Introduktion till Python

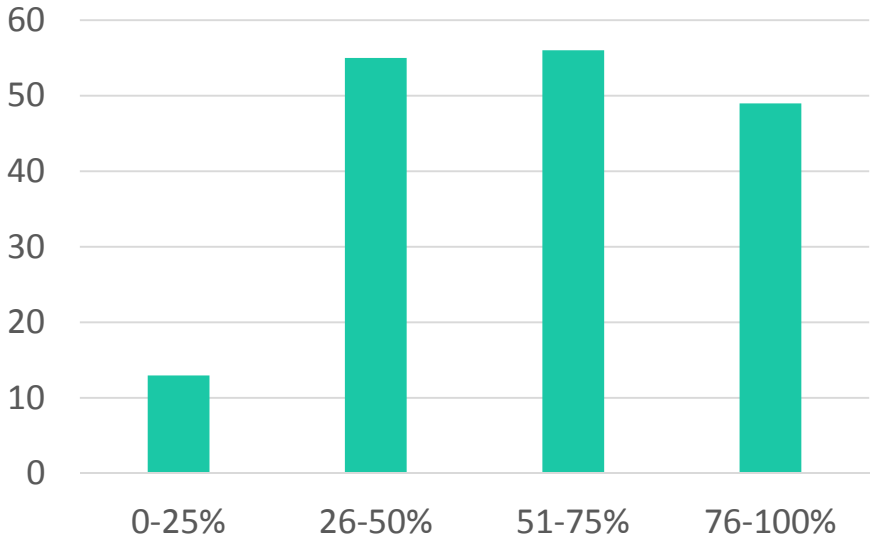
TDDE23 Funktionell och imperativ programmering, del 1

Föreläsning 3

Peter Dalenius
Institutionen för datavetenskap

LINKÖPINGS UNIVERSITET

Status för datorintroduktionen



- Gör färdigt resten hemma eller senare
- Deadline för webbmaterial och slutuppgift är 24 oktober

Översikt

- Varför vi har valt Python
- Hur vi kommer arbeta i kursen
- Vårt första Python-program

Varför vi har valt Python

Programmering i ett nötskal

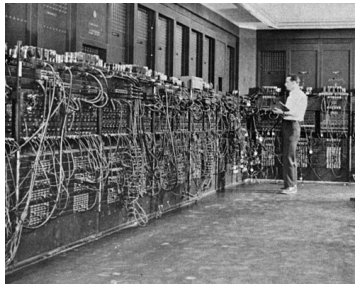
program *en uppsättning regler eller instruktioner med uppgift att styra en dators beräkningar (Nationalencyklopedin)*

komponenterna
programspråket och
dess konstruktioner

processen
principer för hur man
utformar program

resultatet
hur program körs och
hur de påverkar folk

Synen på program har utvecklats



Tidiga datorer,
ofta militära
40-, 50-talet

Persondatorer
för hem och kontor
80-, 90-talet

Mobila
enheter
10-talet

1940 1950 1960 1970 1980 1990 2000 2010

Stordatorer för
industrin
60-, 70-talet

Internet för
vanligt folk
90-,00-talet

Några roliga saker vi *inte* hinner med

- Hur man designar grafiska gränssnitt
- Hur datorer kommunicerar med varandra över nätverk
- Hur man lagrar information i databaser
- Hur man startar trådar och processer för att få saker att hända samtidigt
- Hur man hanterar större programmeringsprojekt
- Hur man gör appar

Vi kommer att fokusera på grunderna i programmering och
hur man etablerar goda vanor för att bli en bra programmerare.
Allt det andra – och mycket mer – kommer längre fram.

Svensk industri är beroende av programvara

Pythons historik

- Skapades i slutet av 1980-talet av holländaren Guido van Rossum, som fortfarande spelar en stor roll för språkets utveckling.
- Det femte mest populära programspråket i världen. [1]
- Åtta av tio toppuniversitet i USA använder Python som förstaspråk. [2]
- Version 3, som vi kommer använda, kom i december 2008. Det var en ganska stor förändring som inte är bakåtkompatibel.
- Fri programvara.

[1] TIOBE Programming Community Index, augusti 2017 http://www.tiobe.com/tiobe-index

[2] IT World, 8 juli 2014 http://www.itworld.com/software/426242/python-bumps-java-top-learning-language

Fördelar med Python

- Lätt att komma igång med och enklare syntax än t.ex. Java och C++
- Har en stabil användarbas både på universitet och i industrin
- Funkar bra både för systemprogram och för tillämpningar
- Neutralt i förhållande till olika programmeringsparadigm
- Kan användas för att illustrera koncept inom programmering

Hur vi kommer arbeta i kursen

Hur ska vi jobba i kursen?

Aktivitet	Tid	Innehåll
Föreläsningar	1 x 2 h	*Introduktion, översikt, sammanfattning*
Seminarer	6 x 2 h	*Genomgångar, förberedelser, exempel, torrövningar*
Laborationer	14 x 2 h	*Arbete med övningar och uppgifter, diskussioner, redovisningar*
Eget arbete	~ 100 h	*Läsa, öva, experiementera, arbeta med övningar och uppgifter*

Seminarier

- Ett seminarium per vecka, från och med nästa vecka
- Samma innehåll i alla fem grupper, men tre olika inriktningar
  - En sak i taget - *För dig som vill ta det steg för steg och känna att du förstår en sak innan du går vidare till nästa.*
  - Experimentering - *För dig som vill lära dig genom att testa dig fram och gärna prövar olika lösningar på medhavd dator.*
  - Diskussion och analys - *För dig som vill diskutera och analysera utifrån det du redan kan och det du kommer att lära dig och som inte är rädd för okända begrepp eller att gå lite utanför kursens ramar.*

Seminarier

- Välj själv vilken inriktning du vill gå på. Det är också okej att byta inriktning mellan gångerna, om det inte verkar passa.
- Alla seminariegrupper ligger parallellt i schemat:
  - En sak i taget (Simon Delvert, Malcolm Vigren)
  - Experimentering (Simon Lindblad, Janos Dani)
  - Diskussion och analys (Fredrik Heintz)
- Seminariematerial och mer information finns på kurswebben.

Laborationer

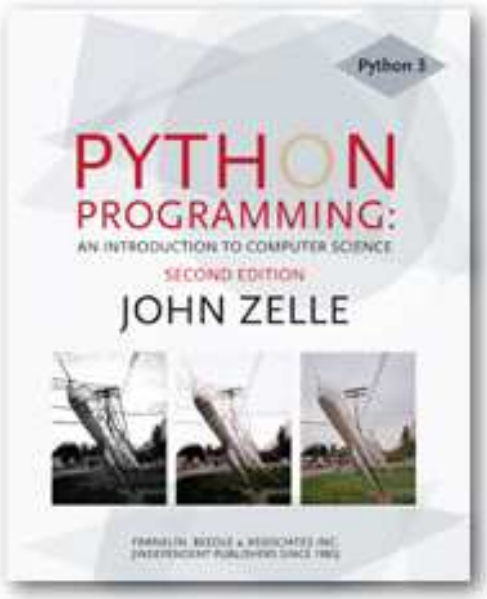
- Under de fyra första veckorna har vi en labbomgång per vecka som genomförs och redovisas individuellt.
- Inför den femte laborationen ska ni bilda par, men det återkommer vi till senare.
- Det finns två labbpass per vecka med tillgång till assistenter. Ni har blivit indelade i grupper (se webben) där varje grupp har olika pass i schemat.
- Utöver detta förväntas man labba på egen hand, antingen i våra salar eller på egen dator.

IDA:s regler för laborationer

- **Det är inte tillåtet** att lämna in lösningar som har kopierats från andra studenter, eller från annat håll, även om modifieringar har gjorts. Om otillåten kopiering eller annan form av fusk misstänks, är läraren skyldig att göra en anmälan till universitetets disciplinnämnd.
- **Du ska kunna** redogöra för detaljer i koden för ett program. Det kan också tänkas att du får förklara varför du har valt en viss lösning. Detta gäller alla i en grupp.
- **Om du förutser** att du inte hinner redovisa i tid, ska du kontakta din lärare. Då kan du få stöd och hjälp och eventuellt kan tidpunkten för redovisningen senareläggas. Det är alltid bättre att diskutera problem än att, t.ex., fuska.
- **Om du inte följer** universitetets och en kurs examinationsregler, utan försöker fuska, t.ex. plagiera eller använda otillåtna hjälpmedel, kan detta resultera i en anmälan till universitetets disciplinnämnd. Konsekvenserna av ett beslut om fusk kan bli varning eller avstängning från studierna.

Kurslitteratur

- John M. Zelle (2010) *Python Programming: An Introduction to Computer Science. Second Edition*. ISBN 978-1-59028-241-0



- Boken har ett bra pedagogiskt upplägg som ansluter väl till det vi vill göra i kursen.
- Den täcker dock inte allt som vi vill gå igenom, så en del saker kommer enbart att tas upp på föreläsningar.
- Om man skaffar en annan Python-bok, se till att den täcker Python 3!

Personal

- **Examinator:** Peter Dalenius
- **Kursadministratör:** Anna Grabska Eklund
- **Laborationsassistenter:**
  - D1A: Johannes Myllylä, Emil Segerbäck
  - D1B: Anders Märak Leffler, Simon Delvert
  - D1C: Malcolm Vigren, Johan Runestam
  - U1A: Niklas Erhard Olsson, Janos Dani
  - U1B: Simon Lindblad, Raymond Leow

Information och kommunikation

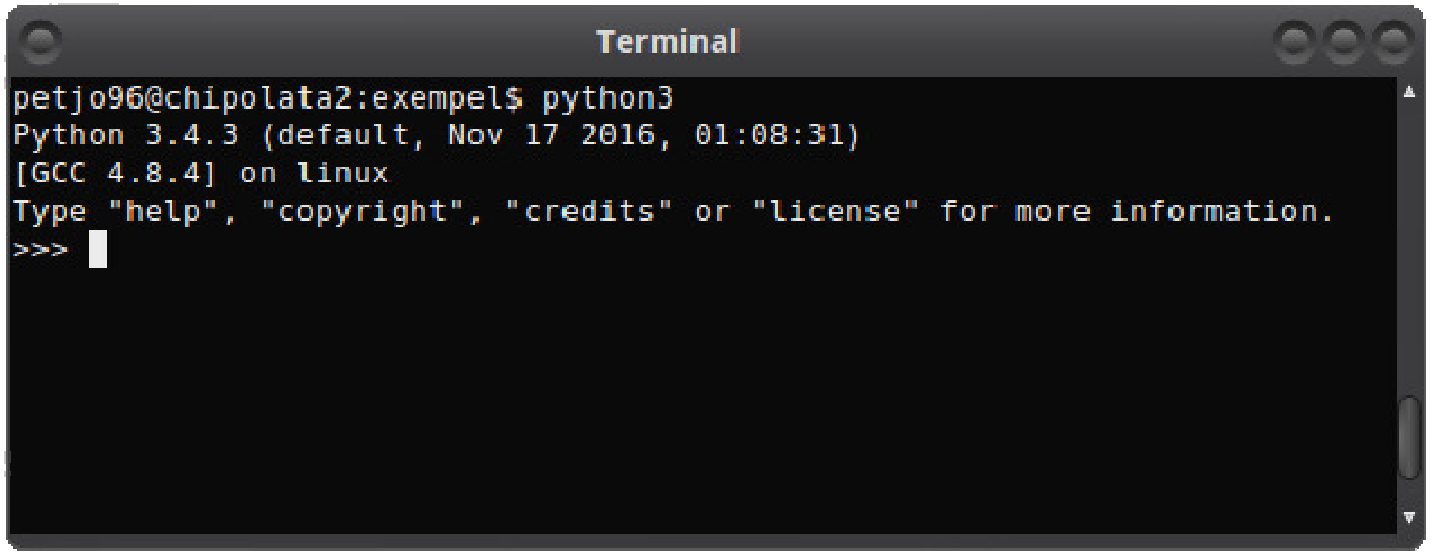
- På kurswebben finns all information om kursen man behöver – *studiematerial, material från föreläsningar och seminarier, instruktioner för laborationer, planering, deadlines, schema, regler för redovisning*

  http://www.ida.liu.se/~TDDE23

- Vi kommer skicka ut kompletterande information till studentmejlen, som ni förväntas läsa varje dag

Vårt första Python-program

Hur man startar Python-interpretatorn



Öppna en terminal och kör kommandot **python3**.
När du ser prompten **>>>** är Python redo att ta emot kommandon.

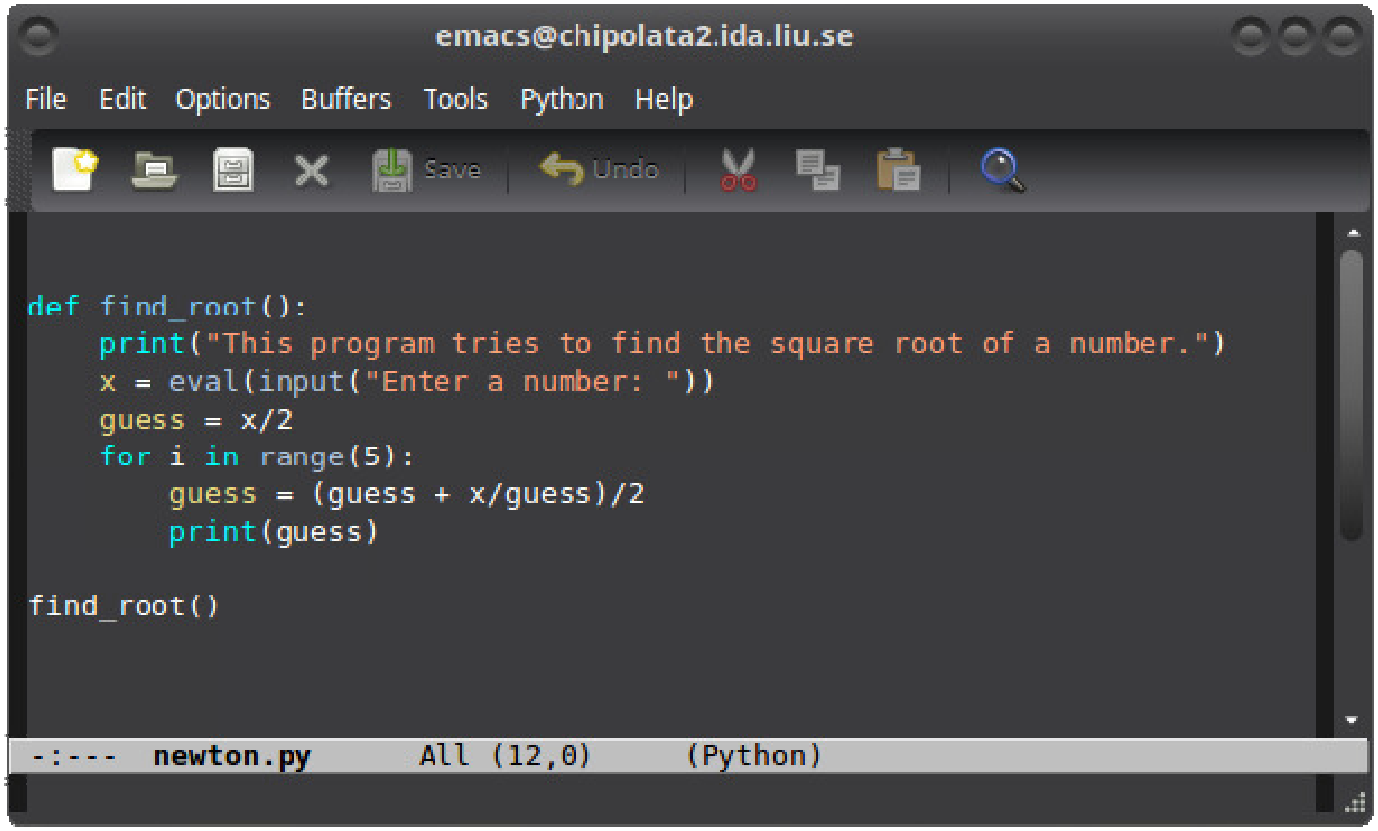
Hur man använder Python-interpretatorn

```
>>> 2+3
5
>>> 2+4*5-6
16
>>> print("Hello world!")
Hello world!
>>> def greeting():
...     print(”Nobody expects the Spanish Inquisition.")
...     print(”Our chief weapon is surprise... and fear.")
...
>>> greeting()
Nobody expects the Spanish Inquisition.
Our chief weapon is surprise... and fear.
>>>
```

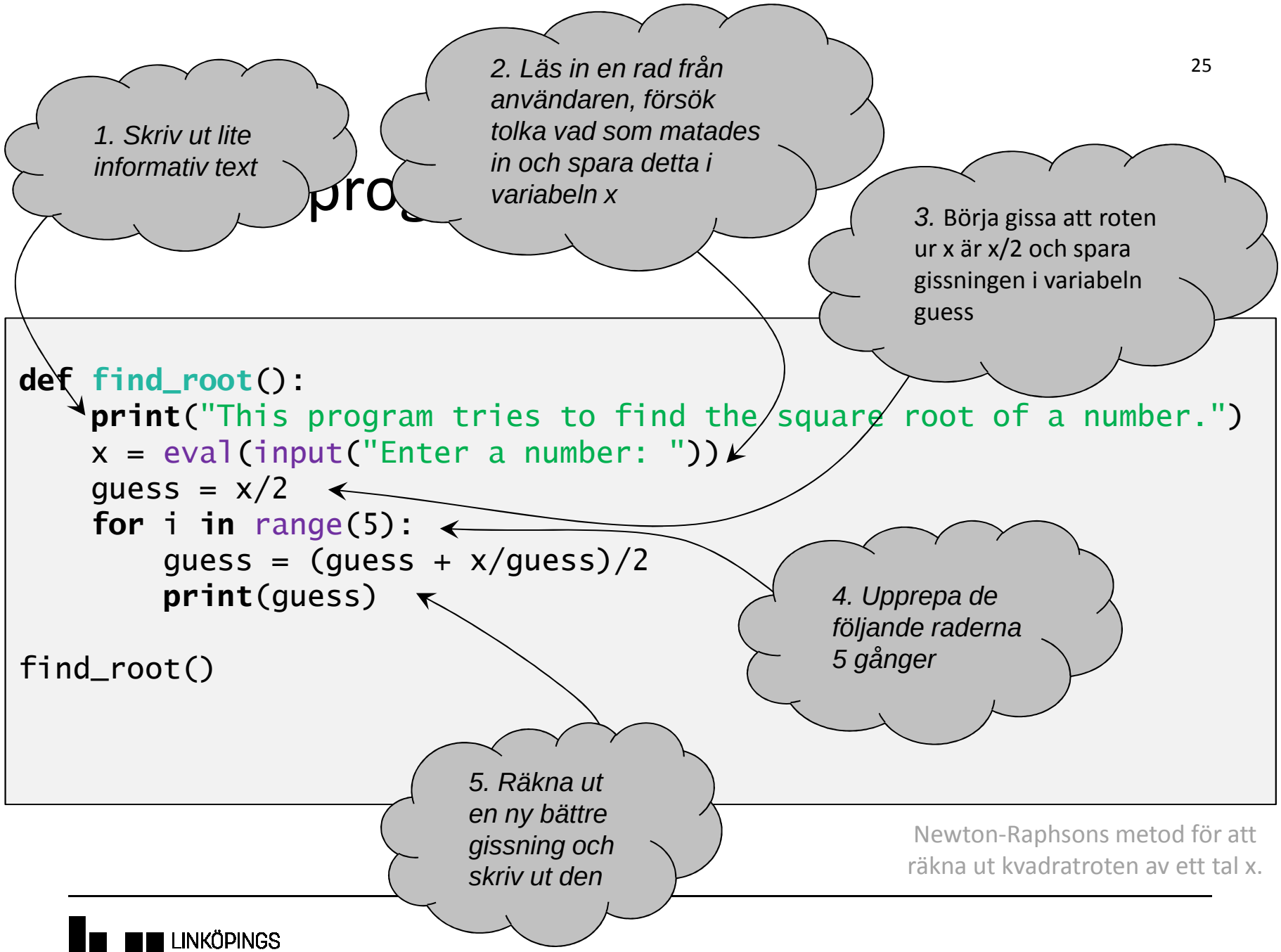
Hur man använder Python-interpretatorn

```
>>> def skryt(namn):
...     print("Det är bra att kunna Python.")
...     print(namn, "är jättebra på Python.")
...
>>> skryt("Peter")
Det är bra att kunna Python.
Peter är jättebra på Python.
>>> skryt("Jenny")
Det är bra att kunna Python.
Jenny är jättebra på Python.
>>> skryt
<function skryt at 0x218390>
>>> print
<built-in function print>
>>>
```

Vårt första Python-program



pro




```
def find_root():
    print("This program tries to find the square root of a number.")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess)/2
        print(guess)

find_root()
```

Newton-Raphsons metod för att räkna ut kvadratroten av ett tal x.

Varför så mycket mellanrum?

```
def find_root():
    print("This program tries to find the square root of a number.")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess)/2
        print(guess)

find_root()
```

Underordnade rader måste skjutas in en bit på raden, ***indenteras****.*

Newton-Raphsons metod för att räkna ut kvadratroten av ett tal x.

Olika sätt att köra vårt program

```
petjo96@chipolata2:python$ python3
Python 3.4.3 (default, Nov 17 2016, 01:08:31)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more
information.
>>> import newton
This program tries to find the square root of a number.
Enter a number: 2
1.5
1.4166666666666665
1.4142156862745097
1.4142135623746899
1.414213562373095
>>>
```

Olika sätt att köra vårt program

```
>>> newton.find_root()
This program tries to find the square root of a number.
Enter a number: 10
3.5
3.178571428571429
3.162319422150883
3.1622776604441363
3.162277660168379
>>> from newton import *
>>> find_root()
This program tries to find the square root of a number.
Enter a number: 3
1.75
1.7321428571428572
1.7320508100147274
1.7320508075688772
1.7320508075688772
>>> exit()
petjo96@chipolata2:python$
```

Ordning och reda på programkoden

Programkod organiseras i *moduler*, och om man inte gör något speciellt kommer varje fil att bli en modul.

`import newton`

Ladda in modulen (filen) newton *som finns i filen* newton.py. *Funktioner i modulen går att anropa med* newton.*funktionsnamn*

`from newton import *`

Ladda in modulen (filen) newton *så att allt som finns i den går att anropa direkt. Detta är bra för våra små program, men blir rörigt när vi kommer lite längre fram.*

Satser – ryggraden i ett program

Ett Python-program består av ett antal *satser (eng. statements)* som utförs i tur och ordning. Vi ska titta på de fem viktigaste, där vi redan sett de fyra första:

- Enkla satser
  - *Tilldelning (eng. assignment)*
  - *Funktionsanrop*
- Sammansatta satser
  - *Iteration (upprepning) med* for
  - *Funktionsdefinition*
  - *Selektion (val) med* if

Olika typer av satser: Tilldelning

Hur ser det ut? namn = uttryck
Vad händer? Uttrycket beräknas och resultatet sparas i den namngivna variabeln.

```
def find_root():
    print("This program tries to find the square root of a number.")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess)/2
        print(guess)

find_root()
```

Olika typer av satser: Funktionsanrop

Hur ser det ut? funktionsnamn(frivilliga argument)
Vad händer? Argumenten skickas till funktionen, underprogrammet, för behandling.

```
def find_root():
    print("This program tries to find the square root of a number.")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess)/2
        print(guess)

find_root()
```

Funktionsanrop som inte är självständiga satser

Olika typer av satser: Iteration med for

Hur ser det ut? **for** namn **in** område: undersatser
Vad händer? Undersatserna körs så många gånger som området specificerar.

```
def find_root():
    print("This program tries to find the square root of a number.")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess)/2
        print(guess)

find_root()
```

Olika typer av satser: Funktionsdefinition

Hur ser det ut? **def** funktionsamn(frivilliga argument): undersatser
Vad händer? Undersatserna sparas för att kunna anropas senare.

```
def find_root():
    print("This program tries to find the square root of a number.")
    x = eval(input("Enter a number: "))
    guess = x/2
    for i in range(5):
        guess = (guess + x/guess)/2
        print(guess)

find_root()
```

Olika typer av satser: Selektion med if

Enkelt val

```
if a > 0:
  print(“positivt")
```

Två alternativ

```
if a > 0:
  print(“positivt")
else:
  print(“negativt")
```

Tre eller fler alternativ

```
if 0 <= a < 10:
  print("ental")
elif a < 0:
  print("negativt")
else:
  print("tio eller mer")
```

Inför första laborationstillfället

- Ta reda på vilken labbgrupp du är med i (se kurswebben).
- Ta reda på när den labbgruppen har sitt första labbtillfälle (se schemat).
- Dyk upp på rätt plats i rätt tid. Våra laborationer brukar börja kvart över. De flesta har sin första laboration imorgon.

www.liu.se

LINKÖPINGS UNIVERSITET