Nothing really works with computer arithmetic

In [5]:
```
1/7/7/7/7/7/7*7*7*7*7*7*7
```
Out[5]:
```
0.9999999999999998
```

There is an "Integer Division"

But it doesn't really work as a division
Math: $\frac{5}{3} \times 3 = 5$
Computer: $\frac{5}{3} \times 3 = 3$

In [2]:
```
(5//3)*3
```
Out[2]:
```
3
```

Elementary numerical algorithms, esp division

Two examples for integer division (both of course implemented internally)

1. Unsigned integer division
2. Signed integer division

In [3]:
```
# unsigned integer division algorithm
# a,b (a >= 0, b > 0) --> (q,r), 0 <= r < b and a = q*b + r
def intDivU(a,b) :
    assert a >= 0 and b > 0, "illegal args to unsigned division"
    q = 0
    r = a
    while r >= b :
        q += 1
        r -= b
    return q,r
```

In [6]:
```
intDivU(5,3)
```
Out[6]:
```
(1, 2)
```

In [4]:
```
# signed integer division (reduce to unsigned and adjust)
def intDivS(a,b) :
    assert b != 0, "dividing by zero"
    # make a and b both >= 0

    # case 1: b < 0
    if b < 0 :
        q, r = intDivS(a,-b)    # well, I'm sure you can unfold this
recursion SOMEHOW...
        return -q,r
    assert b > 0 # because b != 0

    # case 2: a < 0
    if a < 0 :
        q, r = intDivS(-a,b)
        if r == 0 : return -q,0
        return -q - 1, b - r
    assert a >= 0 and b > 0

    # Now we can use this
    return intDivU(a,b)
```

In [5]:
```
intDivS(26,-7)
```

Out[5]: (-3, 5)

Standard Euclidean Algorithm for the gcd (Greatest Common Divisor)

The Euclidean algorithm is based on the principle that the greatest common divisor of two numbers does not change if the larger number is replaced by its difference with the smaller number.

In [6]:
```
# Euclidean algorithm
# Don't use this if you don't have to.
#   It's built into Python
def gcd(a,b) :
    if a == 0 : return b
    if b == 0 : return a
    return gcd(b%a, a)
```

In [7]:
```
gcd(24,15)
```

Out[7]: 3

Extended Euclidean Algorithm

An extension to the Euclidean algorithm that computes, in addition to the greatest common divisor (gcd) of integers a and b, also the coefficients of Bézout's identity, which are integers x and y such that *ax + by = gcd(a,b)*

In [8]:
```
# Extended Euclidean Algorithm
def xgcd(a,b) :
    prevx, x = 1, 0; prevy, y = 0, 1
    while b:
        q = a//b
        x, prevx = prevx - q*x, x
        y, prevy = prevy - q*y, y
        a, b = b, a % b
    return a, prevx, prevy
```

In [9]:
```
xgcd(12,30)
```

Out[9]:
```
(6, -2, 1)
```