

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?





HT2022

Anmälan till inlämningsuppgift 3

[Starta uppgift](#)[Startsida](#)[Kursöversikt](#)[Anslag](#)[Moduler](#)[Omdömen](#)[Personer](#)

Inlämningsdatum 28 nov av 14:00 Poäng 0 Lämnar in en textmatningsruta

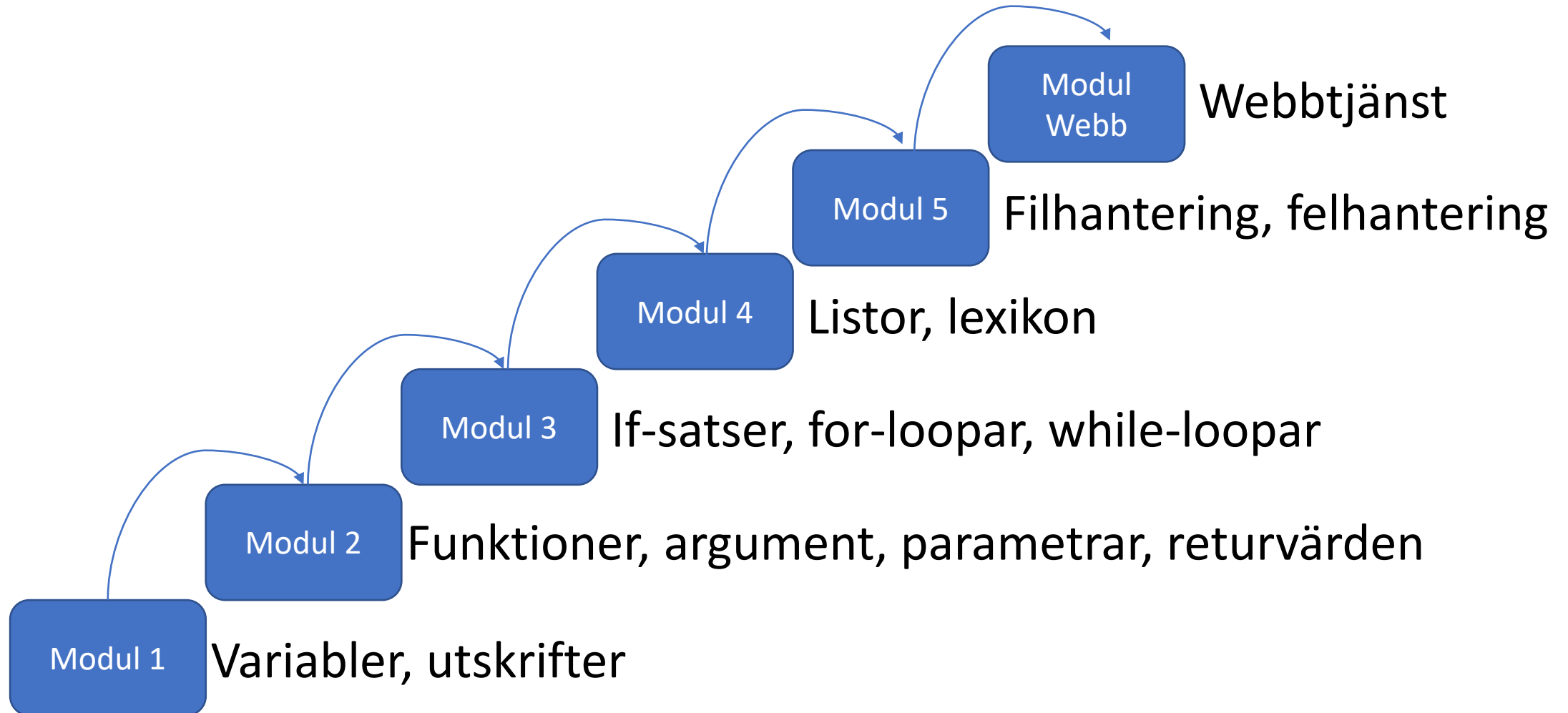
Tillgänglig till 28 nov kl 14:00

Inlämningsuppgift 3 kommer att göras i **grupper** (alltså inte individuellt). För att det inte ska bli några inaktiva studenter i någon av grupperna behöver ni anmäla er genom att lämna in denna uppgift, och kommer inom kort få reda på vem ni ska göra inlämningsuppgiften tillsammans med.

Skulle ni inte anmäla er till någon grupp nu, så kommer kompletteringstillfälle att erbjudas i slutet av kursen (januari).

[◀ Föregående](#)[Nästa ▶](#)

Kursens uppbyggnad





Kursöversikt

Planering

Kursplan →

Canvas →

1. INTRODUKTION TILL PYTHON



2. FUNKTIONER



3. ITERATION OCH SELEKTION



Översikt

Föreläsning

Föreläsning (extra)

Ö.1: Introduktion till selektion

Ö.2: Nästlade if-satser

Ö.3: Introduktion till iterationer

Ö.4: Nästlade iterationer

Ö.5: Iteration och selektion

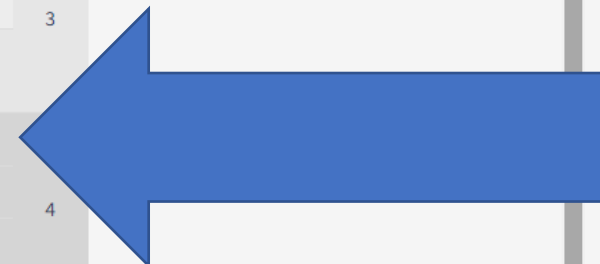
Inlämningsuppgift

4. LISTOR OCH LEXIKON



← Startsidan

Vecka	Datum	Tid	Moment	Lärare/Assistent	Plats	Modul
45	7/11	10-12	Kursintroduktion	Anton & Johan	G8:104	1
	8/11	13-15	F: Introduktion till programmering	Anton	G8:104	
	10/11	08-12	L: Introduktion till programmering	Anton & Lärarassistent	OR:D222, OR:E222	
46	14/11	10-12	F: Funktioner i Python	Anton	G8:104	2
	15/11	13-15	F: Funktioner i Python (2)	Anton	G8:104	
	17/11	08-12	L: Funktioner i Python	Anton & Lärarassistent	OR:B404, OR:E223, OR:E239	
47	21/11	10-12	F: If-satser & Loopar	Anton	G8:104	3
	22/11	13-15	F: If-satser & Loopar (2)	Anton	G8:104	
	24/11	08-12	L: If-satser & Loopar	Anton & Lärarassistent	OR:D222, OR:E222	
48	28/11	10-12	F: Listor & Lexikon	Anton	G8:104	4
	29/11	13-15	F: Listor & Lexikon (2)	Anton	G8:104	
	1/12	08-12	L: Listor & Lexikon	Anton & Lärarassistent	NI:A0306, NI:A0318	
49	5/12	10-12	F: Fil- & Felhantering	Anton	G8:104	5
	6/12	13-15	F: Fil- & Felhantering (2)	Anton	G8:104	
	8/12	08-12	L: Listor & Lexikon	Anton & Lärarassistent	NI:A0318, NI:A0407	
50	12/12	10-12	F: Python & Webben	Anton	G8:104	6
	13/12	13-15	F: Python & Webben (2)	Anton	G8:104	
	15/12	08-12	L: Python & Webben	Anton & Lärarassistent	OR:D222, OR:E223	
2	10/1	13-15	F: Objektorienterad programmering	Anton	Zoom	7
	12/1	08-12	L: Objektorienterad programmering	Anton	Zoom	



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 = "Djungelboken"
```

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

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

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

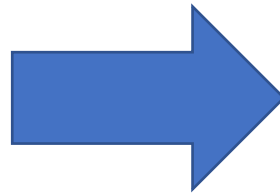
```
movie9 = "Finding Dory"
```

```
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)
```

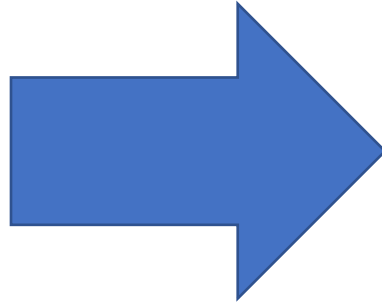


```
Filmer i min samling
*****
Zootropolis
Captain America: Civil war
Deadpool
The nice Guys
Djungelboken
Everybody Wants Some!!
10 Cloverfield Lane
Sing Street
Finding Dory
Batman vs. Superman
>>> |
```

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)
```

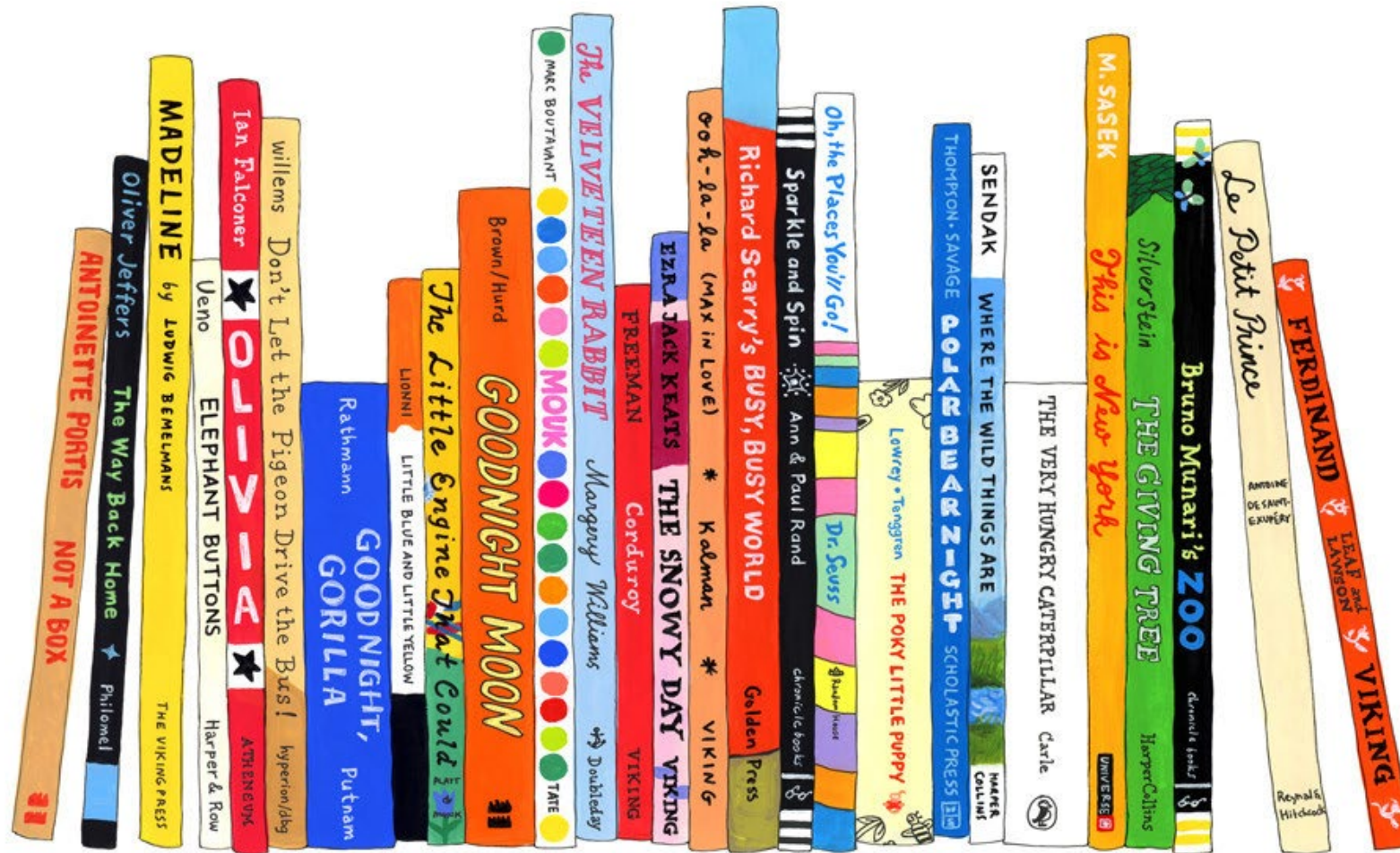
Andra sätt att spara data?

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])

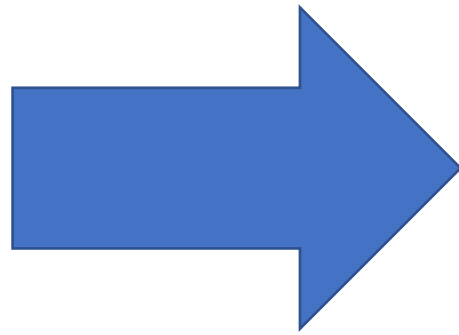
inköpslista

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"]
blue-ray = ["Star Wars", "Titanic", "Jurassic Park"]
movies = dvd + blue-ray
# 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"]
```

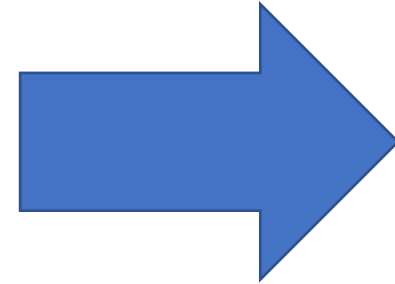
Att gå igenom listor med data

Nu blir for-loopen väldigt häändig!

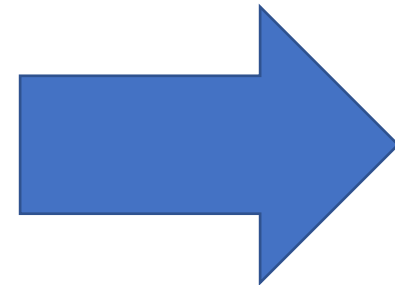
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]



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



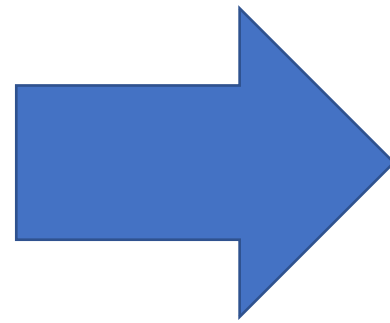
```
for number in range(10):  
    print(number)
```



0
1
2
3
4
5
6
7
8
9


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

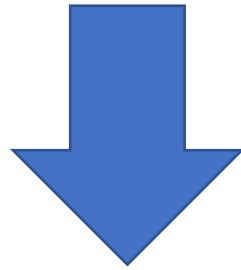
```
for number in numbers:  
    print(number)
```



1
2
3
4
5

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

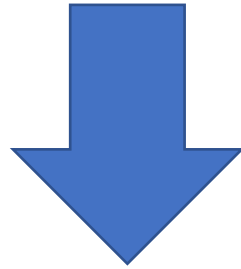
```
for band in bands:  
    print(band)
```



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

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

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



```
Måndag  
Tisdag  
Onsdag  
Torsdag  
Fredag
```

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?