

Felsökning (Debugging)

När man skriver program och testar dem uppstår alltid fel. Det är en naturlig del i själva programmeringsarbetet.

Tre olika typer av fel:

- **Syntaxfel**
Följer ej språkets regler. Upptäcks när programmet skrivas.
- **Exekveringsfel**
Uppstår när programmet körs. Exvis index utanför gränserna, division med noll.
- **Logiska fel**
Programmet kan köras, men ger fel resultat. Dessa fel tar ofta tid att åtgärda

Vid syntaxfel och exekveringsfel ger Python felutskrifter. Dessa lär man sig eftersom att tyda.

Exempel på syntaxfel:

```
s = input('Ge två tal, x och y: ')
s = q.split()
x = int(s[0])
y = int(s[1])
print('x/ (x-y)=', x/ (x-y) )
```

Utskrifter när programmet körs, som avbryts:

```
Ge två tal, x och y: 6 9
```

```
Traceback (most recent call last):
```

```
  File "...debugging.py", line 2, in <module>
    s = q.split()
```

```
NameError: name 'q' is not defined
```

Exempel på exekveringsfel:

```
s = input('Ge två tal, x och y: ')
s = s.split()
x = int(s[0])
y = int(s[1])
print('x/ (x-y)=', x/ (x-y))
```

Utskrifter när programmet körs, som avbryts:

```
Ge två tal, x och y: 5 5
```

```
Traceback (most recent call last):
```

```
  File "...debugging.py", line 5, in <module>
```

```
    print('x/ (x-y)=', x/ (x-y))
```

```
ZeroDivisionError: division by zero
```

Svårare är det med logiska fel. Några tips:

- **Manuell simulerings av programkörningen**

Försök följa programmet sats för sats manuellt genom att läsa koden, uppifrån och ned.

- **Testa med olika indata till programmet**

Detta kan ge vägledning om var felet kan finnas.

- **Skriv ut vilka funktioner/metoder som anropas**

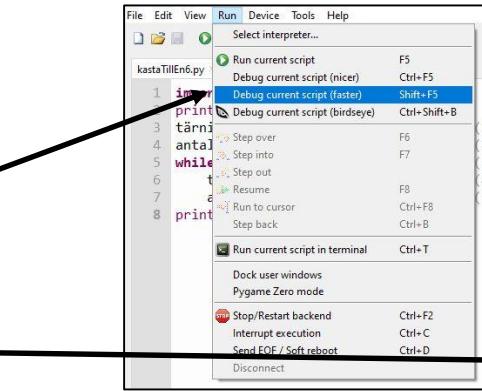
Om du får veta vilka funktioner/metoder som verkligen anropas kan det ge ledtrådar. Lägg in en utskriftssats i varje metod (eller åtminstone i de som är misstänkta att innehålla fel).

- **Skriv ut eller visa variablernas värden (debuggingutskrifter)**
Detta är det bästa sättet eftersom det avslöjar vad som verkligen beräknas i programmet. Det ger vägledning om huruvida variablerna har korrekt värde (som du tror) för att programmet skall fungera.
- Kommentera bort delar av koden. Testkör. Kan då ringa in felet.
- Använd debugging-verktyg. Till ett programmeringsmiljö (IDE) finns ofta ett speciellt program som kan användas för felsökning, en s.k. debugger. Med en sådant program kan man köra (exekvera) programmet på lite olika sätt. Visar detta i Thonny.

Debugging i Thonny

Välj "Debug current script (faster)"

Första satsen i koden gulmarkeras



A screenshot of the Thonny IDE code editor. The first line of the script, 'import random', is highlighted in yellow. The status bar at the bottom right shows 'Step over (F6)'. The code editor displays the same Python script as the previous screenshot.

```
File Edit View Run Device Tools Help  
kastaTillEn6.py  
1 import random  
2 print()  
3 tärning = random.randint(1,6)  
4 antal = 1  
5 while tärning != 6:  
6     tärning = random.randint(1,6)  
7     antal = antal + 1  
8 print('Antal kast = ', antal)
```

Välj att köra välj "Step over".

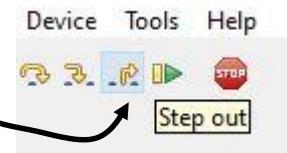
Den första satsen (den gula) körs och nästa sats i koden gulmarkeras.

Varje man gång man väljer "Step Over" kommer nästa sats att köras.

A screenshot of the Thonny IDE code editor. The second line of the script, 'print()', is highlighted in yellow. The code editor displays the same Python script as the previous screenshots.

```
File Edit View Run Device Tools Help  
kastaTillEn6.py  
1 import random  
2 print()  
3 tärning = random.randint(1,6)  
4 antal = 1  
5 while tärning != 6:  
6     tärning = random.randint(1,6)  
7     antal = antal + 1  
8 print('Antal kast = ', antal)
```

Om man vill köra klart programmet, välj Step Out".



Debugging i Thonny...

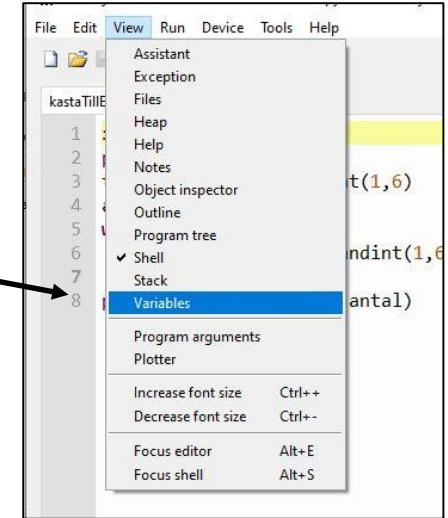
Bevaka variabelvärdet löpande: Välj "View Variables"

Då dyker ett fönster "Variables" upp:

The screenshot shows the Thonny IDE interface. On the left is the script editor with the file 'kastaTillEn6.py' containing the following code:

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

To the right of the script editor is the 'Variables' window, which is currently empty. A callout box with an arrow points from the text 'När programmet körs listas alla variabler och dess värden' to the top-left corner of the 'Variables' window.



Debugging i Thonny...

Alternativ till att köra stegvis:

Sätt stopp-satser i koden: "*Breakpoints*".

Dubbelklicka på radnumret, raden markeras med en röd pkt.

Välj "*Debug current script (faster)*"

Programmet körs till satsen vid brytpunkten.

Kör med stegvis med *Resume*,



programmet kommer att för varje varv i loopen
stanna vid brytpunkten. Exempel på slutresultat
"View Variables".

A screenshot of the Thonny IDE showing a Python script named "kastaTillEn6.py". The code contains a while loop that continues until a random roll is not equal to 6. A red dot at the start of line 7 indicates a breakpoint has been set. The line of code with the breakpoint is highlighted in yellow.

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

A screenshot of the Thonny IDE during debugging. The "Run" menu is open, and the "Resume (F8)" option is selected. The code editor shows the same script with the breakpoint at line 7. The line containing the breakpoint is highlighted in yellow. The "Variables" panel on the right shows the state of variables: antal is 1, random is <module 'random'>, and tärning is 5.

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

Name	Value
antal	1
random	<module 'random'>
tärning	5

A screenshot of the Thonny IDE showing the result of stepping over the breakpoint. The "Run" menu is open, and the "Step Over (F11)" option is selected. The code editor shows the script with the breakpoint now past the point where it was set. The "Variables" panel shows the final values: antal is 6, random is <module 'random'>, and tärning is 6. The shell window at the bottom shows the output "Antal kast = 6".

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

Name	Value
antal	6
random	<module 'random'>
tärning	6

```
Antal kast = 6
```

Debugging i Thonny...

Vi testar följande program

- Kasta tärning tills en sexa, [kastaTillEn6.py](#)
- Kasta tärning tills två sexor i rad, två alternativ, [kastaTillTva6.py](#)
- Beräkna n!, [fakultet.py](#)