och med 1 upp till (men inte med) 3.
`ord[:4]➔"pizz"`	Utelämnat tal *före* kolon betyder början av strängen.
`ord[2:]➔"zza"`	Utelämnat tal *efter* kolon betyder slutet av strängen.

skivning=slicing

Tupler

- En tuppel är en sekvens av data.
- Exempel:
  - primtal = (2,3,5,7,11,13,17,19)
  - dagar = ("måndag","tisdag","lördag")
- En tuppel kan indexeras och skivas precis som en sträng.
- Både tupler och strängar är *omuterbara*
  - `print primtal[0]` Utskrift är OK...
  - ~~`primtal[0]=1`~~ men inte ändring!

omuterbar=immutable

Algoritm för trasselord

Som talgissning, men med ord!

Svårast är att blanda bokstäverna i ordet:

- Börja med ett tomt trasselord
- Så länge som det finns bokstäver kvar i ursprungsordet:
  - Slumpa en plats i ursprungsordet
  - Ta bokstaven på den platsen
  - Lägg till den till trasselordet
  - Plocka bort den från trasselordet

```
# Trasselord
#
# Datorn slumpar ett ord och blandar det,
# spelaren ska gissa ursprungsordet.
#
# Skrivet av Michael Dawson - 1/28/03
# Modifierat av Linda Kann - 29 jan 2007

import random

# Skapa en tuppel med ord
FRUKT = ("persika","citron","jordgubbe","kiwi")
# Slumpa ett ord
ord = random.choice(FRUKT)
# Spara svaret
svaret = ord
```

```
# Skapa ett tomt trasselord
trassel =""
# Så länge som ursprungsordet inte är tomt
while ord:
    # Slumpa en plats i ordet
    plats = random.randrange(len(ord))
    # Ta bokstaven där och lägg den i trassel
    trassel = trassel+ord[plats]
    # Plocka bort bokstaven ur ursprungsordet
    ord = ord[:plats]+ord[(plats+1):]
```

```
# Starta spelet
print \
"""
          Spelet ordtrassel

      Skapa ett ord ur trasslet
        (Avsluta med retur.)
"""
print "Trasselordet:", trassel
gissning = raw_input("Din gissning: ")
gissning = gissning.lower()
```

```
while (gissning!=svaret) and (gissning!=""):
    print "Fel svar."
    gissning = raw_input("Din gissning: ")
    gissning = gissning.lower()

if gissning == svaret:
    print "Javisst! Du klarade det!"

raw_input("\nAvsluta med retur.")
```