

Inleiding tot Programmeren

Toon Calders

toon.calders@uantwerpen.be

Les 7b: Datastructuren

Inhoud

- Geheugen van een computer
 - Binaire en hexadecimale getallen
 - Geheugenlocatie
- Getalrepresentaties
 - Integers
 - Floats
- Strings
 - ASCII code
- Variabelen, objecten, referenties
- List data type

Geheugen van een computer

- Grote array
 - Elke cel bevat een byte (=binair getal van 8 bits)
 - Elke cel heeft een adres
- Binaire getallen

byte	getal	byte	getal	byte	getal	byte	getal	...	byte	getal
0	0	10000	16	100000	32	110000	48	...	11110000	240
1	1	10001	17	100001	33	110001	49	...	11110001	241
10	2	10010	18	100010	34	110010	50	...	11110010	242
11	3	10011	19	100011	35	110011	51	...	11110011	243

Byte

- Byte bestaat uit 8 bits
 - 256 mogelijke waarden : 0 tot 255
- Per 4 bits kunnen we dit ook voorstellen met een hexadecimaal getal

decimaal	binair	hexadecimaal
10	1010	A
16	1 0000	10
31	1 1111	1F
255	1111 1111	FF

Notatie

- Decimaal getal: 10
- Binair getal: 0b1010
- Hexadecimaal getal: 0xA

Schrijf volgend getal in binair en hexadecimaal: 513

```
print(bin(513))  
print(hex(513))
```

```
0b1000000001  
0x201
```

Adressen van geheugenplaatsen

- In 64-bits processor: geheugenplaats is een binair nummer van 64 bits.
- Waardes worden bewaard in het geheugen; hun plaats kan beschreven worden met 64 bits

```
class A:  
    pass
```

```
a=A()  
print(a)
```

<__main__.A object at 0x0000028CA9E09400>

Geheugenbeheer

- Een *memory manager* zorgt voor:
 - Toekennen van geheugen (per byte)
 - Vrijgeven van geheugen (per byte)
- Een programma zal nooit enkel geheugen lezen en schrijven dat door de memory manager werd toegekend.
- Bijvoorbeeld: we moeten een boolean bewaren:
 - 1 byte aanvragen bij de memory manager

Getalrepresentaties

- Integer:
 - 1 bit voor het teken
 - lijst van bytes die samen het getal vormen
- Kunnen zo groot worden als nodig
 - Bvb: 2^{1000} : 160 bytes

Getalrepresentatie: floats

- Binaire komma-getallen
 - Kunnen niet alle decimale komma-getallen voorstellen
- Voorbeeld: $1/10 = 0b0.000110011001100...$

```
print(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1)    0.9999999999999999
```

Getalrepresentatie: floats

- Vlottende komma
 - “Normaliseer” alle binaire komma-getallen tot $1. \dots \times 2^e$ *
 - 1 sign bit
 - Exponent
 - Mantisse

* Wat met 0? Zie later.

Exponent

- 11 bits

bitstring	stelt deze exponent voor
0b00000000000	gereserveerd (o.a. om 0 voor te stellen)
0b00000000001	-1022
0b00000000010	-1021
0b00000000011	-1020
0b00000000100	-1019
...	...
0b01111111110	-1
0b01111111111	0
0b10000000000	1
0b10000000001	2
...	...
0b11111111110	1023
0b11111111111	gereserveerd (o.a. om $-\infty$ en ∞ voor te stellen)

Mantisse

- Resterende 52 bits (64 – 1 sign – 11 exponent)
- 1. wordt weggelaten

Bijvoorbeeld (met minder bits):

$$\begin{aligned} 3.65 &= 11.1010011001100 \\ &= \mathbf{1}.11010011001100 \times 2^1 \end{aligned}$$

Sla 11010011001100 op.

Float – volledige voorstelling

getal	sign	exponent	mantisse	voorstelling
2^{-1}	+	-1	0b1.0...0	0 01111111110 000000000...0
$2^{53} - 1$	+	52	0b1.1...1	0 10000110011 111111111...1
-85.0	-	6	0b1.010101	1 10000000101 010101000...0
-3.65	-	1	0b1.11010011001100	1 10000000101 11010011001100...

0, -inf, +inf ?

[illegible]

Pas op! Cancellation!

```
f1=1.0  
f2=1.0 * 2**-53  
som=f1+f2
```

```
1.0 1.1102230246251565e-16 1.0
```

```
print(f1,f2,som)
```

Pas op! Cancellation!

```
L=[1.0]+[1.0 * 2**-53]*100000+[-1.0]  
print(sum(L))
```

```
L=[1.0 * 2**-53]*100000+[1.0,-1.0]  
print(sum(L))
```

0.0

1.1102230246251565e-11