Skolan för Datavetenskap och kommunikation

DD1311 Programmeringsteknik med PBL

Föreläsning 2

- if-satsen
- while-slingan
- slumptal
- algoritmer
- Kap 3-4 i Dawson

Styrstrukturer

- I ett program utförs satserna i ordning uppifrån och ner:

```
print("So long")
print("and thanks")
print("for all the fish!")
```

- Hur gör man för att hoppa över en sats eller för att upprepa en sats flera gånger?

Exempel

```
talet = 17
print("Gissa talet - fina priser!")
gissning=int(input("Din gissning:"))
if gissning == talet:
    print("Hurra, du vann!")
    print("Tyvärr är priserna slut.")
else:
    print("Fel svar.")
print("\n--- Välkommen åter ---")
```

if-satsen

```
if villkor:
    block1
elif:
    block2
else:
    block3
```

- *if-satsen* används för val mellan två alternativ.
- *elif* och *else* kan användas vid behov.
- Raderna efter kolon bildar ett *block*: en eller flera satser som är *indenterade* (tabbar i början av raden).

Villkor

- Ett villkor har värdet `True` eller `False`.
- Datatypen kallas *boolean*.
- En *boolesk* variabel kan sättas direkt till ett villkorsvärde.
- Exempel: `spara = True`

jämföra=compare
villkor=condition

Operatorer i villkor

Operator	Betyder	Om vi satt `dag=20` blir
`==`	lika med	`dag==20` True
`!=`	skilt från	`dag!=20` 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<=20` True

Jämföra strängar

- Strängar kan jämföras med avseende på likhet:

```
if namn == "Linda":
   print("Inte du nu igen...")
```

- Och även med alfabetisk ordning:

```
if "elefant" < "elmätare":
   print("elefant kommer före\
          elmätare i ordlistan")
```

Jämföra flyttal

- I program där man räknar med flyttal bör man inte jämföra med avseende på likhet.
- Istället för att skriva

```
if radie == 8.13:
```

- så skriver man

```
if abs(radie - 8.13) < 0.001:
```

Kombinera villkor

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

```
if pris < 500 and taltid >= 14:
  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)
print("Gissa talet - fina priser!")
gissning = int(input("Din gissning:")
if gissning == talet:
    print("Hurra, du vann!")
    print("Tyvärr är priserna slut.")
else:
    print("Fel svar.")
print("\n--- Välkommen åter ---")
```

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!

Två exempel

```
vatten = 0
while vatten < 10:
   vatten += 3
```

```
lamm = 0
somnat = False
while not somnat:
   lamm += 1
```

Oändlig slinga

- Om villkoret aldrig uppfylls får man en slinga som upprepas i all oändlighet.
- Kan yttra sig som att programmet "hänger sig" - inget händer
- Eller att massor av text rusar förbi på skärmen (om man har utskrift i slingan).
- Avbryt programmet genom att trycka **Ctrl-C** (Ctrl och C samtidigt).

Uppgift: Hur vet Python vilka satser som ska upprepas i en while-slinga och vilka som ska utföras efteråt?

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 = int(input("Gissa mitt tal:"))
while gissning != tal:
    if gissning > tal:
        gissning = int(input("Lägre:"))
    elif gissning < tal:
        gissning = int(input("Högre:"))
print("Bravo, du gissade rätt!")
```