

Föreläsning 3

Programmeringsteknik

- Listor
- Tupler
- strängar
- for-slingor
- importera moduler
- random

Listor

- En lista är en **föränderlig** samling objekt.
- Listor skapas med hakparenteser:

```
[12, 13, 14, 15]
```

Innehåll i listor

- En lista kan innehålla element av olika typer:

```
["hej" , 151 , 10.59]
```

- En lista kan innehålla en eller flera listor:

```
[ ["hej" , 151] , 10.59]
```

- Man kan åstadkomma godtycklig komplicerade strukturer m.h.a listor:

```
list1 = ["hej"]  
list1 += ["hejdå" , list1]
```

+,* och listor

- Listor kan konkateneras:

```
lista = [1, 2, 3] +  
[4, 5, 6]
```

Vad innehåller variabeln lista?

- Listor kan upprepas:

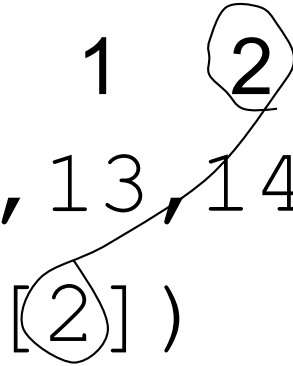
```
lista = [1, 2]  
lista * 3
```

Vad innehåller variabeln lista?

Identifiering

- Ett enskilt element på ett givet index kan identifieras:

```
          0    1    2    3  
lista = [12, 13, 14, 15]  
print(lista[2])
```



Vad skrivs ut?

Dellista

- Ett startindex och ett övre begränsningsindex ger dellistan.

```
0  1  2  3  4  
lista=[12,13,14,15,16]  
lista2= lista[1:4]
```

Vad innehåller lista2?

Dellista

- Ett startindex, ett övre begränsningsindex och ett intervall ger en annan typ av dellista.

```
lista=[6,7,1,8,5,2,4,9]
```

```
lista2= lista[1:8:3]
```

Vad innehåller lista2?

Ändra

- Identifierade element / dellistor kan ändras:

```
lista = [1, 2, 3, 4, 5, 6]  
lista[1:3] = ["a"]  
lista[4] = "b"  
print(lista)
```

Vad skrivs ut?

Borttagning

- Element/dellista kan tas bort:

```
lista = [1, 2, 3, 4, 5, 6]
```

```
del lista[1:3]
```

```
del lista[3]
```

```
print(lista)
```

Vad skrivs ut?

Metoder

- Omvänd ordning på elementen

```
lista = [2, 3, 1, 4, 5, 7, 6]
```

```
lista.reverse()
```

```
print(lista)
```

Vad skrivs ut?

- Sortera elementen

```
lista.sort()
```

```
print(lista)
```

Vad skrivs ut?

Översikt

Listor

Tupler

Strängar

for-slinga

range()

import

Sammanfattning

Metoder

- Lägg till element på slutet
`lista.append(13)`
- Lägg till lista på slutet
`lista.extend([1, 2, 3])`

Tupler

- En tupel är en **omuterbara** lista.
- En tupel skapas med vanliga parenteser:

```
t = (1, 2, 3, 4)
```

+,* och tupler

- Tupler kan konkateneras:

```
t = (1, 2, 3) + (4, 5, 6)
```

```
print(t)
```

Vad skrivs ut?

- Tupler kan upprepas:

```
t = (1, 2, 3)
```

```
print(t * 3)
```

Vad skrivs ut?

Identifiering

- Ett enskilt element på ett givet index kan identifieras:

```
tupel = (12,13,14,15)
```

```
print(tupel[2])
```

Vad skrivs ut?

Deltupel

- Ett startindex och ett övre begränsningsindex ger deltupeln.

```
tupel = (12, 13, 14, 15, 16)  
print(tupel[2:4])
```

Vad skrivs ut?

Omuterbarhet

- En tupel kan **inte** ändras.

```
t1 = (1, [2, 3])
```

```
t1[0] = 33 ger fel
```

```
t1[1] = [2, 3, 33] ger fel
```

- Följande ger **INTE** fel!

```
t1[1].append(33)
```

Varför?

Översikt

Listor

Tupler

Strängar

for-slinga

range()

import

Sammanfattning

Strängar

- En sträng är som en tupel där alla elementen är "bokstäver".

```
nose="En lång näsa"
```

0	1	2	3	4	5	6	7	8	9	10	11
E	n		l	å	n	g		n	ä	s	a

Översikt

Listor

Tupler

Strängar

for-slinga

range()

import

Sammanfattning

Skapa delsträng

Strängar kan anses som en lista med bokstäver:

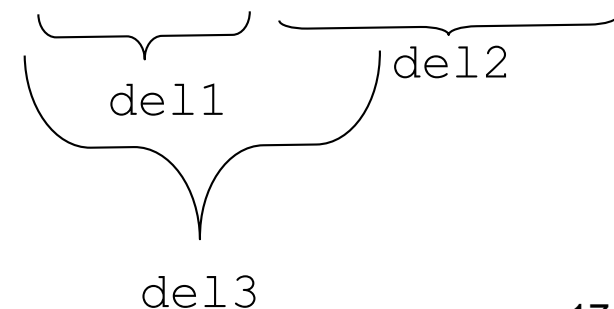
```
state="urbra"
```

```
del1=state[0:2]
```

```
del2=state[2:]
```

```
del3=state[:3]
```

0	1	2	3	4
u	r	b	r	a



Översikt

Listor

Tupler

Strängar

for-slinga

range()

import

Sammanfattning

len()

- Antalet element i listor och tupler:

```
print(len([1, 2, 5]))
```

Vad skrivs ut?

- `in` testar medlemskap i lista, tupel, strängar och dictionary:

```
if 3 in [1, 2, 3]:  
    print("3 finns i listan")  
else:  
    print("3 finns inte i listan")
```

Vad skrivs ut?

Översikt

Listor

Tupler

Strängar

for-slinga

range()

import

Sammanfattning

for-slinga

- Används för att gå igenom alla element i en lista, tupel, eller sträng.

```
lista = [12, 1, 10, 11, 6]
for x in lista:
    print(x, end=" ")
```

Skriver ut:

12 1 10 11 6

Översikt

Listor

Tupler

Strängar

`for`-slinga

`range()`

`import`

Sammanfattning

Strukturen

`for` **en variabel** `in` **en lista** :

Block som exekveras
för varje element.

`else` :

Block som exekveras om
loopen INTE avbryts med
`break`.

Översikt

Listor

Tupler

Strängar

for-slinga

`range()`

import

Sammanfattning

`range()`

- Ofta vill man göra något för varje heltal i ett intervall.
- `range()` är praktisk i detta fall:

```
for x in range(11):  
    print(x, end=" ")
```

Skriver ut:

0 1 2 3 4 5 6 7 8 9 10

Översikt

Listor

Tupler

Strängar

for-slinga

`range()`

import

Sammanfattning

Mer om `range()`

- `range` kan användas på följande sätt:

1. `list(range(3))`

`[0, 1, 2]`

2. `list(range(3, 10))`

`[3, 4, 5, 6, 7, 8, 9]`

3. `list(range(3, 15, 2))`

`[3, 5, 7, 9, 11, 13]`

Mer om range()

- Avtagande heltalslistor

```
list(range(6, 1, -1))  
[6, 5, 4, 3, 2]
```

```
list(range(10, 0, -2))  
[10, 8, 6, 4, 2]
```


Översikt

Listor

Tupler

Strängar

for-slinga

range()

import

Sammanfattning

Importering av moduler

Det finns en hel del funktioner man kan använda genom att importera dem från pythons standard bibliotek. För att importera de så använder man reserverade ordet `import`.

Ett bibliotek av tillgängliga moduler finns i följande sida:

<http://docs.python.org/3/library/index.html>

Example:

```
import random
random.random()
random.randint(0, 6)
random.choice(range(6))
```

Översikt

Listor

Tupler

Strängar

for-slinga

range()

import

Sammanfattning

randrange

- Om man vill slumpa ett heltal i ett givet intervall, så skriver man:

```
import random
```

```
t = random.randrange(3, 6)
```

```
r = random.random()
```

Ett av talen 3, 4 eller 5 slumpas och tilldelas variabeln `t` medan variabeln `r` kommer att tilldelas ett slumpat decimaltal i intervallet `[0, 1)`

Listor

Tupler

Strängar

for-slinga

range()

import

Sammanfattning

Textformattering

- Ett sätt att formatera text är `str.format()`.

```
a = 12
```

```
print("jag är {0:<5d} år".format(a))
```

```
print("jag är {0:>5d} år".format(a))
```

Skriver ut

```
jag är 12___ år
```

```
jag är ___12 år
```

Listor

Tupler

Strängar

`for`-slinga

`range()`

`import`

Sammanfattning

Sammanfattning

- Listor, tupler och strängar är strukturmässigt likt varandra
- Tupler och strängar är omuterbara
- Tupler är snabbare än listor
- Tupler ska användas när man har ett antal värde som ska inte ändras under hela programmet.
- `for`-slingor i kombination med listor, tupler och strängar är en kraftfull konstruktion.
- `range()` underlättar att skapa långa listor av heltal