Loopar/Iteration

En loop används om ett program skall utföra samma sak många gånger

Att upprepa något kallas också för att *iterera*.

Allmänt

while-loopen

for-loopen

Nästlade loopar

Loopar – att upprepa

En loop används om ett program skall utföra samma sak många gånger

Att upprepa något kallas också för att *iterera*.

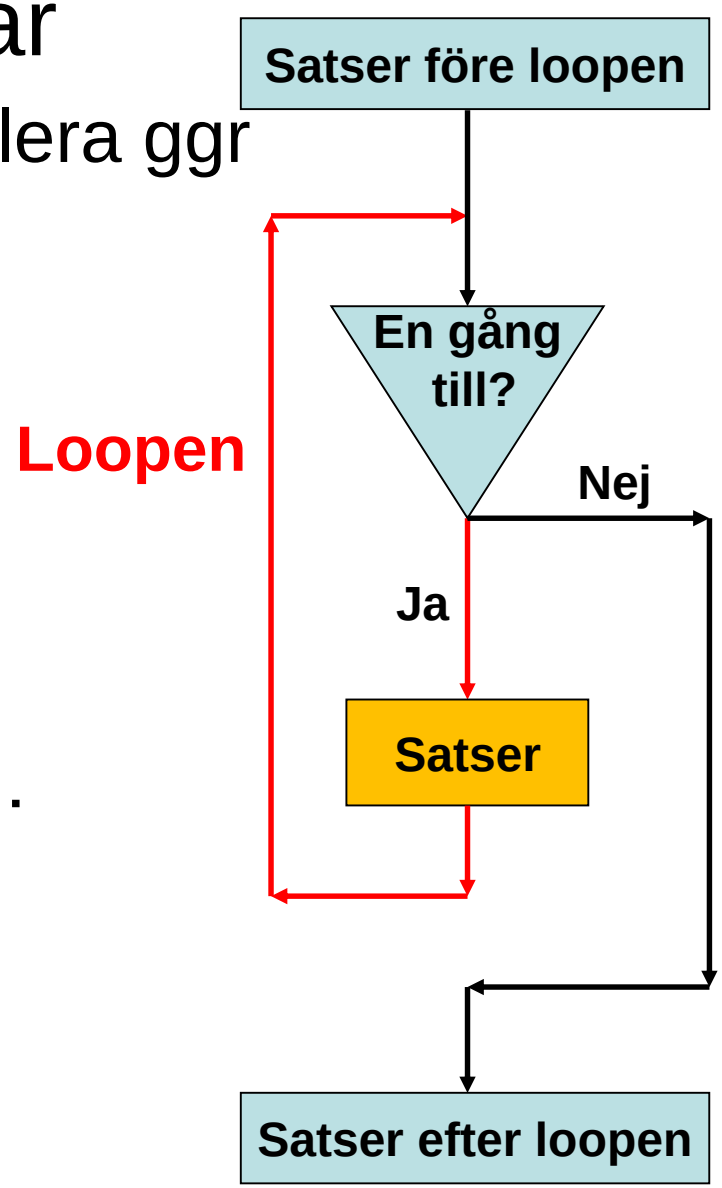
Loopar

Att repetitivt utföra satser flera ggr

Exempel

•gör något 100 ggr

•gör något så länge som …



två typer av loopar

1. **`while`**

2. **`for`**

Vi börjar med **`while`**…

while-loopen

Ett exempel:

```
i = 0
while i < 10:
    i = i + 1
    print(i)
```

Logiskt uttryck som bestämmer om satserna skall upprepas. Avslutas med kolon.

Satserna som skall upprepas *indenteras*

Utskrift:

```
1
2
3
4
5
6
7
8
9
10
```

Alt:

```
i = 0
while i < 10:
    i = i + 1
    print(i, end='')
```

Utskrift:

```
1 2 3 4 5 6 7 8 9 10
```

end='' ersätter radbytet med ett blanktecken

Övning 1: Modifiera loopen så att talen a) 1,3,5,7,9 skrivs ut
b) 10,9,8,7,6,5,4,3,2,1 skrivs ut

Simulera kast med en tärning tills en sexa, räkna antalet kast som krävdes

```
import random  # importera modulen random
tärning = random.randint(1,6)        # (1)
antal = 1                            # (2)
while tärning != 6:                  # (3)
    tärning = random.randint(1,6)    # (4)
    antal = antal + 1                # (5)
print('Antal kast = ', antal)
```

Ex. utskrift

```
Antal kast =  9
```

Vad händer om:

- Sats (1) tas bort?
- Sats (5) tas bort?
- Sats (5) är före sats (4)?
- Sats (3) är `while tärning < 6:`
- Om sats (1) ändras till: `tärning = -1`

Alternativ lösning med **`break`**-sats:

```
antal = 1
while True:  # True är reserverat ord
    tärning = random.randint(1,6)
    if (tärning==6):
        break
    antal = antal + 1
print('Antal kast = ', antal)
```

Vi har löst det med `break`-satsen som hoppar ur loopen.

Men denna lösning är inte lika tydlig som föregående. Det bör framgå av det logiska uttrycket vad som styr loopen. ”while tärning skilt 6” är tydligare än ”while True” där vi egentligen har en loop som snurrar oändligt.

Övning 2: Kasta tärning tills två sexor i rad. Hur skall man tänka?
Så länge som …?. Vilka satser skall repeteras? Satser före while?

Övningar:

3. Fråga användaren om ett tal tills denne anger ett negativt tal. Skriv ut hur många positiva tal som angivits

4. Fråga användaren efter 10 st tal. Beräkna summan, medelvärdet och största och minsta tal.

5. Fakulteten av ett tal n = 1*2*3*…*n. Detta skrivs n! Skriv de satser som frågar efter n och därefter beräknar och skriver ut värdet av n!

Tips:

- Fundera på hur ”Så länge som …” skall formuleras, dvs `while …`
- Fundera på vad som skall repeteras
- Fundera på vad som skall skrivas ut (efter loopen)
- Fundera på vad som skall stå före loopen

`break` och `continue`

- `break`-satsen bryter en loop, dvs hoppar ur loopen.
- `continue`-satsen används för att göra nästa varv i loopen, man gör ej resterande satser i den aktuella repetitionen.

```
while …
    …
    if …
        break
    …
# Efter loopen
```

```
while …
    …
    if …
        continue
    …
# Efter loopen
```

`break` och `continue` fungerar i pythons båda looptyper: `while` och `for`.

for-loopen

Betyder värden
1,2,3,4,5,6,7,8,9,10

Ett exempel:

```
for i in range(1,11):
    print(i, end=' ')
# Efter loopen
```

Satserna som skall upprepas indenteras

Utskrift:

```
1 2 3 4 5 6 7 8 9 10
```

`range` kan allmänt formuleras som: `range(first, last, step)`
Exempel:

```
range(1,10)       1,2,3,4,5,6,7,8,9
range(10,1,-1)    10,9,8,7,6,5,4,3,2
range(1,10,2)     1,3,5,7,9
```

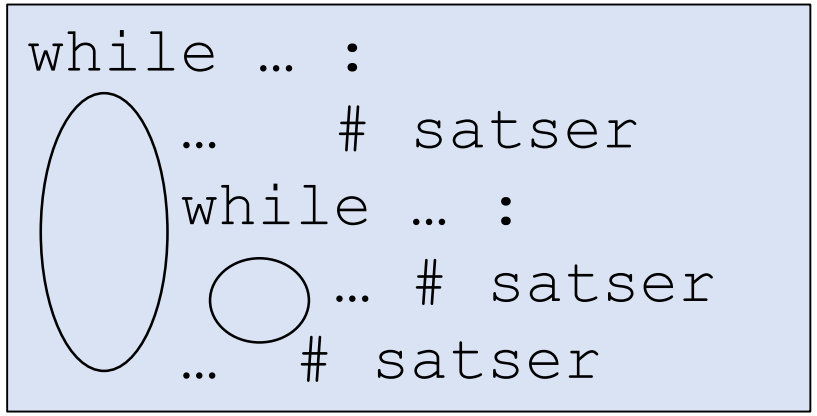
Parametrarna till `range` ersätts med variabler:

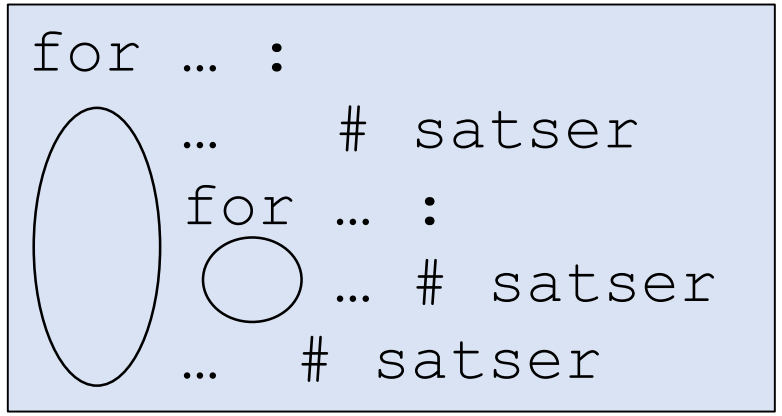
```
first = int(input('Vad skall loopen börja med '))
last = int(input('Vad skall loopen sluta med '))
step = int(input('Steglängden '))
for i in range(first, last+1, step):
    print(i, end='')
```

Ovan kommer fungera om `last > first` och `step > 0`

Nästlade loopar

Ex. Yttre loop med en inre loop





Ex. multiplikationstabell med for-satser

```
m = 5
n = 5
for i in range(1,m+1):
    for j in range(1,n+1):
        prod = i*j
        print(prod, end=' ')
    print()
```

```
1 2 3 4 5
2 4 6 8 10
3 6 9 12 15
4 8 12 16 20
5 10 15 20 25
```

Snyggare utskrift, med TAB

```
for i in range(1,m+1):
    for j in range(1,n+1):
        prod = i*j
        print('\t', prod, end=' ')
    print()
```

```
	1	2	3	4	5
	2	4	6	8	10
	3	6	9	12	15
	4	8	12	16	20
	5	10	15	20	25
```

TAB mellan varje tal plus en blank

Snyggast utskrift, med formattering med f'print:

```
for i in range(1,m+1):
    for j in range(1,n+1):
        prod = i*j
        print(f'{prod:3}', end=' ')
    print()
```

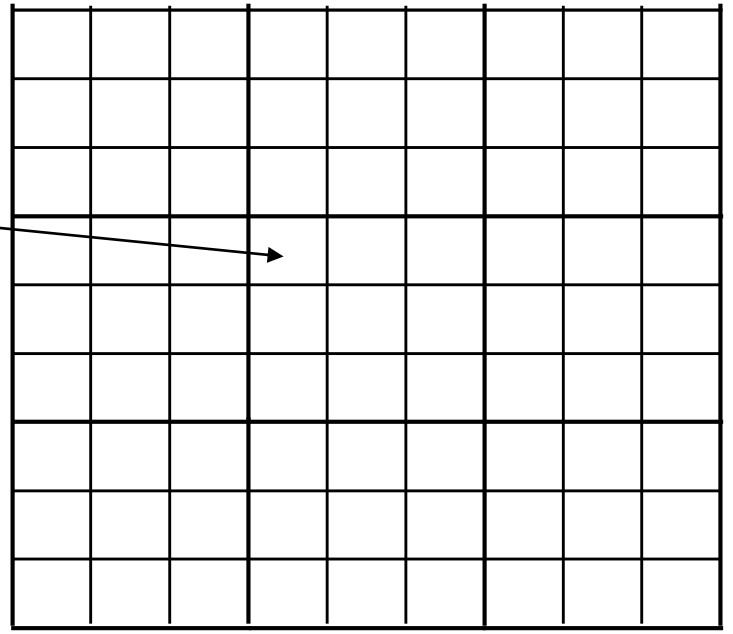
```
  1   2   3   4   5
  2   4   6   8  10
  3   6   9  12  15
  4   8  12  16  20
  5  10  15  20  25
```

Varje tal skrivs ut högerjusterat över 3 positioner plus blank mellan

Exempel där nästlade loopar förekommer:

- Digital bildbehandling
- Brädspel. Exvis schack
- Sudoku

pixel(x,y)



1	2	3	4	5	6	7	8	9