Cheat sheets

Nedan finns referensblad för fyra olika programmeringsspråk, som kan bli aktuella att använda i matematikundervisning.

MATLAB är en välkänd programvara för att göra matematiska beräkningar, som används även på industriell nivå och för tunga beräkningar och simuleringar. Språket har stöd för många matematiska metoder och specialfall, och är relativt lätt att komma in i.

Octave är en gratis och öppen programvara som har mycket stora likheter med MATLAB, och ett språk som är nära identiskt med MATLAB.

Python3 är ett programmeringsspråk som inkluderats framförallt för att det är relativt lätt att använda. Det finns, liksom i Octave/MATLAB möjlighet att hantera komplexa tal.

JavaScript och PHP är två programmeringsspråk som inkluderats framförallt för att de är vanliga inom webbutveckling, och det därför är elever eller lärare använt dem i andra sammanhang.

Två språk inte inkluderats är Java och C++, trots att de är vanliga programmeringsspråk. Anledningen är att de är relativt svåra att komma igång med, samt att personer som använder dessa språk troligtvis hanterar dem så pass väl att referensblad på den här nivån är överflödiga.

Referensbladen är skrivna för att visa grundläggande användning av språken, plus några aspekter som kan vara särskilt intressant för arbete med matematik. Exemplen är valda för att avspegla samma saker i alla språk, vilket betyder att unika och kanske mer användbara aspekter av varje språk inte syns. Det uppmuntras att söka på nätet för att ta reda på mer om hur det språk du väljer används.

Platser för att skriva och testa kod online

I de flesta fall fungerar det bäst att skriva och exekvera (köra) kod lokalt på den egna datorn. Ofta kräver det dock att man installerar särskilda program, och det kan också finnas svårigheter med att använda vissa programmeringsspråk på vissa typer av enheter.

Ett alternativ för att komma igång snabbt är att utnyttja onlinetjänster gjorda just för att skriva och exekvera mindre mängd kod. Några sådana tjänster är:

- repl.it: Stödjer en stor uppsättning språk, och har funktioner för att spara och dela kod. Det finns även guider för att lära sig språken. Stödjer *inte* Octave/MATLAB.
- tutorialspoint.com/codingground.htm: Stödjer en stor uppsättning språk, och har funktioner för att spara och dela kod. Det finns även guider för att lära sig språken. Har stöd även för Octave/MATLAB.
- octave-online.net: Webbplats som endast stödjer Octave/MATLAB. Det finns funktioner för att spara och dela kod.

Cheat sheet – Octave/MATLAB

Det finns ofta fler sätt att hantera språket än vad exemplen nedan visar. Sök på nätet för en mer fullständig referens.

I Octave/MATLAB konverteras variabler i allmänhet till arrayer/vektorer, och den som vill använda Octave eller MATLAB i större omfattning vinner mycket på att hantera arrayer, vektorer och även matriser i språket.

Två platser för att skriva kod online: tutorialspoint.com/codingground.htm och octave-online.net.

Skapa variabler

```
siffervariabel = 1;
textvariabel = "hello";
```

Vanliga operationer

```
% Fyra räknesätt samt 2 upphöjt till 3
siffervariabel = siffervariabel + 2 * 3 - 4 / 5;
siffervariabel = 2 ^ 3;
% Förenklat skrivsätt för att öka en variabels värde med ett
Siffervariabel++
% Sammanslagning av strängar
textvariabel = [textvariabel " world"];
```

Skriva ut värden

```
% Visa resultat i terminal (avsluta raden utan semikolon)
textvariabel
```

Vanliga matematiska konstanter och funktioner

```
pi
e
round(2.5); % Avrundas till 3
abs(-1); % Absolutvärdet av -1
rand(); % Slumptal mellan 0 och 1
max([1, 2, 3, 4]);
min([1, 2, 3, 4]);
sqrt(9);
log10(x);
log(x); % Naturliga logaritmen för x
sin(x); % Vinkeln anges i radianer
cos(x); % Vinkeln anges i radianer
```

Syntax för villkorssatser

```
if (sin(x) == 0)
  'På y-axeln'
elseif (sin(x) > 0)
  'Positivt y-värde'
elseif (sin(x) < 0)
  'Negativt y-värde'
else
  'Kan inte tolka y-värdet'
endif
```

Syntax för loopar

```
% Räkna från 0 till 4
for n = 0:4
  ["Jag har räknat till " mat2str(n)]
endfor

% Loop baserat på villkor istället för uppräkning
while (n <= 100)
  n++
endwhile
```

Syntax för arrayer

```
% Skapa en tom array
arrayvariabel = [];
% Skapa en array med värden
arrayvariabel = ['1', '2', 'femton', 'äpple'];
% Lägga till ett värde i slutet av en array
arrayvariabel = [arrayvariabel 'ny'];
% Välja ut ett enskilt värde i en array -- räknar från 1
arrayvariabel(1);

% Loopa genom alla värden i en array med tal
for varde = arrayvariabel
  varde
endfor
```

Syntax för att skapa och anropa funktioner

```
function returvarde = minFunktion (x, y)
  returvarde = x * sqrt(y);
endfunction

minFunktion(2, 5)
```

Övrigt

```
% Octave/MATLAB hanterar även komplexa tal.
z = (1 + i) * (1 + 3i)
```

Cheat sheet – Python3

Det finns ofta fler sätt att hantera språket än vad exemplen nedan visar. Sök på nätet för en mer fullständig referens.

Två platser för att skriva kod online: repl.it och tutorialspoint.com/codingground.htm.

Skapa variabler

```
siffervariabel = 1
textvariabel = "hello"
```

Vanliga operationer

```
# Fyra räknesätt samt 2 upphöjt till 3
siffervariabel = siffervariabel + 2 * 3 - 4 / 5
siffervariabel = 2 ** 3
# Förenklat skrivsätt för att öka en variabels värde med ett
Siffervariabel += 1
# Sammanslagning av strängar
textvariabel = textvariabel + " world"
```

Skriva ut värden

```
# Visa resultat
Print(siffervariabel)
```

Vanliga matematiska konstanter och funktioner

```
# math- eller random-modulen behövs i vissa fall
import math
import random
math.pi
math.e
round(2.5) # Avrundas till 3
abs(-1) # Absolutvärdet av -1
random.random() # Slumptal mellan 0 och 1
max(1, 2, 3, 4)
min(1, 2, 3, 4)
math.sqrt(9)
math.log10(x)
math.log(x) # Naturliga logaritmen för x
math.sin(x) # Vinkeln anges i radianer
math.cos(x) # Vinkeln anges i radianer
```

Syntax för villkorssatser

```
if math.sin(x) == 0: print('På y-axeln')
elif math.sin(x) > 0: print('Positivt y-värde')
elif math.sin(x) < 0: print('Negativt y-värde')
else: print('Kan inte tolka y-värdet')
```

Syntax för loopar

```
# Räkna från 0 till 4
for n in range(5):
  print('Jag har räknat till ' + str(n))

# Loop baserat på villkor istället för uppräkning
while n <= 100:
  n += 1
```

Syntax för arrayer

```
# Skapa en tom array
arrayvariabel = []
# Skapa en array med värden
arrayvariabel = [1, 2, 'femton', 'äpple']
# Lägga till ett värde i slutet av en array
arrayvariabel.append(x)
# Välja ut ett enskilt värde i en array -- räknar från 0
Print(arrayvariabel[0])

# Loopa genom alla värden i en array
for varde in arrayvariabel:
  print(varde)
```

Syntax för att skapa och anropa funktioner

```
def minFunktion(x, y):
  x = x * math.sqrt(y)
  return x

print(minFunktion(2, 5))
```

Övrigt

```
# cmath-modulen gör det möjligt att hantera komplexa tal.
# "j" (inte "i") används för den imaginära enheten,
# och måste alltid föregås av ett tal.
import cmath
z = (1 + 1j) * (1 + 3j)
```

Cheat sheet – JavaScript

Det finns ofta fler sätt att hantera språket än vad exemplen nedan visar. Sök på nätet för en mer fullständig referens.

Två platser för att skriva kod online: repl.it och tutorialspoint.com/codingground.htm.

Skapa variabler

```
var siffervariabel = 1;
var textvariabel = "hello";
```

Vanliga operationer

```
// Fyra räknesätt samt 2 upphöjt till 3
siffervariabel = siffervariabel + 2 * 3 - 4 / 5;
siffervariabel = Math.pow(2, 3);
// Förenklat skrivsätt för att öka en variabels värde med ett
siffervariabel++;
// Sammanslagning av strängar
textvariabel = textvariabel + " world";
```

Skriva ut värden

```
// Visa resultat i en popup-ruta
window.alert(textvariabel);
// Visa resultat i log
console.log(siffervariabel);
```

Vanliga matematiska konstanter och funktioner

```
Math.PI
Math.E
Math.round(2.5); // Avrundas till 3
Math.abs(-1); // Absolutvärdet av -1
Math.random(); // Slumptal mellan 0 och 1
Math.max(1, 2, 3, 4);
Math.min(1, 2, 3, 4);
Math.sqrt(9);
Math.log10(x);
Math.log(x); // Naturliga logaritmen för x
Math.sin(x); // Vinkeln anges i radianer
Math.cos(x); // Vinkeln anges i radianer
```

Syntax för villkorssatser

```
if (sin(x) == 0) {
  console.log('På y-axeln');
else if (sin(x) > 0) {
  console.log('Positivt y-värde');
} else if (sin(x) < 0) {
  console.log('Negativt y-värde');
} else {
  console.log('Kan inte tolka y-värdet');
}
```

Syntax för loopar

```
// Räkna från 0 till 4
for (n = 0; n < 5; n++) {
  console.log('Jag har räknat till ' + n);
}

// Loop baserat på villkor istället för uppräkning
while (n <= 100) {
  n++;
}
```

Syntax för arrayer

```
// Skapa en tom array
arrayvariabel = [];
// Skapa en array med värden
arrayvariabel = [1, 2, 'femton', 'äpple'];
// Lägga till ett värde i slutet av en array
arrayvariabel.push(x);
// Välja ut ett enskilt värde i en array -- räknar från 0
console.log(arrayvariabel[0]);

// Skapa ett objekt -- till skillnad från arrayer blir värdena inte
sorterade
objektvariabel = {'ett' : 'one', 'två' : 'two', 3 : 'three'};
// Välja ut ett enskilt värde i ett objekt
objektvariabel['ett']; // 'one'
objektvariabel[3]; // 'three'
objektvariabel['3']; // Odefinierat

// Loopa genom alla värden i ett objekt
for (var i in objektvariabel) {
  console.log(objektvariabel[i]);
}
```

Syntax för att skapa och anropa funktioner

```
function minFunktion(x, y) {
  x = x * Math.sqrt(y);
  return x;
}

console.log(minFunktion(2, 5));
```

Cheat sheet – PHP

Det finns ofta fler sätt att hantera språket än vad exemplen nedan visar. Sök på nätet för en mer fullständig referens.

Två platser för att skriva kod online: repl.it och tutorialspoint.com/codingground.htm.

Skapa variabler

```
$siffervariabel = 1;
$textvariabel = "hello";
```

Vanliga operationer

```
// Fyra räknesätt samt 2 upphöjt till 3
$siffervariabel = $siffervariabel + 2 * 3 - 4 / 5;
$siffervariabel = pow(2, 3);
// Förenklat skrivsätt för att öka en variabels värde med ett
$siffervariabel++;
// Sammanslagning av strängar
$textvariabel = $textvariabel . " world";
```

Skriva ut värden

```
// Visa resultat i log
print $siffervariabel;
```

Vanliga matematiska konstanter och funktioner

```
pi();
exp(1); // e upphöjt till 1, dvs. e
round(2.5); // Avrundas till 3
abs(-1); // Absolutvärdet av -1
rand(5, 10); // Slumpat heltal mellan 5 och 10
max(1, 2, 3, 4);
min(1, 2, 3, 4);
sqrt(9);
log10($x);
log($x); // Naturliga logaritmen för $x
sin($x); // Vinkeln anges i radianer
cos($x); // Vinkeln anges i radianer
```

Syntax för villkorssatser

```
if (sin($x) == 0) {
  print 'På y-axeln';
} else if (sin($x) > 0) {
  print 'Positivt y-värde';
} else if (sin($x) < 0) {
  print 'Negativt y-värde';
} else {
  print 'Kan inte tolka y-värdet';
}
```

Syntax för loopar

```
// Räkna från 0 till 4
for ($n = 0; $n < 5; $n++) {
  print 'Jag har räknat till ' . $n;
}

// Loop baserat på villkor istället för uppräkning
while ($n <= 100) {
  $n++;
}
```

Syntax för arrayer

```
// Skapa en tom array
$arrayvariabel = [];
// Skapa en array med värden
$arrayvariabel = [1, 2, 'femton', 'äpple'];
// Lägga till ett värde i slutet av en array
$arrayvariabel[] = $x;
// Välja ut ett enskilt värde i en array -- räknar från 0
print $arrayvariabel[0];

// Loopa genom alla värden i en array
foreach ($arrayvariabel as $index => $varde) {
  print $varde;
}
```

Syntax för att skapa och anropa funktioner

```
function minFunktion($x, $y) {
  $x = $x * sqrt($y);
  return $x;
}

print minFunktion(2, 5);
```

Övrigt

```
// Skapa en radbrytning i utskriven text
Print "\r\n";
```

Tips vid felsökning

En oundviklig del av programmering är att hitta och åtgärda så kallade buggar – kodfel som får program att fungera på oväntade sätt eller inte fungera alls.

Att felsöka kod är en konst i sig, särskilt när program börjar bli komplexa, men det finns ändå några tips på vanliga misstag och metoder för att hitta och åtgärda buggar.

Generella metoder för felsökning

- Att skriva ut kommentarer i koden, som beskriver vad olika avsnitt av koden gör, är ett bra sätt att hålla ordning både på kod och tankar. Det gör det lättare att felsöka själv, även långt i efterhand, och för andra att hjälpa till med felsökning. Titta i referensbladen för att se hur kommentarer skrivs i ditt programmeringsspråk. (Kommentarer är rader som finns med i koden, men ignoreras när programmet körs.)
- Det är ofta användbart att på ett antal ställen i programmet skriva ut/logga värden på variabler som beter sig konstigt, för att ringa in var de spårar ur. (I ordentliga programmeringsmiljöer finns möjlighet att stega genom program och följa värden på uttryck och variabler, men i exempelvis onlinemiljöer för programmering är det ovanligt.)
- Om man är tämligen säker på att felet uppstår i en viss del av koden, men inte kan hitta var, kan det vara användbart att skriva om hela det kodstycket från början. (Du kan då spara den gamla koden, men inleda varje rad med de symboler som används för kommentarer i ditt programmeringsspråk.)

Om programmet inte kan köras

- Vissa programmeringsspråk kräver att kommandon avslutas med ett semikolon. Har du missat att avsluta en rad korrekt?
- I många programmeringsspråk används måsvingar, { och }, för att börja och avsluta block för loopar eller villkor. Har du missat att avsluta något sådant block?
- I många programmeringsspråk används dubbla likhetstecken för villkor (x == 3), medan enkla likhetstecken (x = 3) gör antingen att programmet inte kan köras eller beter sig konstigt. Har du missat att använda dubbla likhetstecken i en villkorssats?

Om programmet ger oväntade resultat

- I många programmeringsspråk används dubbla likhetstecken för villkor (x == 3), medan enkla likhetstecken (x = 3) gör antingen att programmet inte kan köras eller beter sig konstigt. Har du missat att använda dubbla likhetstecken i en villkorssats?
- I många programmeringsspråk börjar arrayer eller listor med värden att räkna från 0, medan andra räknar man från 1. Finns det en miss i hur värden i listor räknas?