Skolan för Datavetenskap och kommunikation

PROGRAMMERINGSTEKNIK

FÖRELÄSNING 12

flyttalsberäkningar, simulering, information om provet

MATTE I PYTHON

I modulen modulen math finns bland annat

- trigonometriska funktioner sin, cos etc
- exponentialfunktioner och logaritmer
- konstanter pi och e

Det finns även separata paket, NumPy och SciPy, för avancerade beräkningar och plottning (finns installerade för Python2 i labbsalarna).

```
#Ekvationslösning med Newton-Raphson
from math import *
def f(x):
    return x*sin(x)-log(pi*x)
def fprim(x):
    return sin(x) + x*cos(x) - 1/x
def newton(f,fprim,x):
    for i in range(10):
        x = x - f(x)/fprim(x)
        print(x)
newton(f, fprim, 0.1)
```

HUR LAGRAS FLYTTAL?

$$mantissa * 2^{exponent}$$

53 bitar för mantissan

10 bitar för exponenten

1 bitar för tecken (+ eller -)

STORLEKAR

- Hur många decimaler? 53 bitar i mantissan

  ```
  >>> 2**53
  9007199254740992
  ```

  (ca 16 siffor)

- Hur stora tal? 10 binära siffor i exponenten

  ```
  >>> bin(1023)
  '0b1111111111'
  >>> 2.0**1023
  8.988465674311158e+307
  ```

VAD HÄNDER SEN?

```
start = 2.0**1023
for i in range(10):
    print(start)
    start = start*1.1
```

```
8.988465674311158e+307
9.887312241742738e+307
1.0876043465917013e+308
1.1963647812508716e+308
1.3160012593759595e+308
1.4476013853135549e+308
1.5923615238449106e+308
1.7515976762294019e+308
inf
inf
```

ÖVERRASKNING. VARFÖR?

```
>>> 0.96875-0.640625
0.328125


>>> 0.4-0.3
0.10000000000000003
```

BINÄRA DECIMALER. BINALER?

Algoritm för konvertering

1. Starta med ett decimaltal d < 1
2. Beräkna d = d*2
3. *Heltalsdelen* blir nästa binära siffra
4. *Decimaldelen* blir nya d
5. Upprepa från början, avbryt om d blir 0

```
def binaler(decimaltal):
    s = "0."
    for i in range(50):
        decimaltal *=2
        heltalsdel = int(decimaltal)
        s += str(heltalsdel)
        decimaltal = decimaltal - heltalsdel
        if decimaltal == 0:
            break
    return s
```

OÄNDLIG BINALUTVECKLING

```
>>> binaler(0.5)
'0.1'
>>> binaler(0.4)
'0.01100110011001100110011001100110011001100110011001'
```

JÄMFÖRELSE KNEPIG

```
x = 0
while x != 1:
    x += 0.1
    print(x)
```

```
0.1
0.2
0.30000000000000004
0.4
0.5
0.6
0.7
0.7999999999999999
0.8999999999999999
0.9999999999999999
1.0999999999999999
1.2
1.3
1.4000000000000001
```

KOLLA DIFFERENSEN ISTÄLLET

```
x = 0
while abs(x-1) > 0.01:
    x += 0.1
    print(x)
```

```
0.1
0.2
0.30000000000000004
0.4
0.5
0.6
0.7
0.7999999999999999
0.8999999999999999
0.9999999999999999
>>>
```

SIMULERING

- Skapa en modell
- Skriv ett program som simulerar modellen
- Använd random för slump

TÄRNINGSKAST

```
def kasta():
    return random.randint(1,6)
```

REPRODUCERBARA RESULTAT?

- Använd random.seed för att få samma slumpföljd ur random.randint()
- random.seed(1) ger 2 5 1 3 1...
- random.seed(2) ger 1 1 1 3 2...

osv

MODULEN TURTLE

- `sam = turtle.Turtle()` skapar Turtle objekt
- `sam.forward(20)` går 20 steg
- `sam.right(30)` vrider 30 grader åt höger
- `sam.left(60)` vrider 60 grader åt vänster

```
import turtle
import random

sam = turtle.Turtle()
print("Sam promenerar...")
for i in range(200):
    sam.forward(20)
    if random.random() < 0.5:
        sam.left(30)
    else:
        sam.right(30)
print("Framme!")
```