DD1331

Grundläggande programmering för F1

Några bilder till föreläsning 3

Innehåll

Innehåll

- Repetition med for

Innehåll

- Repetition med for
  - Över talföljder eller strängar
  - Nästlade repetitioner

Innehåll

- Repetition med for
  - Över talföljder eller strängar
  - Nästlade repetitioner
- Typen str för texter

Innehåll

- Repetition med for
  - Över talföljder eller strängar
  - Nästlade repetitioner
- Typen str för texter
  - Jämförelser
  - Delsträngar

Innehåll

- Repetition med for
  - Över talföljder eller strängar
  - Nästlade repetitioner
- Typen str för texter
  - Jämförelser
  - Delsträngar
- Funktioner, introduktion

Innehåll

- Repetition med for
  - Över talföljder eller strängar
  - Nästlade repetitioner
- Typen str för texter
  - Jämförelser
  - Delsträngar
- Funktioner, introduktion
  - Inbyggda
  - Parametrar
  - Anrop
  - Att definiera egna
  - Returvärden

Talföljder

Talföljder

Funktionen `range()` ger en följd heltal:

`range(10)` $\rightarrow$ `0,1,2,3,4,5,6,7,8,9`

Talföljder

Funktionen `range()` ger en följd heltal:

`range(10)` $\rightarrow$ `0,1,2,3,4,5,6,7,8,9`

Ange både start och slut:

`range(9,15)` $\rightarrow$ `9,10,11,12,13,14`

Talföljder

Funktionen `range()` ger en följd heltal:

`range(10)` $\rightarrow$ `0,1,2,3,4,5,6,7,8,9`

Ange både start och slut:

`range(9,15)` $\rightarrow$ `9,10,11,12,13,14`

Med en tredje parameter anges ett steg:

Talföljder

Funktionen `range()` ger en följd heltal:

`range(10)` $\rightarrow$ `0,1,2,3,4,5,6,7,8,9`

Ange både start och slut:

`range(9,15)` $\rightarrow$ `9,10,11,12,13,14`

Med en tredje parameter anges ett steg:

`range(14,25,3)` $\rightarrow$ `14,17,20,23`

Talföljder

Funktionen `range()` ger en följd heltal:

`range(10)` $\rightarrow$ `0,1,2,3,4,5,6,7,8,9`

Ange både start och slut:

`range(9,15)` $\rightarrow$ `9,10,11,12,13,14`

Med en tredje parameter anges ett steg:

`range(14,25,3)` $\rightarrow$ `14,17,20,23`

Steget kan vara negativt:

`range(25,14,-3)` $\rightarrow$ `25,22,19,16`

for– repetition

for– repetition

För varje element i en **sekvens**

    utför ett block med satser.

for– repetition

För varje element i en **sekvens**

    utför ett block med satser.

Exempel:

for– repetition

För varje element i en **sekvens**

utför ett block med satser.

Exempel:

- För varje x i `[1,4,7,10,13]`:
  – Beräkna och skriv $x^5 - x - 11$

for– repetition

För varje element i en **sekvens**

utför ett block med satser.

Exempel:

- För varje x i `[1,4,7,10,13]`:
  – Beräkna och skriv $x^5 - x - 11$
- För varje ord i boken:
  – Öka ord-räknare med 1

for– repetition

För varje element i en **sekvens**

utför ett block med satser.

Exempel:

- För varje x i `[1,4,7,10,13]`:
  – Beräkna och skriv $x^5 - x - 11$
- För varje ord i boken:
  – Öka ord-räknare med 1
- För varje vagn i tåget:

for– repetition

För varje element i en **sekvens**

utför ett block med satser.

Exempel:

- För varje x i `[1,4,7,10,13]`:
  – Beräkna och skriv $x^5 - x - 11$
- För varje ord i boken:
  – Öka ord-räknare med 1
- För varje vagn i tåget:
  – För varje plats i vagnen:

for– repetition

För varje element i en **sekvens**

utför ett block med satser.

Exempel:

- För varje x i `[1,4,7,10,13]`:
  – Beräkna och skriv $x^5 - x - 11$
- För varje ord i boken:
  – Öka ord-räknare med 1
- För varje vagn i tåget:
  – För varje plats i vagnen:
    – Kontrollera biljetten

for – repetition

```
for variabel in sekvens:
    block
```

for – repetition

```
for variabel in sekvens:
    block
```

variabel antar värdena i sekvens, i tur och ordning

for – repetition

```
for variabel in sekvens:
    block
```

variabel antar värdena i sekvens, i tur och ordning
För varje värde på variabel, utför satserna i blocket!

for – repetition

```
for variabel in sekvens:
    block
```

variabel antar värdena i sekvens, i tur och ordning
För varje värde på variabel, utför satserna i blocket!

Skriv ut kvadraterna på 1,2,...10 på en rad

for – repetition

```
for variabel in sekvens:
    block
```

variabel antar värdena i sekvens, i tur och ordning
För varje värde på variabel, utför satserna i blocket!

Skriv ut kvadraterna på 1,2,...10 på en rad

```
for x in range(1,11):
    print(x*x, end = "  ")    #inget radbyte
print()
```

for – repetition

```
for variabel in sekvens:
    block
```

variabel antar värdena i sekvens, i tur och ordning
För varje värde på variabel, utför satserna i blocket!

Skriv ut kvadraterna på 1,2,...10 på en rad

```
for x in range(1,11):
    print(x*x, end = "  ")    #inget radbyte
print()
```

Utskriften blir

for – repetition

```
for variabel in sekvens:
    block
```

variabel antar värdena i sekvens, i tur och ordning
För varje värde på variabel, utför satserna i blocket!

Skriv ut kvadraterna på 1,2,...10 på en rad

```
for x in range(1,11):
    print(x*x, end = "  ")    #inget radbyte
print()
```

Utskriften blir

```
1  4  9  16  25  36  49  64  81  100
```

Jämför **for** och **while**

Jämför **for** och **while**

Skriv ut kvadraterna på 1,2,...10

Jämför **for** och **while**

Skriv ut kvadraterna på 1,2,...10

```
for x in range(1,11):
    print(x*x, end = "  ")
print()
```

Jämför **for** och **while**

Skriv ut kvadraterna på 1,2,...10

```
for x in range(1,11):
    print(x*x, end = "  ")
print()
```

```
x = 1
while x < 11:
    print(x*x, end = "  ")
    x += 1
print()
```

Jämför **for** och **while**

Skriv ut kvadraterna på 1,2,...10

```
for x in range(1,11):
    print(x*x, end = "  ")
print()
```

```
x = 1
while x < 11:
    print(x*x, end = "  ")
    x += 1
print()
```

for ger kompaktare kod än **while**

for och **while** ej alltid utbytbara i Python

for och **while** ej alltid utbytbara i Python

Använd **for** för att gå igenom en **följd** eller **sekvens**.

for och **while** ej alltid utbytbara i Python

Använd **for** för att gå igenom en **följd** eller **sekvens**.

Använd **while** för andra repetitioner.

for och **while** ej alltid utbytbara i Python

Använd **for** för att gå igenom en **följd** eller **sekvens**.

Använd **while** för andra repetitioner.

Vilken multipel av 2 är närmast större än 1729 ?

for och **while** ej alltid utbytbara i Python

Använd **for** för att gå igenom en **följd** eller **sekvens**.

Använd **while** för andra repetitioner.

Vilken multipel av 2 är närmast större än 1729 ?

for eller **while** ?

for och **while** ej alltid utbytbara i Python

Använd **for** för att gå igenom en **följd** eller **sekvens**.

Använd **while** för andra repetitioner.

Vilken multipel av 2 är närmast större än 1729 ?

for eller **while** ? Använd **while** !

for och **while** ej alltid utbytbara i Python

Använd **for** för att gå igenom en **följd** eller **sekvens**.

Använd **while** för andra repetitioner.

Vilken multipel av 2 är närmast större än 1729 ?

for eller **while** ? Använd **while** !

```
x = 1729
mul = 1
while mul < x:
    mul *= 2
print("Svar:", mul)
```

Nästlade repetitioner, förberedelse

Nästlade repetitioner, förberedelse

Skriv ut sjuans multiplikationstabell

Nästlade repetitioner, förberedelse

Skriv ut sjuans multiplikationstabell

```
for m in range(11):
    print(m*7, end = "  ")
print()
```

Nästlade repetitioner, förberedelse

Skriv ut sjuans multiplikationstabell

```
for m in range(11):
    print(m*7, end = "  ")
print()
```

Lägg sju i en variabel, tal

```
tal = 7
for m in range(11):
    print(m*tal, end = "  ")
print()
```

Nästlade repetitioner, förberedelse

Skriv ut sjuans multiplikationstabell

```
for m in range(11):
    print(m*7, end = "  ")
print()
```

Lägg sju i en variabel, tal

```
tal = 7
for m in range(11):
    print(m*tal, end = "  ")
print()
```

Båda exemplen ger utskrift

Nästlade repetitioner, förberedelse

Skriv ut sjuans multiplikationstabell

```
for m in range(11):
    print(m*7, end = "  ")
print()
```

Lägg sju i en variabel, tal

```
tal = 7
for m in range(11):
    print(m*tal, end = "  ")
print()
```

Båda exemplen ger utskrift

```
0  7  14  21  28  35  42  49  56  63  70
```

```
    for m in range(11):
        print(m*tal, end = "  ")
    print()
```

Upprepa radutskriften för tal=0,1,2,...10

```
for m in range(11):
    print(m*tal, end = "  ")
print()
```

Upprepa radutskriften för tal=0,1,2,...10

```
for tal in range(11):
```

```
    for m in range(11):
        print(m*tal, end = "  ")
    print()
```

Upprepa radutskriften för tal=0,1,2,...10

```
for tal in range(11):
```

```
    for m in range(11):
        print(m*tal, end = "  ")
    print()
```

Utskrift:

```
0   0   0   0   0   0   0   0   0   0   0          tal=0
0   1   2   3   4   5   6   7   8   9   10         tal=1
0   2   4   6   8   10  12  14  16  18  20         tal=2
0   3   6   9   12  15  18  21  24  27  30         tal=3
0   4   8   12  16  20  24  28  32  36  40         tal=4
0   5   10  15  20  25  30  35  40  45  50         tal=5
0   6   12  18  24  30  36  42  48  54  60         tal=6
0   7   14  21  28  35  42  49  56  63  70         tal=7
0   8   16  24  32  40  48  56  64  72  80         tal=8
0   9   18  27  36  45  54  63  72  81  90         tal=9
0   10  20  30  40  50  60  70  80  90  100        tal=10
```

Hitta alla Pythagoras-trianglar med sidor ≤ 100

Hitta alla Pythagoras-trianglar med sidor ≤ 100

$$a^2 + b^2 = c^2$$

a, b, c är heltal

Hitta alla Pythagoras-trianglar med sidor ≤ 100

$$a^2 + b^2 = c^2$$

a, b, c är heltal

Prova alla kombinationer av värden!

Hitta alla Pythagoras-trianglar med sidor ≤ 100

$$a^2 + b^2 = c^2$$

a, b, c är heltal

Prova alla kombinationer av värden!

```
for a in range(1,101):
    for b in range(1,101):
        for c in range(1,101):
            if a*a + b*b == c*c:
                print(a,b,c)
```

Hitta alla Pythagoras-trianglar med sidor ≤ 100

$$a^2 + b^2 = c^2$$

a, b, c är heltal

Prova alla kombinationer av värden!

```
for a in range(1,101):
    for b in range(1,101):
        for c in range(1,101):
            if a*a + b*b == c*c:
                print(a,b,c)
```

Hur många gånger utförs if-satsen ovan?

Hitta alla Pythagoras-trianglar med sidor ≤ 100

$$a^2 + b^2 = c^2$$

a, b, c är heltal

Prova alla kombinationer av värden!

```
for a in range(1,101):
    for b in range(1,101):
        for c in range(1,101):
            if a*a + b*b == c*c:
                print(a,b,c)
```

Hur många gånger utförs if-satsen ovan?
Hur kan man slippa dubletter, t.ex. (3,4,5) och (4,3,5) ?

Hitta alla Pythagoras-trianglar med sidor ≤ 100

$$a^2 + b^2 = c^2$$

a, b, c är heltal

Prova alla kombinationer av värden!

```
for a in range(1,101):
    for b in range(1,101):
        for c in range(1,101):
            if a*a + b*b == c*c:
                print(a,b,c)
```

Hur många gånger utförs if-satsen ovan?
Hur kan man slippa dubletter, t.ex. (3,4,5) och (4,3,5) ?
Hur räknar man antalet funna trianglar?

Hitta alla Pythagoras-trianglar med sidor ≤ 100

$$a^2 + b^2 = c^2$$

a, b, c är heltal

Prova alla kombinationer av värden!

```
for a in range(1,101):
    for b in range(1,101):
        for c in range(1,101):
            if a*a + b*b == c*c:
                print(a,b,c)
```

Hur många gånger utförs if-satsen ovan?
Hur kan man slippa dubletter, t.ex. (3,4,5) och (4,3,5) ?
Hur räknar man antalet funna trianglar?
Svar ges på webbsidan för F3 och (om vi hinner) på föreläsningen.

Typen str

Typen str

- Följd av tecken inom ' ' eller " ", t.ex.

Typen str

- Följd av tecken inom ' ' eller " ", t.ex.

```
'abrakadabra'  "Grace Hopper"
```

Typen str

- Följd av tecken inom ' ' eller " ", t.ex.

```
'abrakadabra'  "Grace Hopper"
'The Knights Who Say "Ni!"'
```

Typen str

- Följd av tecken inom ' ' eller " ", t.ex.

```
'abrakadabra'  "Grace Hopper"
'The Knights Who Say "Ni!"'
```

- Konkateneras med +

```
"boll" + "kalle" → "bollkalle"
```

Typen str

- Följd av tecken inom ' ' eller " ", t.ex.

  ```
  'abrakadabra'  "Grace Hopper"
  'The Knights Who Say "Ni!"'
  ```

- Konkateneras med +

  `"boll" + "kalle"` $\rightarrow$ `"bollkalle"`

- Upprepas med *

  `'Spam'*5` $\rightarrow$ `'SpamSpamSpamSpamSpam'`

Typen str, forts

Typen str, forts

- Enskilda tecken och delsträngar nås med index

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:    01234567
```

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:    01234567
text[0] → "v"
```

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:    01234567
text[0] → "v"
text[0:4] → "vind"
```

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:     01234567
text[0] → "v"
text[0:4] → "vind"
text[4:] → "ruta"
```

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:     01234567
text[0] → "v"
text[0:4] → "vind"
text[4:] → "ruta"
text[-1] → "a"
```

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:     01234567
text[0] → "v"
text[0:4] → "vind"
text[4:] → "ruta"
text[-1] → "a"
text[-2:] → "ta"
```

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:     01234567
```

`text[0]` → `"v"`

`text[0:4]` → `"vind"`

`text[4:]` → `"ruta"`

`text[-1]` → `"a"`

`text[-2:]` → `"ta"`

`text[6]` = "v" → **NEJ, str KAN EJ ÄNDRAS!**

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:     01234567
text[0] → "v"
text[0:4] → "vind"
text[4:] → "ruta"
text[-1] → "a"
text[-2:] → "ta"
text[6] = "v" → NEJ, str KAN EJ ÄNDRAS!
text[0:6] + "VA" → "vindruVA"
```

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:     01234567
text[0] → "v"
text[0:4] → "vind"
text[4:] → "ruta"
text[-1] → "a"
text[-2:] → "ta"
text[6] = "v" → NEJ, str KAN EJ ÄNDRAS!
text[0:6] + "VA" → "vindruVA"
```

Utelämnat index efter : betyder "sista" +1

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:     01234567
text[0] → "v"
text[0:4] → "vind"
text[4:] → "ruta"
text[-1] → "a"
text[-2:] → "ta"
text[6] = "v" → NEJ, str KAN EJ ÄNDRAS!
text[0:6] + "VA" → "vindruVA"
```

Utelämnat index efter : betyder "sista" +1

Utelämat index före : betyder "första" (index 0)

Typen str, forts

- Enskilda tecken och delsträngar nås med index

```
text = "vindruta"
index:     01234567
text[0] → "v"
text[0:4] → "vind"
text[4:] → "ruta"
text[-1] → "a"
text[-2:] → "ta"
text[6] = "v" → NEJ, str KAN EJ ÄNDRAS!
text[0:6] + "VA" → "vindruVA"
```

Utelämnat index efter : betyder "sista"+1

Utelämat index före : betyder "första" (index 0)

Alla uttryck av typen `text[x:y]` ger en kopia

Typen str, forts

Typen str, forts

- `len(s)` ger antalet tecken

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` $\rightarrow$ 11

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` $\rightarrow$ 11

  `len("")` $\rightarrow$ 0

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` $\rightarrow$ 11

  `len("")` $\rightarrow$ 0

- Jämförelse

  `"Abbe" < "Alva"` $\rightarrow$ `True`

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` $\rightarrow$ 11

  `len("")` $\rightarrow$ 0

- Jämförelse

  `"Abbe" < "Alva"` $\rightarrow$ `True`

  `"årstid" < "ädelsten"` $\rightarrow$ `False`

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` $\rightarrow$ `11`

  `len("")` $\rightarrow$ `0`

- Jämförelse

  `"Abbe" < "Alva"` $\rightarrow$ `True`

  `"årstid" < "ädelsten"` $\rightarrow$ `False`

  `"ärlig" < "öra"` $\rightarrow$ `True`

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` $\rightarrow$ 11

  `len("")` $\rightarrow$ 0

- Jämförelse

  `"Abbe" < "Alva"` $\rightarrow$ `True`

  `"årstid" < "ädelsten"` $\rightarrow$ `False`

  `"ärlig" < "öra"` $\rightarrow$ `True`

  Svenska bokstävernas ordning är ä å ö

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` $\rightarrow$ 11

  `len("")` $\rightarrow$ 0

- Jämförelse

  `"Abbe" < "Alva"` $\rightarrow$ `True`

  `"årstid" < "ädelsten"` $\rightarrow$ `False`

  `"ärlig" < "öra"` $\rightarrow$ `True`

  Svenska bokstävernas ordning är ä å ö

- Innehåller

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` $\rightarrow$ 11

  `len("")` $\rightarrow$ 0

- Jämförelse

  `"Abbe" < "Alva"` $\rightarrow$ `True`

  `"årstid" < "ädelsten"` $\rightarrow$ `False`

  `"ärlig" < "öra"` $\rightarrow$ `True`

  Svenska bokstävernas ordning är ä å ö

- Innehåller

  `"elst" in "ädelsten"` $\rightarrow$ `True`

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` $\rightarrow$ 11

  `len("")` $\rightarrow$ 0

- Jämförelse

  `"Abbe" < "Alva"` $\rightarrow$ `True`

  `"årstid" < "ädelsten"` $\rightarrow$ `False`

  `"ärlig" < "öra"` $\rightarrow$ `True`

  Svenska bokstävernas ordning är ä å ö

- Innehåller

  `"elst" in "ädelsten"` $\rightarrow$ `True`

  `"vinter" in "årstid"` $\rightarrow$ `False`

Typen str, forts

- `len(s)` ger antalet tecken

  `len('abrakadabra')` → 11

  `len("")` → 0

- Jämförelse

  `"Abbe" < "Alva"` → `True`

  `"årstid" < "ädelsten"` → `False`

  `"ärlig" < "öra"` → `True`

  Svenska bokstävernas ordning är ä å ö

- Innehåller

  `"elst" in "ädelsten"` → `True`

  `"vinter" in "årstid"` → `False`

  `"i" in "årstid"` → `True`

for-repetition över textsträng

for-repetition över textsträng

Vad blir ditt namn baklänges?
(generell metod)

for-repetition över textsträng

Vad blir ditt namn baklänges?
(generell metod)

```
namn = "TUNSTROM"
bakofram = ""
for c in namn:
    bakofram = c + bakofram
print(bakofram)
```

for-repetition över textsträng

Vad blir ditt namn baklänges?
(generell metod)

```
namn = "TUNSTROM"
bakofram = ""
for c in namn:
    bakofram = c + bakofram
print(bakofram)
```

I Python kan man skriva `namn[::-1]`
för att få texten baklänges

Funktion (metod, underprogram, procedur)

Funktion (metod, underprogram, procedur)

- Namn på programavsnitt

Funktion (metod, underprogram, procedur)

- Namn på programavsnitt
- Används vid **strukturering**
  för att dela upp ett stort problem i delar

Funktion (metod, underprogram, procedur)

- Namn på programavsnitt
- Används vid **strukturering**
  för att dela upp ett stort problem i delar
- Andvänds för att **undvika koduppprepning**

Funktion (metod, underprogram, procedur)

- Namn på programavsnitt
- Används vid **strukturering**
  för att dela upp ett stort problem i delar
- Andvänds för att **undvika koduppprepning**
- Kan ha **parametrar**

Funktion (metod, underprogram, procedur)

- Namn på programavsnitt
- Används vid **strukturering**
  för att dela upp ett stort problem i delar
- Andvänds för att **undvika koduppprepning**
- Kan ha **parametrar** (men måste inte)

Funktion (metod, underprogram, procedur)

- Namn på programavsnitt
- Används vid **strukturering**
  för att dela upp ett stort problem i delar
- Andvänds för att **undvika koduppprepning**
- Kan ha **parametrar** (men måste inte)
- Kan ha **returvärde**

Funktion (metod, underprogram, procedur)

- Namn på programavsnitt
- Används vid **strukturering**
  för att dela upp ett stort problem i delar
- Andvänds för att **undvika koduppprepning**
- Kan ha **parametrar** (men måste inte)
- Kan ha **returvärde** (men måste inte)

Funktioner

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")`

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")` `namn = input("Vad heter du?")`

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")` `namn = input("Vad heter du?")` `y = math.sin(0.63*0.9)` `z = round(y,2)`

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")` `namn = input("Vad heter du?")` `y = math.sin(0.63*0.9)` `z = round(y,2)`
- Anrop "ropa på" funktionen, dess satser utförs

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")` `namn = input("Vad heter du?")` `y = math.sin(0.63*0.9)` `z = round(y,2)`
- Anrop "ropa på" funktionen, dess satser utförs
- Returvärden

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")` `namn = input("Vad heter du?")` `y = math.sin(0.63*0.9)` `z = round(y,2)`
- Anrop "ropa på" funktionen, dess satser utförs
- Returvärden

  `round()` JA

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")` `namn = input("Vad heter du?")` `y = math.sin(0.63*0.9)` `z = round(y,2)`
- Anrop "ropa på" funktionen, dess satser utförs
- Returvärden
  - `round()` JA
  - `input()` JA

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")` `namn = input("Vad heter du?")` `y = math.sin(0.63*0.9)` `z = round(y,2)`
- Anrop "ropa på" funktionen, dess satser utförs
- Returvärden
  - `round()` JA
  - `input()` JA
  - `print()` NEJ

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")` `namn = input("Vad heter du?")` `y = math.sin(0.63*0.9)` `z = round(y,2)`
- Anrop "ropa på" funktionen, dess satser utförs
- Returvärden
  - `round()` JA
  - `input()` JA
  - `print()` NEJ
  - `math.sin()` JA

Funktioner

- Inbyggda t.ex `input()` `print()` `round()` `math.cos()` `time.sleep()`
- Parametrar `print("Get real")` `namn = input("Vad heter du?")` `y = math.sin(0.63*0.9)` `z = round(y,2)`
- Anrop "ropa på" funktionen, dess satser utförs
- Returvärden
  - `round()` JA
  - `input()` JA
  - `print()` NEJ
  - `math.sin()` JA
  - `time.sleep()` NEJ

Definiera egna funktioner

Definiera egna funktioner

- Definiera funktioner överst i programmet

Definiera egna funktioner

- Definiera funktioner överst i programmet
- Skriv

```
def funkNamn(parametrar):
```

Definiera egna funktioner

- Definiera funktioner överst i programmet
- Skriv

```
def funkNamn(parametrar):
```

- Därefter, indenterat, kommer funktionens satser

```
def funkNamn(parametrar):
    sats
    sats
     :
```

Definiera egna funktioner

- Definiera funktioner överst i programmet
- Skriv

```
def funkNamn(parametrar):
```

- Därefter, indenterat, kommer funktionens satser

```
def funkNamn(parametrar):
    sats
    sats
     :
```

- Parametrar kan utelämnas men () krävs

```
def funkNamn():
```

Första funktionen

Funktion som skriver ut tre rader Python-gillande:

Första funktionen

Funktion som skriver ut tre rader Python-gillande:

```
def likePython():
```

Första funktionen

Funktion som skriver ut tre rader Python-gillande:

```
def likePython():

    print("Python is fun")
    print("Python is fun")
    print("Python is fun")
```

Första funktionen

Funktion som skriver ut tre rader Python-gillande:

```
def likePython():

    print("Python is fun")
    print("Python is fun")
    print("Python is fun")
```

Anrop

```
likePython() #Nu skrivs tre rader
```

Andra funktionen

Välj hur många rader som ska skrivas

Andra funktionen

Välj hur många rader som ska skrivas

```
def likePython(n):
```

Andra funktionen

Välj hur många rader som ska skrivas

```
def likePython(n):

    for i in range(n):
        print("Python is fun")
```

Andra funktionen

Välj hur många rader som ska skrivas

```
def likePython(n):

    for i in range(n):
        print("Python is fun")
```

Anrop

Andra funktionen

Välj hur många rader som ska skrivas

```
def likePython(n):

    for i in range(n):
        print("Python is fun")
```

Anrop

```
likePython(9)  #Nio rader skrivs
likePython(0)  #Ingenting skrivs
likePython(-1) #Ingenting skrivs
```

Så här ser hela programmet ut

Så här ser hela programmet ut

```
def likePython(n):
    for i in range(n):
        print("Python is fun")


likePython(9)  #Nio rader skrivs
likePython(0)  #Ingenting skrivs
likePython(-1) #Ingenting skrivs
```

Python-gillande med variation

Python-gillande med variation

Låt slumpen välja ett av orden: awesome, brilliant, great, cool

Python-gillande med variation

Låt slumpen välja ett av orden: awesome, brilliant, great, cool

0 1 2 3

Python-gillande med variation

Låt slumpen välja ett av orden: awesome, brilliant, great, cool

0 1 2 3

```
from random import randrange

def likePythonRandom():
    rand = randrange(4)    # slumpa 0,1,2,3
```

Python-gillande med variation

Låt slumpen välja ett av orden: awesome, brilliant, great, cool

0 1 2 3

```
from random import randrange

def likePythonRandom():
    rand = randrange(4)    # slumpa 0,1,2,3

    if rand == 0:
        opinion = "awesome"
    elif rand == 1:
        opinion = "brilliant"
    elif rand == 2:
        opinion = "great"
    else: opinion = "cool"        # rand == 3
```

Python-gillande med variation

Låt slumpen välja ett av orden: awesome, brilliant, great, cool

0 1 2 3

```
from random import randrange

def likePythonRandom():
    rand = randrange(4)    # slumpa 0,1,2,3

    if rand == 0:
        opinion = "awesome"
    elif rand == 1:
        opinion = "brilliant"
    elif rand == 2:
        opinion = "great"
    else: opinion = "cool"        # rand == 3

    print("Python is " + opinion)
```

Python-gillande med variation

Låt slumpen välja ett av orden: awesome, brilliant, great, cool

0 1 2 3

```
from random import randrange

def likePythonRandom():
    rand = randrange(4)    # slumpa 0,1,2,3

    if rand == 0:
        opinion = "awesome"
    elif rand == 1:
        opinion = "brilliant"
    elif rand == 2:
        opinion = "great"
    else: opinion = "cool"        # rand == 3

    print("Python is " + opinion)
```

Anrop av `likePythonRandom()` ger EN utskrift

Python-gillande med variation, forts

Funktionen `manyPythonLikes(n)` anropar funktionen `likePythonRandom()` **n** gånger så att **n** st slumpmässiga utskrifter görs.

Python-gillande med variation, forts

Funktionen `manyPythonLikes(n)` anropar funktionen `likePythonRandom()` **n** gånger så att **n** st slumpmässiga utskrifter görs.

```
from random import randrange

def likePythonRandom():
    rand = randrange(4)   # slumpa 0,1,2,3
        :                 # kod som tidigare

def manyPythonLikes(n):
    for i in range(n):
        likePythonRandom()

# Anropa funktionen
manyPythonLikes(7)
```

Funktion med returvärde

Funktion med returvärde

```
def funkNamn(parametrar):
        :
        :
    return varde
```

Funktion med returvärde

```
def funkNamn(parametrar):
        :
        :
    return varde
```

Anropsexempel:

```
v = funkNamn(aktuellaParametrar)

print(funkNamn(aktuellaParametrar))
```

Funktion med returvärde

```
def funkNamn(parametrar):
        :
        :
    return varde
```

Anropsexempel:

```
v = funkNamn(aktuellaParametrar)

print(funkNamn(aktuellaParametrar))
```

Tag hand om funktionsvärdet!

Funktion med returvärde

Exempel: Fahrenheit $\rightarrow$ Celsius

$c = (f - 32) * 5/9$

Funktion med returvärde

Exempel: Fahrenheit $\rightarrow$ Celsius

$c = (f - 32) * 5/9$

Gamla F-till-C-programmet:

```
# Fran Fahrenheit till Celsius

print()
f = float(input("Fahrenheit = "))
c = (f-32)*5/9
c = round(c,1)
print("Celsius = ",c)
print()
```

Funktion för Fahrenheit → Celsius

```
# Funktion fran Fahrenheit till Celsius

def celsius(f):
    c = (f-32)*5/9
    c = round(c,1)
    return c
```

Funktion för Fahrenheit → Celsius

```
# Funktion fran Fahrenheit till Celsius

def celsius(f):
    c = (f-32)*5/9
    c = round(c,1)
    return c
```

Kompaktare versioner:

```
def celsius(f):
    return round((f-32)*5/9,1)
```

Funktion för Fahrenheit → Celsius

```
# Funktion fran Fahrenheit till Celsius

def celsius(f):
    c = (f-32)*5/9
    c = round(c,1)
    return c
```

Kompaktare versioner:

```
def celsius(f):
    return round((f-32)*5/9,1)
```

Låt antalet decimaler vara parameter också:

```
def celsius(f,dec):
    return round((f-32)*5/9,dec)
```

Fahrenheit $\rightarrow$ Celsius - funktion, forts

Testa funktionen med några anrop

Fahrenheit $\rightarrow$ Celsius - funktion, forts

Testa funktionen med några anrop

```
def celsius(f,dec):
    return round((f-32)*5/9, dec)

# Funktionen testas
while True:
    fa = float(input("GraderFahrenheit:"))
    print("GraderCelsius:", celsius(fa,2))
    print()
```

Fahrenheit → Celsius - funktion, forts

Testa funktionen med några anrop

```
def celsius(f,dec):
    return round((f-32)*5/9, dec)

# Funktionen testas
while True:
    fa = float(input("GraderFahrenheit:"))
    print("GraderCelsius:", celsius(fa,2))
    print()
```

```
# Tabell over celsius och fahrenheit
print("fa   celsius")
for fa in range(10,101,10):
    print(fa, "  ", celsius(fa,1))
```

Fahrenheit $\rightarrow$ Celsius - funktion, forts

Tabellutskriften:

```
fa   celsius
10    -12.2
20    -6.7
30    -1.1
40    4.4
50    10.0
60    15.6
70    21.1
80    26.7
90    32.2
100    37.8
```