

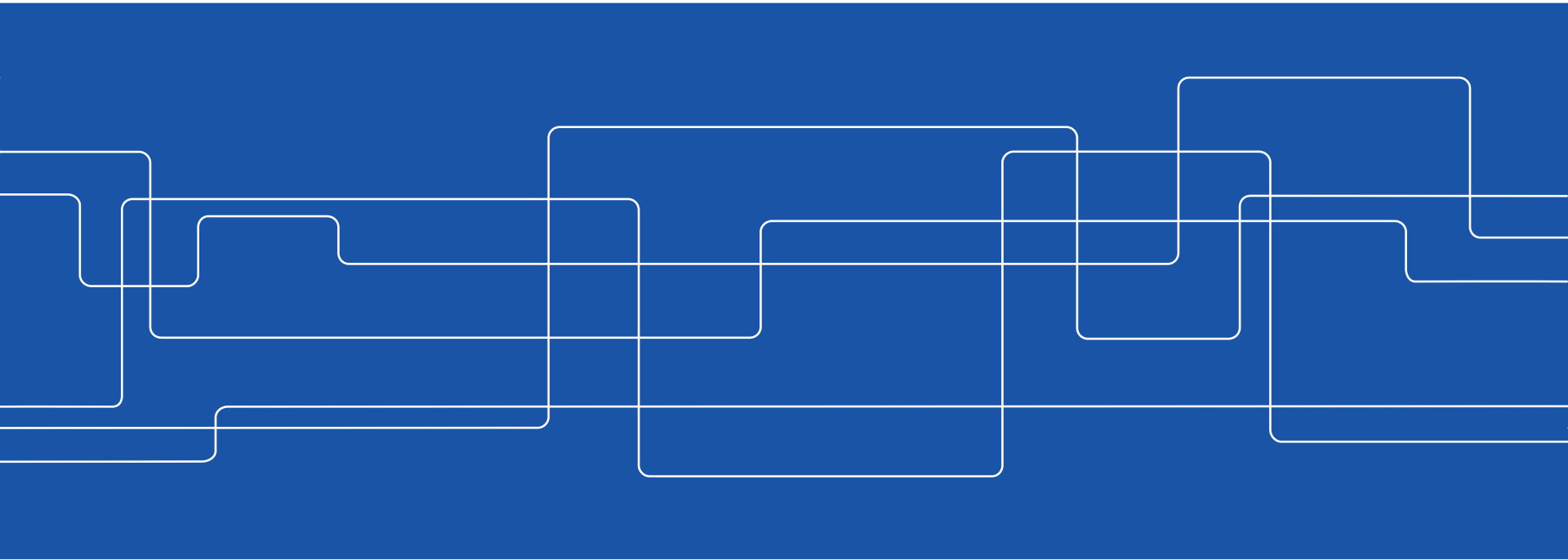


Föreläsning 2

Programmeringsteknik och C

DD1316

Mikael Djurfeldt <mdj@kth.se>





Föreläsning 2

Programmeringsteknik och C

- Python—introduktion
- Utskrift
- Inläsning
- Variabler
- Datatyp
- Aritmetiska operatorer
- Funktioner
- Omvandling av typer
- Reserverade ord
- Logiska operatorer
- If-sats
- While-sats
- Kommentarer



Programmering

- Programmering betyder att instruera en dator
- Ett program är en sekvens av instruktioner till en dator
- Datorer kan i grunden endast förstå maskinkod d.v.s kombinationer av ettor och nollor
- Att skriva maskinkod, d.v.s instruktioner i form av kombination av ettor och nollor, är svårt och tidskrävande



Programspråk

- Programspråk används av människor för att skriva datorprogram på ett mer begripligt och lätthanterligt sätt (än maskinkod)
- Programspråk tolkas eller översätts till maskinkod av maskinkodsprogram (som i sin tur kan vara *skrivna* i ett programspråk)
- En instruktion som är skriven i ett programspråk måste följa språkets regler för att kunna tolkas eller omvandlas till maskinkod

Utskrift på skärmen

1

```
print("Hej")  
print("Hur är det?")
```

Hej

Hur är det?

2

```
print("Hej", end="")  
print("Hur är det?")
```

HejHur är det?

Syntax och semantik

- **Syntax** är programspråkets struktur, dvs vilka element som får förekomma och i vilken ordning.
- Exempel på syntaktiska fel:

```
print "Hej"
```

```
print ("Hej")
```

- **Semantik** är innebörden hos ett stycke programkod, tex en sats (i Python ungefär "programrad") eller ett program.
- Exempel på semantiskt fel:

```
print ("Hälften av x är", x/3)
```



Variabel

- Med hjälp av variabler kan man lagra ett värde i datorns minne så att programmet kan komma åt det lagrade värdet vid senare tillfällen i programmet.
- Exempel:

```
age = 20
```

```
name = "Mikael"
```

```
length = 1.87
```

Variabelnamn

- Använd beskrivande variabelnamn som talar om vilken roll variabeln har

Bra: `nTurns`, `n_turns`, `numberOfTurns`

Dåligt: `pluttilutt`, `zz45`

- Undvik *alltför* långa namn
- Var konsekvent vid val av variabelnamn (tex antingen genomgående “camelCase” eller “under_score”)
- Försök att följa språkets tradition för val av variabelnamn (Ex: I Python brukar ett variabelnamn inledas med gemen)
- Python är ett skiftlägekänligt (case sensitive) språk

Reserverade ord

- En del ord får inte användas som variabelnamn eftersom de har särskild betydelse för Python. Dessa kallas **reserverade ord**:

<code>is</code>	<code>elif</code>	<code>import</code>	<code>global</code>
<code>if</code>	<code>from</code>	<code>pass</code>	<code>class</code>
<code>in</code>	<code>return</code>	<code>except</code>	<code>continue</code>
<code>def</code>	<code>and</code>	<code>else</code>	<code>raise</code>
<code>del</code>	<code>or</code>	<code>continue</code>	<code>assert</code>
<code>for</code>	<code>while</code>	<code>break</code>	<code>exec</code>
<code>try</code>	<code>not</code>	<code>finally</code>	<code>lambda</code>



Datatyper

- Datatyp för text (Sträng):
 - str t.ex: "hej", "12"
- Numeriska datatyper:
 - int t.ex: 12
 - float t.ex: 12.0

Inläsning


- Inläsning från tangentbordet görs m.h.a funktionen

```
input()
```

Datorn läser inmatade tecken tills användaren trycker på "Enter" och resultatet blir en sträng med dessa tecken

- Om man ger en textsträng som argument till input skrivs denna ut som prompt innan input väntar på inmatning:

```
input("Vad heter du?")
```

A horizontal curly brace is drawn under the text "Vad heter du?" in the code snippet above, pointing to the word "argument" below it.
argument



Omvandling av typer

Omvandling till typerna str, integer och float görs m.h.a följande funktioner:

`str(x)`

`int(x)`

`float(x)`

Exempel:

```
age_str = input("ange ålder:")
```

```
age = int(age_str)
```



Kommentarer

```
# This is a comment  
  
print("Learning Python is easy!")
```

Learning Python is easy!

- Varför och när skriver man kommentarer i ett program?
 - Programkoden själv räcker inte för att göra koden begriplig för en annan programmerare
 - Efter ett par veckor är man själv en annan programmerare!
 - Dokumentera med kommentarer allt som inte är självklart
 - Funktioner/metoder (senare): Dokumentera vad de gör, deras parametrar och funktionens/metodens returvärde

Operatorerna + och *

- Operatoren + används för att konkatenera två strängar.

"ab"+"ba" → "abba"

- Man kan använda * följd av ett heltal för att upprepa en sträng ett antal gånger.

"mam"*2 → "mammam"



Minilabb

Skriv ett program som frågar efter användarens ålder och beräknar och skriver ut examensålder.

```
age = input("Hur gammal är du?")  
age = int(age)  
examinationAge = age + 4.5  
print("Du kommer att vara",  
      examinationAge,  
      "när du tar examen!")
```



Jämförelseoperatorer

Följande är operatorer som används för att jämföra värden. De har ett boolskt värde d.v.s. sant (True) eller falskt (False):

==

!=

<

<=

>

>=



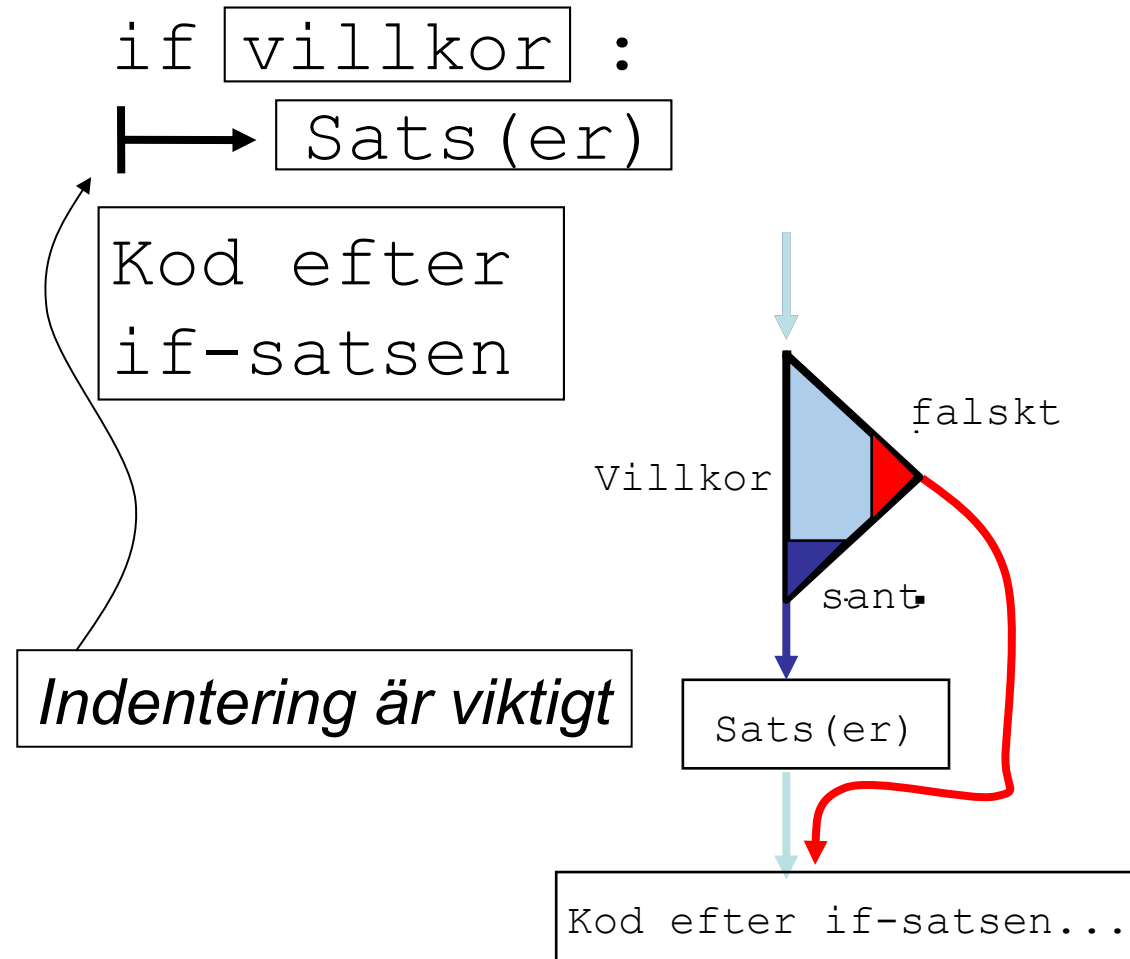
If-sats (villkorssats)

En if-sats används för villkorlig exekvering av en eller flera satser.

Exempel:

```
if bokpris > 500:  
    print("dyr bok!")  
    print("ingen affär")  
print("hejdå")
```

If-satsens struktur



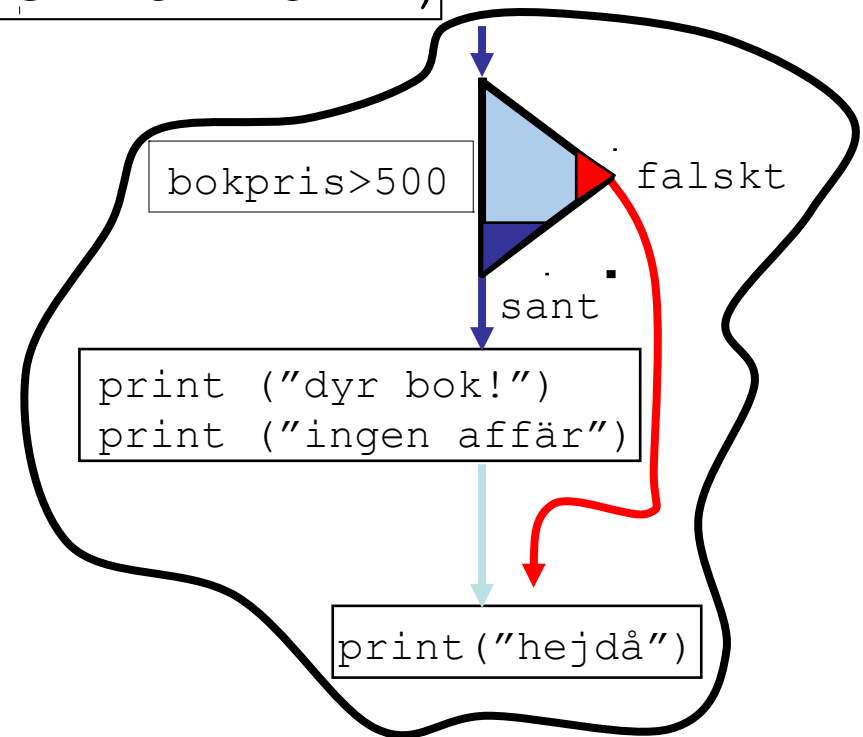
Exempel

```
if bokpris > 500 :
```

```
    print("dyr bok!")
```

```
    print("ingen affär")
```

```
print("hejdå")
```





Exempel

```
age = input("ålder:")  
age = float(age)  
if age < 20:  
    print ("Åldersgräns är 20 för att kunna vara"  
           "systembolagets kund")
```



elif och else

If-satser kan kombineras med `elif` och `else`.

Exempel:

```
if bokpris > 500:
    print("dyrbok, ingen affär!")
elif bokpris > 300:
    print("dyr men jag behöver boken!")
else:
    print("billig bok, köp snabbt!")
```



While-loop (while-slinga)

while-slinga används för att exekvera en eller flera satster ett antal gånger.

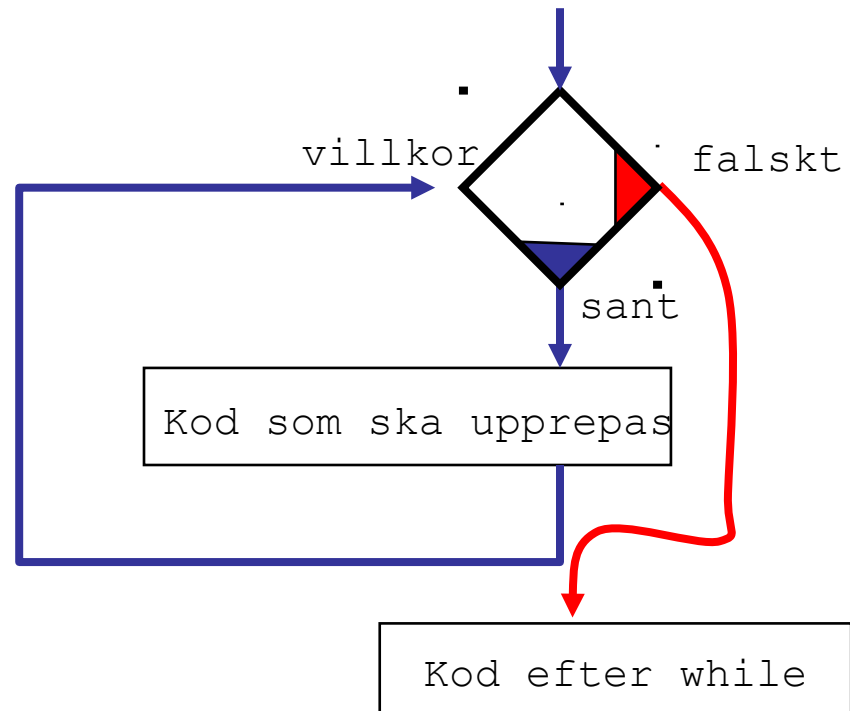
Exempel:

```
varv = 0  
  
while varv < 3:  
    print("Hej")  
    varv = varv + 1
```

While-satsens struktur

```
while villkor:
```

```
    kod som ska upprepas
```



Indentering

Indentering (indragning av kod) har stor betydelse i python.

Hur många gånger skrivs Hej ut av följande program?

```
varv = 0
while varv < 3:
    print ("Hej")
varv = varv + 1
```


Logiska operatorer

A	B	A and B	A or B	not A
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

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

```
if pris < 1000 and taltid >= 100:  
    print("Telefonfynd!")
```

Formatering

a="förförkortad diverse:%5.3s"% "diverse"

string

förförkortad diverse: div
5
3



Formatering


t="A%11.2f number"%52687.43456

float

A number

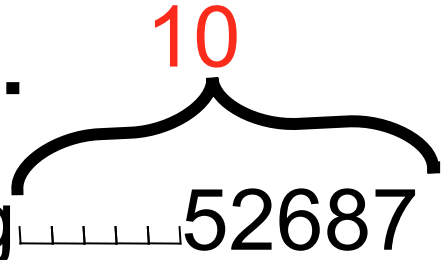
Formatering

t="A big%10d number"%52687.



digit

A big ¹⁰52687 number





Funktioner

Exempel:

```
# Compute the square of x
# Parameters:
# x: number
# Returns square of x

def square (x):
    return x * x

y = 3

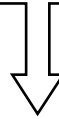
print ("Kvadraten på", y, "är", square (y))
```

Kvadraten på 3 är 9

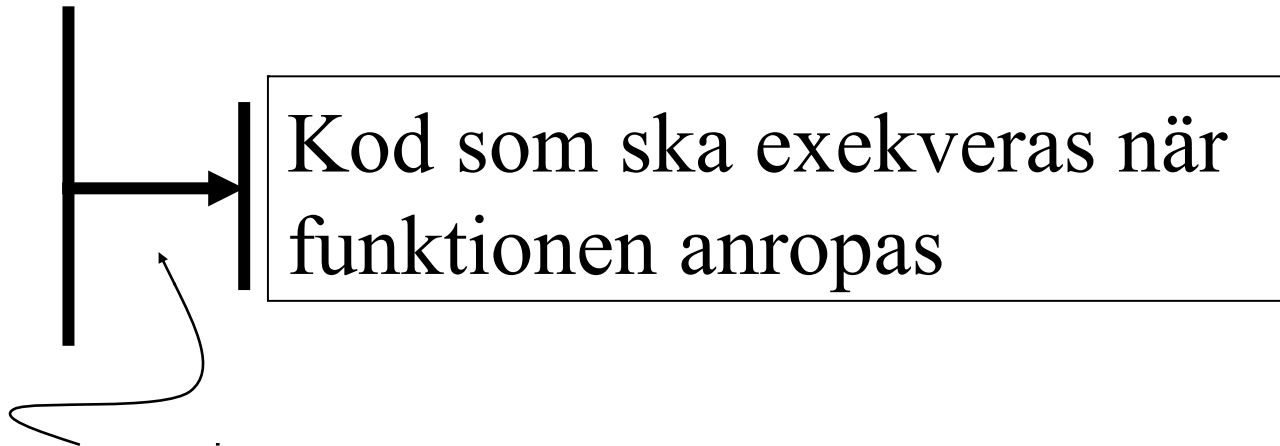
Funktioner

Syntax:

parametrar



```
def funktionensnamn ( ) :
```



Indragning är viktigt!



Funktioner

- Indata skickas till funktioner via funktionens **formella parametrar** (tex `x` i `def square (x)`)
- Värdet man skickar till funktionen vid ett anrop kallas **argument** (tex `3` i `square (3)`)
- Funktioner returnerar utdata med hjälp av `return`-satsen
- Om en funktion inte har `return`-sats i kroppen kommer funktionen att returnera `None`. (`None` betyder ingenting i python.)



Funktioner

- Används för att dela upp ett program i **naturliga** och **återanvändbara** delar
- En funktion tar oftast **indata** och ger **utdata**, men en funktion kan även utföra operationer med bieffekter
- Man kan undvika upprepning av kod genom att använda egna funktioner och parametrar
- Med hjälp av egna funktioner inför man abstraktion i sitt program



Sammanfattning

- Använd beskrivande variabelnamn
- Var noggrann med datatyper (skilj mellan sträng och tal)
- `input()` används för inmatning
- Planera ditt program innan du börjar skriva kod för det
- If-satser används för att villkorligt köra en eller flera satser



Sammanfattning

- while-slinga (while-sats) används för att upprepa en eller flera satser
- Genom att använda while-satser får man kortare kod, mer genomskådlig kod och ett mer flexibelt program
- När två eller fler while-slingor hamnar innanför varandra kallas det nästlade slingor
- Funktioner används för att dela upp ett program i återanvändbara delar