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 skrivs.

- **Exekveringsfel**
  Uppstår när programmet körs. Exvis konvertering mellan datatyper, 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:

```
x = int (input('Ge ett heltal x: '))
t = int (input('Ge ett heltal y: '))
print('x/(x-y)=', x/(x-y))
```

Utskrifter när programmet körs, som avbryts:

```
Ge ett heltal x: 5
Ge ett heltal y: 9
Traceback (most recent call last):
  File "…ex1.py", line 3, print('x/(x-y)=', x/(x-y))
NameError: name 'y' is not defined
```

Exempel på exekveringsfel:

```
x = int (input('Ge ett heltal x: '))
y = int (input('Ge ett heltal y: '))
print('x/(x-y)=', x/(x-y))
```

Utskrifter när programmet körs, som avbryts:

```
Ge ett heltal x: 5
Ge ett heltal y: 5
Traceback (most recent call last):
  File "…ex1.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 simulering av programkörningen**
  Försök följa programmet sats för sats manuellt genom att läsa koden, uppifrån och ned. Går det igenom looparna.

- **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).

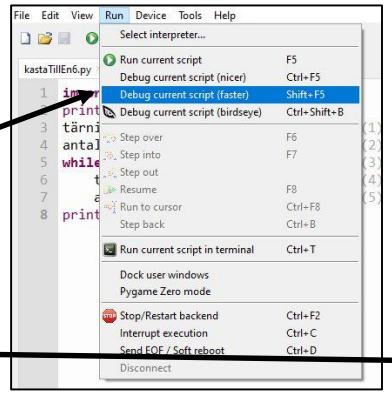
- **Skriv ut eller visa variablernas värden (debuggingutskrifter)** Detta är det bästa sättet eftersom det avslöjar vad som verkligen beräknas i programmet. Det ger vägledning om huruvida variablerna har korrekt värde (som du tror) för att programmet skall fungera.

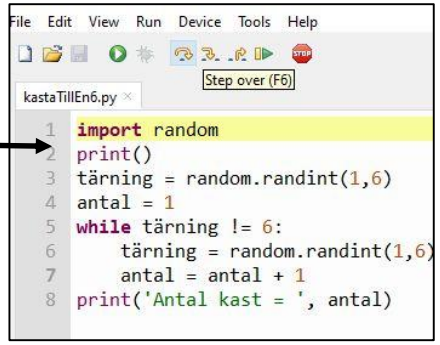
- Kommentera bort delar av koden. Testkör. Kan då ringa in felet.

- 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.

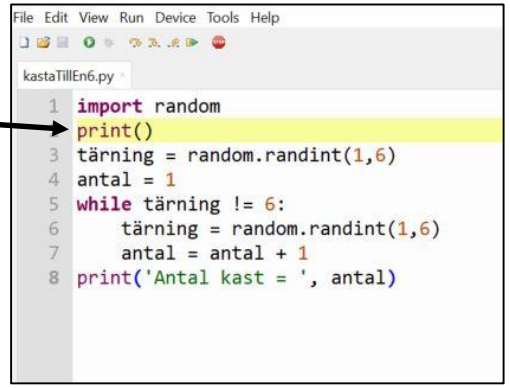
Debugging i Thonny

Välj ”*Debug current script (faster)”*
(eller lusen  *)*
Första satsen i koden gulmarkeras

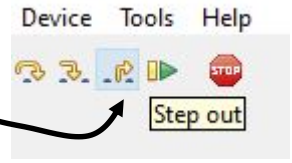




Välj att köra *välj ”Step over”. (el F6)*
Den första satsen (den gula) körs och nästa sats i koden gulmarkeras.
Varje man gång man väljer *”Step Over”* kommer nästa sats att köras.



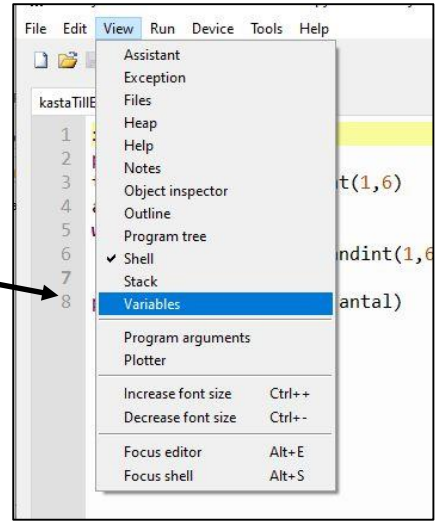
Om man vill köra klart programmet, välj *Step Out”.*



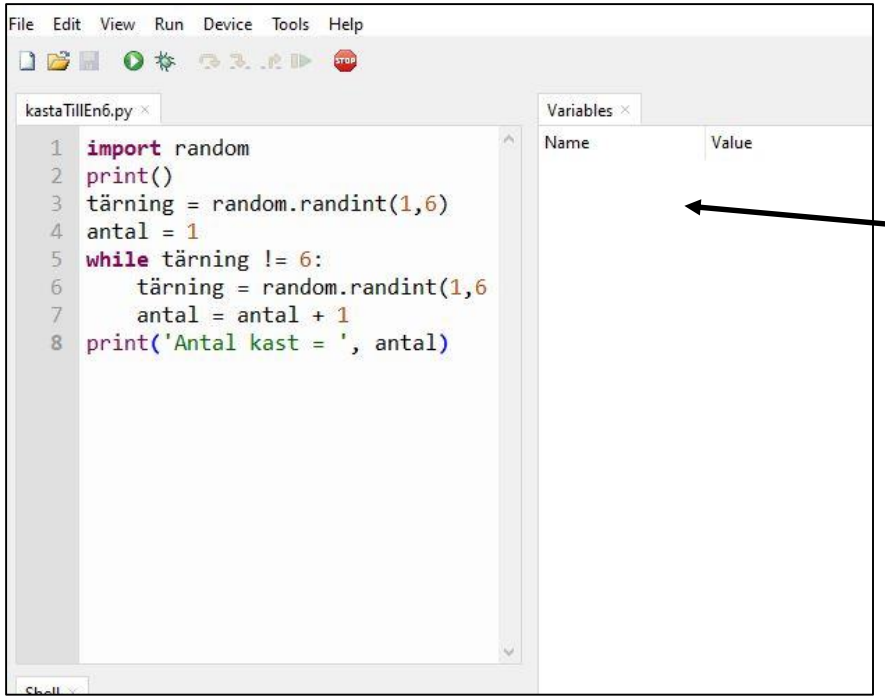
Om man önskar köra mer noggrant, exvis in i en loop, *välj ”Step into”. (el F7)*

Debugging i Thonny…

Bevaka variabelvärden löpande: Välj *”View Variables”*



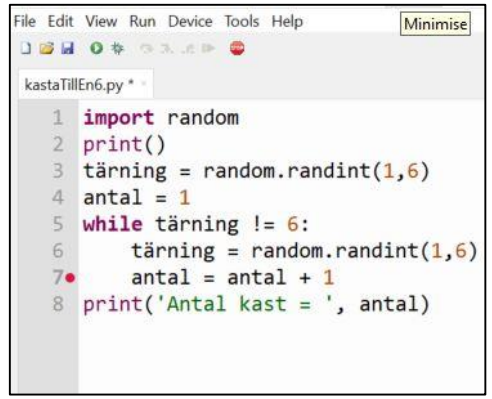
Då dyker ett fönster ”Variables” upp:



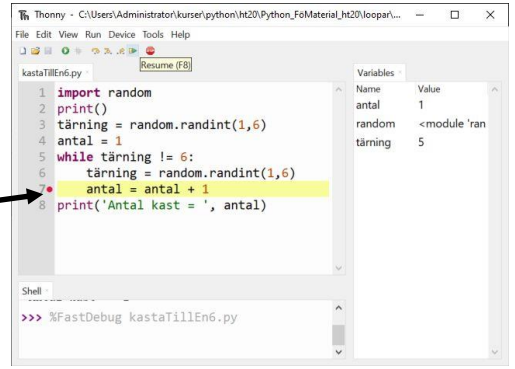
När programmet körs listas alla variabler och dess värden

Debugging i Thonny…

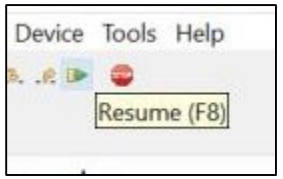
Alternativ till att köra stegvis:
Sätt stopp-satser i koden: *”Breakpoints”.*
Dubbelklicka på radnumret, raden markeras med en röd pkt.



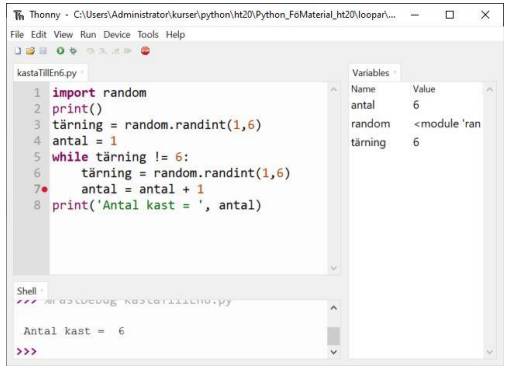
Välj ”*Debug current script (faster)”*
Programmet körs till satsen vid brytpunkten.



Kör med stegvis med *Resume*,



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



Debugging i Thonny…

Vi testar följande program

- Kasta tärning tills en sexa, kastaTillEn6.py
- Beräkna n!, fakultet_fkn.py (mha en egendefinierad funktion)

Mer om debugging i Thonny: Se video, ”minilektion/Referens”, kurswebben