

Matematik i Python

2024/2025

Innehåll

Tal

int & float
Basen 2
Decimaltal
Bråk
Avrundningsfel

Att räkna

Prioritering
Division
Rest vid division

Bitwise XOR

Matematik bibliotek

math
Scipy & Numpy

Funktioner

Funktion i matematik
Funktioner i Python
Funktion i en funktion

Övningar

Del A
Del B

Tal

Två sorters tal

Python har två grundläggande sätt att spara ner tal på:

1. Heltal, `int`
2. Flyttal, `float`

Basen 2

Heltal

Datorn jobbar i ettor och nollor — alltså basen 2. Det går fin fint att beskriva alla heltal i bas 2:

$$1_{10} = 1_2$$

$$2_{10} = 10_2$$

$$3_{10} = 11_2$$

$$4_{10} = 100_2$$

Basen 2

Heltal

Datorn jobbar i ettor och nollor — alltså basen 2. Det går fin fint att beskriva alla heltal i bas 2:

$$1_{10} = 1_2$$

$$2_{10} = 10_2$$

$$3_{10} = 11_2$$

$$4_{10} = 100_2$$

$$\underbrace{207}_{2 \cdot 10^2 + 7 \cdot 10^0}_{10} = \underbrace{1100111}_{{2^7 + 2^6 + 2^3 + 2^2 + 2^1 + 2^0}}_2$$

Basen 2

Decimaltal

Precis som heltal beskrivs i bas 10 med $27_{10} = 2 \cdot 10^1 + 7 \cdot 10^0$ så beskrivs decimaltal likadant:

$$0.1_{10} = \frac{1}{10^1} = 10^{-1}$$

$$0.27_{10} = 2 \cdot \frac{1}{10^1} + 7 \cdot \frac{1}{10^2} = 2 \cdot 10^{-1} + 7 \cdot 10^{-2}$$

Basen 2

Decimaltal

Precis som heltal beskrivs i bas 10 med $27_{10} = 2 \cdot 10^1 + 7 \cdot 10^0$ så beskrivs decimaltal likadant:

$$0.1_{10} = \frac{1}{10^1} = 10^{-1}$$

$$0.27_{10} = 2 \cdot \frac{1}{10^1} + 7 \cdot \frac{1}{10^2} = 2 \cdot 10^{-1} + 7 \cdot 10^{-2}$$

I bas 2:

$$0.1_2 = 2^{-1} = \frac{1}{2}$$

$$0.101_2 = 2^{-1} + 2^{-3} = \frac{1}{2} + \frac{1}{2^3} = \frac{5}{8} = 0.625_{10}$$

Problematik

Bråkiga bråk

Bas 10 klarar inte av att representera alla bråk vi kan föreställa oss. Ett typexempel är:

$$\frac{1}{3} = 0.33333333...$$

Problematik

Bråkiga bråk

Bas 10 klarar inte av att representera alla bråk vi kan föreställa oss. Ett typexempel är:

$$\frac{1}{3} = 0.33333333...$$

I bas 3 är tredjedelar en baggis:

$$\frac{1}{3} = 0.1_3, \frac{5}{3} = 1.2_3$$

Problematik

Bråkiga bråk

Bas 10 har problem med bråk som inte innehåller nämnarna 2 eller 5.

På samma sätt har bas 2 problem med alla bråk som inte har nämnaren 2. Till exempel går det inte att beskriva $0.1_{10} = \frac{1}{10}$ som ett bråk i bas 2. Skulle vi försöka skulle det bli något så här:

$$0.1_{10} = \frac{1}{2^4} + \frac{1}{2^5} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{1}{2^{12}} \dots = 0.000110011001_2 \dots$$

Problematik

Talet 0.1_{10}

Detta kan leda till en del avrundningsfel.

```
1 tal = 0
2 a = 0
3 while a < 10:
4     tal += 0.1
5     a += 1
6 print(tal)
```

```
1 0.9999999999999999
```

Men det borde ha blivit 1.

Innehåll

Tal

- int & float
- Basen 2
- Decimaltal
- Bråk
- Avrundningsfel

Att räkna

- Prioritering
- Division
- Rest vid division

Bitwise XOR

Matematik bibliotek

- math
- Scipy & Numpy

Funktioner

- Funktion i matematik
- Funktioner i Python
- Funktion i en funktion

Övningar

- Del A
- Del B

Matematiska beräkningar

Det fyra räknesätten

När vi jobbar med `int` och `float` vill vi ibland använda oss utav vanlig matte:

Addition:	+
Subtraktion:	-
Multiplikation:	*
Division:	/

De här kommandona följer de vanliga prioriteringsreglerna:

1. Paranteser
2. Potenser
3. Multiplikation & division
4. Addition & subtraktion

Matematiska beräkningar

De fyra räknesätten

```
1 >>> 6+2
2 8
3 >>> 6-2
4 4
5 >>> 6*2
6 12
7 >>> 6/2
8 3
```

Matematiska beräkningar

Prioritering

```
1 >>> 5*(3+2) - 6/3+1  
2 24.0
```

Matematiska beräkningar

Prioritering

```
1 >>> 5*(3+2) - 6/3+1  
2 24.0
```

$$\begin{aligned} & 5 \cdot (3 + 2) - \frac{6}{3} + 1 = \\ & = 5 \cdot 5 - \frac{6}{3} + 1 = \\ & = 25 - 2 + 1 = \\ & = 24 \end{aligned}$$

Matematiska beräkningar

Division

Division i Python genererar som regel en **float**. Vill man undvika detta och tvinga resultatet att bli ett heltal finns det två sätt:

```
1 >>> int(8/3)
2 2
3 >>> 8//3 # Detta kallas för floor-division
4 2
```

Värt att notera är att Python stryker alla decimaler och inte avrundar. Om man vill avrunda uppåt kan man

Matematiska beräkningar

Division

Division i Python generar som regel en **float**. Vill man undvika detta och tvinga resultatet att bli ett heltal finns det två sätt:

```
1 >>> int(8/3)
2 2
3 >>> 8//3 # Detta kallas för floor-division
4 2
```

Värt att notera är att Python stryker alla decimaler och inte avrundar. Om man vill avrunda uppåt kan man lägga till en halv:

```
1 >>> int(8/3+0.5)
2 3
```

Matematiska beräkningar

Rest vid division

I vissa sammanhang är man mest intresserad av resten när man dividerar:

```
1 >>> 4%3
2 1
3 >>> 9%3
4 0
```

Matematiska beräkningar

Potenser

Potenser skrivs i Python med `**`:

```
1 >>> 2**3
2 8
3 >>> 25**(1/2)
4 5.0
5 >>> 2**(-2)
6 0.25
```

Matematiska beräkningar

Potenser

Potenser skrivs i Python med `**`:

```
1 >>> 2**3
2 8
3 >>> 25**(1/2)
4 5.0
5 >>> 2**(-2)
6 0.25
```

Använder man `^` istället för `**` händer följande:

```
1 >>> 2^5
2 7
3 >>> 9^10
4 3
```

Potenser

^-kommandot

Det som händer när man skriver 2^7 är att Python jämför de båda talens binära representation bit för bit.

$$2_{10} = 010_2$$

$$7_{10} = 111_2$$

Potenser

^-kommandot

Det som händer när man skriver 2^7 är att Python jämför de båda talens binära representation bit för bit.

$$2_{10} = 010_2$$

$$7_{10} = 111_2$$

010 De är olika

$$111 \rightarrow 1$$

Potenser

^-kommandot

Det som händer när man skriver 2^7 är att Python jämför de båda talens binära representation bit för bit.

$$2_{10} = 010_2$$

$$7_{10} = 111_2$$

010 De är olika

$$111 \rightarrow 1$$

010 De är lika

$$111 \rightarrow 10$$

Potenser

^-kommandot

Det som händer när man skriver 2^7 är att Python jämför de båda talens binära representation bit för bit.

$$2_{10} = 010_2$$

$$7_{10} = 111_2$$

010 De är olika

$$111 \rightarrow 1$$

010 De är lika

$$111 \rightarrow 10$$

010 De är olika

$$111 \rightarrow 101$$

$$101_2 = 5_{10}$$

Potenser

Bitwise X-OR

```
1 >>> 2^7  
2 5
```

Detta kallas för en "bitwise x-or", (exclusive or). XOR innebär att BARA den ena parten ska vara sann. Ett vardagsexempel är att lampor med flera lampknappar. Vi pratade om detta när vi pratade om `if`-satser. En bitwise operator jämför två objekt, i det här fallet tal, en bit i taget.

Innehåll

Tal

- int & float
- Basen 2
- Decimaltal
- Bråk
- Avrundningsfel

Att räkna

- Prioritering
- Division
- Rest vid division

Bitwise XOR

Matematik bibliotek

- math

- Scipy & Numpy

Funktioner

- Funktion i matematik
- Funktioner i Python
- Funktion i en funktion

Övningar

- Del A
- Del B

Math

Vill man ha lite större matematiska muskler i Python kan man importera biblioteket `Math` som följer med när man installerar Python.

```
1 import math # Laddar biblioteket  
2 print(math.pi) # Skriver ut en approximation på pi
```

```
1 3.141592653589793
```

Saknar man de komplexa talen kan man importera `cmath` istället

Scipy & Numpy

Utöver `math` och `cmath` så är de två biblioteken `scipy` och `numpy` populära. Namnen står för "Scientific Python" och "Numerical Python" och används bland annat på Lunds universitet i matematikforskningen.

- ▶ Numpy: är tänkt att innehålla arrayer och allt som är nödvändigt för dessa
- ▶ Scipy: är tänkt att innehålla matematiska funktioner

Innehåll

Tal

- int & float
- Basen 2
- Decimaltal
- Bråk
- Avrundningsfel

Att räkna

- Prioritering
- Division
- Rest vid division

Bitwise XOR

Matematik bibliotek

- math

- Scipy & Numpy

Funktioner

- Funktion i matematik

- Funktioner i Python

- Funktion i en funktion

Övningar

- Del A

- Del B

Funktioner

Matematisk funktion

$$f(x) = 5x^2 + 3$$

$$g(x) = 2x - 1$$

Funktioner

Matematisk funktion

$$f(x) = 5x^2 + 3$$

$$g(x) = 2x - 1$$

$$f(0) = 5 \cdot 0^2 + 3 = 3$$

Funktioner

Matematisk funktion

$$f(x) = 5x^2 + 3$$

$$g(x) = 2x - 1$$

$$f(0) = 5 \cdot 0^2 + 3 = 3$$

$$g(4) = 2 \cdot 4 - 1 = 8 - 1 = 7$$

Funktioner

Matematisk funktion

$$f(x) = 5x^2 + 3$$

$$g(x) = 2x - 1$$

$$f(0) = 5 \cdot 0^2 + 3 = 3$$

$$g(4) = 2 \cdot 4 - 1 = 8 - 1 = 7$$

$$f(g(x)) = 5 \cdot g(x)^2 + 3 = 5 \cdot (2x - 1)^2 + 3$$

Funktioner

Funktioner i Python

För att skapa en funktion i Python skriver vi:

```
1 def f(x):  
2     return 5*x**2+3  
3  
4 def g(x):  
5     return 2*x-1
```

Funktioner

Använda funktioner

```
1 def dubbla(a):  
2     return 2*a  
3  
4 ålder = input("Hur gammal är du? ")  
5 ålder = int(ålder)  
6  
7 d_ålder = dubbla(ålder)  
8 print(d_ålder, "är dubbelt så gammalt")
```

Derivering

$$g(x) = x^2 + 3x + 4$$

```
1 def g(x):  
2     return x**2+3*x+4  
3  
4 def derivata(f, x):  
5     h = 2**(-10)  
6     return (f(x+h)-f(x))/h
```

Varför skriver jag $h = 2^{-10}$ istället för $h = 0.001$?

Innehåll

Tal

- int & float
- Basen 2
- Decimaltal
- Bråk
- Avrundningsfel

Att räkna

- Prioritering
- Division
- Rest vid division

Bitwise XOR

Matematik bibliotek

- math
- Scipy & Numpy

Funktioner

- Funktion i matematik
- Funktioner i Python
- Funktion i en funktion

Övningar

- Del A
- Del B

Övningar

Del A

Utgå från filen `Matematik.py`

1. Justera programmet så att det tar emot två decimaltal och räknar ut medelvärdet av dem.
2. Utveckla programmet så att det också delar `tal1` med `tal2`.
3. Låt programmet nu ta medelvärdet upphöjt till det delade talet (`tal1/tal2`).
4. Kom ihåg $0.1+0.1\dots$. Vart uppstår det första synliga avrundningsfelet? När blir det rätt igen?
5. Skriv ett program som konverterar ett heltal i bas tio till bas 2.

Övningar

Del B

6. Använd funktionen `omkrets` och skriv kod som tar emot en radie från användaren och skriver ut omkretsen
7. Skapa en funktion som tar emot ett heltal och returnerar resten när man delar med tre.
8. Skapa en funktion som tar emot två tal och skriver ut resten när man delar första talet med det andra talet.