

# Introduktion till programmering

Listor och lexikon

# Dagens föreläsning

- Vad består ett program av
  - och hur lagrar vi data?
- Listor
  - lagring i sekvenser
  - i form av strängar
  - i form av värdesekvenser
- Lexikon
  - Lagring genom nyckel/värde-principen



# Frågor innan vi börjar?






## Gruppindelning - Inlämningsuppgift 3

[Anton Tibblin](#)

Alla sektioner

27 nov kl 9.15

Hej,

Här kommer gruppindelningen inför inlämningsuppgift 3 - jag trodde att Canvas hade ett smidigt verktyg för detta (**fel!**) eller att man enkelt kunde exportera de studenter som lämnat in en uppgift i vettigt format (**fel!**) - för den intresserade finns lite mer info om hur grupperna skapades nedan. Här kommer grupperna i alla fall. Tips är att kontakta varandra via Discord eller Canvas så fort som möjligt för att lägga upp en plan (inlämningsuppgiften hittar ni här: <https://mau-webb.github.io/resurser/da354a-ht23/4-listor-och-lexikon/assignment/> .

När ni lämnar in sen, så lämnar en person in för hela gruppen.

### Grupper:

- Grupp 1: Adam Areberg, Arissa Rakluea, Matina El-Obari
- Grupp 2: Robin Garmy, Tobias Björklund, Cornelia Rönnemyr Papic
- Grupp 3: Joel Bengtsson, Medina Muratagic, Anton Lind
- Grupp 4: Tim Emander, Mira El-Malla, Artem Blinkov
- Grupp 5: William Brand, Hassanine Hussein, Bianca Rydenhov
- Grupp 6: Helin Zamar, Alhakam Ezzeddin, Adrian Begic
- Grupp 7: Abdullah Hamid, Benjamin Aston, Sofie Friholm
- Grupp 8: Malcolm Hejde, Zakariya Daham, Jacob Katz
- Grupp 9: David Miladinovic, Wilmer Nordblad Lundin, Teodor Aronsson
- Grupp 10: Shaho Murad Bradostin, Anton Johansson, Simon Sjöholm
- Grupp 11: Linnéa Närhi, Jacob Lindvall, Jezza Bushra Mahmood
- Grupp 12: Mustafa Mahamud Mohammed, Julia Franzén, My Cronqvist
- Grupp 13: Linnea Bengtsson Jonsson, Ines Suhonjic, Reyhan Doski
- Grupp 14: Tilda Olsson, Nils Norén, Malik Tallouzi
- Grupp 15: Algot Olofsson, Andreh Ghassan Hami, Simon Gorny
- Grupp 16: Amjad Sharqawi, Joyce Kromah, Aya Oweidat
- Grupp 17: Max Krüger, Thomas Iordanescu, Andreas Persson
- Grupp 18: Ellen Gembäck, Vanessa Maras
- Grupp 19: Marcus Dentz, Arian Bagher

```

import random
from bs4 import BeautifulSoup # pip install beautifulsoup4
from bs4 import UnicodeDammit # pip install beautifulsoup4
from os import listdir

def main():
    # Läs instudenterna från canvas export-mapp (som innehåller HTML-filer...)
    students = get_students_from_file()

    # Blanda studenterna
    random.shuffle(students)

    # Skapa grupper
    groups = generate_groups(students)

    # Skriv ut grupperna
    print_groups(groups)

def get_students_from_file():
    folder = "DA354A-Inl3"
    files = listdir(folder)
    students = []
    for file in files:
        with open(f"{folder}/{file}", encoding="utf8") as student_file:
            file_content = student_file.read()
            soup = BeautifulSoup(file_content, 'html.parser')
            html_title = soup.title.string.split("Anmälan till inlämningsuppgift 3: ")
            student = html_title[1]
            students.append(student)

    return students

def generate_groups(students):
    groups = []
    while len(students) > 0:
        group = []
        if len(students) > 4 or len(students) == 3:
            for i in range(3):
                group.append(students.pop())
        else:
            for i in range(2):
                group.append(students.pop())

        groups.append(group)

    return groups

def print_groups(groups):
    for i, group in enumerate(groups):
        print(f"Grupp {i+1}: {' '.join(group)}")

main()

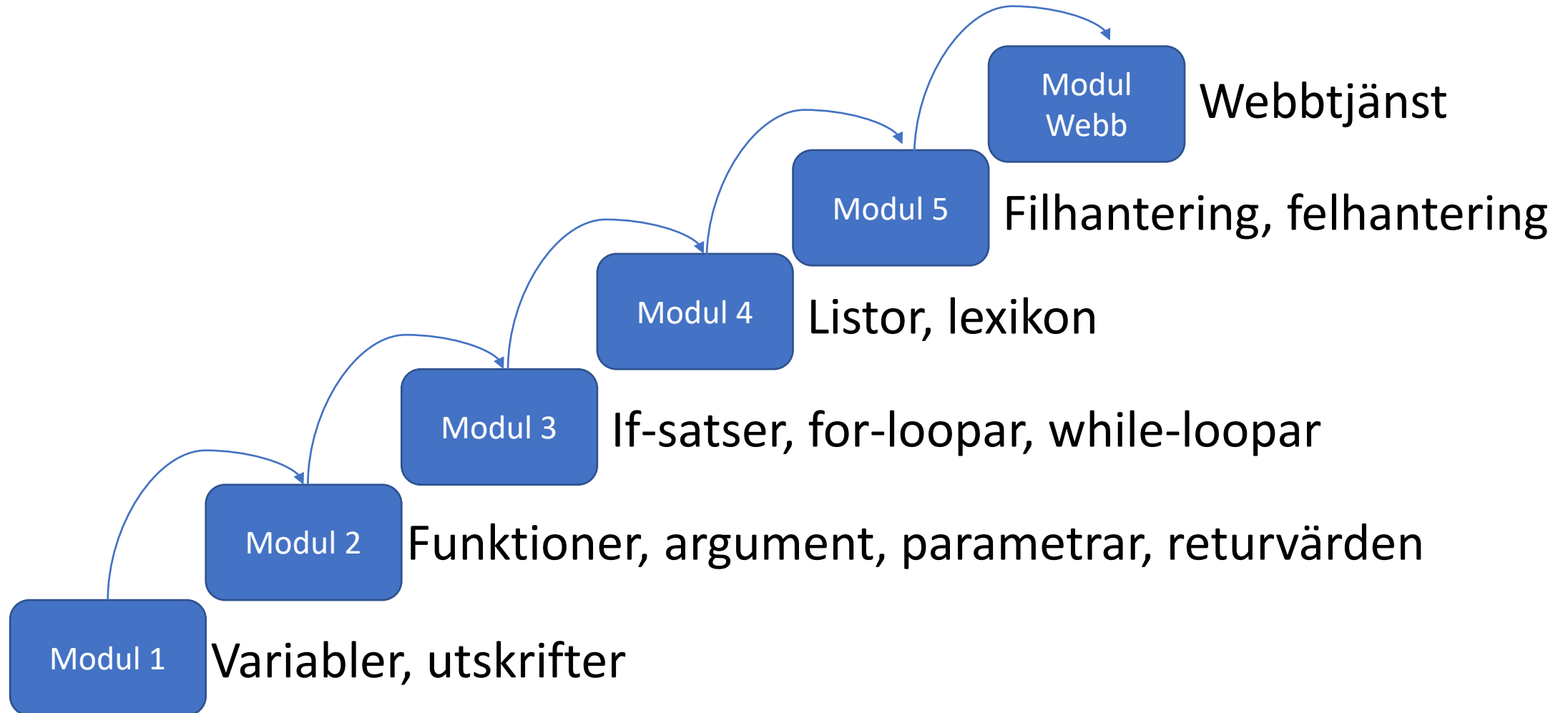
```

## Exempelkörning

```
*****
Välkommen till Bingo!
*****
Välj ett utav alternativen
1. Spela bingo!
2. Statistik
3. Avsluta
Ditt val: 1
*****
Ange fem siffror [1-25] (avgränsa med ','): 1,2,3,4,5
*****
Dragning
*****
  1  [ 2]   3   4  [ 5]
-----
  6  [ 7]   8  [ 9]  10
-----
 11  [12]  13   14   15
-----
 16  [17]  18  [19]  20
-----
[21]  22  [23]  24  [25]
-----
*****
Resultat:
Du fick 2 poäng!
*****
```

```
Välj ett utav alternativen
1. Spela bingo!
2. Statistik
3. Avsluta
Ditt val: 2
*****
Statistik
*****
Spel 1: 2 poäng
Spel 2: 2 poäng
Spel 3: 3 poäng
Spel 4: 2 poäng
*****
```

# Kursens uppbyggnad



Vecka	Datum	Tid	Moment	Lärare/Assistent	Plats	Modul
45	6/11	15-17	Kursintroduktion	Anton & Johan	NI:A0406	1
	7/11	13-15	F: Introduktion till programmering	Anton (inspelad föreläsning)	NI:A0406	
	9/11	08-12	L: Introduktion till programmering	Ellie & Hussan	NI:A0314, NI:A0318, NI:B0314	
46	13/11	10-12	F: Funktioner i Python	Anton (inspelad föreläsning)	NI:A0406	2
	14/11	13-15	Övrig tid i klassrum	-	NI:A0406	
	16/11	08-12	L: Funktioner i Python	Ellie & Hussan	NI:A0314, NI:A0318, NI:B0314	
47	21/11	13-15	F: If-satser & Loopar	Anton (inspelad föreläsning)	NI:A0406	3
	22/11	08-10	F: If-satser & Loopar (2)	Anton	NI:A0406	
	23/11	08-12	L: If-satser & Loopar	Anton & Hussan	NI:A0314, NI:A0318, NI:B0314	
48	27/11	10-12	F: Listor & Lexikon	Anton	NI:A0306	
	28/11	13-15	F: Listor & Lexikon (2)	Anton	NI:A0307	
	30/11	08-12	L: Listor & Lexikon	Anton & Hussan	NI:A0314, NI:A0318, NI:B0314	
49	4/12	10-12	F: Fil- & Felhantering	Anton	NI:A0406	5
	6/12	10-12	F: Fil- & Felhantering (2)	Anton	NI:B0E15	
	7/12	08-12	L: Listor & Lexikon	Anton & Hussan	NI:A0314, NI:A0318, NI:B0314	
50	11/12	13-15	F: Python & Webben	Anton	NI:A0607	6
	12/12	13-15	F: Python & Webben (2)	Anton	NI:A0406	
	14/12	08-12	L: Python & Webben	Anton & Hussan	NI:A0314, NI:A0318, NI:B0314	
51	19/12	13-15	F: Extra	Anton	NI:A0406	7
2	9/1	13-15	F: Objektorienterad programmering	Anton	NI:A0306	
	11/1	08-12	L: Objektorienterad programmering	Hussan	NI:A0314, NI:A0318, NI:B0314	





# Modulerna hittills

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!"

```
# Om man är 18 år eller mer
if age >= 18:
    print("Över 18år")
else:
    print("Under 18år")
```

```
# En lopp som går 10 gånger
for i in range(10):
    print(i)
```

```
# En loop som går 10 gånger
i = 0
while i < 10:
    print(i)
    i = i + 1
```

# Vad gör ett program egentligen?

Input ↔ Från användare

Output ↔ Till användare

Beräkningar ↔ Beräkningar

Konditional exekvering ↔ If-satser

Repetition ↔ Iterationer



**Spara data när  
programmet körs?**

**Men vi har ju sparat  
data innan?**

**Ex. 1 – Min filmsamling**

**Om vi vill spara 10st filmer,  
hur gör vi då?**

---

```
# Mina filmer
movie1 = "Zootropolis"
movie2 = "Captain America: Civil war"
movie3 = "Deadpool"
movie4 = "The nice Guys"
movie5 = "Djungleboken"
movie6 = "Everybody Wants Some!!"
movie7 = "10 Cloverfield Lane"
movie8 = "Sing Street"
movie9 = "Finding Dory"
movie10 = "Batman vs. Superman"
```



Om vi vill spara 50st filmer,  
hur gör vi då?

---

```
# Mina filmer
```

```
movie1 = "Zootropolis"
```

```
movie2 = "Captain America: Civil war"
```

```
movie3 = "Deadpool"
```

```
movie4 = "The nice Guys"
```

```
movie5 = "Djungleboken"
```

```
movie6 = "Everybody Wants Some!!"
```

```
movie7 = "10 Cloverfield Lane"
```

```
movie8 = "Sing Street"
```

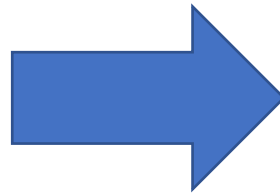
```
movie9 = "Finding Dory"
```

```
movie10 = "Batman vs. Superman"
```

**Om jag vill skriva ut mina  
10st filmer, hur gör vi då?**

```
# Mina filmer
movie1 = "Zootropolis"
movie2 = "Captain America: Civil war"
movie3 = "Deadpool"
movie4 = "The nice Guys"
movie5 = "Djungelboken"
movie6 = "Everybody Wants Some!!"
movie7 = "10 Cloverfield Lane"
movie8 = "Sing Street"
movie9 = "Finding Dory"
movie10 = "Batman vs. Superman"
```

```
# Skriver ut alla filmer
print("Filmer i min samling")
print("*"*40)
print(movie1)
print(movie2)
print(movie3)
print(movie4)
print(movie5)
print(movie6)
print(movie7)
print(movie8)
print(movie9)
print(movie10)
```

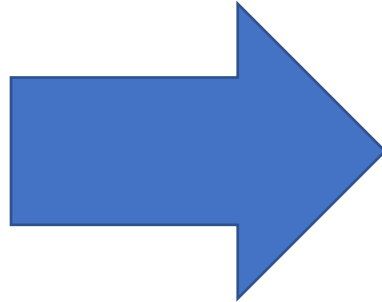


```
Filmer i min samling
*****
Zootropolis
Captain America: Civil war
Deadpool
The nice Guys
Djungelboken
Everybody Wants Some!!
10 Cloverfield Lane
Sing Street
Finding Dory
Batman vs. Superman
>>> |
```

# Vad finns det för problem med detta?

```
# Mina filmer
movie1 = "Zootropolis"
movie2 = "Captain America: Civil war"
movie3 = "Deadpool"
movie4 = "The nice Guys"
movie5 = "Djungelboken"
movie6 = "Everybody Wants Some!!"
movie7 = "10 Cloverfield Lane"
movie8 = "Sing Street"
movie9 = "Finding Dory"
movie10 = "Batman vs. Superman"
```

```
# Skriver ut alla filmer
print("Filmer i min samling")
print("*"*40)
print(movie1)
print(movie2)
print(movie3)
print(movie4)
print(movie5)
print(movie6)
print(movie7)
print(movie8)
print(movie9)
print(movie10)
```



```
Filmer i min samling
*****
Zootropolis
Captain America: Civil war
Deadpool
The nice Guys
Djungelboken
Everybody Wants Some!!
10 Cloverfield Lane
Sing Street
Finding Dory
Batman vs. Superman
>>> |
```

# Vad finns det för problem med detta?

## 1) Jag måste in i källkoden för att lägga till en ny film

- Vi vill ju att användaren ska kunna lägga in under programmets körning

## 2) Jag måste in i källkoden för att skriva ut den nya filmen

- Vi vill ju att filmen automatiskt ska skrivas ut när användaren lagt till den

## 3) Jag behöver individuellt för varje film skriva "print()"

- Vi vill ju skriva ut hela filmsamlingen, snarare än varje film individuellt

## 4) Den blir en variabel, en rad kod, per film

- Vi vill ju automatisera processen så att alla filmerna ligger i samma variabel

## 5) Hur hanterar vi 100st filmer?

```
# Mina filmer
movie1 = "Zootropolis"
movie2 = "Captain America: Civil war"
movie3 = "Deadpool"
movie4 = "The nice Guys"
movie5 = "Djungelboken"
movie6 = "Everybody Wants Some!!"
movie7 = "10 Cloverfield Lane"
movie8 = "Sing Street"
movie9 = "Finding Dory"
movie10 = "Batman vs. Superman"
```

```
# Skriver ut alla filmer
print("Filmer i min samling")
print("*"*40)
print(movie1)
print(movie2)
print(movie3)
print(movie4)
print(movie5)
print(movie6)
print(movie7)
print(movie8)
print(movie9)
print(movie10)
```

# Andra sätt att spara data?

Än som värde: strängar / boolean / nummer

# Listor!





# Listor – Sekvens av data



En **sekvens** är ett objekt som innehåller flera värden, som lagras en efter den andra. Du kan utföra operationer på en sekvens, för att undersöka och manipulera de värden som lagrats i sekvensen.

# Strängar, listor, (tupler)

Olika typer av sekvenser

# Strängar som sekvenser – med index

' R o s e s       a r e       r e d '

↑ ↑ ↑ ↑ ↑    ↑    ↑    ↑    ↑    ↑

0 1 2 3 4 5    6 7 8 9    10 11 12

```
s = "My name is Bob."
```

```
s[0] # 'M'
```

```
s[4] # 'a'
```

```
s[0:7] # "My name"
```

```
s[:7] # "My name"
```

```
s[8:15] # "is Bob."
```

```
s[8:] # "is Bob."
```

"slicing"

"substrings"

```
s = "My name is Bob."
```

```
s[-1] # '.'
```

```
s[-7] # 'i'
```

```
s[3:-1] # 'name is Bob'
```

# Metoder för strängar - test

**Table 8-1** Some string testing methods

Method	Description
<code>isalnum()</code>	Returns true if the string contains only alphabetic letters or digits and is at least one character in length. Returns false otherwise.
<code>isalpha()</code>	Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.
<code>isdigit()</code>	Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.
<code>islower()</code>	Returns true if all of the alphabetic letters in the string are lowercase, and the string contains at least one alphabetic letter. Returns false otherwise.
<code>isspace()</code>	Returns true if the string contains only whitespace characters, and is at least one character in length. Returns false otherwise. (Whitespace characters are spaces, newlines ( <code>\n</code> ), and tabs ( <code>\t</code> ).
<code>isupper()</code>	Returns true if all of the alphabetic letters in the string are uppercase, and the string contains at least one alphabetic letter. Returns false otherwise.



# Metoder för strängar - modifieringar

**Table 8-2** String Modification Methods

Method	Description
<code>lower()</code>	Returns a copy of the string with all alphabetic letters converted to lowercase. Any character that is already lowercase, or is not an alphabetic letter, is unchanged.
<code>lstrip()</code>	Returns a copy of the string with all leading whitespace characters removed. Leading whitespace characters are spaces, newlines ( <code>\n</code> ), and tabs ( <code>\t</code> ) that appear at the beginning of the string.
<code>lstrip(char)</code>	The <i>char</i> argument is a string containing a character. Returns a copy of the string with all instances of <i>char</i> that appear at the beginning of the string removed.
<code>rstrip()</code>	Returns a copy of the string with all trailing whitespace characters removed. Trailing whitespace characters are spaces, newlines ( <code>\n</code> ), and tabs ( <code>\t</code> ) that appear at the end of the string.
<code>rstrip(char)</code>	The <i>char</i> argument is a string containing a character. The method returns a copy of the string with all instances of <i>char</i> that appear at the end of the string removed.
<code>strip()</code>	Returns a copy of the string with all leading and trailing whitespace characters removed.
<code>strip(char)</code>	Returns a copy of the string with all instances of <i>char</i> that appear at the beginning and the end of the string removed.
<code>upper()</code>	Returns a copy of the string with all alphabetic letters converted to uppercase. Any character that is already uppercase, or is not an alphabetic letter, is unchanged.



**Exempel på strängar**

# Listor i Python

- Hittills har vi bara sparat ett värde i varje variabel, t.ex.

```
nr_1 = 5  
nr_2 = 3  
nr_3 = 6
```

- Vi skulle istället kunna spara dessa som en lista:

```
numbers = [5, 3, 6]
```

- På detta sätt kan vi enkelt spara flera värden på samma plats

# Listor

- En lista är en datatyp som kan innehålla flera värden
- Listor är förändringsbara, vilket gör att vi kan modifiera dem under ett programs körning
  - Tupler är inte detta
- Listor är en dynamisk datastruktur, vilket gör att vi kan:
  - Lägga till värden
  - Modifiera värden
  - Ta bort värden
- Man kan använda metoder för att modifiera listor

# To Do List



# inköpslista

ägg

ost

bröd

paprika

tomat

sallad

marmelad

```
inköpslista = [  
    "ägg",  
    "ost",  
    "bröd",  
    "paprika",  
    "tomat",  
    "sallad",  
    "marmelad"  
]
```

# inköpslista

0	→	ägg
1	→	ost
2	→	bröd
3	→	paprika
4	→	tomat
5	→	sallad
6	→	marmelad

```
inköpslista = [  
0 → "ägg",  
1 → "ost",  
2 → "bröd",  
3 → "paprika",  
4 → "tomat",  
5 → "sallad",  
6 → "marmelad"  
]
```



```
inköpslista = [  
0 → "ägg",  
1 → "ost",  
2 → "bröd",  
3 → "paprika",  
4 → "tomat",  
5 → "sallad",  
6 → "marmelad"  
]
```

**print(inköpslista[3])**

# inköpslista

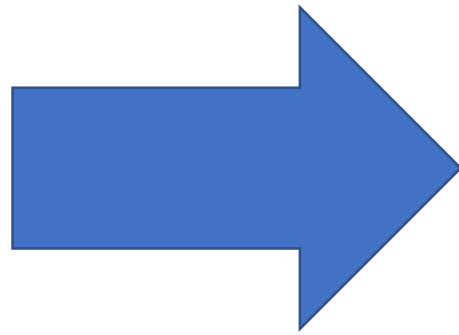
- 0 → ägg
- 1 → ost
- 2 → bröd
- 3 → paprika
- 4 → tomat
- 5 → sallad
- 6 → marmelad

```
inköpslista = [  
0 → "ägg",  
1 → "ost",  
2 → "bröd",  
3 → "paprika",  
4 → "tomat",  
5 → "sallad",  
6 → "marmelad"  
]
```

```
inköpslista = ["ägg", "ost", "bröd", "paprika", "tomat", "sallad", "marmelad"]
```



```
inköpslista = [  
    "ägg",  
    "ost",  
    "bröd",  
    "paprika",  
    "tomat",  
    "sallad",  
    "marmelad"  
]  
  
for item in inköpslista:  
    print(item)
```




```
ägg  
ost  
bröd  
paprika  
tomat  
sallad  
marmelad
```

```
# Explicit angivelse av varje element:  
l1 = [5, 10, 15, 20]
```

```
# Elementen kan vara av godtycklig typ:  
l2 = [3, "abc", 32.4, "def"]
```

```
# range() returnerar en lista  
l3 = range(3, 7) # [3, 4, 5, 6]
```



Nja, en sekvens  
av tal...

```
# Skapa med operatoren *  
l4 = [0] * 5 # [0, 0, 0, 0, 0]
```

# Uppdelning (slicing) av listor

```
# Lista på dagar i veckan
```

```
days = ["Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag", "Lördag", "Söndag"]
```

Måndag	Tisdag	Onsdag	Torsdag	Fredag	Lördag	Söndag
0	1	2	3	4	5	6
-7	-6	-5	-4	-3	-2	-1

```
# Lista på dagar i veckan
```

```
days = ["Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag", "Lördag", "Söndag"]
```

```
days[0] # Måndag
```

```
days[1] # Tisdag
```

```
days[4] # Fredag
```

```
days[-1] # Söndag
```

```
days[0:5] # Vardagar (Måndag - Fredag)
```

```
days[5:7] # Helg (Lördag, Söndag)
```

```
days[5:] # Helg (Lördag, Söndag)
```

```
days[-2:] # Helg (Lördag, Söndag)
```

[6:10]

0 1 2 3 4 5 6 7 8 9 10 11

M o n t y P y t h o n

-12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1

[-12:-7]

# Lägga ihop listor (konkatenering)

```
# Lägga ihop listor
l1 = [1, 2, 3]
l2 = [4, 5, 6]
l3 = l1 + l2
# l3 => [1, 2, 3, 4, 5, 6]

dvd = ["Fight Club", "American Beauty", "Inception"]
blueray = ["Star Wars", "Titanic", "Jurassic Park"]
movies = dvd + blueray
# movies => ["Fight Club", "American Beauty", "Inception", "Star Wars", "Titanic", "Jurassic Park"]
```

# In <sekvens>

- Vi kan kontrollera om ett värde finns i en lista genom **in**

```
>>> people = ["Anton", "Fredrik", "Kristina", "Johan"]
>>> "Johan" in people
True
>>> "Kalle" in people
False
>>> "Fred" in people
False

>>> name = "Anton"
>>> "n" in name
True
>>> "a" in name
False
>>> "ton" in name
True
```

# Funktioner för listor

**Table 8-4** A few of the list methods

Method	Description
<code>append(<i>item</i>)</code>	Adds <i>item</i> to the end of the list.
<code>index(<i>item</i>)</code>	Returns the index of the first element whose value is equal to <i>item</i> . A <code>ValueError</code> exception is raised if <i>item</i> is not found in the list.
<code>insert(<i>index</i>, <i>item</i>)</code>	Inserts <i>item</i> into the list at the specified <i>index</i> . When an item is inserted into a list, the list is expanded in size to accommodate the new item. The item that was previously at the specified index, and all the items after it, are shifted by one position toward the end of the list. No exceptions will occur if you specify an invalid index. If you specify an index beyond the end of the list, the item will be added to the end of the list. If you use a negative index that specifies an invalid position, the item will be inserted at the beginning of the list.
<code>sort()</code>	Sorts the items in the list so they appear in ascending order (from the lowest value to the highest value).
<code>remove(<i>item</i>)</code>	Removes the first occurrence of <i>item</i> from the list. A <code>ValueError</code> exception is raised if <i>item</i> is not found in the list.
<code>reverse()</code>	Reverses the order of the items in the list.

# Lägga till saker i en lista

```
numbers = [1, 2, 3]
numbers.append(4)
numbers.append(10)
print(numbers) # [1, 2, 3, 4, 10]
```

```
movies = ["Star Wars", "Titanic", "Jurassic Park"]
movies.append("Fight Club")

print(movies) # ["Star Wars", "Titanic", "Jurassic Park", "Fight Club"]
```



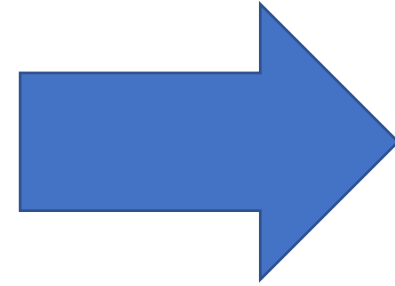
# Att gå igenom listor med data

Nu blir for-loopen väldigt häändig!

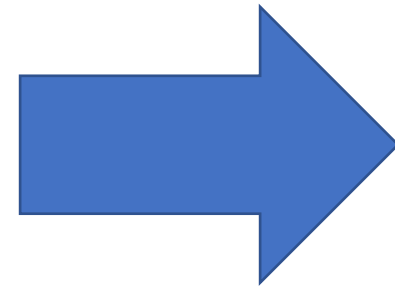
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]



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



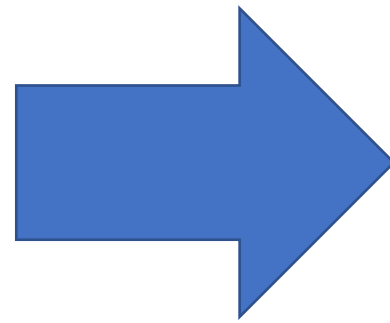
```
for number in range(10):  
    print(number)
```



0  
1  
2  
3  
4  
5  
6  
7  
8  
9

```
numbers = [1, 2, 3, 4, 5]
```

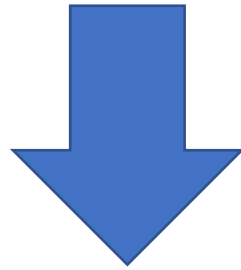
```
for number in numbers:  
    print(number)
```



1  
2  
3  
4  
5

```
bands = ["Iron Maiden", "The Killers", "Queen", "AC/DC"]
```

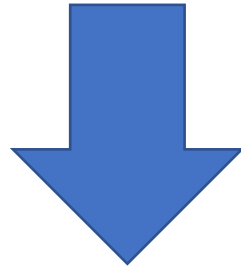
```
for band in bands:  
    print(band)
```



```
Iron Maiden  
The Killers  
Queen  
AC/DC
```

```
days = ["Måndag", "Tisdag", "Onsdag", "Torsdag", "Fredag"]
```

```
for day in days:  
    print(day)
```



```
Måndag  
Tisdag  
Onsdag  
Torsdag  
Fredag
```

A still from the movie Toy Story showing Woody and Buzz Lightyear. Woody is on the left, looking slightly concerned or skeptical. Buzz is on the right, looking enthusiastic and holding up a small yellow object. The word "LISTS" is superimposed in large white letters with a black outline over the top of the image.

# LISTS

LISTS EVERYWHERE

**En lista på böcker**

**I LOVE**

**LISTS**



# Lexikon

- Vi använder lexikon för att strukturera upp våra värden genom **nycklar**
- Ett tydligt exempel på detta är en klassisk kontaktlista:

```
# Skapar vår telefonbok  
phone_book = {}
```

```
# Lägg till tre personer, med vars ett nummer  
phone_book["Anton"] = "070-000000"  
phone_book["Fredrik"] = "070-111111"  
phone_book["Johanna"] = "070-222222"
```

```
# Skapar vår telefonbok  
phone_book = {}
```

```
# Lägg till tre personer, med vars två nummer  
phone_book["Anton"] = ["070-000000", "070-101010"]  
phone_book["Fredrik"] = ["070-111111", "070-121212"]  
phone_book["Johanna"] = ["070-222222", "070-232323"]
```

**Demo på lexikon**

# Listor är förändringsbara datatyper

Vad innebär detta?