KAIZEN TALK
COMMENT DEVENIR RICHE
À L'AIDE DES MATHÉMATIQUES

KAIZEN

Depositphotos

KAIZEN

KAIZEN

ManuEla79 sur Twitter
Sarah Titus

Dreamstime

KAIZEN

Wikimedia Commons

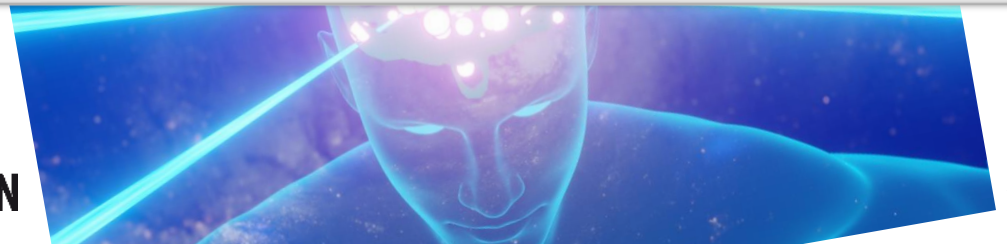
KAIZEN

COMMENT VOLER 1 CENTIME À L'AIDE DES MATHÉMATIQUES

(OU COMMENT ÊTRE PÉDANT DE FAÇON CLASSE)

KAIZEN

ETAT DE L'ART



KAIZEN

Wikimedia Commons

L'ART ~~DU VOL~~ DES MATHS

WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE

KIM ZETTER SECURITY JUN 28, 2010 3:34 PM

FTC: Scammers Stole Millions Using Micro Charges to Credit Cards

A gang of unknown thieves has stolen nearly $10 million using micro charges made to more than a million credit and debit cards in an elaborate multiyear scam, according to a lawsuit filed by the Federal Trade Commission in March. The fraudulent charges went unnoticed by the majority of card owners because they were made [...]

Babelio

[skeptics]

Home

PUBLIC

Questions

Tags

Has a programmer ever embezzled money by shaving fractions of a cent from many bank transactions?

Asked 9 years, 9 months ago Modified 7 years, 9 months ago Viewed 27k times

YouTube

KAIZEN

LE PROBLÈME

couleur™

$$\begin{array}{rr} & 22,67 \\ + & 7,33 \\ = & \square \end{array}$$

$$\begin{array}{rr} & {\scriptstyle 1\ 1\ 1}\ 22,67 \\ + & 7,33 \\ = & 30,00 \end{array}$$

Quel est le résultat ?
(niveau primaire)

KAIZEN

POURTANT C'EST FACILE !

Combien font 1 + 1 ?

- 1 + 1 = 2 (gens normaux)
- 1 + 1 = 11 (concaténation)
- 1 + 1 = 10 (base 2)
- 1 + 1 = 0 (modulo 2)
- 1 + 1 = 1 (algèbre booléenne)
- 1 + 1 = 10,01 (base **φ**-naire)
- 1 + 1 = √2 (distance Manhattan)
- 1 + **𝟙** = syntax error (Unicode)

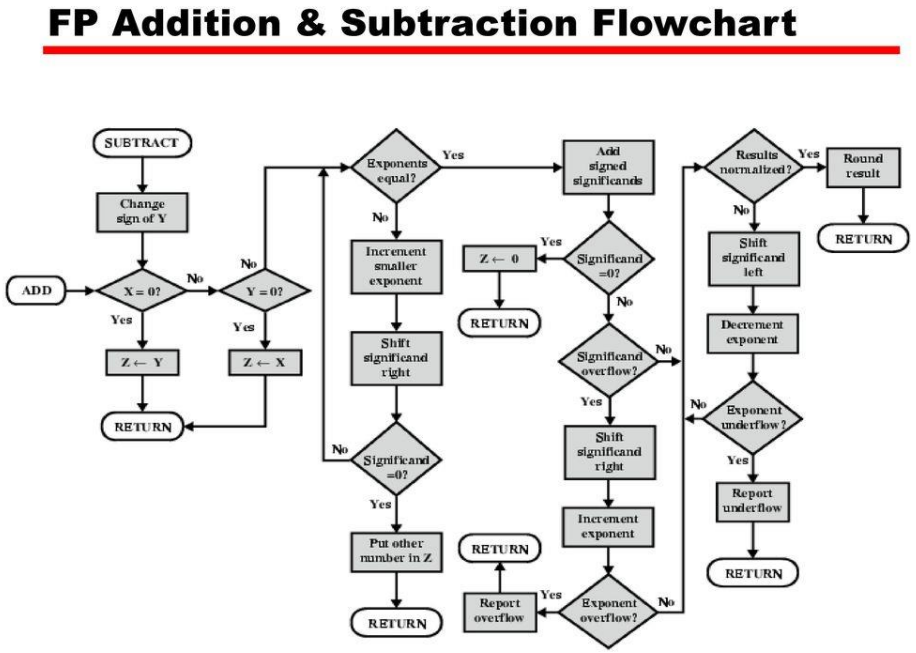
KAIZEN

DOES NOT COMPUTE !!1!

Humain

Dreamstime

Ordinateur



Thomasine Glenn

KAIZEN

ILS SONT PARTOUT !

- Python :

```
>>> print(format(0.1 , ".20f"))
0.10000000000000000555
```

- JS :

```
>> console.log((0.1).toFixed(20))
0.10000000000000000555
```

- C :

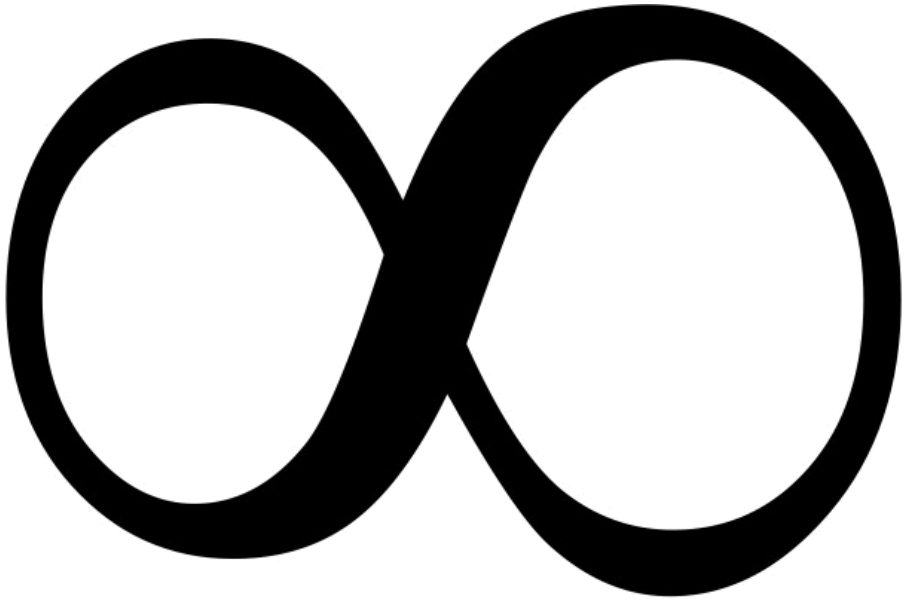
```
int main()
{
    printf("%.10g", (float)0.1);
}
0,1000000015
```

```
x = (
    0.85
    - 0.15  # 0.70
    - 0.30  # 0.40
    - 0.22  # 0.18
    - 0.18  # 0.00
)
print(format(x, ".25f"))
print(f"{(x == 0)!r}")
print(f"{(x > 0)!r}")
```

```
-0.0000000000000000277555756
False
False
```

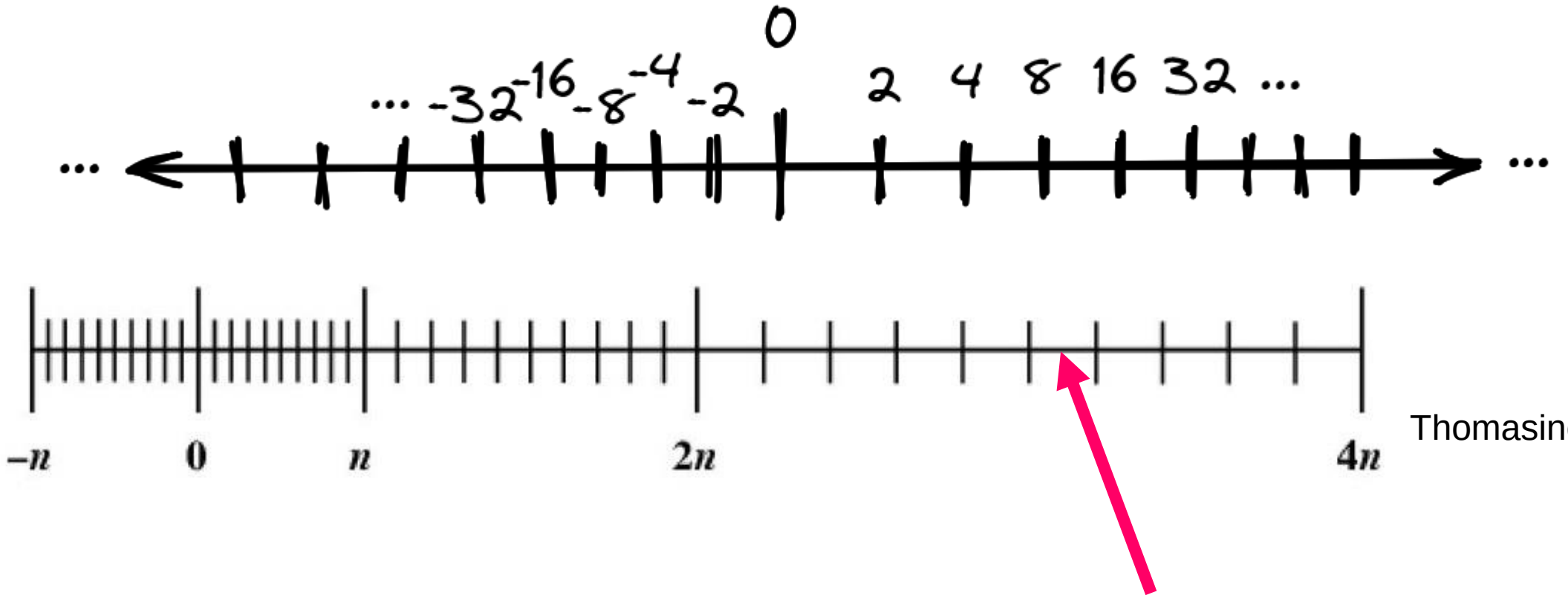
KAIZEN

TOO MANY COOKS



KAIZEN

PARLONS TECHNIQUE (ENFIN!)

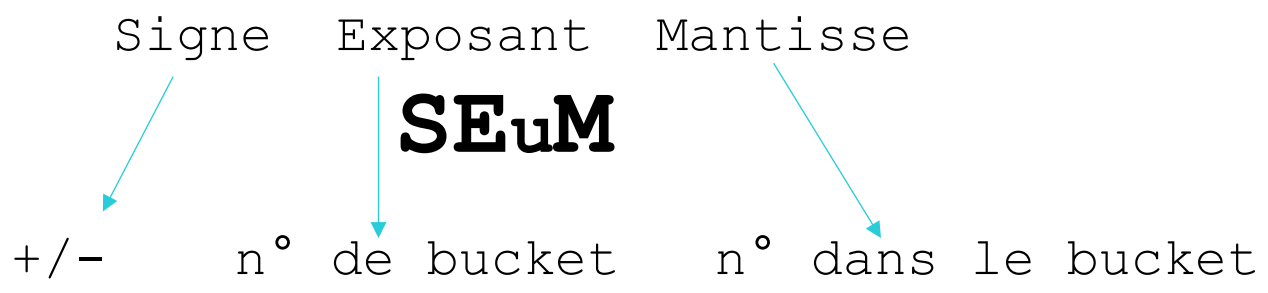


Thomasine Glenn

KAIZEN

IEEE-754 : LE SEUL STANDARD IEEE CONNU ?

```
32 = 1 + 8  + 23 (+1) -> float32
64 = 1 + 11 + 52 (+1) -> float64
```



Maximum :
32: 2^8=256, donc exposants de -126 à 127, donc valeurs absolues jusqu'à ~10^38
64: 2^11=2048, donc exposants de -1022 à 1023, donc valeurs absolues jusqu'à ~10^308
Minimum :
32: 2^-126 ~= 10^-38 64: 2^-1022 ~= 10^-308

KAIZEN

HISTORIQUE RAPIDOS

- Virgule fixe
- 1914 : théorisation pour un calculateur électro-mécanique
- 1940 : premières implèm dans des calculateurs programmables
- 1950 : commercialisation grandissante, formats divergents
- 1985 : standard IEE-754
- 1989 : Prix Turing

KAIZEN

UN FORMAT MIRACLE ?

- Précision limitée, variable selon la magnitude
- Entiers manquants (premiers : 2^24+1 et 2^53+1)
- Sans notion de chiffres significatifs
- Ne peut représenter que des rationels
- « desastrous cancellation »
- printf qui cache la poussière sous le tapis

KAIZEN

QUELLES ALTERNATIVES ?

- Utiliser les libs « Decimal » ?
- Utiliser des entiers pour représenter des centimes ?
- N’utiliser que des float64 ?
- Analyser les calculs ?
- Se résigner ?

KAIZEN

POUR ALLER PLUS LOIN

- Standard IEE-754
  - ulp, fma, fsum
  - epsilon machine
  - subnormals
  - NaN
    - quiet/signalling NaNs
- formats exotiques
  - virgule fixe
- FPU rounding mode, flush-to-zero
  - « round to nearest, ties to even »
- endianness
- analyse numérique
  - « desastrous cancellation »
  - arithmétiques par intervalle
  - Lemme de Sterbenz
- Odoo Decimal
- chiffres significatifs
- Performance
- …

Sources :

- Wikipedia
- doc Oracle
- doc Python
- What Every Computer Scientist Should Know About Floating-Point Arithmetic
- IEE-754
- Yet another tutorial on floating-points
- Floating point converter

KAIZEN

QUIZZZ !!

```rust
fn main() {
    let mut i: f32 = 0.0;
    loop {
        i += 0.1;
        println!("{}", i);

        if i == 1.0 {
            println!("good");
            break;
        }
        if i > 2.0 {
            println!("too high !!");
            break;
        }
    }
}
```

```
0.1
0.2
0.3
0.4
0.5
0.6
0.70000005
0.8000001
0.9000001
1.0000001
1.1000001
1.2000002
1.3000002
1.4000002
1.5000002
1.6000003
1.7000003
1.8000003
1.9000003
2.0000002
too high !!
```

KAIZEN