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?

Anton Tibblin SKAPARE | LÄRARE

Skapad 1 dec 9.50 | Publicerades 1 dec 9.50

Gruppindelning - Inlämningsuppgift 3

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/da354b-ht25/4-listor-och-lexikon/assignment/).

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

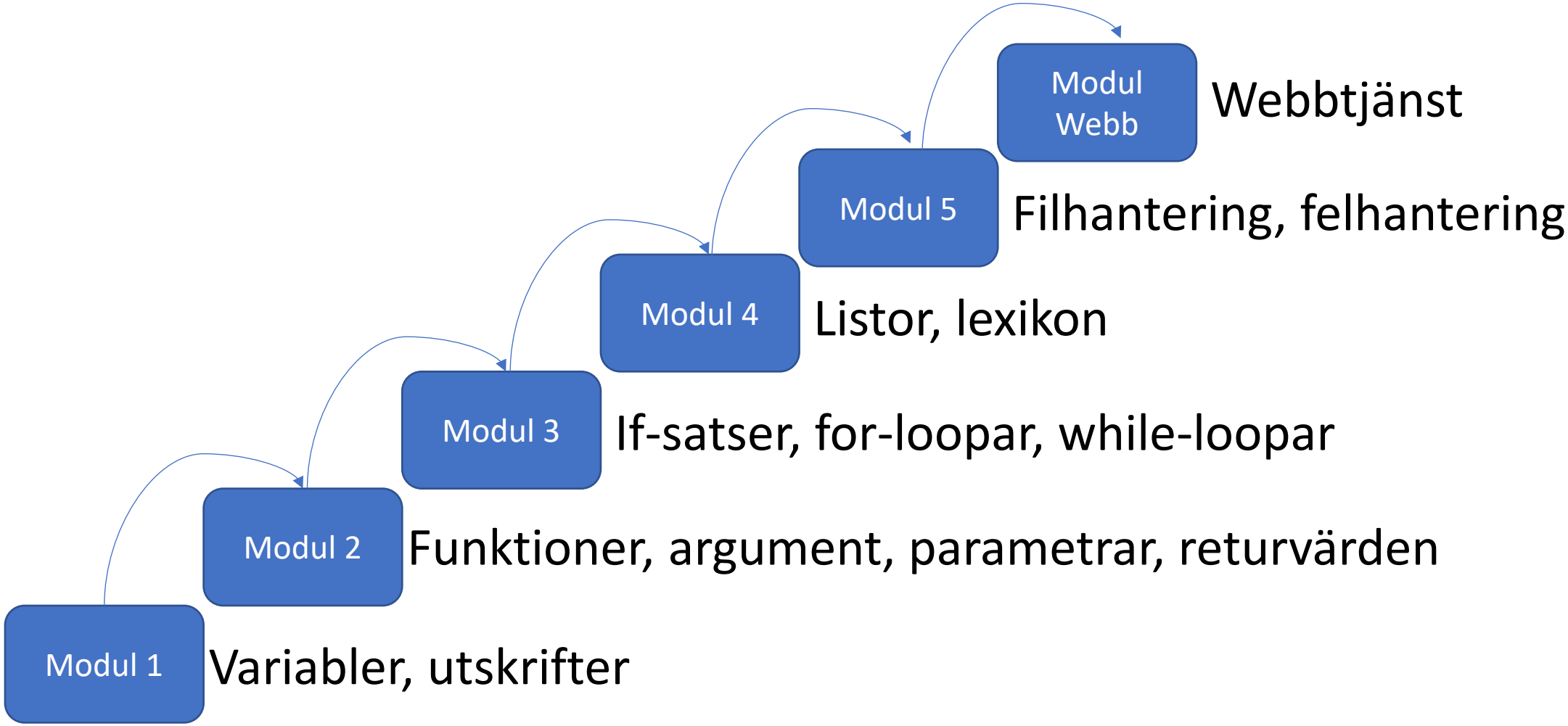
*Har du **inte** anmält dig, men ändå vill göra uppgiften, hör av dig till mig senast tisdag (2/12) 23.59, så ska jag se om det finns möjlighet för att skapa nya grupper på onsdag.*

Grupper:

Grupp 1: Viktor Valfridsson, Felicia Kotris, Maja Lidman
Grupp 2: Imre Gábor, Klara Svensson, Ronja Blücher Suneson
Grupp 3: Maiwand Saleh, Yasmine Wael Abdel Al, Alina Björkén
Grupp 4: Kai Almagharbel, Nathalie Do, Philippe Gynning Vivien
Grupp 5: Seniha Hangar, Joel Årsköld, Eren Alhussein
Grupp 6: Mauritz Krylander, Yosof Alobaidi, Tilda Pettersson
Grupp 7: Stefan-Axel Ndikumana, Viggo Salcedo Bengtsson, Fredrik Hamilton
Grupp 8: Carl Larsen, Aya Saeed Saeed, Wictor Nyman
Grupp 9: Linnéa Dahlén, Jon Westring, Philip Beckvall Karson
Grupp 10: Stephen Doan, Maja Ahlm, Esmir Islamovic
Grupp 11: Ene Bylykbashi, Emma George, Sara Boughiout
Grupp 12: Marwan Abu Matar, Auss Mohamed Jasim Al-Obaidi, Tae-Wha Lee Furugren
Grupp 13: Josefin Bäck, Najeh Al Najjar, Zakariya Daham
Grupp 14: Adrian Scholl, Jonah Bolin, Ayad Mahamoud Salim
Grupp 15: Mohamad Subhi Hajjar, Axel Wollin, Anas Abedrabbo
Grupp 16: Viktor Lindberg, April Blanco Barreta

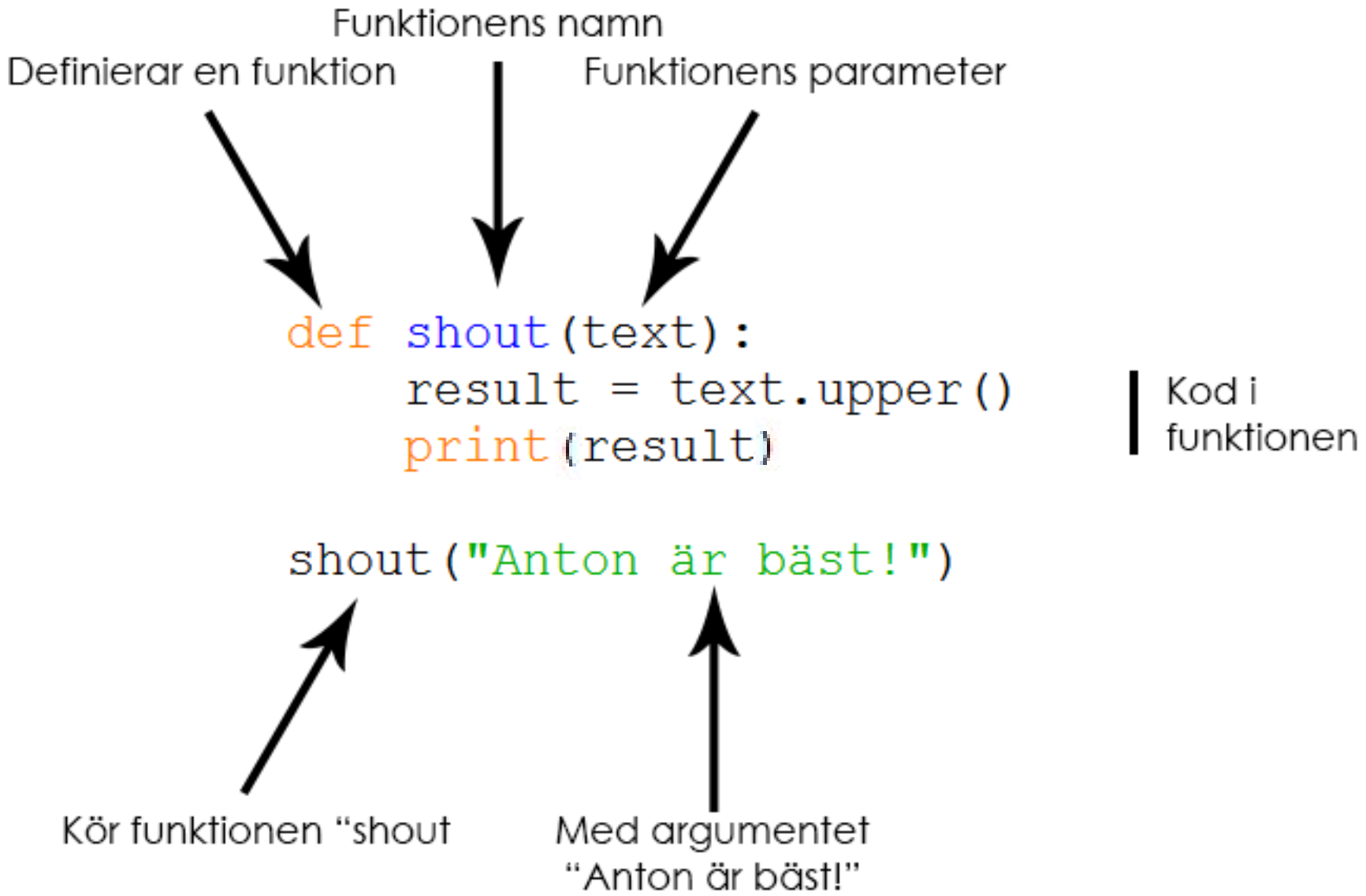
Nu för den intresserade, som tycker att problemlösning är roligt.

Kursens uppbyggnad



Vecka	Datum	Tid	Moment	Lärare/Assistent	Plats	Modul
46	10/11	10-12	Kursintroduktion	Anton & Johan	NI:A0306	
	11/11	13-15	F: Introduktion till programmering	Anton	NI:A0306	1
	13/11	08-12	L: Introduktion till programmering	Anton & assistenter	NI:A0314, NI:C0319	
47	17/11	10-12	F: Funktioner i Python	Anton	NI:A0507	
	18/11	13-15	F: Funktioner i Python (2)	Anton	NI:A0506	2
	20/11	08-12	L: Funktioner i Python	Anton & assistenter	NI:A0314, NI:A0407	
48	24/11	10-12	F: If-satser & Loopar	Anton	OR:D222	
	26/11	10-12	F: If-satser & Loopar (2)	Anton	NI:B0E07	3
	27/11	08-12	L: If-satser & Loopar	Anton & assistenter	NI:A0314, NI:A0406	
49	1/12	13-15	F: Listor & Lexikon	Anton	OR:D328	
	2/12	13-15	F: Listor & Lexikon (2)	Anton	NI:A0606	
	4/12	08-12	L: Listor & Lexikon	Anton & assistenter	NI:A0311, NI:A0318	4
	5/12	13-15	Redovisning: Inl. 2 (frivillig)	Lärarassistenter	NI:A0304, NI:C0309, NI:C0325	
50	8/12	10-12	F: Fil- & Felhantering	Anton	OR:C127	
	9/12	13-15	F: Fil- & Felhantering (2)	Anton	NI:A0306	5
	11/12	08-12	L: Fil- & Felhantering	Anton & assistenter	NI:A0307, NI:A0314	
51	15/12	10-12	F: Python & Webben	Anton	NI:A0306	
	16/12	13-15	F: Python & Webben (2)	Anton	NI:A0306	
	18/12	08-12	L: Python & Webben	Anton & assistenter	NI:A0306, NI:A0314	6
	19/12	13-15	Redovisning: Inl. 4 (frivillig)	Lärarassistenter	NI:A0304, NI:A0305, NI:B0303	
2	8/1	08-12	L: TBA	Anton & assistenter	NI:A0306, NI:A0314	
3	12/1	08-12	L: TBA	Anton & assistenter	OR:D222, OR:E222	7

Modulerna hittills


Funktionens namn
Definierar en funktion
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!”

```python
# 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?

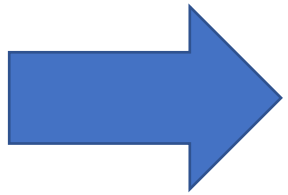
```
# Mina filmer
movie1 = "Zootropolis"
movie2 = "Captain America: Civil war"
movie3 = "Deadpool"
movie4 = "The nice Guys"
movie5 = "Djungelboken"
movie6 = "Everybody Wants Some!!"
movie7 = "10 Cloverfield Lane"
movie8 = "Sing Street"
movie9 = "Finding Dory"
movie10 = "Batman vs. Superman"
```

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?

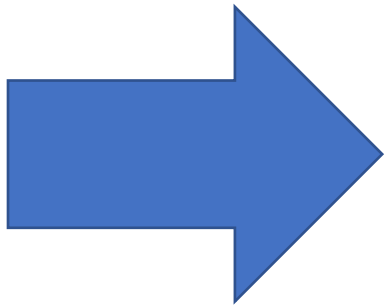
```python
# Mina filmer
movie1 = "Zootropolis"
movie2 = "Captain America: Civil war"
movie3 = "Deadpool"
movie4 = "The nice Guys"
movie5 = "Djungelboken"
movie6 = "Everybody Wants Some!!"
movie7 = "10 Cloverfield Lane"
movie8 = "Sing Street"
movie9 = "Finding Dory"
movie10 = "Batman vs. Superman"

# Skriver ut alla filmer
print("Filmer i min samlilng")
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 samlilng
****************************************
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 samlilng")
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 samlilng
****************************************
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 filmen ligger i samma variabel
5) **Hur hanterar vi 100st filmer?**

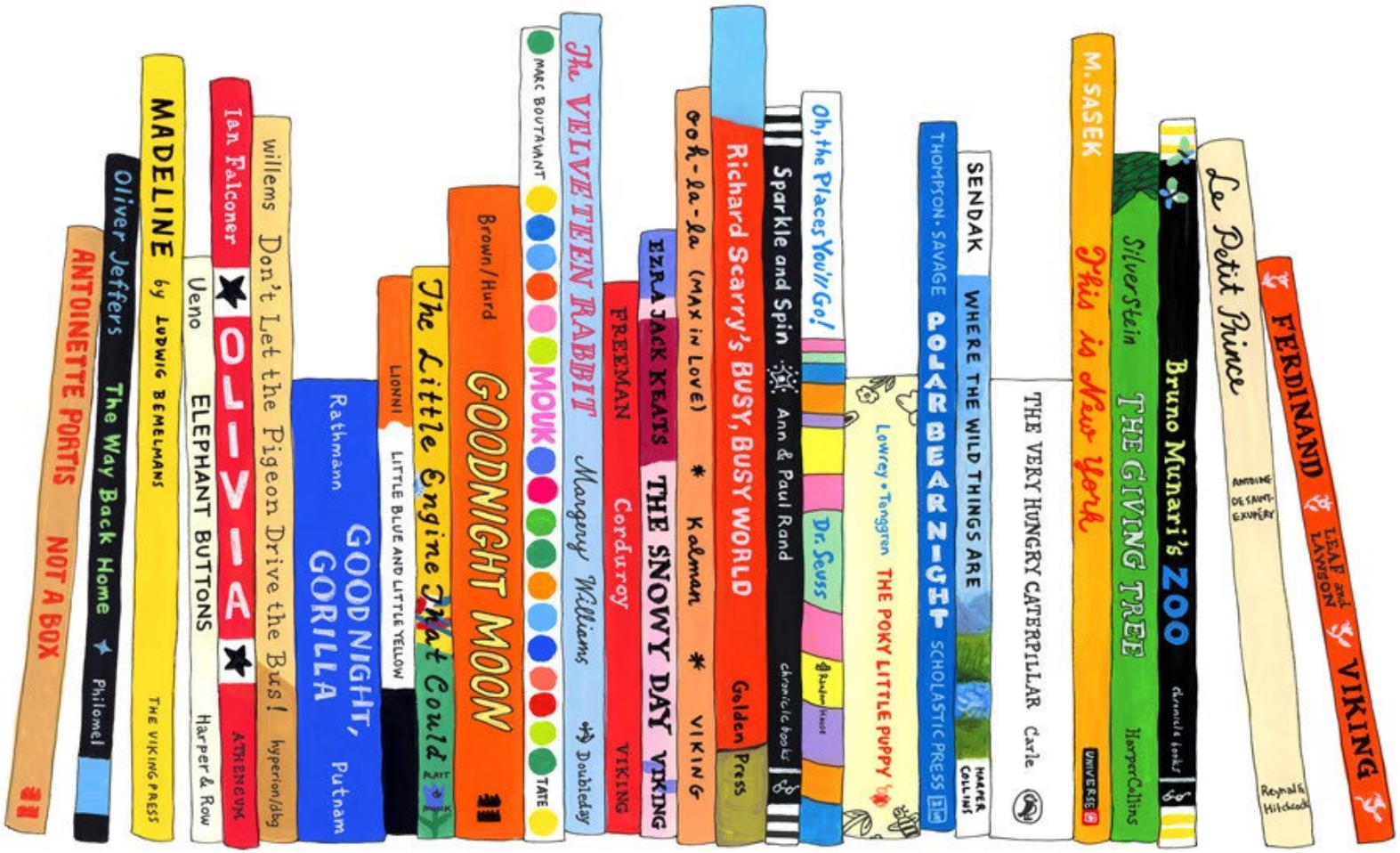
```
# Mina filmer
movie1 = "Zootropolis"
movie2 = "Captain America: Civil war"
movie3 = "Deadpool"
movie4 = "The nice Guys"
movie5 = "Djungelboken"
movie6 = "Everybody Wants Some!!"
movie7 = "10 Cloverfield Lane"
movie8 = "Sing Street"
movie9 = "Finding Dory"
movie10 = "Batman vs. Superman"

# Skriver ut alla filmer
print("Filmer i min samlilng")
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

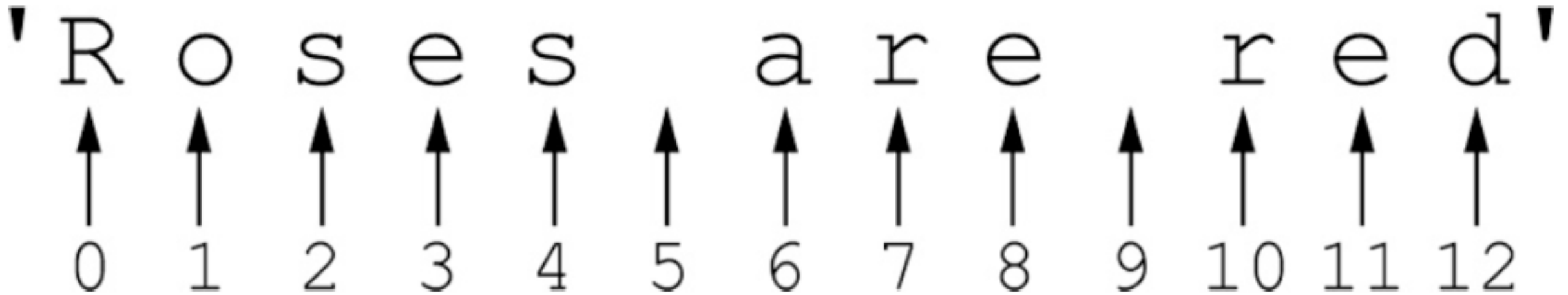


En sekvens är ett objekt som innehåller flera värden, som lagras en efter den andra. Du kan utföra operationer på en sekvens, för att undersöka och manipulera de värden som lagrats i sekvensen.

Strängar, listor, (tupler)

Olika typer av sekvenser

Strängar som sekvenser – med index



```
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”

”substings”

```
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
`isalnum()`	Returns true if the string contains only alphabetic letters or digits and is at least one character in length. Returns false otherwise.
`isalpha()`	Returns true if the string contains only alphabetic letters, and is at least one character in length. Returns false otherwise.
`isdigit()`	Returns true if the string contains only numeric digits and is at least one character in length. Returns false otherwise.
`islower()`	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.
`isspace()`	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 (`\n`), and tabs (`\t`).
`isupper()`	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
`lower()`	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.
`lstrip()`	Returns a copy of the string with all leading whitespace characters removed. Leading whitespace characters are spaces, newlines (`\n`), and tabs (`\t`) that appear at the beginning of the string.
`lstrip(char)`	The *`char`* argument is a string containing a character. Returns a copy of the string with all instances of *`char`* that appear at the beginning of the string removed.
`rstrip()`	Returns a copy of the string with all trailing whitespace characters removed. Trailing whitespace characters are spaces, newlines (`\n`), and tabs (`\t`) that appear at the end of the string.
`rstrip(char)`	The *`char`* argument is a string containing a character. The method returns a copy of the string with all instances of *`char`* that appear at the end of the string removed.
`strip()`	Returns a copy of the string with all leading and trailing whitespace characters removed.
`strip(char)`	Returns a copy of the string with all instances of *`char`* that appear at the beginning and the end of the string removed.
`upper()`	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**

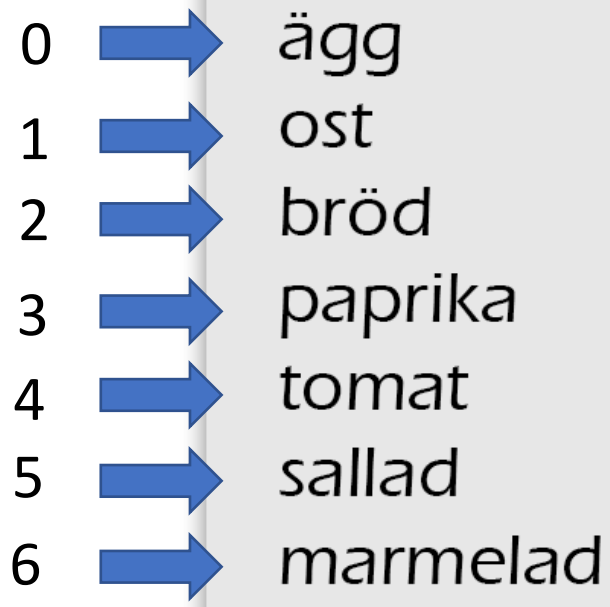

To Do List

inköpslista

ägg
ost
bröd
paprika
tomat
sallad
marmelad

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

inköpslista




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

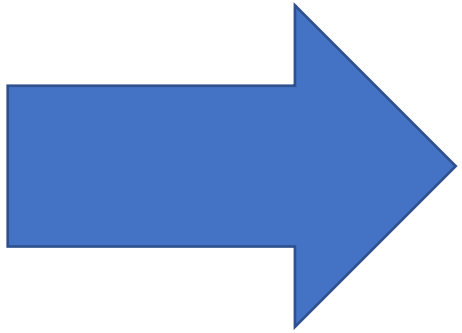

```
inköpslista = [
0   "ägg",
1   "ost",
2   "bröd",
3   "paprika",
4   "tomat",
5   "sallad",
6   "marmelad"
]
```

print(inköpslista[3])


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

```
inköpslista = ["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
```

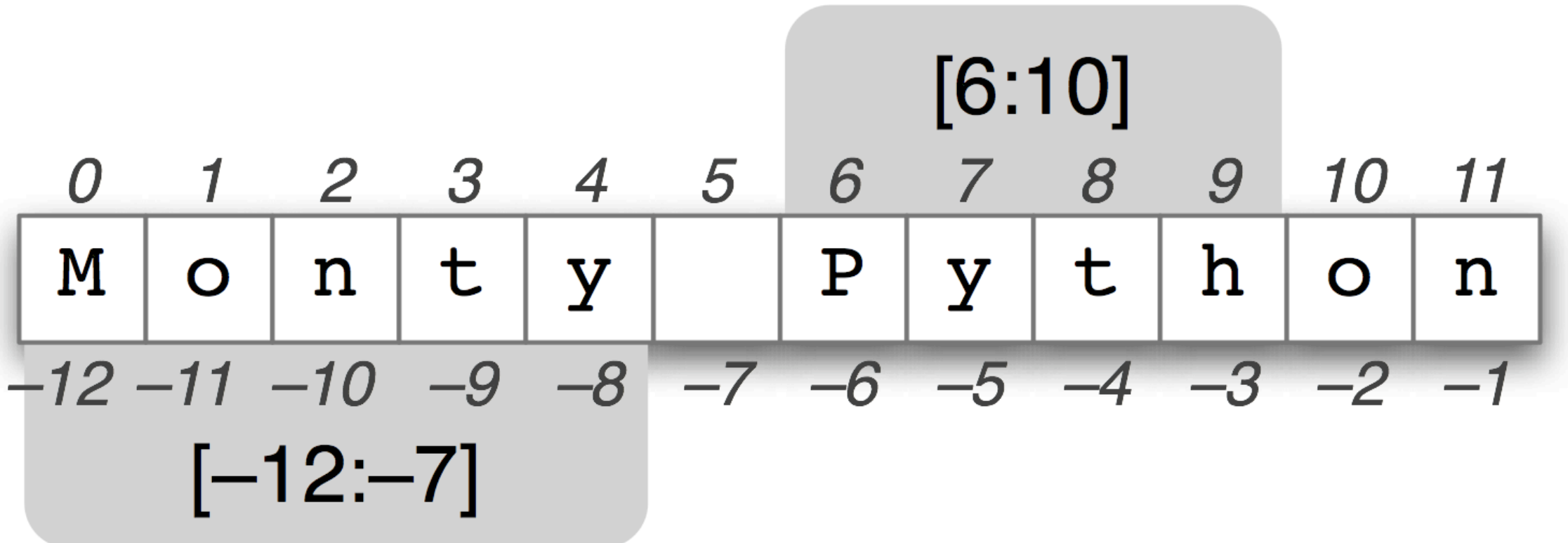
```
# Explicit angivelse av varje element:
l1 = [5, 10, 15, 20]

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

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

# Skapa med operatorn *
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", "Fredik", "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
`append(item)`	Adds *item* to the end of the list.
`index(item)`	Returns the index of the first element whose value is equal to item. A `ValueError` exception is raised if item is not found in the list.
`insert(index, item)`	Inserts *item* into the list at the specified *index*. 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.
`sort()`	Sorts the items in the list so they appear in ascending order (from the lowest value to the highest value).
`remove(item)`	Removes the first occurrence of *item* from the list. A `ValueError` exception is raised if item is not found in the list.
`reverse()`	Reverses the order of the items in the list.

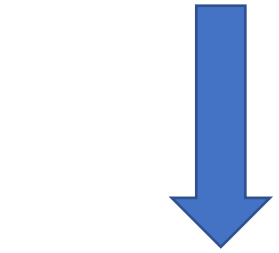
Lägga till saker i en lista

```
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"]
```

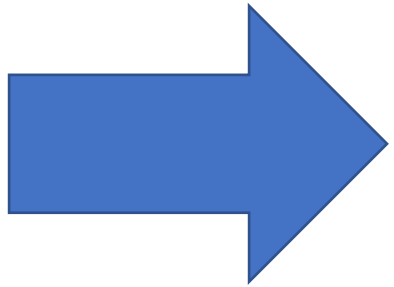
Att gå igenom listor med data

Nu blir for-loopen väldigt händig!

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

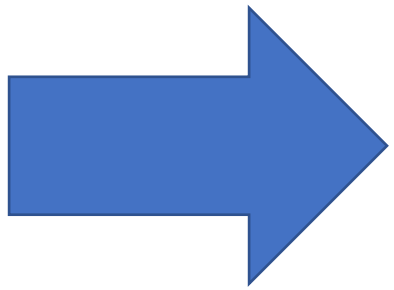


```
for i in range(10):
    print(i)
```

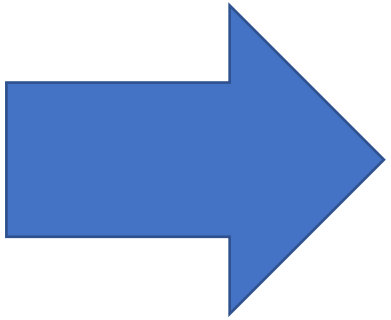


0
1
2
3
4
5
6
7
8
9

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

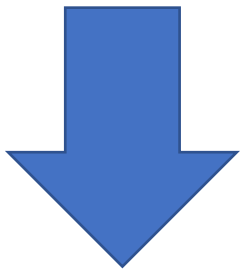


```
numbers = [1, 2, 3, 4, 5]

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



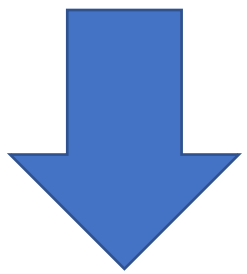
```
1
2
3
4
5
```

```python
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
memegenerator.net

En lista på böcker


I LOVE
LISTS
memegenerator.net

Lexikon

- Vi använder lexikon för att strukturer 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