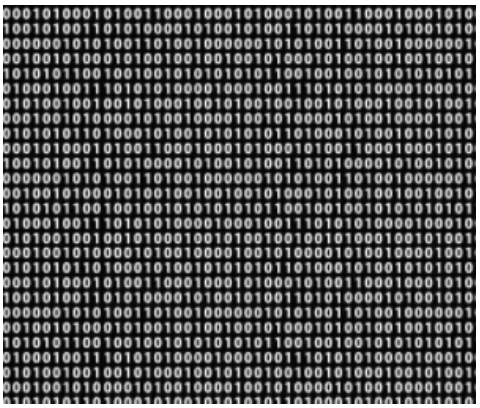
Olika talsystem



Omvandling till och från binära tal heltal i Python

Våra olika talsystem har namn efter det antal symboler eller tecken som används för att representera något numeriskt värde. Det talsystem som du vanligtvis använder kallas *decimal*. I decimal-systemet använder du tio olika symboler: 0, 1, 2, 3, 4, 5, 6, 7, 8 och 9. Med dessa tio symboler kan du representera vilket tal som helst. Andra talsystem är det binära, hexadecimala, och oktala talsystemet.

När du får slut på symboler går du till nästa sifferplacering. För att representera tal större än 9 i decimalsystemet använder du 10 som betyder en enhet av tio och noll enheter av ett. Det är delvis annorlunda i andra talsystem. I ett binärt system använder du bara två symboler: 0 och 1. När du får slut på symboler måste du gå till nästa sifferplacering. Talen 0, 1, 2, 3, 4 och 5 blir då så här i det binära talsystemet: 0, 1, 10, 11, 100 och 101.

Låt oss titta lite närmare på några av talsystemen!

Konvertera till binärt tal

Binära heltal är tal som representeras med bas två. Det betyder att i det binära talsystemet finns det bara två symboler som används för att representera tal: 0 och 1. När du räknar upp från noll binärt får du snabbare slut på symboler eftersom det bara finns två symboler. Du kan inte gå till siffran 2 eftersom 2 inte finns i det binära systemet. Istället använder du en speciell kombination av ettor och nollor. I ett binärt system är exempel det binära talet 1010 lika med 10 i decimal form. I binär form använder du potenser av två. Vi tar talet 10 som exempel. Det kan skrivas så här med potenser av två:

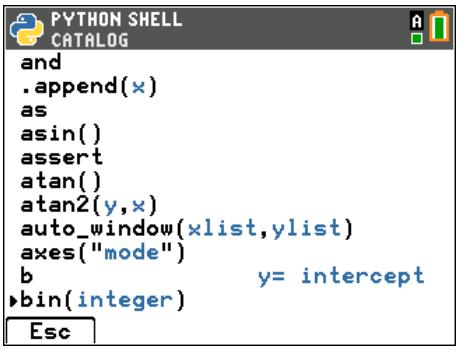
$$1\cdot 2^3+0\cdot 2^2+1\cdot 2^1+0\cdot 2^0=10$$

Nu skriver vi de röda sifforna från vänster och får då 1010.

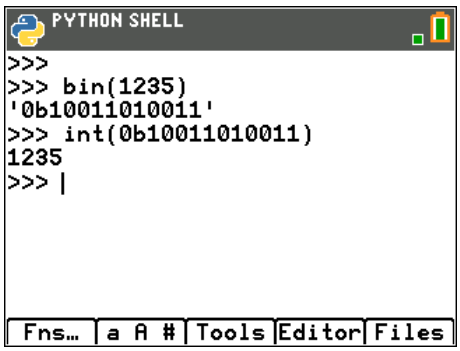
Vi tar ett svårare exempel. Vi väljer det binära talet 1100111

$$1\cdot 2^6+1\cdot 2^5+0\cdot 2^4+0\cdot 2^3+1\cdot 2^2+1\cdot 2^1+1\cdot 2^0=$$
$$64+32+0+0+4+2+1=103$$

I Python kan man använda bin()-funktionen för att konvertera från ett decimalvärde till dess motsvarande binära värde. Tryck på 2nd [catalog] och gå till bin i listan.



Här visar vi nu beräkningarna i ett Shell-fönster. Kom ihåg att skriva 0b, där b står för binär, framför det binära talet när du omvandlar till decimaltal.



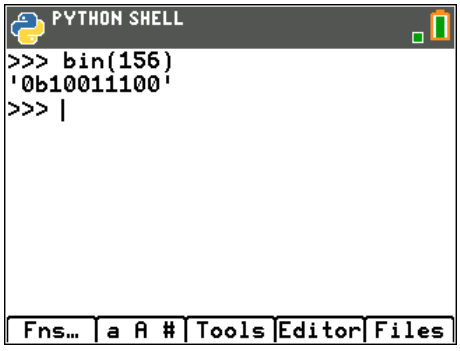
Hur gör man om man inte har tillgång till något hjälpmedel och snabbt vill ta fram den binära representationen. Jo, man dividerar med två och skriver upp resten på en lodrät rad. Låt oss ta ett exempel.

Start med 156	rest
156/2=78	0
78/2	0
39/2	1
19/2	1
9/2	1
4/2	0
2/2	0
1/2	1

Sedan så läser man raden rest *nedifrån och uppåt*. I detta exempel så blir det

10011100

Vi kollar om det stämmer. Vi går in i Shell-fönstret i Pythonappen.

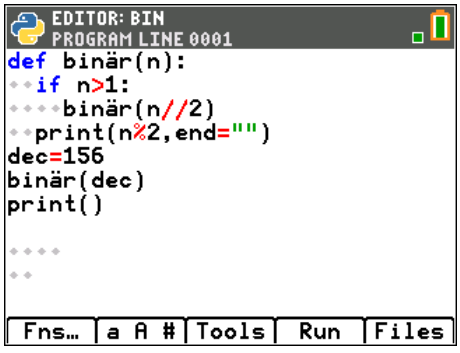


Sammanfattningsvis så skriver man

$156_{10} = 10011100_2$

Nu ska vi gå över till att redovisa ett kort program i Python som gör allt det här. Funktion bin är ju inbyggd i Python men vi ska nu göra vår egen lösning.

Programmet ser du här till höger



Några kommentarer

- n//2 betyder att vi får *heltalsdelen* vid division. 29//2 ger resultatet 14.
- n%2 betyder att vi beräknar *resten* vid division. 29%2 ger resultatet 1.
- end="" gör att siffrorna i det binära talet skrivs ut vågrätt. Pröva att ta bort den instruktionen och titta på resultatet vid körning.
- Programmet är ett bra exempel på *rekursion*. I detta exempel så anropar binär() sig själv. Läs mer om rekursiva funktioner på svenska och andra Python-kurser som finns på webben.

Man skulle också kunna använda upprepade beräkningar (iteration) för att göra samma beräkning. På nästa sida har vi listat ett program som använder while och for-satser

Här är programmet. Här finns alltså både en while- och en for-sats. floor(*n*) betyder det största heltal som inte är större än *n*. floor(3.5) är alltså 3.

```
EDITOR: BIN2
PROGRAM LINE 0001
from math import *
num=int(input("Skriv in ett tal
    : "))
rem=""
while num>=1:
    rem+=str(num%2)
    num=floor(num/2)
binär=""
for i in range(len(rem)-1,-1,-1)
    :
    binär = binär + rem[i]

print("I binär form {0} ".format
    (binär))
Fns… | a A # | Tools | Run | Files
```

Om vi kör detta program blir resultatet så här:

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # Running BIN2
>>> from BIN2 import *
Skriv in ett tal : 156
I binär form 10011100
>>>
Fns… | a A # | Tools | Editor | Files
```

Fördjupning: Diskutera andra talsystem med eleverna, t ex det hexadecimala talsystemet. Det finns **16** hexadecimala symboler. De är desamma som decimalsiffrorna upp till 9 men sedan kommer bokstäverna A, B, C, D, E och F i stället för decimaltalen 10 till 15. En hexadecimal siffra kan alltså representera 16 olika värden istället för 10 som vi är vana vid.
Det hexadecimala talsystemet används ofta av programmerare för att förenkla det binära talsystemet. Eftersom 16 motsvarar 2^4 finns det ett linjärt förhållande mellan siffrorna 2 och 16. Detta innebär att en hexadecimal siffra motsvarar fyra binära siffror. Datorer använder det binära talsystemet medan människor använder det hexadecimala talsystemet för att förkorta binärt och göra det lättare att förstå. Hexadecimala symboler används bland annat för att definiera färger på sidor på nätet. Python har också en inbyggd funktion för att omvandla från det decimala till det hexadecimala talsystemet. 0x betyder att det är hexadecimal visning.

```
PYTHON SHELL
>>> hex(15)
'0xf'
>>>
>>> hex(16)
'0x10'
>>>
>>> hex(32)
'0x20'
>>>
Fns… | a A # | Tools | Editor | Files
```