

TSRT04 Introduktionskurs i MATLAB

Föreläsning

Johan Löfberg

Reglerteknik, ISY
LiTH

E-mail: tsrt04@isy.liu.se

WWW: <http://www.control.isy.liu.se/student/tsrt04/>

16 januari, 2011

Kursupplägg

- 1 föreläsning, 2×2 h lektioner, 4×2 h laboration

Kursupplägg

- 1 föreläsning, 2×2 h lektioner, 4×2 h laboration
 - Självstudier — betoning på egna upptäckter.

Kursupplägg

- 1 föreläsning, 2×2 h lektioner, 4×2 h laboration
 - Självstudier — betoning på egna upptäckter.
 - Kursmaterial på kurshemsida (MATLAB laddas ner från Studentportalen)



Kursupplägg

- 1 föreläsning, 2×2 h lektioner, 4×2 h laboration
 - Självstudier — betoning på egna upptäckter.
 - Kursmaterial på kurshemsida (MATLAB laddas ner från Studentportalen)
- Labbar!



Kursupplägg

- 1 föreläsning, 2×2 h lektioner, 4×2 h laboration
 - Självstudier — betoning på egna upptäckter.
 - Kursmaterial på kurshemsida (MATLAB laddas ner från Studentportalen)
- Labbar!
 - Man labbar i par.
 - Labba med någon på samma kunskapsnivå.
 - Vi kräver att båda av er skall sköta programmeringen, dvs tangentbordet!

Examination

- Labbar = Examination, men också tillfälle att lära sig!

Examination

- Labbar = Examination, men också tillfälle att lära sig!

"Miniprojekt"

- Delas ut på första labtillfället, redovisas senast på fjärde.
- Däremellan: Hemarbete!
- Obligatorisk närvaro tills projektet redovisats och godkänts.
- Om du ej kan närvara: Kontakta examinator Johan Löfberg (tsrt04@isy.liu.se).



Examination

- Labbar = Examination, men också tillfälle att lära sig!

"Miniprojekt"

- Delas ut på första labtillfället, redovisas senast på fjärde.
- Däremellan: Hemarbete!
- Obligatorisk närvaro tills projektet redovisats och godkänts.
- Om du ej kan närvara: Kontakta examinator Johan Löfberg (tsrt04@isy.liu.se).

Plottuppgift

- Delas ut på andra lektionen och första labben.
- Plotta och visa ett dataset på olika sätt.

Examination

- Labbar = Examination, men också tillfälle att lära sig!

"Miniprojekt"

- Delas ut på första labtillfället, redovisas senast på fjärde.
- Däremellan: Hemarbete!
- Obligatorisk närvaro tills projektet redovisats och godkänts.
- Om du ej kan närvara: Kontakta examinator Johan Löfberg (tsrt04@isy.liu.se).

Plottuppgift

- Delas ut på andra lektionen och första labben.
- Plotta och visa ett dataset på olika sätt.

Quiz

- Görs individuellt innan projektredovisning.

Tidigare kursutvärderingar

- Väldigt enkel

Tidigare kursutvärderingar

- Väldigt enkel
- Den var för enkel och ytlig

Tidigare kursutvärderingar

- Väldigt enkel
- Den var för enkel och ytlig
- YATZY-projektet var väldigt enkelt

Tidigare kursutvärderingar

- Väldigt enkel
- Den var för enkel och ytlig
- YATZY-projektet var väldigt enkelt
- Examineringen var väldigt enkel, för enkel

Tidigare kursutvärderingar

- Väldigt enkel
- Den var för enkel och ytlig
- YATZY-projektet var väldigt enkelt
- Examineringen var väldigt enkel, för enkel
- Det var kul men lite väl lätt trots att jag inte har någon programmeringsbakgrund.



Tidigare kursutvärderingar

- Väldigt enkel
- Den var för enkel och ytlig
- YATZY-projektet var väldigt enkelt
- Examineringen var väldigt enkel, för enkel
- Det var kul men lite väl lätt trots att jag inte har någon programmeringsbakgrund.
- Jag fann kursen omöjlig trots tidigare programmeringserfarenhet

Tidigare kursutvärderingar

- Väldigt enkel
- Den var för enkel och ytlig
- YATZY-projektet var väldigt enkelt
- Examineringen var väldigt enkel, för enkel
- Det var kul men lite väl lätt trots att jag inte har någon programmeringsbakgrund.
- Jag fann kursen omöjlig trots tidigare programmeringserfarenhet
- Förenkla kursen!

Tidigare kursutvärderingar

- Väldigt enkel
- Den var för enkel och ytlig
- YATZY-projektet var väldigt enkelt
- Examineringen var väldigt enkel, för enkel
- Det var kul men lite väl lätt trots att jag inte har någon programmeringsbakgrund.
- Jag fann kursen omöjlig trots tidigare programmeringserfarenhet
- Förenkla kursen!
- Projekten var för svåra



Tidigare kursutvärderingar

- Väldigt enkel
- Den var för enkel och ytlig
- YATZY-projektet var väldigt enkelt
- Examineringen var väldigt enkel, för enkel
- Det var kul men lite väl lätt trots att jag inte har någon programmeringsbakgrund.
- Jag fann kursen omöjlig trots tidigare programmeringserfarenhet
- Förenkla kursen!
- Projekten var för svåra
- För lite poäng för arbetsinsatsen

Tidigare kursutvärderingar

- Väldigt enkel
- Den var för enkel och ytlig
- YATZY-projektet var väldigt enkelt
- Examineringen var väldigt enkel, för enkel
- Det var kul men lite väl lätt trots att jag inte har någon programmeringsbakgrund.
- Jag fann kursen omöjlig trots tidigare programmeringserfarenhet
- Förenkla kursen!
- Projekten var för svåra
- För lite poäng för arbetsinsatsen

Notera: 2 hp \approx 55 h studier. Bara 14 h schemalagda...



Vad är MATLAB?

- MATrix LABoratory
- Från början: ett verktyg för linjär algebra.
- Nu:
 - Avancerad miniräknare/beräkningsprogram
 - Programspråk
 - Kan användas för många sorters beräkningar och simuleringar
 - Många användningsområden: Fysik, statistik, telekom, elektronik, ekonomi. . .

Navigering av obemannad helikopter

Mål: Att beräkna position och bygga upp en karta av omgivningen utan att använda GPS.

Sensorer:

- Kamera
- Accelerometrar
- Gyro
- Lufttrycksgivare (höjd)

MATLAB: Används för att utvärdera och implementera våra matematiska algoritmer.



MATLAB som räknedosa

```
>> 2e3 + 300
```



MATLAB som räknedosa

```
>> 2e3 + 300
```

```
ans = 2300
```

MATLAB som räknedosa

```
>> 2e3 + 300
```

```
ans = 2300
```

```
>> cos(pi)
```



MATLAB som räknedosa

```
>> 2e3 + 300
```

```
ans = 2300
```

```
>> cos(pi)
```

```
ans = -1
```

MATLAB som räknedosa

```
>> 2e3 + 300
```

```
ans = 2300
```

```
>> cos(pi)
```

```
ans = -1
```

```
>> abs(1+i)
```

MATLAB som räknedosa

```
>> 2e3 + 300
```

```
ans = 2300
```

```
>> cos(pi)
```

```
ans = -1
```

```
>> abs(1+i)
```

```
ans = 1.4142
```



Viktigaste funktionen: help

- "help kommando" ger en hjälptext om "kommando"

Viktigaste funktionen: help

- "help kommando" ger en hjälptext om "kommando"
- "doc kommando" ger en mer utförlig hjälp

Viktigaste funktionen: help

- "help kommando" ger en hjälptext om "kommando"
- "doc kommando" ger en mer utförlig hjälp
- "helpdesk" startar en hjälpbrowser

Viktigaste funktionen: help

- "help kommando" ger en hjälptext om "kommando"
- "doc kommando" ger en mer utförlig hjälp
- "helpdesk" startar en hjälpbrowser
- "help" ger en lista över "toolboxar" (samlingar av kommandon ordnade ämnesvis)

Minst viktiga funktionen: why

- “why” ger en totalt oanvändbar förklaring till varför saker och ting inte går som man vill.



Variabler

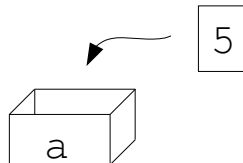
- En "behållare" att spara värden i.
- Har ett **namn** och ett **värde**.

```
>> a = 5
```

Variabler

- En "behållare" att spara värden i.
- Har ett **namn** och ett **värde**.

```
>> a = 5
```

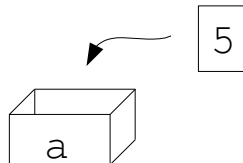


Variabler

- En "behållare" att spara värden i.
- Har ett **namn** och ett **värde**.

```
>> a = 5
```

```
a = 5
```



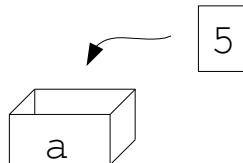
Variabler

- En "behållare" att spara värden i.
- Har ett **namn** och ett **värde**.

```
>> a = 5
```

```
a = 5
```

```
>> b = a + 3
```



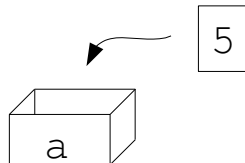
Variabler

- En "behållare" att spara värden i.
- Har ett **namn** och ett **värde**.

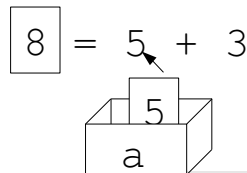
```
>> a = 5
```

```
a = 5
```

```
>> b = a + 3
```



(HL beräknas först, och resultatet lagras i b.)



Variabler

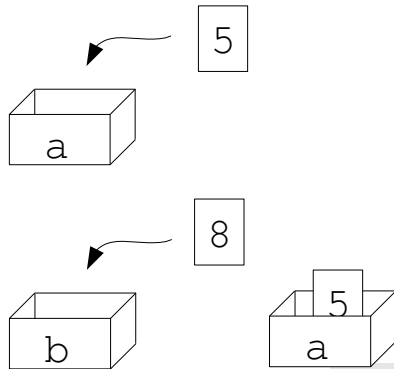
- En "behållare" att spara värden i.
- Har ett **namn** och ett **värde**.

```
>> a = 5
```

```
a = 5
```

```
>> b = a + 3
```

```
b = 8
```



(HL beräknas först, och resultatet lagras i b.)



Variabler

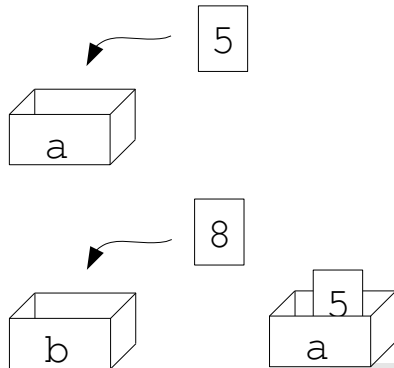
- En "behållare" att spara värden i.
- Har ett **namn** och ett **värde**.

```
>> a = 5
```

```
a = 5
```

```
>> b = a + 3
```

```
b = 8
```



(HL beräknas först, och resultatet lagras i b.)

Vad händer om man skriver

```
>> a = a + 2?
```

Variabler

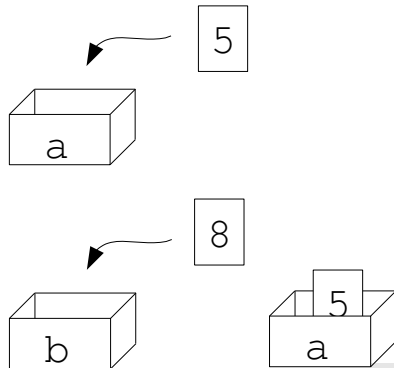
- En "behållare" att spara värden i.
- Har ett **namn** och ett **värde**.

```
>> a = 5
```

```
a = 5
```

```
>> b = a + 3
```

```
b = 8
```



(HL beräknas först, och resultatet lagras i b.)

Vad händer om man skriver

```
>> a = a + 2?
```

```
a = 7
```

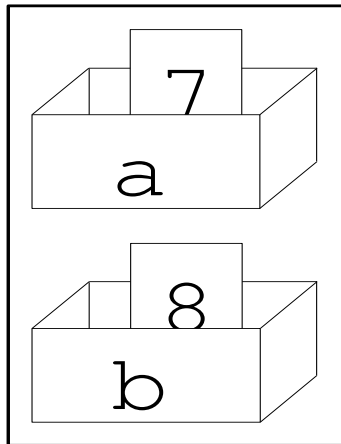
Workspace

Platsen där variablerna sparas kallas "workspace" (jmf. ett arkivskåp).

Workspace (MATLAB)

```
a = 7
```

```
b = 8
```



Vektorer och matriser

Vektorer och matriser är en viktig del av MATLAB.

- $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ skrivs `>> A = [1 2;3 4]`

Vektorer och matriser

Vektorer och matriser är en viktig del av MATLAB.

- $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ skrivs `>> A = [1 2;3 4]`
- $a = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ skrivs `>> a = [1; 2; 3]`



Vektorer och matriser

Vektorer och matriser är en viktig del av MATLAB.

- $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ skrivs `>> A = [1 2;3 4]`
- $a = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ skrivs `>> a = [1; 2; 3]`
- $b = (4 \ 5 \ 6)$ skrivs `>> b = [4 5 6]`
(eller `[4, 5, 6]`)



Vektorer och matriser

- Transponat skrivs med `.'` (räcker med `'` på reella matriser) :



Vektorer och matriser

- Transponat skrivs med `.'` (räcker med `'` på reella matriser) :
`>> b.'`



Vektorer och matriser

- Transponat skrivs med `.'` (räcker med `'` på reella matriser) :

```
>> b.'
```

```
ans =
```

```
4
```

```
5
```

```
6
```

Vektorer och matriser

- Transponat skrivs med `.'` (räcker med `'` på reella matriser) :

```
>> b.'
```

```
ans =
```

```
4
```

```
5
```

```
6
```

Speciella matriser och vektorer:

Vektorer och matriser

- Transponat skrivs med `.'` (räcker med `'` på reella matriser) :

```
>> b.'
```

```
ans =
```

```
4
```

```
5
```

```
6
```

Speciella matriser och vektorer:

- `>> C = eye(2)` ger $C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.

Vektorer och matriser

- Transponat skrivs med `.'` (räcker med `'` på reella matriser) :

```
>> b.'
```

```
ans =
```

```
4
```

```
5
```

```
6
```

Speciella matriser och vektorer:

- `>> C = eye(2)` ger $C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.
- `>> x = 3:6` ger $x = (3 \ 4 \ 5 \ 6)$.



Vektorer och matriser

- Transponat skrivs med `.'` (räcker med `'` på reella matriser) :

```
>> b.'
```

```
ans =
```

```
4
```

```
5
```

```
6
```

Speciella matriser och vektorer:

- `>> C = eye(2)` ger $C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.
- `>> x = 3:6` ger $x = (3 \ 4 \ 5 \ 6)$.
- `>> x = 2:3:11` ger $x = (2 \ 5 \ 8 \ 11)$.



Matrisoperationer

Vi kan räkna med matriser:

```
>> A = [1 2;3 4];
```

```
>> B = eye(2);
```

```
>> A*B
```

Matrisoperationer

Vi kan räkna med matriser:

```
>> A = [1 2;3 4];
```

```
>> B = eye(2);
```

```
>> A*B
```

```
ans =
```

```
1 2
```

```
3 4
```

Matrisoperationer

Vi kan räkna med matriser:

```
>> A = [1 2;3 4];
```

```
>> B = eye(2);
```

```
>> A*B
```

```
ans =
```

```
1 2
```

```
3 4
```

$$\text{dvs } \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

(vanlig matrismultiplikation)

Matrisoperationer

Vi kan räkna med matriser:

```
>> A = [1 2;3 4];
```

```
>> B = eye(2);
```

```
>> A*B
```

```
>> A.*B
```

```
ans =
```

```
1 2
```

```
3 4
```

$$\text{dvs } \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

(vanlig matrismultiplikation)



Matrisoperationer

Vi kan räkna med matriser:

```
>> A = [1 2;3 4];
```

```
>> B = eye(2);
```

```
>> A*B
```

```
ans =
```

```
1 2  
3 4
```

$$\text{dvs } \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

(vanlig matrismultiplikation)

```
>> A.*B
```

```
ans =
```

```
1 0  
0 4
```



Matrisoperationer

Vi kan räkna med matriser:

```
>> A = [1 2;3 4];
```

```
>> B = eye(2);
```

```
>> A*B
```

```
ans =
```

```
1 2
3 4
```

$$\text{dvs } \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

(vanlig matrismultiplikation)

```
>> A.*B
```

```
ans =
```

```
1 0
0 4
```

$$\text{dvs } \begin{pmatrix} 1 \cdot 1 & 2 \cdot 0 \\ 3 \cdot 0 & 4 \cdot 1 \end{pmatrix}$$

(komponentvis multiplikation)

Matrisoperationer

Andra funktioner jobbar ofta komponentvis:

```
>> x = 0:(pi/2):(2*pi)
```

```
x = 0      1.5708      3.1416      4.7124      6.2832
```

```
>> y = sin(x)
```

Matrisoperationer

Andra funktioner jobbar ofta komponentvis:

```
>> x = 0:(pi/2):(2*pi)
```

```
x = 0    1.5708    3.1416    4.7124    6.2832
```

```
>> y = sin(x)
```

```
y = 0    1.0000    0.0000   -1.0000   -0.0000
```



Indexering av Matriser

Hur får man tag i enskilda komponenter i vektorer och matriser?

```
>> y(4)
```

Indexering av Matriser

Hur får man tag i enskilda komponenter i vektorer och matriser?

```
>> y(4)
```

```
ans = -1
```

Indexering av Matriser

Hur får man tag i enskilda komponenter i vektorer och matriser?

```
>> y(4)
```

```
ans = -1
```

```
>> A = [3 5 2;7 8 6];
```

$$A = \begin{pmatrix} 3 & 5 & 2 \\ 7 & 8 & 6 \end{pmatrix}$$

Indexering av Matriser

Hur får man tag i enskilda komponenter i vektorer och matriser?

```
>> y(4)
```

```
ans = -1
```

```
>> A = [3 5 2;7 8 6];
```

```
>> A(1,2)
```

$$A = \begin{pmatrix} 3 & 5 & 2 \\ 7 & 8 & 6 \end{pmatrix}$$



Indexering av Matriser

Hur får man tag i enskilda komponenter i vektorer och matriser?

```
>> y(4)
```

```
ans = -1
```

```
>> A = [3 5 2;7 8 6];
```

```
>> A(1,2)
```

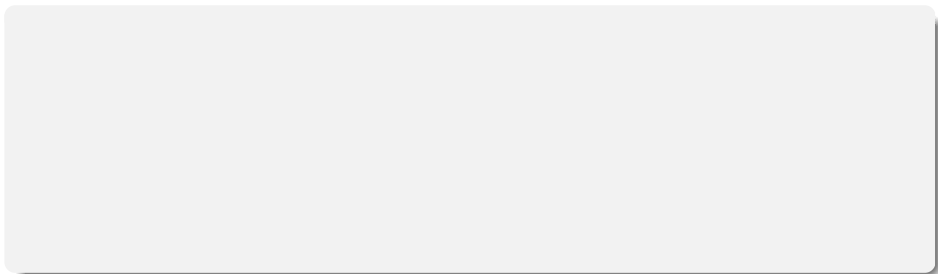
```
ans = 5
```

$$A = \begin{pmatrix} 3 & 5 & 2 \\ 7 & 8 & 6 \end{pmatrix}$$



Grafik

Anta att vi vill plotta $y = \sin(x)$ för $0 \leq x \leq 10$:



Grafik

Anta att vi vill plotta $y = \sin(x)$ för $0 \leq x \leq 10$:

```
x = 0:0.1:10; % The x for which y should be computed
```

Grafik

Anta att vi vill plotta $y = \sin(x)$ för $0 \leq x \leq 10$:

```
x = 0:0.1:10; % The x for which y should be computed  
y = sin(x);
```

Grafik

Anta att vi vill plotta $y = \sin(x)$ för $0 \leq x \leq 10$:

```
x = 0:0.1:10; % The x for which y should be computed  
y = sin(x);
```

```
plot(x,y) % Plot y vs. x
```

Grafik

Anta att vi vill plotta $y = \sin(x)$ för $0 \leq x \leq 10$:

```
x = 0:0.1:10; % The x for which y should be computed  
y = sin(x);
```

```
plot(x,y) % Plot y vs. x  
xlabel('x')
```

Grafik

Anta att vi vill plotta $y = \sin(x)$ för $0 \leq x \leq 10$:

```
x = 0:0.1:10; % The x for which y should be computed  
y = sin(x);
```

```
plot(x,y) % Plot y vs. x  
xlabel('x')  
ylabel('y = sin(x)')
```

Grafik

Anta att vi vill plotta $y = \sin(x)$ för $0 \leq x \leq 10$:

```
x = 0:0.1:10; % The x for which y should be computed  
y = sin(x);
```

```
plot(x,y) % Plot y vs. x  
xlabel('x')  
ylabel('y = sin(x)')  
title('My first plot')
```

Skript

- Ett sätt att utföra flera kommandon på en gång.

Skript

- Ett sätt att utföra flera kommandon på en gång.
- Spara ett antal kommandon i en **m-fil** (vars namn slutar på `.m`) och kör alla på en gång genom att bara skriva filnamnet vid kommandoraden.

Skript

- Ett sätt att utföra flera kommandon på en gång.
- Spara ett antal kommandon i en **m-fil** (vars namn slutar på `.m`) och kör alla på en gång genom att bara skriva filnamnet vid kommandoraden.
- `>> edit` startar en editor där man kan skriva m-filen.

Skript

- Ett sätt att utföra flera kommandon på en gång.
- Spara ett antal kommandon i en **m-fil** (vars namn slutar på `.m`) och kör alla på en gång genom att bara skriva filnamnet vid kommandoraden.
- `>> edit` startar en editor där man kan skriva m-filen.
- Kommentarer kan skrivas med `%`

Exempel på skript

Lina har sprungit 5 km på 23 min 15 s, och vill räkna ut tiden per km.

Exempel på skript

Lina har sprungit 5 km på 23 min 15 s, och vill räkna ut tiden per km.

m-fil (runpacescript.m)

```
dist = 5;    % Distance in km
min = 23;    % Total time expressed in
s = 15;     % minutes and seconds

% Compute time per km in minutes:
totalminutes = min + s/60;
minperkm = totalminutes/dist
```



Funktioner

- Nackdel med skript: Kan skriva över variabler man har definierat i Workspace.

Funktioner

- Nackdel med skript: Kan skriva över variabler man har definierat i Workspace.
- Annat koncept: **Funktioner** — skapar sin egen Workspace.

Funktioner

- Nackdel med skript: Kan skriva över variabler man har definierat i Workspace.
- Annat koncept: **Funktioner** — skapar sin egen Workspace.
- Fungerar precis som MATLABs egna funktioner.

Function Example

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km in minutes, given
% the distance, and the total time expressed
% in minutes and seconds.

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```



Function Example

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km in minutes, given
% the distance, and the total time expressed
% in minutes and seconds.

totalminutes = min + s/60;
minperkm = totalminutes/dist;
end
```

- `function` — talar om att det är en funktion

Function Example

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km in minutes, given
% the distance, and the total time expressed
% in minutes and seconds.

totalminutes = min + s/60;
minperkm = totalminutes/dist;
end
```

- *funktionsnamn* — samma som m-filens namn



Function Example

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km in minutes, given
% the distance, and the total time expressed
% in minutes and seconds.

totalminutes = min + s/60;
minperkm = totalminutes/dist;
end
```

- *inargument* — de värden som funktionen behöver



Function Example

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km in minutes, given
% the distance, and the total time expressed
% in minutes and seconds.

totalminutes = min + s/60;
minperkm = totalminutes/dist;
end
```

- *utargument* — det resultat som funktionen ska lämna

Hur körs en funktion

>>

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Workspace (runpace)

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Workspace (runpace)

```
dist = 5
```

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Workspace (runpace)

```
dist = 5
min = 23
```

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Workspace (runpace)

```
dist = 5
min = 23
s = 15
```

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Workspace (runpace)

```
dist = 5
min = 23
s = 15
totalminutes = 23.25
```

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Workspace (runpace)

```
dist = 5
min = 23
s = 15
totalminutes = 23.25
minperkm = 4.65
```

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Workspace (runpace)

```
dist = 5
min = 23
s = 15
totalminutes = 23.25
minperkm = 4.65
```

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

Workspace (MATLAB)

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Workspace (runpace)

```
dist = 5
min = 23
s = 15
totalminutes = 23.25
minperkm = 4.65
```

Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

```
mpkm = 4.65
```

Workspace (MATLAB)

```
mpkm = 4.65
```

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Workspace (runpace)

```
dist = 5
min = 23
s = 15
totalminutes = 23.25
minperkm = 4.65
```



Hur körs en funktion

```
>> mpkm=runpace(5,23,15)
```

```
mpkm = 4.65
```

Workspace (MATLAB)

```
mpkm = 4.65
```

m-fil (runpace.m)

```
function minperkm = runpace(dist, min, s)
% Computes the time per km...

    totalminutes = min + s/60;
    minperkm = totalminutes/dist;
end
```

Skript vs. funktioner

Skript

- En sekvens av kommandon, inget annat

Funktioner

- Funkar som MATLABs egna funktioner

Skript vs. funktioner

Skript

- En sekvens av kommandon, inget annat
- Använder MATLABs Workspace

Funktioner

- Funkar som MATLABs egna funktioner
- Skapar egen Workspace

Skript vs. funktioner

Skript

- En sekvens av kommandon, inget annat
- Använder MATLABs Workspace
- Måste skriva in "startvärden" i skriptet eller i kommandoraden

Funktioner

- Funkar som MATLABs egna funktioner
- Skapar egen Workspace
- Kan ta in- och utargument

Ett lite större exempel

Exempel ("Platespotting")

- Se registrerings skyltar i nummerordning (leta först efter en bil med 001, sedan efter en med 001, osv)
- Hur många bilar måste man se i genomsnitt innan man har nått till 999?



Ett lite större exempel

Exempel ("Platespotting")

- Se registrerings skyltar i nummerordning (leta först efter en bil med 001, sedan efter en med 001, osv)
- Hur många bilar måste man se i genomsnitt innan man har nått till 999?
- Anta: Vi ser bilar slumpmässigt, med lika stor sannolikhet för alla nummer (likformig fördelning/uniform distribution).



Ett lite större exempel

Exempel ("Platespotting")

- Se registrerings skyltar i nummerordning (leta först efter en bil med 001, sedan efter en med 001, osv)
- Hur många bilar måste man se i genomsnitt innan man har nått till 999?
- Anta: Vi ser bilar slumpmässigt, med lika stor sannolikhet för alla nummer (likformig fördelning/uniform distribution).
- Med sannolikhetslära kan man räkna ut resultatet analytiskt.

Ett lite större exempel

Exempel ("Platespotting")

- Se registrerings skyltar i nummerordning (leta först efter en bil med 001, sedan efter en med 001, osv)
- Hur många bilar måste man se i genomsnitt innan man har nått till 999?
- Anta: Vi ser bilar slumpmässigt, med lika stor sannolikhet för alla nummer (likformig fördelning/uniform distribution).
- Med sannolikhetslära kan man räkna ut resultatet analytiskt.
- Vi provar att simulera istället!

Skript för platespotting

1. Skapa ett slumpmässigt registreringsnummer:

```
plate = 1+floor(999*rand);    % Create random plate
```

Skript för platespotting

2. Kolla om det var det rätta registreringsnumret:

```
carfound = false;

plate = 1+floor(999*rand);      % Create random plate

if plate == 1                  % If right plate is found
    carfound = true;
end
```

Skript för platespotting

3. Kolla flera bilar, tills det rätta registreringsnumret hittats:

```
carfound = false;

while carfound == false           % Search until found
    plate = 1+floor(999*rand);     % Create random plate

    if plate == 1                 % If right plate is found
        carfound = true;
    end
end
```

Skript för platespotting

4. Räkna hur många bilar vi kollat:

```
carfound = false;
carsseen = 0;

while carfound == false           % Search until found
    plate = 1+floor(999*rand);     % Create random plate
    carsseen = carsseen+1;
    if plate == 1                 % If right plate is found
        carfound = true;
    end
end
```

Skript för platespotting

5. Upprepa för alla registreringsnummer från 0 till 999:

```
totalcarsseen = 0;

for targetplate = 1:999           % Search plate number 001-999
    carfound = false;
    carsseen = 0;

    while carfound == false      % Search until found
        plate = 1+floor(999*rand); % Create random plate
        carsseen = carsseen+1;
        if plate == targetplate  % If right plate is found
            carfound = true;
        end
    end

    totalcarsseen = totalcarsseen + carsseen;
end
```

Skript för platespotting

6. Färdigt!

```
totalcarsseen = 0;

for targetplate = 1:999           % Search plate number 001-999
    carfound = false;
    carsseen = 0;

    while carfound == false      % Search until found
        plate = 1+floor(999*rand); % Create random plate
        carsseen = carsseen+1;
        if plate == targetplate  % If right plate is found
            carfound = true;
        end
    end

    totalcarsseen = totalcarsseen + carsseen;
end
```

Kontrollstrukturer

I exemplet användes olika *kontrollstrukturer*:

- if-satser
- while-loopar
- for-loopar

if-satser

- Om man vill göra olika beräkningar beroende på resultatet av vissa delberäkningar:

```
if plate == targetplate  
    carfound = true;  
end
```

- Generell struktur:

```
if villkor  
    kommandon om villkor=sant  
else  
    kommandon om villkor=falskt  
end
```

if-satser

Exempel

Ett kundkort ger 2% ränta på inestående saldo och tar 14% ränta för krediter. Skriv en funktion som beräknar räntan för ett givet belopp.



if-satser

Exempel

Ett kundkort ger 2% ränta på inestående saldo och tar 14% ränta för krediter. Skriv en funktion som beräknar räntan för ett givet belopp.

m-fil (cardinterest.m)

```
function interest = cardinterest(amount)
% Computes annual interest for a given amount

if amount >= 0
    interest = 0.02*amount;
else
    interest = 0.14*amount;
end
```

while-loopar

- Om man vill upprepa liknande beräkningar ett okänt antal gånger:

```
while carfound == false
    plate = 1+floor(999*rand);
    carcount = carcount+1;
    if plate == targetplate
        carfound = true;
    end
end
```

- Generell struktur:

```
while villkor
    kommandon
end
```

for-loopar

- Om man vill upprepa liknande beräkningar ett bestämt antal gånger:

```
for targetplate = 1:999
```

```
    ...
```

```
end
```

- Generell struktur:

```
for var = vektor med olika värden
```

```
    kommandon
```

```
end
```

Tips för framtiden (och examinationen)

Dela upp stora problem i mindre delar!

Försök dela upp stora problem i små välavgränsade delproblem som är lättare att testa var för sig.

Tips för framtiden (och examinationen)

Dela upp stora problem i mindre delar!

Försök dela upp stora problem i små välavgränsade delproblem som är lättare att testa var för sig.

Bli vän med debuggern!

Använd MATLABs debugger, den ger bättre förståelse för vad som händer, vad de kommandon man använder faktiskt gör, och varför man får de resultat man får.

Tips för framtiden (och examinationen)

Dela upp stora problem i mindre delar!

Försök dela upp stora problem i små välavgränsade delproblem som är lättare att testa var för sig.

Bli vän med debuggern!

Använd MATLABs debugger, den ger bättre förståelse för vad som händer, vad de kommandon man använder faktiskt gör, och varför man får de resultat man får.

Läs felmeddelanden!

Om MATLAB-koden avbryts och felmeddelande skrivs ut, läs dem noga. Vanligaste felet: stavfel **??? Undefined function or variable**

Sammanfattning

- MATLAB är bra för många sorters beräkningar.

Sammanfattning

- MATLAB är bra för många sorters beräkningar.
- Använd skript och funktioner, *det sparar tid* för dig och assistenten.

Sammanfattning

- MATLAB är bra för många sorters beräkningar.
- Använd skript och funktioner, *det sparar tid* för dig och assistenten.
- Kontrollstrukturer:

Sammanfattning

- MATLAB är bra för många sorters beräkningar.
- Använd skript och funktioner, *det sparar tid* för dig och assistenten.
- Kontrollstrukturer:
 - if-satser — gör olika beräkningar beroende på delresultat.

Sammanfattning

- MATLAB är bra för många sorters beräkningar.
- Använd skript och funktioner, *det sparar tid* för dig och assistenten.
- Kontrollstrukturer:
 - if-satser — gör olika beräkningar beroende på delresultat.
 - for-loopar — upprepar beräkningar ett visst antal gånger.

Sammanfattning

- MATLAB är bra för många sorters beräkningar.
- Använd skript och funktioner, *det sparar tid* för dig och assistenten.
- Kontrollstrukturer:
 - if-satser — gör olika beräkningar beroende på delresultat.
 - for-loopar — upprepar beräkningar ett visst antal gånger.
 - while-loopar — upprepar beräkningar ett obestämt antal gånger.

Sammanfattning

- MATLAB är bra för många sorters beräkningar.
- Använd skript och funktioner, *det sparar tid* för dig och assistenten.
- Kontrollstrukturer:
 - if-satser — gör olika beräkningar beroende på delresultat.
 - for-loopar — upprepar beräkningar ett visst antal gånger.
 - while-loopar — upprepar beräkningar ett obestämt antal gånger.
- Använd hjälpfunktionerna för att ta reda på mer!!!

Lycka till med resten av kursen!
Ha kul med MATLAB!