

D0009E

Introduktion till programmering

Kurskompendium från Pelles Moln · 23 noter · 2026-10-19
pellesmoln.org/kurser/D0009E

Innehåll

Grunder

Variabler	3
Datatyper	4
Aritmetik	6
Logiska uttryck	7
Vägval	8

Iteration och funktioner

While-loopar	9
For-loopar	10
Funktioner	12
Parametrar	20
Returvärden	21
Rekursion	22
Scope	23

Datastrukturer

Strängar	24
Listor	25
Tupler	26
Kataloger	27

IO

Filer	28
Felhantering	30

Objektorientering

Objektorientering	31
Klasser	32
Objekt	33
Metoder	34
Konstruktörer	35

Variabler

19 okt. 2026, 1 min läsning

UTKAST

#programming #python

Kurs: D0009E

En **variabel** är ett namngivet referensutrymme för data. I Python skapas den vid tilldelning:

```
1 x = 42
2 name = "Pelle"
3 pi = 3.14
```

Regler

- Namn får bestå av bokstäver, siffror och `_`, men ska inte börja med siffra.
- Skiftlägeskänsligt: `x` och `X` är olika.
- Följ konventionen `snake_case` för variabler.

Dynamisk typning

Typen bestäms vid tilldelningen och kan ändras:

```
1 x = 5          # int
2 x = "hej"      # nu str
```

Se även

- Datatyper
- Scope

Datatyper

19 okt. 2026, 1 min läsning

UTKAST

#programming #python

Kurs: D0009E

Grundläggande typer i Python

Typ	Exempel	Kommentar
<code>int</code>	<code>42</code>	Heltal
<code>float</code>	<code>3.14</code>	Flyttal
<code>bool</code>	<code>True</code> , <code>False</code>	Boolesk
<code>str</code>	<code>"hej"</code>	Sträng
<code>list</code>	<code>[1, 2, 3]</code>	Föränderlig sekvens
<code>tuple</code>	<code>(1, 2)</code>	Oföränderlig sekvens
<code>dict</code>	<code>{"a": 1}</code>	Nyckel-värde
<code>set</code>	<code>{1, 2, 3}</code>	Mängd
<code>NoneType</code>	<code>None</code>	Saknar värde

Kontrollera typ

```
1 type(x)
2 isinstance(x, int)
```

Konvertering

```
1 int("5"), float("3.14"), str(42)
```

Se även

- [Variabler](#)
 - [Strängar](#)
 - [Listor](#)
-

Aritmetik

19 okt. 2026, 1 min läsning

UTKAST

#programming #python

Kurs: D0009E

Operatorer

Operator	Betydelse	Exempel
+	addition	3+2
-	subtraktion	3-2
*	multiplikation	3*2
/	division (float)	7/2 → 3.5
//	heltalsdivision	7//2 → 3
%	rest (modulo)	7%2 → 1
**	potens	2**10 → 1024

Ordningsregler

Som i matematiken: ****** först, sedan *****, **/**, **//**, **%**, sist **+**, **-**. Använd parenteser vid tveksamhet.

Se även

- Datatyper
- Logiska uttryck

Logiska uttryck

19 okt. 2026, 1 min läsning

UTKAST

#programming #python

Kurs: D0009E

Jämförelseoperatorer

`==` `!=` `<` `>` `<=` `>=`

Resultatet är `True` eller `False`.

Logiska operatorer

`and`, `or`, `not`.

```
1 x > 0 and x < 10
2 not (x == 0)
```

Sanningsvärden

I Python är `0`, `""`, `[]`, `{}`, `None` **falskt**. Allt annat är sant.

Kortslutningsutvärdering

`and` returnerar första falska värdet (eller sista); `or` returnerar första sanna.

Se även

- Vägval
- Aritmetik

Vägval

19 okt. 2026, 1 min läsning

UTKAST

#programmering #python #kontrollflöde

Kurs: D0009E

`if/elif/else` låter programmet välja väg baserat på villkor.

```
1 if x > 0:
2     print("positiv")
3 elif x == 0:
4     print("noll")
5 else:
6     print("negativ")
```

Kortform (villkorsuttryck)

```
1 sign = "+" if x > 0 else "-"
```

Se även

- Logiska uttryck
- For-loopar
- While-loopar

While-loopar

19 okt. 2026, 1 min läsning

UTKAST

#programmering #python #kontrollflöde

Kurs: D0009E

`while` upprepar så länge villkoret är sant.

```
1 x = 10
2 while x > 0:
3     print(x)
4     x -= 1
```

Vanliga fel

- Oändlig loop om villkoret aldrig blir falskt.
- Glöm inte uppdatera variablerna i loopen.

Använd `while` när

- antalet iterationer inte är känt i förväg
- du väntar på en händelse / giltig inmatning

Se även

- For-loopar
- Välgval

For-loopar

19 okt. 2026, 1 min läsning

UTKAST

#programmering #python #kontrollflöde

Kurs: D0009E

`for` itererar över ett **iterabelt** objekt (lista, sträng, range, dict, ...).

```
1 for x in [1, 2, 3]:
2     print(x)
3
4 for i in range(10): # 0..9
5     print(i)
6
7 for tecken in "abc":
8     print(tecken)
```

break och continue

- `break` avbryter loopen.
- `continue` hoppar till nästa iteration.

enumerate och zip

```
1 for i, x in enumerate(lista):
2     ...
3
4 for a, b in zip(l1, l2):
5     ...
```

Se även

- While-loopar
- Listor

- Rekursion
-

Funktioner

19 okt. 2026, 6 min läsning

UTKAST

#envariabelanalys #funktioner

Kurs: M0065M Förkunskaper: Inga

1. Grundidé

Idén. En funktion är analysens grundobjekt: ett samband där varje tillåtet indata bestämmer exakt ett utdata — sträckan som funktion av tiden, arean som funktion av radien. Allt vi senare gör i kursen är verktyg för att undersöka sådana samband.

Definition

En funktion $f : A \rightarrow B$ tilldelar varje $x \in A$ exakt ett värde $f(x) \in B$.

Det viktiga är att varje tillåtet x har **ett och endast ett** funktionsvärde.

$$f : A \rightarrow B, \quad x \mapsto f(x)$$

2. Centrala begrepp

Begrepp	Betydelse
Definitionsmängd D_f	Alla tillåtna x
Värdemängd V_f	Alla värden funktionen faktiskt antar, $V_f = \{f(x) : x \in D_f\}$
Målmängd	Den mängd funktionen sägs avbilda till
Funktionsvärde $f(a)$	Värdet när $x = a$

 Note

Värdemängden behöver inte vara samma sak som målmängden. Är de lika, $V_f = B$, säger vi att f är **surjektiv**.

Exempel: För $f(x) = x^2$ med $f : \mathbb{R} \rightarrow \mathbb{R}$ är målmängden $\mathbb{R}$, men värdemängden är $[0, \infty)$. Funktionen $f(x) = x^3$ med $f : \mathbb{R} \rightarrow \mathbb{R}$ är däremot surjektiv.

 Underförstådd definitions mängd

Står definitions mängden inte uttryckligen utskriven antar vi att den är **den största mängd där uttrycket är definierat**. Två funktioner f och g är **lika** om $D_f = D_g$ och $f(x) = g(x)$ för alla $x \in D_f$.

Därför är $f(x) = x + 1$ och $g(x) = \frac{x^2 - 1}{x - 1}$ **inte** lika: de har samma värden överallt utom i $x = 1$, där g inte är definierad.

 Bestäm definitions mängden

För $f(x) = \frac{1}{x - 2}$ måste nämnaren vara skild från noll.

$$D_f = \mathbb{R} \setminus \{2\}$$

 Bestäm definitions mängd och värdemängd

För $f(x) = \sqrt{4 - x^2}$ krävs $4 - x^2 \geq 0$, alltså $-2 \leq x \leq 2$.

$$D_f = [-2, 2], \quad V_f = [0, 2]$$

 Exempel – Definitions- och värdemängd för $g(x) = \frac{1}{1 - \sqrt{x - 2}}$

Vi bestämmer definitions mängd och värdemängd för

$$g(x) = \frac{1}{1 - \sqrt{x - 2}}.$$

 Lösning

Definitionsmängd. För $x < 2$ är talet under rottecknet negativt. För $x = 3$ är nämnaren $1 - \sqrt{1} = 0$. Alltså

$$D_g = [2, 3) \cup (3, \infty)$$

Värdemängd. Vi följer nämnaren $1 - \sqrt{x - 2}$:

- $x = 2$ ger $g(2) = 1$. När x växer från 2 mot 3 minskar nämnaren från 1 mot 0 men är positiv, så $g(x)$ växer från 1 mot $+\infty$.
- För $x > 3$ är nämnaren negativ. När $x \rightarrow 3^+$ går nämnaren mot 0 och $g(x) \rightarrow -\infty$; när $x \rightarrow +\infty$ går nämnaren mot $-\infty$ och $g(x) \rightarrow 0$ underifrån.

$$V_g = (-\infty, 0) \cup [1, \infty)$$

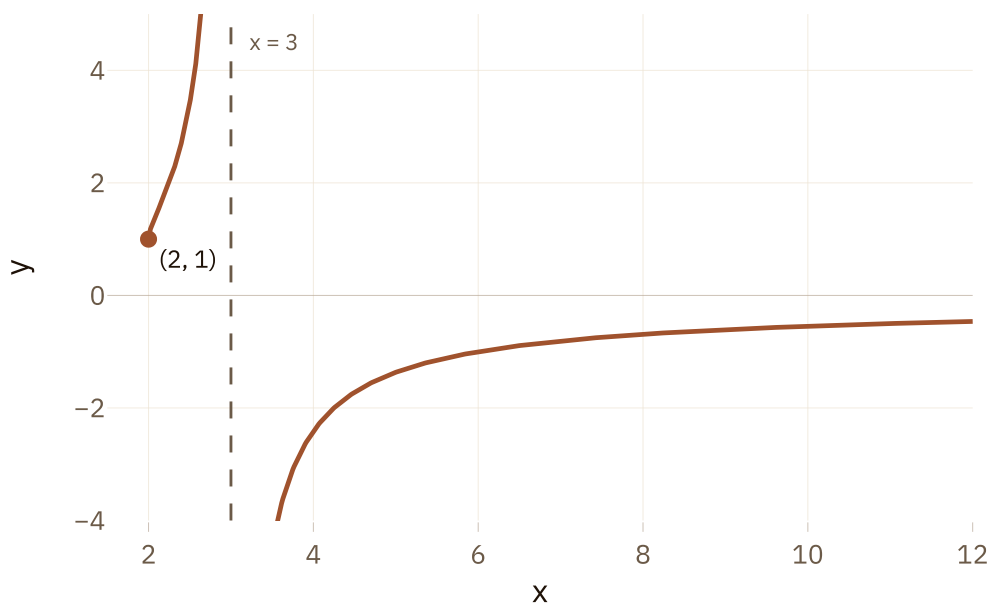
Stämmer med det handskrivna lösningsförslaget (föreläsning 6).

3. Grafen till en funktion

Grafen består av alla punkter $(x, f(x))$ där $x \in D_f$.

$$\text{graf}(f) = \{(x, f(x)) : x \in D_f\}$$

Grafen till $g(x) = 1/(1 - \sqrt{x - 2})$: definitionsmängden börjar i $x = 2$ och har ett hål i $x = 3$, där grafen har en lodrät asymptot.



Tip

Vertikala linjetestet: en kurva är grafen till en funktion om varje vertikal linje skär kurvan i högst en punkt.

Cirkeln $x^2 + y^2 = 4$ klarar inte testet: att lösa ut y ger $y = \pm\sqrt{4 - x^2}$, alltså två värden för varje $x \in (-2, 2)$.

4. Sammansättning av funktioner

Om f och g är funktioner definieras sammansättningen som

$$(f \circ g)(x) = f(g(x))$$

Här appliceras g först och sedan f – en sammansatt funktion evalueras alltid inifrån och ut.

Definitionsmängden blir

$$D_{f \circ g} = \{x \in D_g : g(x) \in D_f\}.$$

Exempel på sammansättning

Låt $f(x) = x^2$ och $g(x) = \sin x$.

$$(f \circ g)(x) = \sin^2 x, \quad (g \circ f)(x) = \sin(x^2)$$

Alltså är sammansättning i allmänhet **inte kommutativ**.

≡ Exempel – Fyra sammansättningar och deras definitionsmängder

Låt $f(x) = \frac{2}{x}$ och $g(x) = \frac{x}{1-x}$. Vi bildar $f \circ f, f \circ g, g \circ f$ och $g \circ g$ och anger definitionsmängderna.

Lösning

Här är $D_f = \mathbb{R} \setminus \{0\}$ och $D_g = \mathbb{R} \setminus \{1\}$. För varje sammansättning utesluter vi dels de x där den inre funktionen inte är definierad, dels de x där den inre funktionen hamnar utanför den yttre funktionens definitionsmängd.

a) $(f \circ f)(x) = f\left(\frac{2}{x}\right) = \frac{2}{2/x} = x$, med $D_{f \circ f} = \mathbb{R} \setminus \{0\}$. Trots att uttrycket x är definierat överallt ärver vi $x \neq 0$ från den inre funktionen.

b) $(f \circ g)(x) = f\left(\frac{x}{1-x}\right) = \frac{2(1-x)}{x}$, med $D_{f \circ g} = \mathbb{R} \setminus \{0, 1\}$: $x = 1$ utesluts av den inre funktionen och $x = 0$ eftersom $g(0) = 0 \notin D_f$.

c) $(g \circ f)(x) = g\left(\frac{2}{x}\right) = \frac{2/x}{1-2/x} = \frac{2}{x-2}$, med $D_{g \circ f} = \mathbb{R} \setminus \{0, 2\}$.

d) $(g \circ g)(x) = \frac{x/(1-x)}{1-x/(1-x)} = \frac{x}{1-2x}$, med $D_{g \circ g} = \mathbb{R} \setminus \{\frac{1}{2}, 1\}$.

Stämmer med det handskrivna lösningsförslaget (föreläsning 6).

5. Varför funktioner är centrala i analys

I envariabelanalys studerar vi hur funktioner beter sig:

- nära en punkt → **Gränsvärden**
- utan hopp → **Kontinuitet**
- hur snabbt de förändras → **Derivata**
- hur deras areor och ackumulation beskrivs → **Integraler**

⚠ Vanliga fel

- **Att förenkla bort en definitionslucka.** $\frac{x^2 - 1}{x - 1} = x + 1$ gäller bara för $x \neq 1$. Samma sak i sammansättningar: $(f \circ f)(x) = x$ ovan är inte definierad i $x = 0$.
- **Att glömma villkoret** $g(x) \in D_f \cap D_{f \circ g}$. Det räcker inte att titta på det förenklade slututtrycket.
- **Fel ordning i** $f \circ g$. g verkar först; $(f \circ g)(x)$ och $(g \circ f)(x)$ är oftast olika.
- **Rot av en kvot.** $\sqrt{\frac{3-x}{x+2}}$ kräver att *kvoten* är ≥ 0 och att nämnaren är $\neq 0$ — inte att täljare och nämnare var för sig är positiva. Teckentabell ger $(-2, 3]$.

6. Lösta övningar

≡ Övningsuppgift 3.1 — Definierar ekvationen en funktion av x ?

Vi avgör vilka av följande uttryck som definierar y som en funktion av x :

- a) $x + y = 1$ b) $x^2 + y = 1$ c) $x + y^2 = 1$ d) $x^2 + y^2 = 1$ e) $x^2 + y^2 = 0$ f) $x^2 - y^2 = 0$ g) $x^4 + y = 0$
 h) $x^4 - y = 0$ i) $x^2 + 4x + y = 5$ j) $x + 4y + y^2 = 5$ k) $\sqrt{x} + y = 0$ l) $x + \sqrt{y} = 0$ m) $x + y^2 = 1, y \geq 0$.

Lösning

Metod. Vi löser ut y . Får vi ett entydigt y för varje tillåtet x är det en funktion; får vi två värden (ett $\pm$) för något x är det inte en funktion — det är det vertikala linjetestet i algebraisk form.

	Löst för y	Funktion?
a	$y = 1 - x$	ja
b	$y = 1 - x^2$	ja
c	$y = \pm\sqrt{1 - x}$	nej
d	$y = \pm\sqrt{1 - x^2}$	nej
e	enda punkten är $x = 0, y = 0$	ja, med $D = \{0\}$

	Löst för y	Funktion?
f	$y = \pm x$	nej
g	$y = -x^4$	ja
h	$y = x^4$	ja
i	$y = 5 - 4x - x^2$	ja
j	$(y + 2)^2 = 9 - x$, alltså $y = -2 \pm \sqrt{9 - x}$	nej
k	$y = -\sqrt{x}$	ja
l	$\sqrt{y} = -x$ kräver $x \leq 0$, och då $y = x^2$	ja
m	$y = \sqrt{1 - x}$ (villkoret $y \geq 0$ väljer en gren)	ja

Funktion: a, b, e, g, h, i, k, l, m. Ej funktion: c, d, f, j.

Facit: a) Funktion b) Funktion c) Ej funktion [$y = \pm \sqrt{1-x}$] d) Ej funktion [$y = \pm \sqrt{1-x^2}$] e) Funktion [$y = 0, x = 0$] f) Ej funktion [$y = \pm x$] g) Funktion h) Funktion i) Funktion j) Ej funktion [$y = -2 \pm \sqrt{9-x}$] k) Funktion l) Funktion [$y = x^2, x \leq 0$] m) Funktion [$y = \sqrt{1-x}$].

Övningsuppgift 3.2 – Definitionsmängder

Vi bestämmer definitionsmängden för följande funktioner av x :

a) $y = 2 - x$ b) $y = 4x^2$ c) $y = x^2 - 2x - 3$ d) $y = \sqrt{x^2 - 2x - 3}$ e) $y = \frac{3-x}{x+2}$ f) $y = \sqrt{\frac{3-x}{x+2}}$ g) $x^2 + y^2 = 16, y \geq 0$ h) $\frac{y}{x} = 1$.

Lösning

a–c) Polynom är definierade för alla reella x : $D_f = \mathbb{R}$.

d) Vi kräver $x^2 - 2x - 3 = (x + 1)(x - 3) \geq 0$. Parabolen är negativ bara mellan nollställena:

$$D_f = \{x : x \leq -1 \text{ eller } x \geq 3\}$$

e) Nämnaren får inte vara noll: $D_f = \{x : x \neq -2\}$.

f) Vi kräver $\frac{3-x}{x+2} \geq 0$ och $x \neq -2$. Teckentabell:

	$x < -2$	$-2 < x < 3$	$x = 3$	$x > 3$
$3 - x$	+	+	0	-
$x + 2$	-	+	+	+
kvot	-	+	0	-

$$D_f = \{x : -2 < x \leq 3\}$$

g) $y = \sqrt{16 - x^2}$ (villkoret $y \geq 0$ väljer den övre halvcirkeln), som kräver $16 - x^2 \geq 0$:

$$D_f = \{x : -4 \leq x \leq 4\}.$$

h) $y = x$, men bara där ekvationen $y/x = 1$ är meningsfull: $D_f = \{x : x \neq 0\}$. Samma fälla som $\frac{x^2 - 1}{x - 1}$ ovan.

Facit: a) $D_f = \mathbb{R}$ b) $D_f = \mathbb{R}$ c) $D_f = \mathbb{R}$ d) $D_f = \{x : (x \leq -1) \vee (x \geq 3)\}$ e) $D_f = \{x : x \neq -2\}$ f) $D_f = \{x : -2 < x \leq 3\}$ g) $D_f = \{x : -4 \leq x \leq 4\}$ h) $D_f = \{x : x \neq 0\}$.

Läsning

- [P.4 Functions and Their Graphs](#)
- [P.5 Combining Functions](#)

Se även

- [Funktioner och invers](#)
- [Gränsvärden](#)
- [Kontinuitet](#)
- [Derivata](#)
- [Integraler](#)

Resurser

- [Khan Academy: What is a function?](#)
- [Wikipedia: Function \(mathematics\)](#)

Parametrar

19 okt. 2026, 1 min läsning

UTKAST

#programming #python #funktion

Kurs: D0009E

En funktion deklarerar med parametrar och anropas med argument.

```
1 def greet(name, greeting="Hej"):
2     print(f"{greeting} {name}")
3
4 greet("Pelle")           # positionellt
5 greet(name="Pelle")      # nyckelord
6 greet("Pelle", greeting="Tja") # blandat
```

Defaultvärden

Skrivs `param=värde`. Muterbara default (`[]`, `{}`) **undviks** — använd `None` och sätt inuti funktionen.

`*args` och `**kwargs`

```
1 def f(*args, **kwargs):
2     ...
```

`args` blir en tuple, `kwargs` en dict.

Se även

- Returvärden
- Scope
- Metoder

Returvärden

19 okt. 2026, 1 min läsning

UTKAST

#programming #python #funktion

Kurs: D0009E

`return` skickar tillbaka ett värde från en funktion.

```
1 def square(x):  
2     return x * x  
3  
4 y = square(5) # 25
```

Utan return

En funktion utan `return` (eller `return` utan värde) returnerar `None`.

Flera värden

Pythons funktioner returnerar en tuple:

```
1 def divmod_(a, b):  
2     return a // b, a % b  
3  
4 q, r = divmod_(7, 2)
```

Se även

- Parametrar
- Tupler

Rekursion

19 okt. 2026, 1 min läsning

UTKAST

#programming #python #algorithm

Kurs: D0009E

En funktion som anropar sig själv. Används när problemet har en enklare delproblem-struktur.

```
1 def faktorial(n):  
2     if n <= 1:  
3         return 1  
4     return n * faktorial(n - 1)
```

Två delar

1. **Basfall** — enklaste fall som returnerar direkt.
2. **Rekursivt steg** — anrop på ett “mindre” problem.

Warning

Glöm inte basfallet → oändlig rekursion → `RecursionError`.

Se även

- Matematisk induktion
- For-loopar

Resurser

- [3Blue1Brown: Towers of Hanoi and recursion](#) 

Scope

19 okt. 2026, 1 min läsning

UTKAST

#programming #python

Kurs: D0009E

Scope (räckvidd) bestämmer **var** en variabel är synlig.

LEGB-regeln

Python slår upp namn i ordningen:

1. **Local** — funktionens egna namn
2. **Enclosing** — omgivande funktion
3. **Global** — modulnivå
4. **Built-in** — `print`, `len`, ...

```
1 x = 1
2 def f():
3     x = 2          # lokal
4     print(x)       # 2
5 f()
6 print(x)           # 1
```

`global` och `nonlocal`

Används för att skriva till en yttre variabel inifrån en funktion (bör användas sparsamt).

Se även

- Variabler
- Parametrar

Strängar

19 okt. 2026, 1 min läsning

UTKAST

#programming #python #datastruktur

Kurs: D0009E

Oföränderlig sekvens av tecken.

```
1 s = "Hej världen"
2 s[0]          # "H"
3 s[-1]         # "n"
4 s.upper()
5 s.split()     # ["Hej", "världen"]
6 len(s)
7 "hej" in s
```

f-strängar

```
1 name = "Pelle"
2 print(f"Hej {name}, 2+2 = {2+2}")
```

Se även

- Listor
- Felhantering

Listor

19 okt. 2026, 1 min läsning

UTKAST

#programming #python #datastruktur

Kurs: D0009E

Ordnad, **föränderlig** sekvens.

```
1 xs = [1, 2, 3]
2 xs[0]      # 1
3 xs[-1]     # 3
4 xs[1:3]    # [2, 3]
5 xs.append(4)
6 xs.remove(2)
7 len(xs)
```

Listomfattning (list comprehension)

```
1 kvadrater = [x*x for x in range(10)]
2 jamna = [x for x in xs if x % 2 == 0]
```

Se även

- Tupler
- Strängar
- For-loopar

Tupler

19 okt. 2026, 1 min läsning

UTKAST

#programming #python #datastruktur

Kurs: D0009E

Ordnad, **oföränderlig** sekvens.

```
1 p = (3, 4)
2 x, y = p      # tuple unpacking
3 p[0]          # 3
```

När använder man tuple vs list?

- **Tuple:** data som hör ihop som en enhet (koordinater, returvärden).
- **List:** växande/ändras, homogent innehåll.

Se även

- Listor
- Returvärden

Kataloger

19 okt. 2026, 1 min läsning

UTKAST

#programming #python #io

Kurs: D0009E

Kataloger (mappar) manipuleras via standardmodulerna `os` och `pathlib`.

`pathlib` (modernt)

```
1 from pathlib import Path
2
3 p = Path("data")
4 p.mkdir(exist_ok=True)
5
6 for fil in p.iterdir():
7     print(fil.name)
8
9 for txt in p.glob("*.txt"):
10     print(txt)
```

`os` (traditionellt)

```
1 import os
2 os.listdir(".")
3 os.makedirs("ny/mapp", exist_ok=True)
4 os.path.exists("data.txt")
```

Se även

- Filer
- Felhantering

Filer

19 okt. 2026, 1 min läsning

UTKAST

#programming #python #io

Kurs: D0009E

Öppna och läsa

```
1 with open("data.txt", "r", encoding="utf-8") as f:
2     text = f.read()
3     # eller rad för rad:
4     # for rad in f:
5     #     ...
```

Skriva

```
1 with open("ut.txt", "w", encoding="utf-8") as f:
2     f.write("hej\n")
```

Öppningslägen

Läge	Betydelse
r	läsa
w	skriva (sriver över)
a	lägga till
r+	läs + skriv
b	binärt (t.ex. rb)



Använd alltid `with` — filen stängs automatiskt även vid fel.

Se även

- [Kataloger](#)
 - [Felhantering](#)
-

Felhantering

19 okt. 2026, 1 min läsning

UTKAST

#programming #python

Kurs: D0009E

Python signalerar fel med **undantag** (exceptions). Fångas med `try / except`.

```
1 try:
2     x = int(input("Tal: "))
3 except ValueError:
4     print("Inte ett heltal")
5 except Exception as e:
6     print("Annat fel:", e)
7 finally:
8     print("Alltid")
```

Kasta själv

```
1 if x < 0:
2     raise ValueError("x måste vara ickenegativ")
```

Vanliga undantag

`ValueError`, `TypeError`, `KeyError`, `IndexError`, `FileNotFoundError`, `ZeroDivisionError`.

Se även

- Filer
- Kataloger

Objektorientering

19 okt. 2026, 1 min läsning

UTKAST

#programmering #python #objektorientering

Kurs: D0009E

Paradigm där data och beteende buntas ihop i **objekt** skapade från **klasser**.

Kärnkoncept

- **Inkapsling** — data och metoder hör ihop; detaljer döljs.
- **Arv** — en klass ärver från en annan.
- **Polymorfism** — samma metodnamn, olika beteende beroende på typ.
- **Abstraktion** — exponera vad, dölj hur.

Varför?

- Återanvändning genom arv.
- Lättare att modellera domäner (bil har hjul, spelare har hp, ...).
- Naturlig gruppering för stora program.

Se även

- [Klasser](#)
- [Objekt](#)
- [Metoder](#)
- [Konstruktorer](#)

Klasser

19 okt. 2026, 1 min läsning

UTKAST

#programmering #python #objektorientering

Kurs: D0009E

En **klass** är en mall för att skapa objekt med gemensam struktur och beteende.

```
1 class Cirkel:
2     def __init__(self, r):
3         self.r = r
4     def area(self):
5         return 3.14159 * self.r ** 2
6
7 c = Cirkel(2)
8 c.area()    # 12.566...
```

Arv

```
1 class Enhetscirkel(Cirkel):
2     def __init__(self):
3         super().__init__(1)
```

Se även

- Objekt
- Konstruktorer
- Metoder
- Objektorientering

Objekt

19 okt. 2026, 1 min läsning

UTKAST

#programming #python #objektorientering

Kurs: D0009E

Ett **objekt** är en instans av en klass — en konkret datastruktur med egna attribut.

```
1 c1 = Cirkel(1)
2 c2 = Cirkel(2)
3 c1.r      # 1
4 c2.r      # 2
```

I Python är **allt** ett objekt: tal, strängar, funktioner, moduler.

Identitet vs likhet

- `is` testar om två namn refererar till samma objekt.
- `==` testar värdelikheter (kan definieras i klassen via `__eq__`).

Se även

- Klasser
- Objektorientering

Metoder

19 okt. 2026, 1 min läsning

UTKAST

#programmering #python #objektorientering

Kurs: D0009E

En **metod** är en funktion som hör till ett objekt (eller klass). Anropas med punktnotation:

```
1 s = "hej"
2 s.upper()      # "HEJ"
3 xs = [1, 2]
4 xs.append(3)
```

I en klass

Första parametern är alltid `self` (det aktuella objektet):

```
1 class Cirkel:
2     def __init__(self, r):
3         self.r = r
4     def area(self):
5         return 3.14 * self.r ** 2
```

Se även

- Klasser
- Objekt
- Parametrar

Konstruktörer

19 okt. 2026, 1 min läsning

UTKAST

#programmering #python #objektorientering

Kurs: D0009E

`__init__` är den specialmetod som körs när ett objekt skapas. Den initierar objektets attribut.

```
1 class Punkt:
2     def __init__(self, x, y):
3         self.x = x
4         self.y = y
5
6 p = Punkt(3, 4)  # anropar __init__
```

Arv och `super()`

```
1 class Punkt3D(Punkt):
2     def __init__(self, x, y, z):
3         super().__init__(x, y)
4         self.z = z
```

Se även

- Klasser
- Objekt
- Metoder