

# Introduktion till programmering

Loopar och if-satser

# Dagens föreläsning

- Vad vi gjort hittills – snabb översikt
- Selektion genom **if-satser**
- Iteration genom **loopar**
  - Loopen: **for**
  - Loopen: **while**
- Kombinera detta i olika exempel
  - Bygga en meny i ett program
- Välja programrepresentanter

**Frågor innan vi börjar?**

# Mentimeter

En snabb statuskoll!



38%

HT18

73%

HT19

34%

HT20



43%

HT20

**38% => 73% => 34% => 43%**

HT18

HT19

HT20

HT21

# Varför är detta viktigt?



[Sammanfattning](#)

[Kursplan](#)

## SAMMANFATTNING

Kursens syfte är att studenten ska utveckla kunskaper i användandet av databaser i programmeringssammanhang, såväl design och konstruktion av databaser, som användning av databaser för att lagra och hämta information.

### BEHÖRIGHETSKRAV

1,5 hp programmering.

Utöver ovanstående formella förkunskapskrav förutsätts även att studenten har kunskaper totalt motsvarande inom programmering.



[Sammanfattning](#)

[Kursplan](#)

[Kontakt](#)

## SAMMANFATTNING

### BEHÖRIGHETSKRAV

9 hp från någon eller flera av kurserna:

DA339A Objektorienterad programmering

DA315A Objektorienterad spelprogrammering

DA156A Introduktion till webbutveckling

DA354A Introduktion till programmering

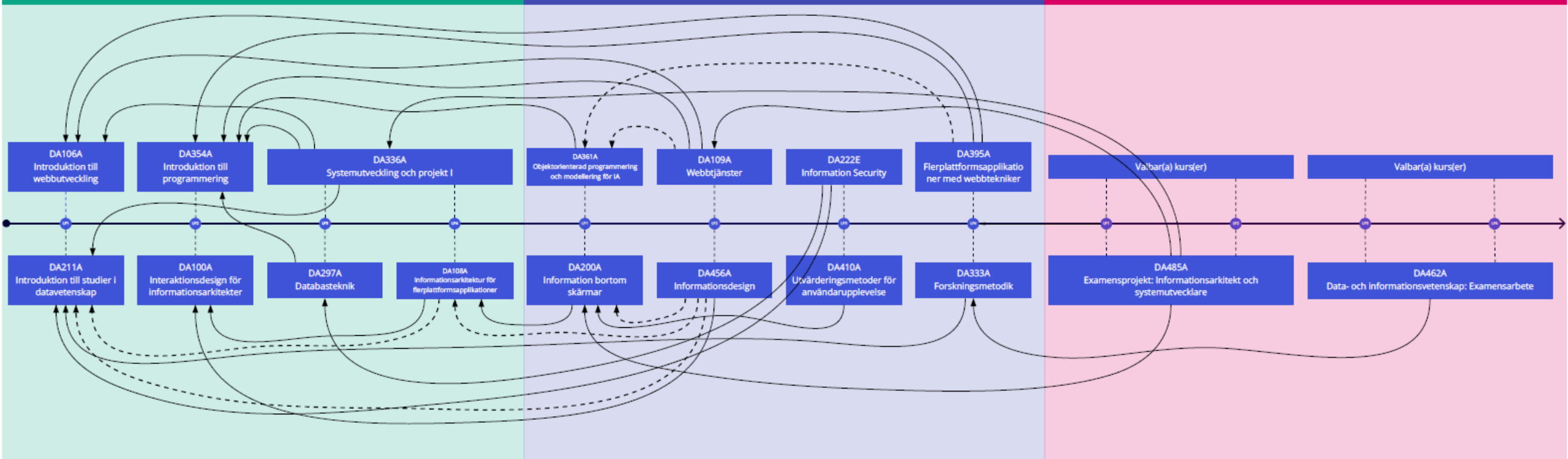
Utöver ovanstående formella förkunskapskrav förutsätts även att studenten har kunskaper från kurserna:

DA211A Introduktion till studier i datavetenskap/DA154A Introduktion till datavetenskap , DA156A Introduktion till webbutveckling/DA106A Introduktion till webbutveckling och DA354A Introduktion till programmering

År 1

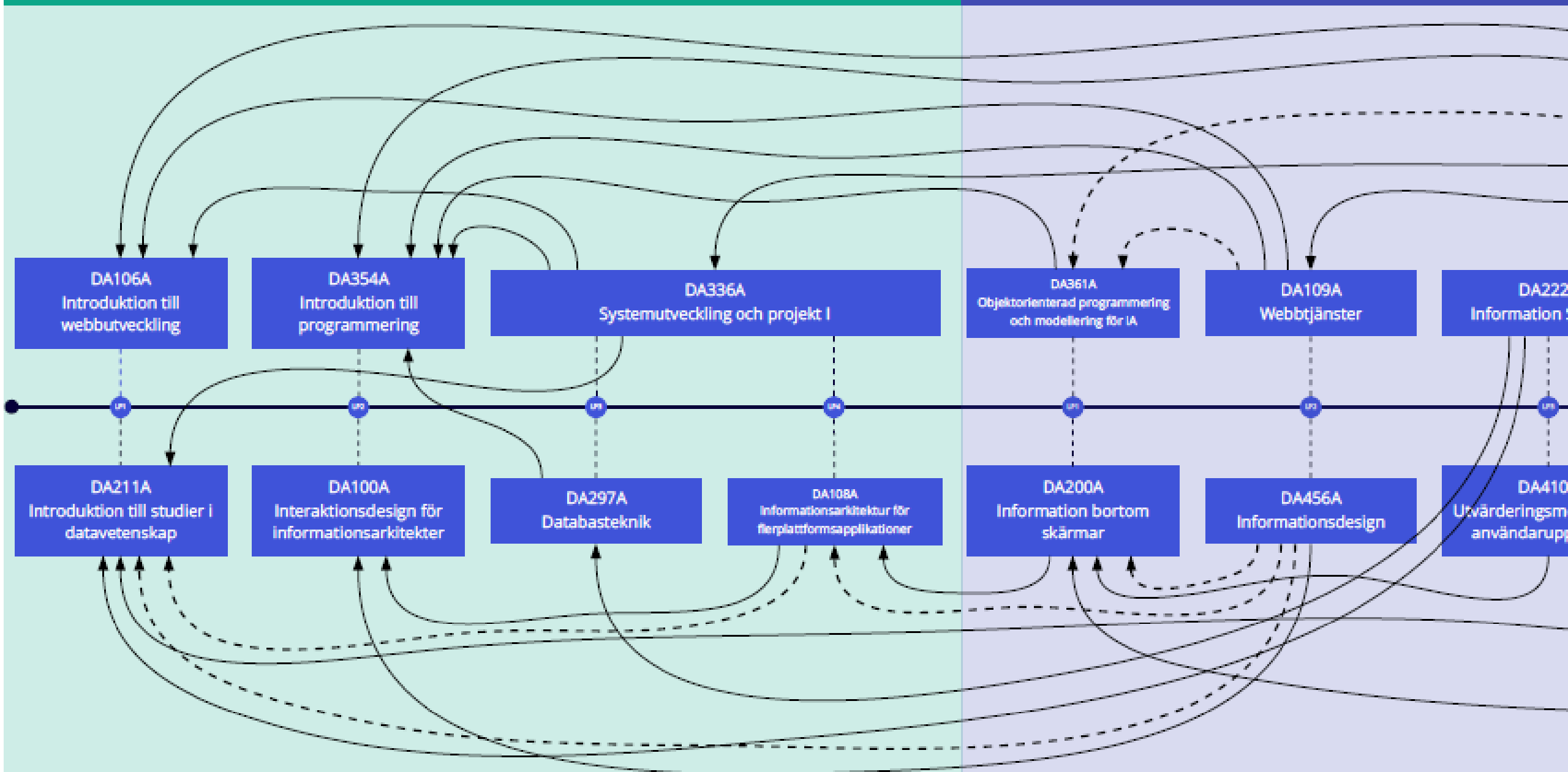
År 2

År 3



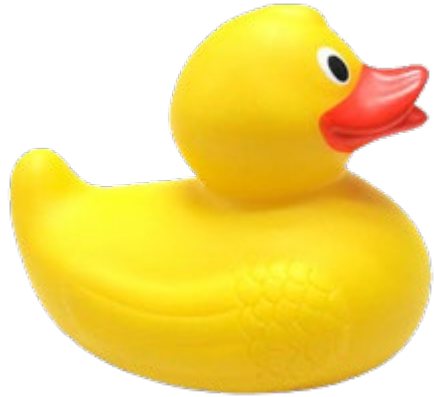
År 1

År 2



# Varför är detta viktigt?

- Behörighetskrav till kommande kurser
- Man glömmer snabbt – mycket jobbigare att göra klart en kurs i efterhand
- Man slipper ev. problem med CSN
- Det är skönt att inte ha saker släpande efter sig



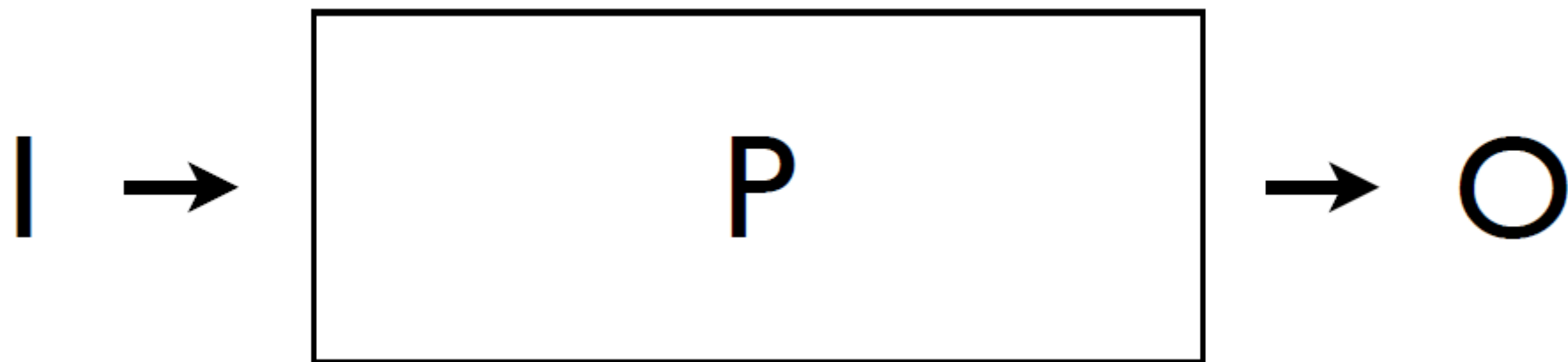
# Johans tips



En **lat** programmerare är  
en **bra** programmerare







32

str

"32"

# Att skapa en funktion, och köra den

```
print("_"*40)
print("Welcome")
print("_"*40)
```



```
def welcome():
    print("_"*40)
    print("Welcome")
    print("_"*40)

welcome()
```

Definierar en funktion      Funktionens namn      Funktionens parameter

```
def shout(text):  
    result = text.upper()  
    print(result)
```

Kod i  
funktionen

```
shout("Anton är bäst!")
```

Kör funktionen "shout"

Med argumentet  
"Anton är bäst!"

# Returvärden

- Funktioner genomför sina instruktioner – och kan sedan avsluta med att returnera ett resultat.
- T.ex. en funktion som omvandlar meter till yards.
  - Parameter: meters
  - Returvärde: yards
- Skulle kunna se ut på följande sätt:

```
def meters_to_yards(meters):  
    yard_per_meter = 0.9144  
    result = meters/yard_per_meter  
    return result
```



```
109.36132983377078  
218.72265966754156  
328.0839895013123
```

```
print(meters_to_yards(100))  
print(meters_to_yards(200))  
print(meters_to_yards(300))
```

# Att använda sig utav moduler

- Det är väldigt enkelt att använda sig utav dessa inbyggda moduler. Vill vi använda oss utav modulen "math" skriver vi:

```
# Importerar alla funktioner från modulen "math"
from math import *
print(floor(4.321))
print(ceil(4.321))
```



4.0  
5.0

```
# Importerar funktionerna "floor" & "ceil" from modulen "math"
from math import floor, ceil
print(floor(4.321))
print(ceil(4.321))
```



4.0  
5.0

```
# Importerar modulen "math"
import math
print(math.floor(4.321))
print(math.ceil(4.321))
```



4.0  
5.0

# Yardskonverterare

```
1  def meters_to_yards(meter):
2      '''Konverterar meter till yards'''
3      yards_per_meter = 0.9144
4      result = meter/yards_per_meter
5      return result
6
7
8  def main():
9      '''Frågar användaren efter meter, resultatet
10     visas sedan upp'''
11     user_meters = input("Hur många meter vill du omvandla till yards? ")
12     yards = meters_to_yards(int(user_meters))
13     print(user_meters, "meter är", round(yards), "yards")
14
15  main()
```

# Men... namngivning? Varför är det viktigt? Och varför är det svårt?

**A**

```
def shout(text):  
    result = text.upper()  
    print result  
  
shout("Anton är bäst!")
```

**C**

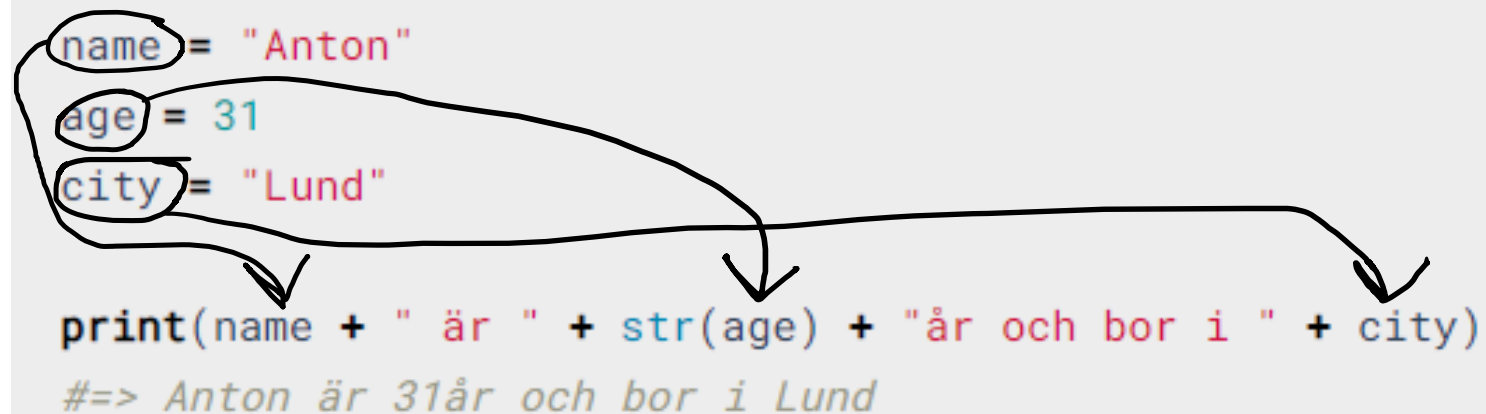
```
def hejsan(hoppsan):  
    tjosan = hoppsan.upper()  
    print tjosan  
  
hejsan("Anton är bäst!")
```

**B**

```
def a(b):  
    c = b.upper()  
    print c  
  
a("Anton är bäst!")
```

```
name = "Anton"
age = 31
city = "Lund"

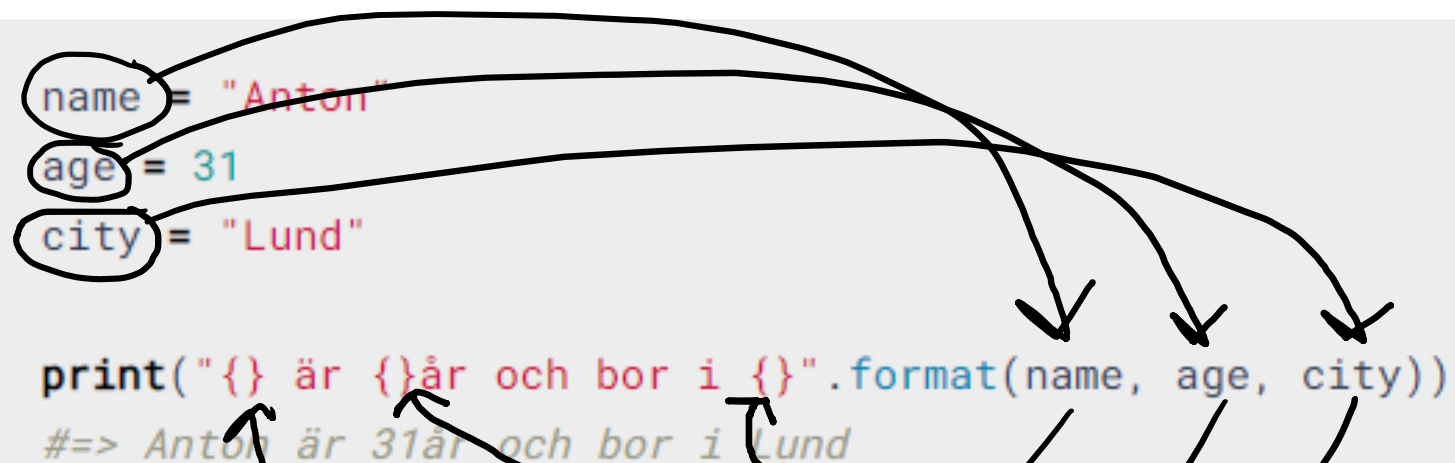
print(name + " är " + str(age) + "år och bor i " + city)
#=> Anton är 31år och bor i Lund
```



A diagram illustrating variable references in a string concatenation operation. Three variables are defined at the top: `name = "Anton"`, `age = 31`, and `city = "Lund"`. Each variable name is circled in black. Arrows originate from these circles and point to their respective placeholders in the `print` statement: `name` points to `name`, `age` points to `str(age)`, and `city` points to `city`. The output of the code is shown as `#=> Anton är 31år och bor i Lund`.

```
name = "Anton"
age = 31
city = "Lund"

print("{} är {}år och bor i {}".format(name, age, city))
#=> Anton är 31år och bor i Lund
```

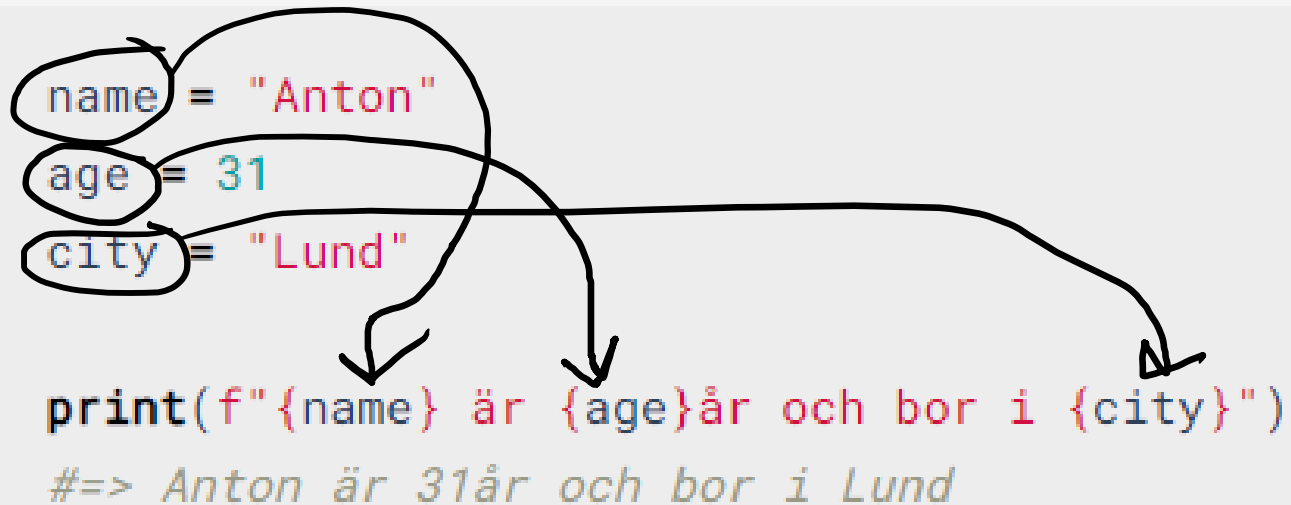


A diagram illustrating variable references in a `.format()` call. Three variables are defined at the top: `name = "Anton"`, `age = 31`, and `city = "Lund"`. Each variable name is circled in black. Arrows originate from these circles and point to their respective placeholders in the `print` statement: `name` points to the first `{}`, `age` points to the second `{}`, and `city` points to the third `{}`. The output of the code is shown as `#=> Anton är 31år och bor i Lund`.



## Utskrift med **f** (Formatted string literals)

Sedan Python version 3.6 så kan man även använda s.k. *Formatted string literals* för att skriva ut variabler i en sträng. Detta gör det möjligt att direkt ange variabler (värden) i en sträng, vilket kan snabba upp hur vi gör utskrifter. Detta "aktiveras" genom att man skriver **f** framför den sträng man vill använda variabler i, t.ex.



The diagram illustrates the use of formatted string literals in Python. It shows three variable assignments at the top: `name = "Anton"`, `age = 31`, and `city = "Lund"`. Each variable name is circled. Arrows from these circles point to the corresponding placeholders in the `print` statement below: `print(f"{name} är {age}år och bor i {city}")`. The `f` at the start of the string indicates it is a formatted string literal. Below the code, the output is shown: `#=> Anton är 31år och bor i Lund`.

```
name = "Anton"
age = 31
city = "Lund"

print(f"{name} är {age}år och bor i {city}")

#=> Anton är 31år och bor i Lund
```

Programming is  
**10%** writing code  
and **90%**  
understanding why  
it's not working.

# Vad gör ett program egentligen?

Input	↔	Från användare
Output	↔	Till användare
Beräkningar	↔	Beräkningar
Konditional exekvering	↔	If-satser
Repetition	↔	Iterationer



12:26 RMA 0 0 ATM

Santander FIFA.com FIFA.com FIFA.com FIFA.com FIFA.com



 7. E. HAZARD



18. FELIPE 







# if-satser i Python





**How I Answer Every**

False

**True or False Quiz**

**Frågor ska vara JA eller NEJ**

Om det regnar

Sant

Ta ett paraply

Om det är kallt

Sant

Ta på dig en jacka

Om du är hungrig

Sant

Ät en macka



# 20 år eller äldre?

.....

Denna webbsida innehåller information om alkoholdrycker. För inköp och besök på denna webbplats måste du vara 20 år eller äldre.

**JAG ÄR UNDER 20 ÅR**

**JAG ÄR 20 ÅR ELLER ÄLDRE**

När jag bekräftar att jag är 20 år eller äldre godkänner jag också att systembolaget.se använder cookies. [Vad är cookies?](#)

[Varför är det åldersgräns](#) på systembolaget.se?

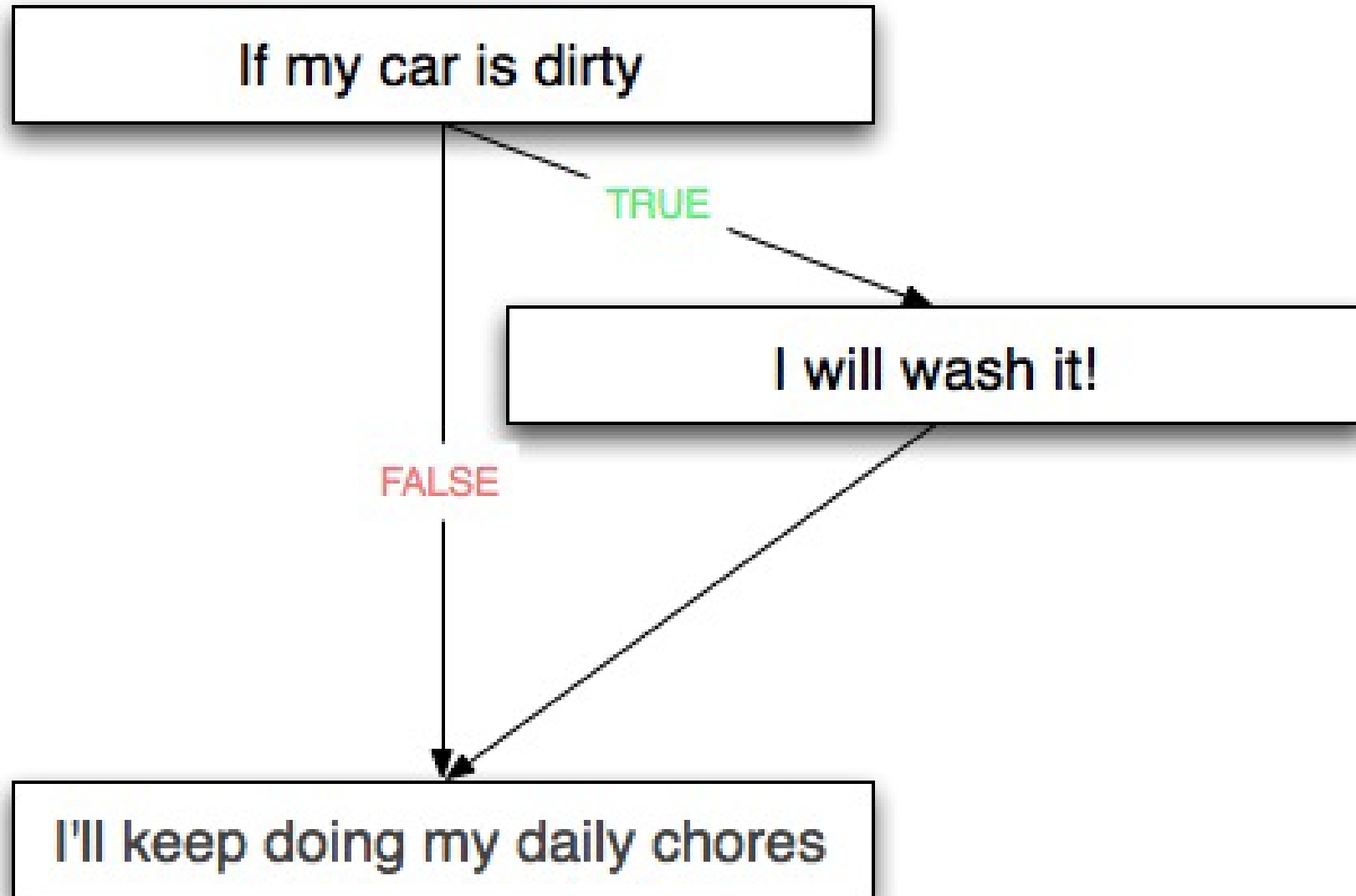
# Använder den ljusa sidan av kraften

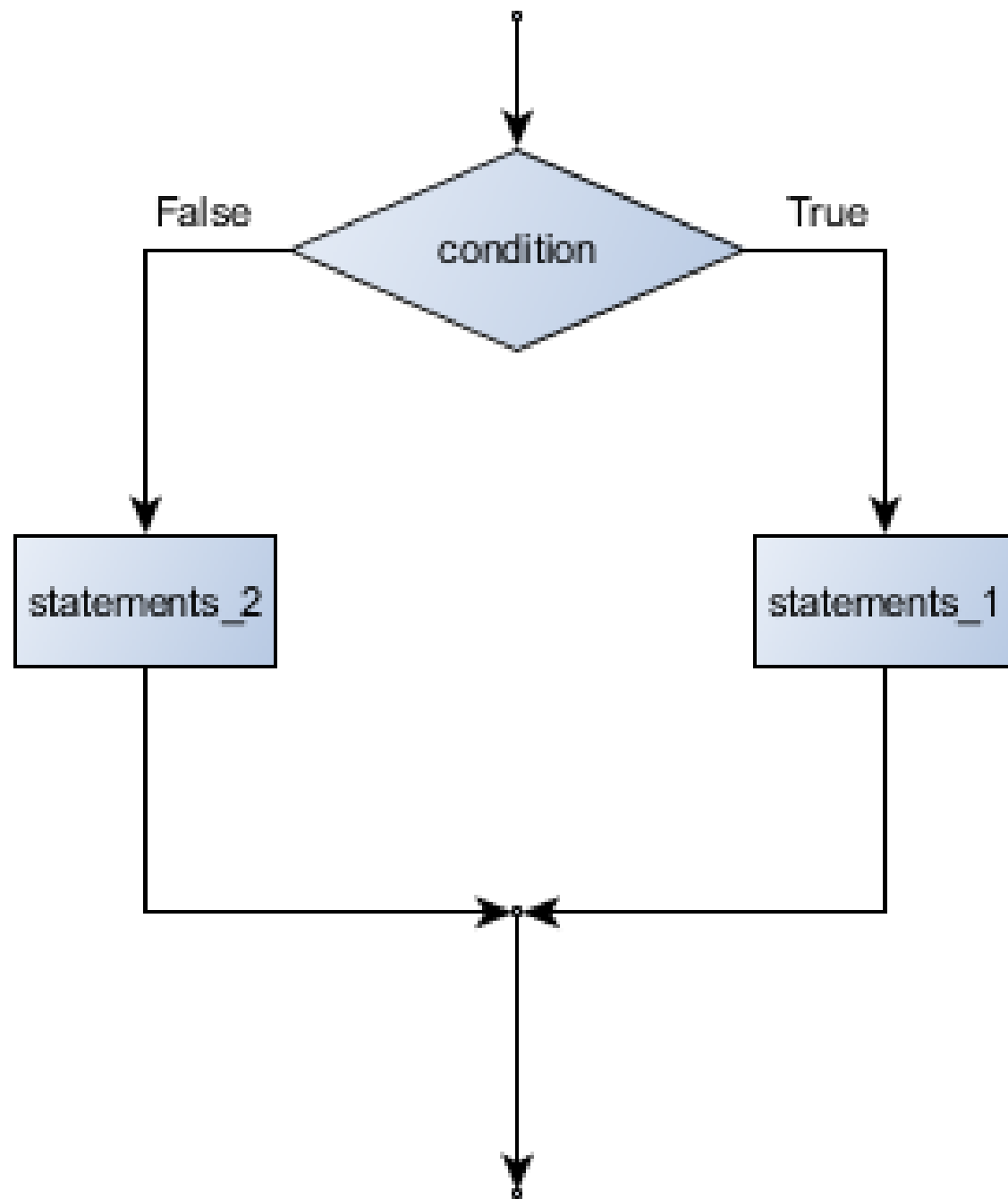
Sant



Falskt







# Uttryck i Python

- Vi vill veta om något är **sant**

```
if True:
    # Om sant, gör detta
else:
    # Annars, gör detta
```

Som falskt räknas, förtutom **False**:

- **None**
  - Siffran 0
  - Tomma strängar, eller datasamlingar: "", [], {}
- De flesta andra värden räknas som sanna

Uttryck	Sant/Falskt
<b>True</b>	Sant
<b>False</b>	Falskt
5 > 2	Sant
2 > 5	Falskt
5 == 5	Sant
5 == 6	Falskt
5 != 6	Sant
"Anton" == "anton"	Falskt
"Anton" == "Anton"	Sant
5 == "5"	Falskt
0 == <b>False</b>	Sant
20 == <b>True</b>	Falskt



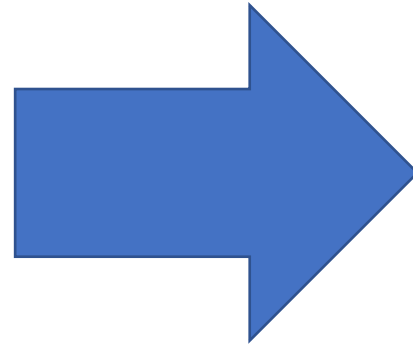
# Operatorer i Python

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	<code>a + b = 30</code>
- Subtraction	Subtracts right hand operand from left hand operand.	<code>a - b = -10</code>
* Multiplication	Multiplies values on either side of the operator	<code>a * b = 200</code>
/ Division	Divides left hand operand by right hand operand	<code>b / a = 2</code>
% Modulus	Divides left hand operand by right hand operand and returns remainder	<code>b % a = 0</code>
** Exponent	Performs exponential (power) calculation on operators	<code>a**b = 10 to the power 20</code>
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.	<code>9//2 = 4</code> and <code>9.0//2.0 = 4.0</code>

Operator	Description	Example
<code>==</code>	If the values of two operands are equal, then the condition becomes true.	<code>(a == b)</code> is not true.
<code>!=</code>	If values of two operands are not equal, then condition becomes true.	
<code>&lt;&gt;</code>	If values of two operands are not equal, then condition becomes true.	<code>(a &lt;&gt; b)</code> is true. This is similar to <code>!=</code> operator.
<code>&gt;</code>	If the value of left operand is greater than the value of right operand, then condition becomes true.	<code>(a &gt; b)</code> is not true.
<code>&lt;</code>	If the value of left operand is less than the value of right operand, then condition becomes true.	<code>(a &lt; b)</code> is true.
<code>&gt;=</code>	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	<code>(a &gt;= b)</code> is not true.
<code>&lt;=</code>	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	<code>(a &lt;= b)</code> is true.

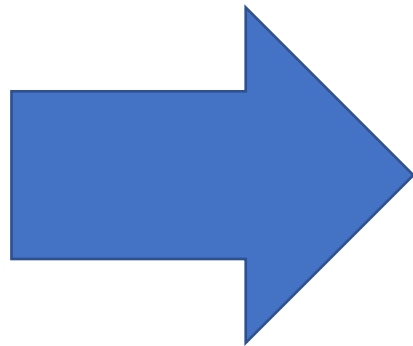
<code>x == y</code>	<code># x is equal to y</code>
<code>x != y</code>	<code># x is not equal to y</code>
<code>x &gt; y</code>	<code># x is greater than y</code>
<code>x &lt; y</code>	<code># x is less than y</code>
<code>x &gt;= y</code>	<code># x is greater than or equal to y</code>
<code>x &lt;= y</code>	<code># x is less than or equal to y</code>

```
villkor = False
if villkor == True:
    print("Ja, det är sant!")
else:
    print("Nej, det är falskt")
```



Nej, det är falskt!

```
villkor = False
if villkor:
    print("Ja, det är sant!")
else:
    print("Nej, det är falskt")
```



Nej, det är falskt!

# Vi vill kontrollera om ett uttryck är sant

## Om det regnar

```
# -*- coding: cp1252 -*-  
rains = True
```

```
print("Det var en helt vanlig dag och Kalle skulle gå till parken.")  
if rains:  
    print("Då det regnade, tog Kalle med sig ett paraply.")  
print("Framme i parken drack han saften som haft med sig.")  
print("Kalle var mycket nöjd")
```



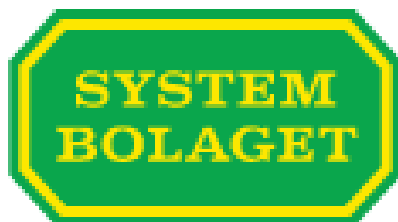
Det var en helt vanlig dag och Kalle skulle gå till parken.  
Då det regnade, tog Kalle med sig ett paraply.  
Framme i parken drack han saften som haft med sig.  
Kalle var mycket nöjd

```
# -*- coding: cp1252 -*-  
rains = False
```

```
print("Det var en helt vanlig dag och Kalle skulle gå till parken.")  
if rains:  
    print("Då det regnade, tog Kalle med sig ett paraply.")  
print("Framme i parken drack han saften som haft med sig.")  
print("Kalle var mycket nöjd")
```



Det var en helt vanlig dag och Kalle skulle gå till parken.  
Framme i parken drack han saften som haft med sig.  
Kalle var mycket nöjd



# 20 år eller äldre?

.....

Denna webbsida innehåller information om alkoholdrycker. För inköp och besök på denna webbplats måste du vara 20 år eller äldre.

**JAG ÄR UNDER 20 ÅR**

**JAG ÄR 20 ÅR ELLER ÄLDRE**

När jag bekräftar att jag är 20 år eller äldre godkänner jag också att systembolaget.se använder cookies. [Vad är cookies?](#)

[Varför är det åldersgräns](#) på systembolaget.se?

# Får du handla på systemet?

```
print("Välkommen till systembolaget!")
age = int(input("Hur gammal är du? "))
if (age < 20):
    print("Du är tyvärr inte tillräckligt gammal för att få handla här")
else:
    print("Välkommen in, hoppas du hittar det du söker efter.")
```

---

Välkommen till systembolaget!

Hur gammal är du? 18

Du är tyvärr inte tillräckligt gammal för att få handla här

---

Välkommen till systembolaget!

Hur gammal är du? 22

Välkommen in, hoppas du hittar det du söker efter.

```
- -  
if condition:  
    # Om sant gör detta
```

```
if age < 7:  
    print("Du får bara se barntillåtna filmer med vuxet sällskap")
```

```
if condition:  
    # Om sant gör detta  
else:  
    # Annars gör detta
```

```
if age >= 15:  
    print("Du får se alla filmer själv")  
else:  
    print("Kontrollera noga vilka filmer du får se")
```

```
if condition_1:  
    # Om sant gör detta och avsluta if-stats  
elif condition_2:  
    # Om sant gör detta och avsluta if-stats  
elif condition_3:  
    # Om sant gör detta och avsluta if-stats  
else:  
    # Annars gör detta
```

```
if age >= 15:  
    print("Du får se alla filmer själv")  
elif age >= 11:  
    print("Du får se filmer med 11-års-gräns själv")  
elif age >= 7:  
    print("Du får se filmer med 11-års-gräns med vuxet sällskap,", \  
          "och filmer med 7-års-gräns själv")  
else:  
    print("Du får bara se barntillåtna filmer med vuxet sällskap")
```

# Vilka filmer får man se på bio?

Genom att kontrollera ålder



```
age = int(input("Hur gammal är du? "))

if age >= 15:
    print("Du får se alla filmer själv")
elif age >= 11:
    print("Du får se filmer med 11-års-gräns själv")
elif age >= 7:
    print("Du får se filmer med 11-års-gräns med vuxet sällskap,", \
          "och filmer med 7-års-gräns själv")
else:
    print("Du får bara se barntillåtna filmer med vuxet sällskap")
```

---

```
>>> ===== RESTART =====
>>>
Hur gammal är du? 10
Du får se filmer med 11-års-gräns med vuxet sällskap, och filmer med 7-års-gräns själv
>>> ===== RESTART =====
>>>
Hur gammal är du? 25
Du får se alla filmer själv
>>> ===== RESTART =====
>>>
Hur gammal är du? 4
Du får bara se barntillåtna filmer med vuxet sällskap
```

and  
or  
not

Expression	Meaning
<code>x &gt; y and a &lt; b</code>	Is x greater than y AND is a less than b?
<code>x == y or x == z</code>	Is x equal to y OR is x equal to z?
<code>not (x &gt; y)</code>	Is the expression <code>x &gt; y</code> NOT true?

!false

it's funny because  
it's true.

Exempel

# Loopar i Python

Om och om ingen...



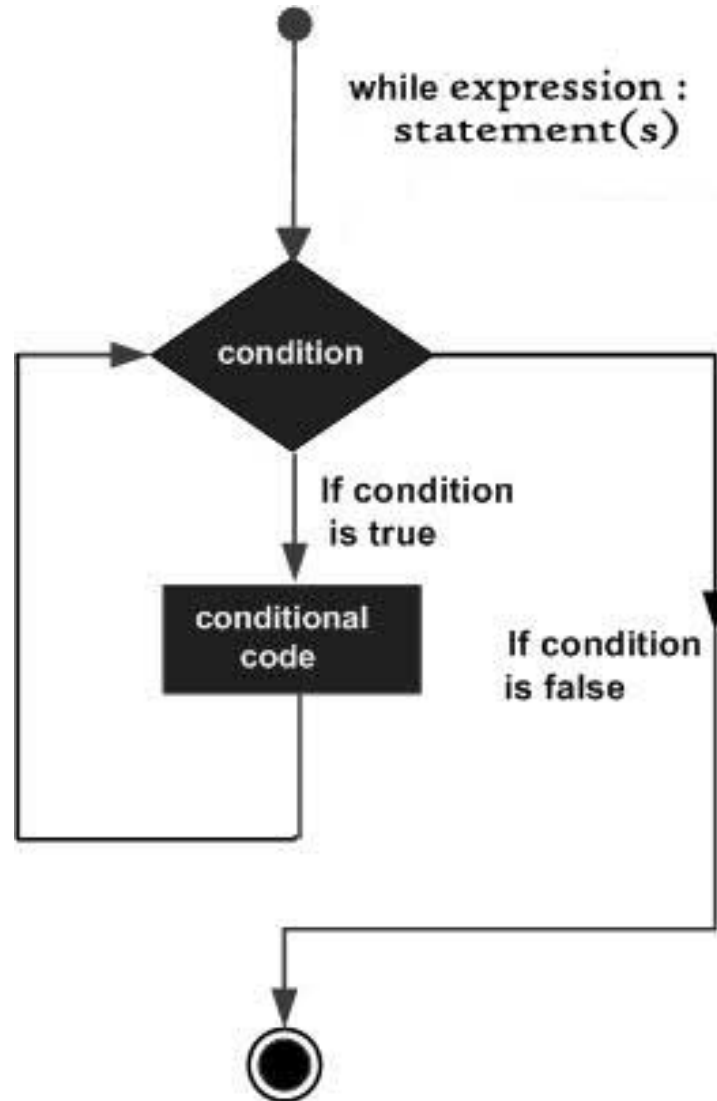


THE SIMPSONS—TM AND ©1990 TWENTIETH CENTURY FOX FILM CORPORATION. ALL RIGHTS RESERVED. THE TOPPS COMPANY, INC.

# Iteration

- Iteration = upprepning
- Upprepa en beräkning eller annan operation tills ett önskat resultat har uppnåtts
- Typer av loopar: **while** & **for**
  - **Villkorsloop**
  - **Uppräkningsloop**
- Nyckelord vi iterationer: **break** & **continue**

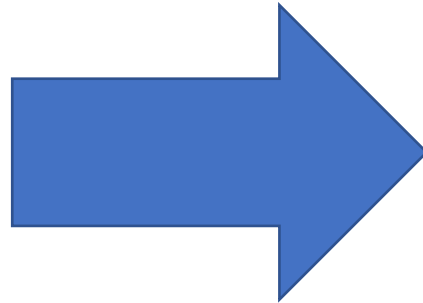
# Iteration: while-loop







```
tal = 1
while tal <= 7:
    print(tal)
    tal = tal + 1
```



1  
2  
3  
4  
5  
6  
7

# Iteration: for-loop

- Bästa när man vet antalet gånger looper ska köras
- Passar bra med datasamlingar (lister, lexikon)
- Används ofta tillsammans med funktionen **range()**

```
for i in range(0, 7):  
    print(i)
```



0  
1  
2  
3  
4  
5  
6

**Demo - loopar**