Konrad Gawda

Swiss Python Summit

Float

Everything You Wanted to Know About

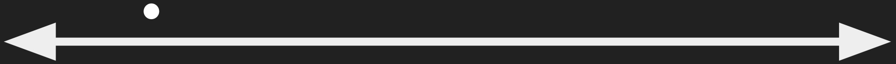
Konrad Gawda

Python - programmer and trainer

Cloud Evangelist

Videocast host

1231231231231231231

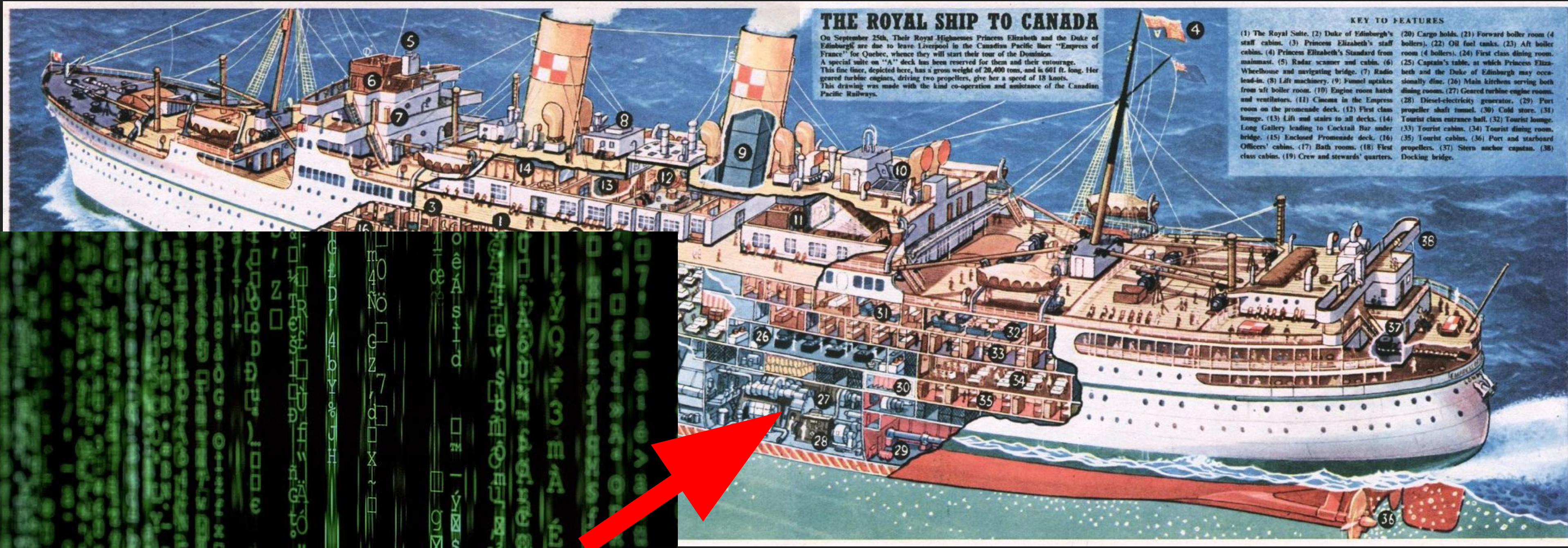


import numbers

$$\begin{tikzcd}
\texttt{numbers.Number} \arrow[d] & & & \\
\texttt{numbers.Complex} \arrow[r] & \texttt{numbers.Real} \arrow[r] \arrow[d] & \texttt{numbers.Rational} \arrow[r] & \texttt{numbers.Integral} \\
 & \texttt{float} & &
\end{tikzcd}$$

```
1.0
1e0

float('1.0')
float(1)

1/1
```

Binary representation



THE ROYAL SHIP TO CANADA

On September 25th, Their Royal Highnesses Princess Elizabeth and the Duke of Edinburgh are due to leave Liverpool in the Canadian Pacific liner "Empress of France" for Quebec, whence they will start their tour of the Dominion.
A special suite on "A" deck has been reserved for them and their entourage.
This fine liner, depicted here, has a gross weight of 20,400 tons, and is 601 ft. long. Her geared turbine engines, driving two propellers, give her a speed of 18 knots.
This drawing was made with the kind co-operation and assistance of the Canadian Pacific Railways.

KEY TO FEATURES

(1) The Royal Suite. (2) Duke of Edinburgh's staff cabins. (3) Princess Elizabeth's staff cabins. (4) Princess Elizabeth's Standard from mainmast. (5) Radar scanner and cabin. (6) Wheelhouse and navigating bridge. (7) Radio lead-in. (8) Lift machinery. (9) Funnel uptakes from aft boiler room. (10) Engine room hatch and ventilators. (11) Cinema in the Empress room on the promenade deck. (12) First class lounge. (13) Lift and stairs to all decks. (14) Long Gallery leading to Cocktail Bar under bridge. (15) Enclosed Promenade deck. (16) Officers' cabins. (17) Bath rooms. (18) First class cabins. (19) Crew and stewards' quarters. (20) Cargo holds. (21) Forward boiler room (4 boilers). (22) Oil fuel tanks. (23) Aft boiler room (4 boilers). (24) First class dining room. (25) Captain's table, at which Princess Elizabeth and the Duke of Edinburgh may occasionally dine. (26) Main kitchens serving both dining rooms. (27) Geared turbine engine rooms. (28) Diesel-electricity generator. (29) Port propeller shaft tunnel. (30) Cold store. (31) Tourist class entrance hall. (32) Tourist lounge. (33) Tourist cabins. (34) Tourist dining room. (35) Tourist cabins. (36) Port and starboard propellers. (37) Stern anchor capstan. (38) Docking bridge.

Almost all platforms map Python floats to IEEE 754 binary64 "double precision" values

https://docs.python.org/3/tutorial/floatingpoint.html

```
typedef struct {
    PyObject_HEAD
    double ob_fval;
} PyFloatObject;
```

https://github.com/python/cpython/blob/main/Include/cpython/floatobject.h

```
import sys, ctypes
f = 1.0

sys.getsizeof(f)
24

ctypes.string_at(id(f), sys.getsizeof(f)).hex()
'030000000000000000599500000000000000000000000f03f'

ctypes.string_at(id(f), sys.getsizeof(f))[-8:].hex()
'000000000000f03f'

bin(int.from_bytes(ctypes.string_at(id(f), sys.getsizeof(f))[-8:]))
'0b1111000000111111'

bin(int.from_bytes(ctypes.string_at(id(f), sys.getsizeof(f))[-8:],\
byteorder=sys.byteorder))
'0b11111111110000000000000000000000000000000000000000000000000000'

bin(int.from_bytes(ctypes.string_at(id(f), sys.getsizeof(f))[-8:],\
byteorder=sys.byteorder)).replace('0b','').zfill(64)
'0011111111110000000000000000000000000000000000000000000000000000'
```

0.625_{10}

$0.a_{16}$

0.101_{2}

$$9\text{e}13 = 9 \times 10 ** 13$$

$$1.625\text{e}6 = 1.625 \times 10 ** 6$$

Rather not: 1625e3, 0.1625e7

$$1.625\text{e-}6 = 1.625 \times 10 ** {-6} = 1.625 \,/\, 10 ** {-6}$$

$$x^{-n} = 1 / x^{n}$$

Time for binary numbers

$$1.101\ p\ 101 = 1.101_2 \times 2 ** 101_2$$

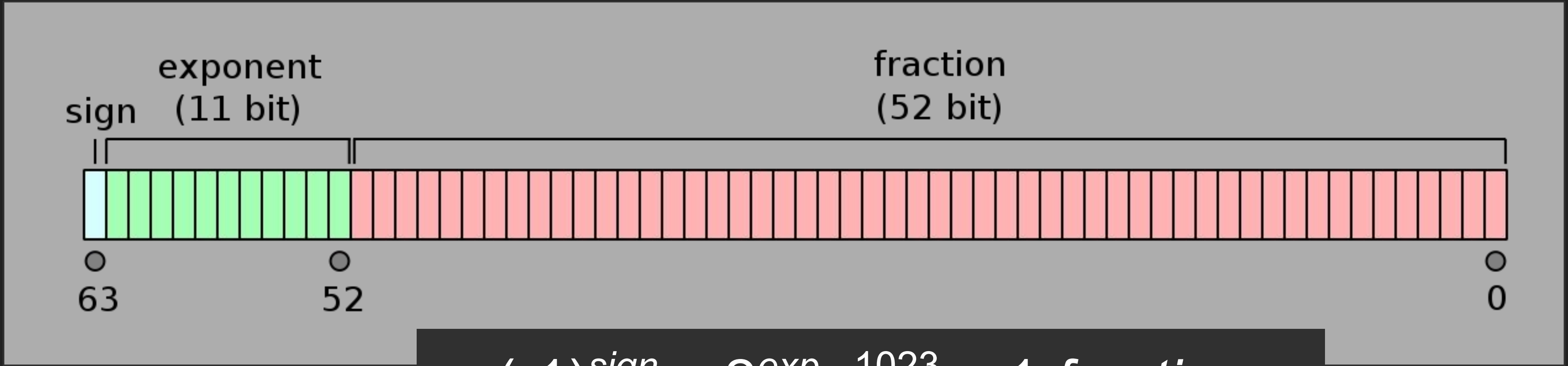
$$1.625 * 2{*}{*}5 = 52$$

$$1.101\ p\ {-101} = 1.101_2 \times 2 ** {-101}_2$$

$$1.101\ \ 11 \rightarrow 1.101_2 \times 2 ** (11-1000)_2$$

$$(\cancel{1}).101\ \ 11 \rightarrow 1.101_2 \times 2 ** (11-1000)_2$$

IEEE 754 binary64 ("double")



$$(-1)^{sign} \times 2^{exp - 1023} \times 1.\mathit{fraction}$$

- Sign bit: 1 bit
- Exponent: 11 bits
- Significand / fraction: 53 bits (52 explicitly stored)

IEEE 754…

William "Velvel" Kahan

A primary architect of the Intel 80x87 floating-point coprocessor and IEEE 754 floating-point standard.

Exponent

$(-1)^{sign} \times 2^{\mathbf{exp} - 1023} \times 1.\mathit{fraction}$

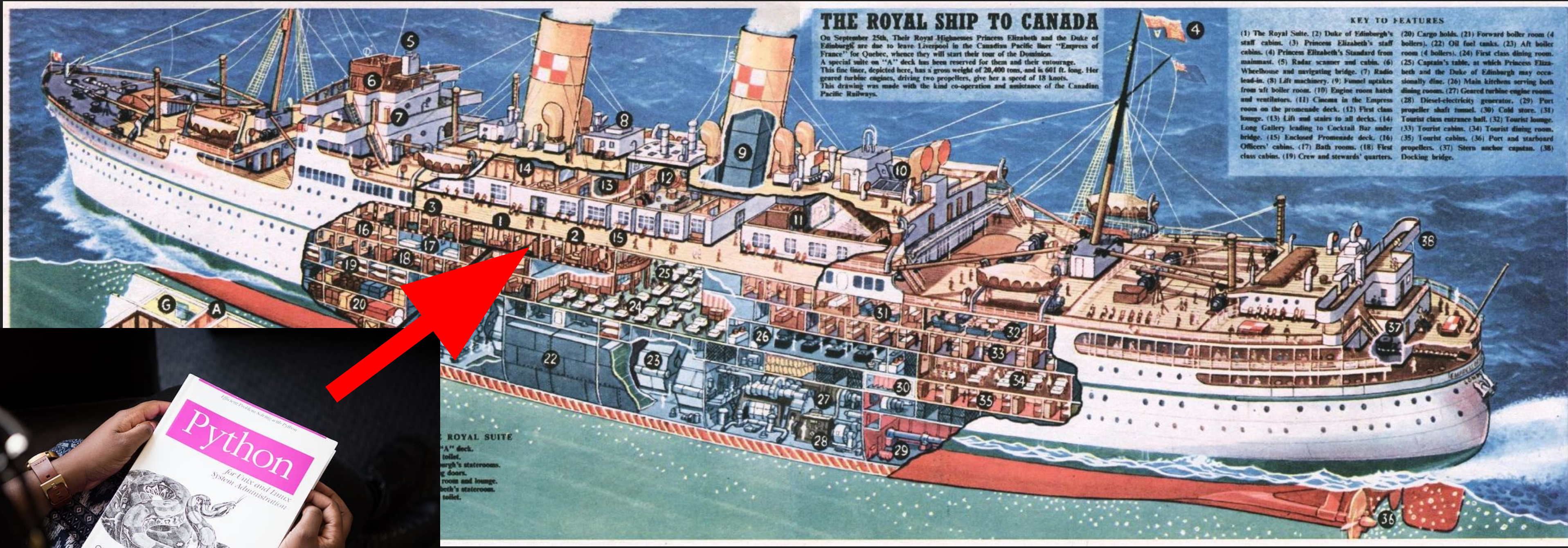
- 000_{16}
  - zero (when fraction == 0)
  - *subnormal (denormalized) numbers*
- 001_{16} - $7fe_{16}$
  - 001 → $2**-2022_{10}$
  - 3ff (1023_{10}) → 2**0
  - 7fe (2046_{10}) → $2*2023_{10}$
- $7ff_{16}$
  - infinity (when fraction == 0)
  - NaN

```
00000000000

00000000001
01111111111
11111111110

11111111111
```

```
0 00000000000 (0).0000000000000000000000000000000000000000000000000000
 ↪️ 0.0
0 01111111111 (1).0000000000000000000000000000000000000000000000000000
 ↪️ 1.0
0 10000000000 (1).0000000000000000000000000000000000000000000000000000
 ↪️ 2.0
0 10000000001 (1).1100000000000000000000000000000000000000000000000000
 ↪️ 7.0
0 10000000100 (1).1111100000000000000000000000000000000000000000000000
 ↪️ 63.0
0 10000001000 (1).1111111110000000000000000000000000000000000000000000
 ↪️ 1023.0
0 10000001001 (1).0000000000000000000000000000000000000000000000000000
 ↪️ 1024.0
```

```
0 01111111111 (1).0000000000000000000000000000000000000000000000000000
 ↪️ 1.0
0 01111111110 (1).0000000000000000000000000000000000000000000000000000
 ↪️ 0.5
0 01111111100 (1).0000000000000000000000000000000000000000000000000000
 ↪️ 0.125
0 01111111011 (1).1001100110011001100110011001100110011001100110011010
 ↪️ 0.1
1 01111111011 (1).1001100110011001100110011001100110011001100110011010
 ↪️ -0.1
```

Programmer's point of view

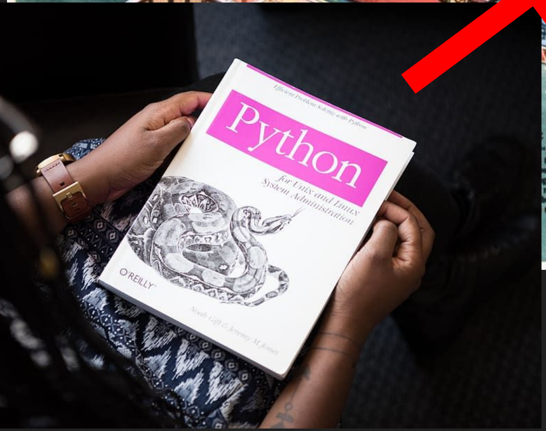


THE ROYAL SHIP TO CANADA

On September 25th, Their Royal Highnesses Princess Elizabeth and the Duke of Edinburgh are due to leave Liverpool in the Canadian Pacific liner "Empress of France" for Quebec, whence they will start their tour of the Dominion.
A special suite on "A" deck has been reserved for them and their entourage.
This fine liner, depicted here, has a gross weight of 20,400 tons, and is 601 ft. long. Her geared turbine engines, driving two propellers, give her a speed of 18 knots.
This drawing was made with the kind co-operation and assistance of the Canadian Pacific Railways.

KEY TO FEATURES

(1) The Royal Suite. (2) Duke of Edinburgh's staff cabins. (3) Princess Elizabeth's staff cabins. (4) Princess Elizabeth's Standard from mainmast. (5) Radar scanner and cabin. (6) Wheelhouse and navigating bridge. (7) Radio lead-in. (8) Lift machinery. (9) Funnel uptakes from aft boiler room. (10) Engine room hatch and ventilators. (11) Cinema in the Empress room on the promenade deck. (12) First class lounge. (13) Lift and stairs to all decks. (14) Long Gallery leading to Cocktail Bar under bridge. (15) Enclosed Promenade deck. (16) Officers' cabins. (17) Bath rooms. (18) First class cabins. (19) Crew and stewards' quarters. (20) Cargo holds. (21) Forward boiler room (4 boilers). (22) Oil fuel tanks. (23) Aft boiler room (4 boilers). (24) First class dining room. (25) Captain's table, at which Princess Elizabeth and the Duke of Edinburgh may occasionally dine. (26) Main kitchens serving both dining rooms. (27) Geared turbine engine rooms. (28) Diesel-electricity generator. (29) Port propeller shaft tunnel. (30) Cold store. (31) Tourist class entrance hall. (32) Tourist lounge. (33) Tourist cabins. (34) Tourist dining room. (35) Tourist cabins. (36) Port and starboard propellers. (37) Stern anchor capstan. (38) Docking bridge.


Efficient Problem-Solving with Python
Python
for Unix and Linux System Administration
O'REILLY
Noah Gift & Jeremy M. Jones

float methods

```
float.is_integer()
```

```
float.as_integer_ratio()
(0.5).as_integer_ratio() → (1, 2)
(0.1).as_integer_ratio() → (3602879701896397, 36028797018963968)
```

So why Python does not print 1/10 as 0.1000000000000000055511151231 ?

Float is by default printed rounded to *machine precision*.

float methods

```
float.hex()
float.fromhex(s)  # classmethod
```

IEEE 754

C's %a format character

Java's Double.toHexString

e.g. `'-0x1.8000000000000p+0'`

```
[sign] ['0x'] integer ['.' fraction] ['p' exponent]
```

*exponent: 2**exp*$_{10}$

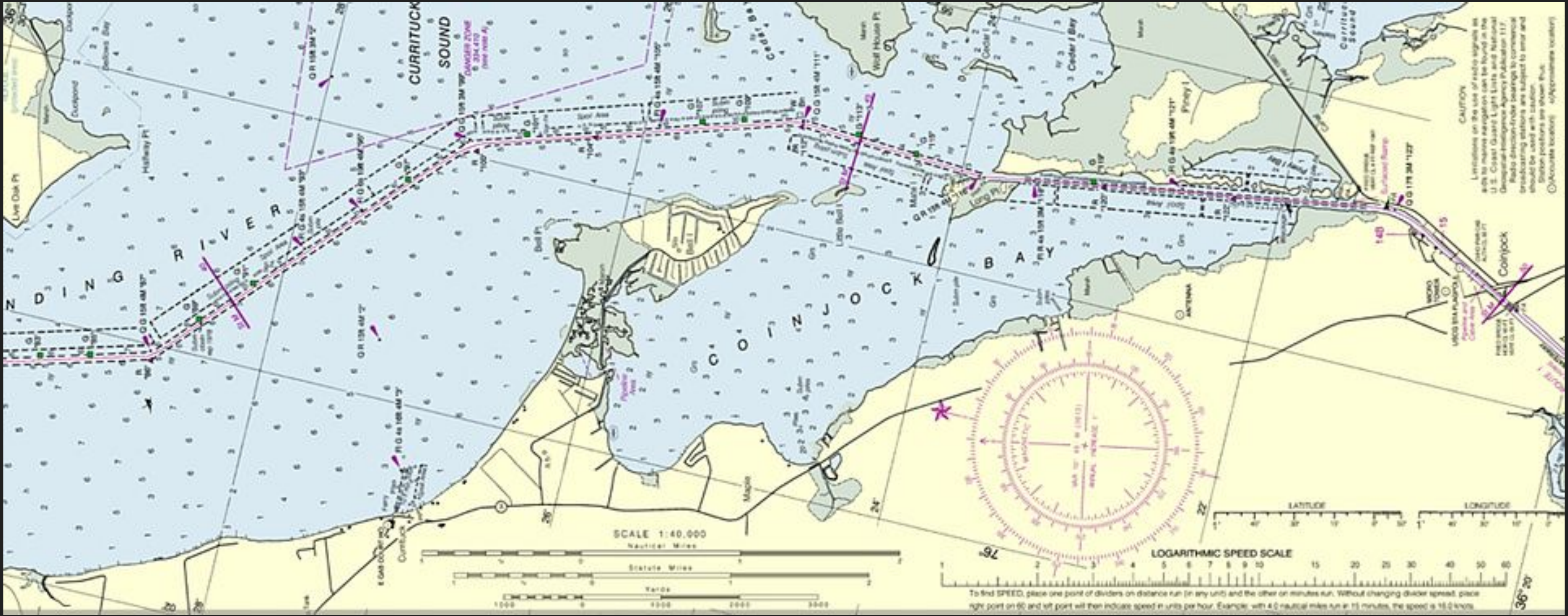
```
'0x0.0p+0'
                0.0
'0x1.0000000000000p+0'
                1.0
'0x1.c000000000000p+2'
                7.0
'0x1.f800000000000p+5'
                63.0
'0x1.ff80000000000p+9'
                1023.0
'0x1.0000000000000p+10'
                1024.0
```

$1c_{16} = 28_{10}$

```
'0x1.0000000000000p-1'
                 0.5
'0x1.0000000000000p-3'
                 0.125
'0x1.999999999999ap-4'
                 0.1
'-0x1.999999999999ap-4'
                 -0.1
```

$9_{16} = 1001_2$

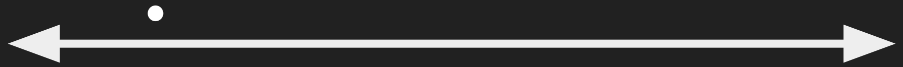
Precision



NOAA, https://oceanservice.noaa.gov/news/mar14/pdf-charts.html sys.float_info

```
sys.float_info(
    max=1.7976931348623157e+308,
    max_exp=1024,
    max_10_exp=308,
    min=2.2250738585072014e-308,
    min_exp=-1021,
    min_10_exp=-307,
    dig=15,
    mant_dig=53,
    epsilon=2.220446049250313e-16,
    radix=2,
    rounds=1
)
```

1231231231231231231



September 2022
CVE-2020-10735: Prevent DoS by large int<->str conversions

`sys.float_info.dig` → 15
`sys.get_int_max_str_digits()` → 4300

`math.ulp(x)` - *Unit in the Last Place* (since Python 3.9)

```python
for x in range(100):
    print(f"ulp(2**{x}) = {math.ulp(2**x)}")
```

```
ulp(2**0) = 2.220446049250313e-16
ulp(2**1) = 4.440892098500626e-16
ulp(2**2) = 8.881784197001252e-16
ulp(2**3) = 1.7763568394002505e-15
ulp(2**4) = 3.552713678800501e-15
ulp(2**5) = 7.105427357601002e-15
ulp(2**6) = 1.4210854715202004e-14
ulp(2**7) = 2.842170943040401e-14
ulp(2**8) = 5.684341886080802e-14
ulp(2**9) = 1.1368683772161603e-13
ulp(2**10) = 2.2737367544323206e-13
ulp(2**11) = 4.547473508864641e-13
ulp(2**12) = 9.094947017729282e-13
ulp(2**13) = 1.8189894035458565e-12
ulp(2**14) = 3.637978807091713e-12
ulp(2**15) = 7.275957614183426e-12
ulp(2**16) = 1.4551915228366852e-11
ulp(2**17) = 2.9103830456733704e-11
ulp(2**18) = 5.820766091346741e-11
ulp(2**19) = 1.1641532182693481e-10
ulp(2**20) = 2.3283064365386963e-10
ulp(2**21) = 4.656612873077393e-10
ulp(2**22) = 9.313225746154785e-10
ulp(2**23) = 1.862645149230957e-09
ulp(2**24) = 3.725290298461914e-09
ulp(2**25) = 7.450580596923828e-09
ulp(2**26) = 1.4901161193847656e-08
ulp(2**27) = 2.9802322387695312e-08
ulp(2**28) = 5.960464477539063e-08
ulp(2**29) = 1.1920928955078125e-07
ulp(2**30) = 2.384185791015625e-07
ulp(2**31) = 4.76837158203125e-07
ulp(2**32) = 9.5367431640625e-07
ulp(2**33) = 1.9073486328125e-06
ulp(2**34) = 3.814697265625e-06
ulp(2**35) = 7.62939453125e-06
ulp(2**36) = 1.52587890625e-05
ulp(2**37) = 3.0517578125e-05
ulp(2**38) = 6.103515625e-05
ulp(2**39) = 0.0001220703125
ulp(2**40) = 0.000244140625
ulp(2**41) = 0.00048828125
ulp(2**42) = 0.0009765625
ulp(2**43) = 0.001953125
ulp(2**44) = 0.00390625
ulp(2**45) = 0.0078125
ulp(2**46) = 0.015625
ulp(2**47) = 0.03125
ulp(2**48) = 0.0625
ulp(2**49) = 0.125
ulp(2**50) = 0.25
ulp(2**51) = 0.5
ulp(2**52) = 1.0
ulp(2**53) = 2.0
ulp(2**54) = 4.0
ulp(2**55) = 8.0
ulp(2**56) = 16.0
ulp(2**57) = 32.0
ulp(2**58) = 64.0
ulp(2**59) = 128.0
ulp(2**60) = 256.0
ulp(2**61) = 512.0
...
```

*Any float x within **2**52 - 2**53** has int precision*

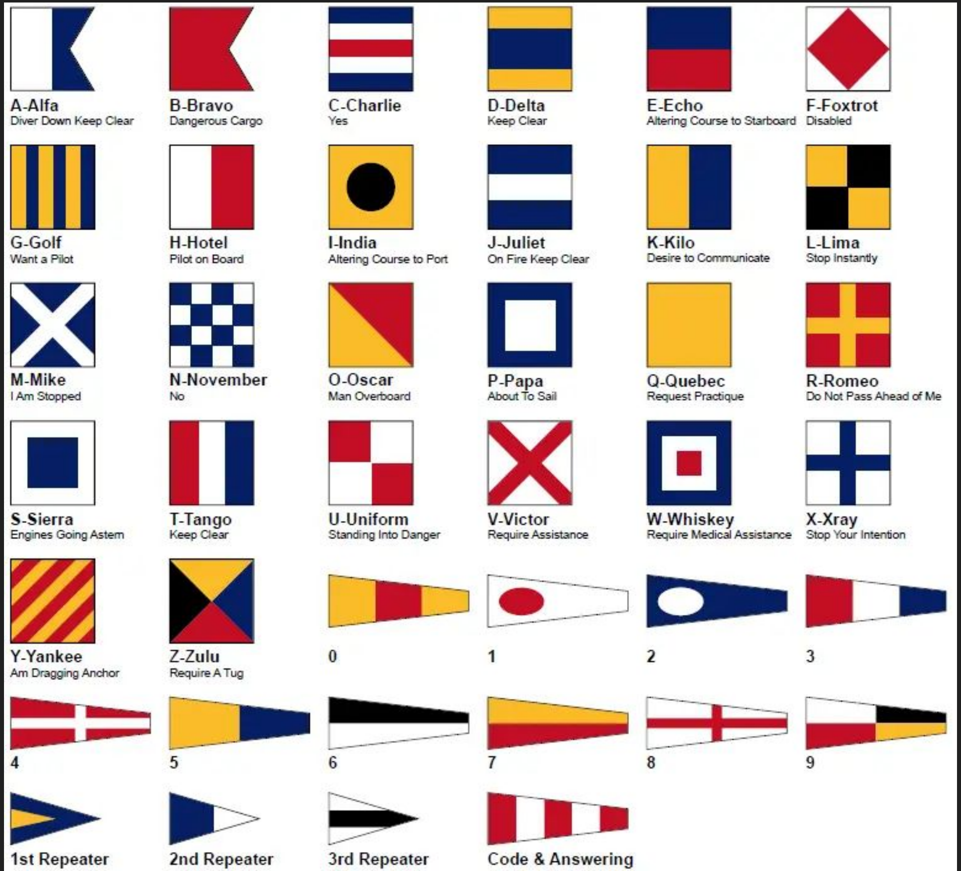
4 503 599 627 370 496 … 9 007 199 254 740 992

4.503600e+15 … 9.007199e+15

0x1.0000000000000p+52 … 0x1.0000000000000p+53

4.5 … 9 × {Ger./Fr./It./Pol. billiard*, Eng. quadrillion, SI: Peta}

Special symbols



Exponent

- `000`$_{16}$
  - zero (when fraction == 0)
  - *subnormal (denormalized) numbers*
- `001`$_{16}$ `-` `7fe`$_{16}$
  - `001` → `2**-2022`$_{10}$
  - `3ff` (1023$_{10}$) → `2**0`
  - `7fe` (2046$_{10}$) → `2*2023`$_{10}$
- `7ff`$_{16}$
  - infinity (when fraction == 0)
  - NaN

```
00000000000

00000000001
01111111111
11111111110

11111111111
```

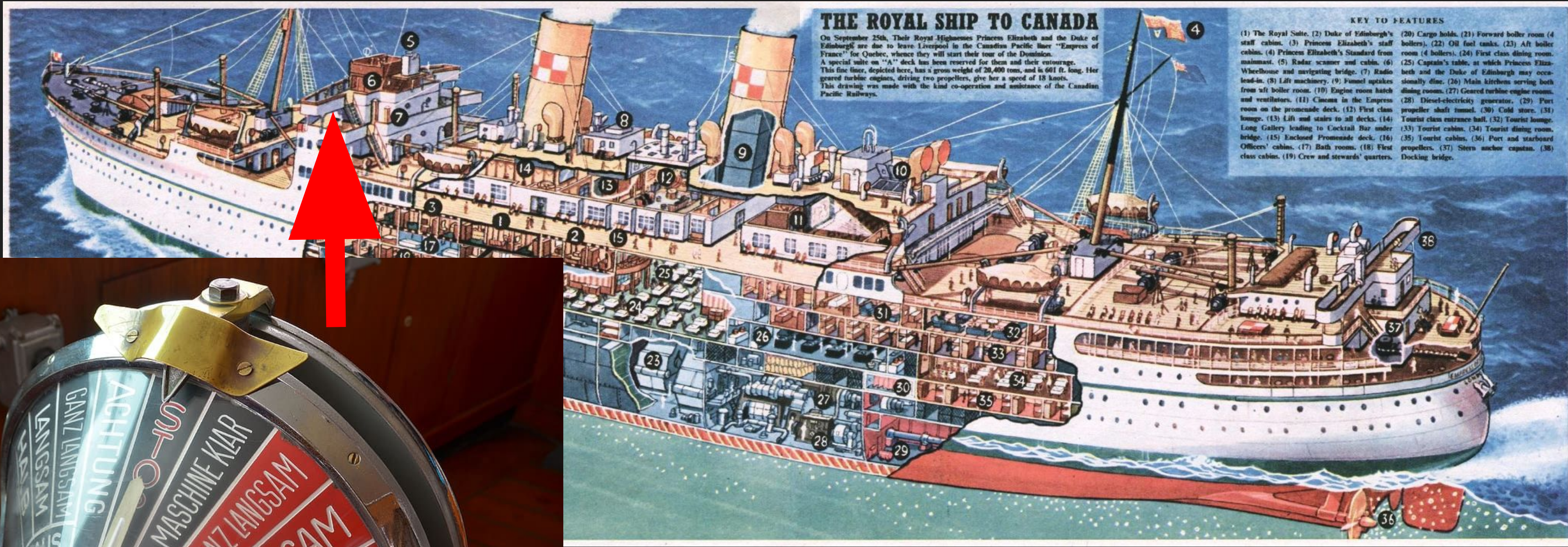
This is still *float*

IEEE 754

```
float('Inf')   # inf, Infinity, …
float('-Inf')  # -inf, -Infinity, …
math.inf, -math.inf
math.isinf(x)

-0.0, 0/-1, -1/float('inf')
0.0 == -0.0
-0.0 + 0.0 == 0.0

float('NaN')  # nan, -nan, … - Not a Number
math.nan
math.isnan(x)
bool(math.nan) == True
```

Comparisons like =, <, >=, … with NaN - **all return False**

Processor modes



THE ROYAL SHIP TO CANADA

On September 25th, Their Royal Highnesses Princess Elizabeth and the Duke of Edinburgh are due to leave Liverpool in the Canadian Pacific liner "Empress of France" for Quebec, whence they will start their tour of the Dominion.
A special suite on "A" deck has been reserved for them and their entourage.
This fine liner, depicted here, has a gross weight of 20,400 tons, and is 601 ft. long. Her geared turbine engines, driving two propellers, give her a speed of 18 knots.
This drawing was made with the kind co-operation and assistance of the Canadian Pacific Railways.

KEY TO FEATURES

(1) The Royal Suite. (2) Duke of Edinburgh's staff cabins. (3) Princess Elizabeth's staff cabins. (4) Princess Elizabeth's Standard from mainmast. (5) Radar scanner and cabin. (6) Wheelhouse and navigating bridge. (7) Radio lead-in. (8) Lift machinery. (9) Funnel uptakes from aft boiler room. (10) Engine room hatch and ventilators. (11) Cinema in the Empress room on the promenade deck. (12) First class lounge. (13) Lift and stairs to all decks. (14) Long Gallery leading to Cocktail Bar under bridge. (15) Enclosed Promenade deck. (16) Officers' cabins. (17) Bath rooms. (18) First class cabins. (19) Crew and stewards' quarters. (20) Cargo holds. (21) Forward boiler room (4 boilers). (22) Oil fuel tanks. (23) Aft boiler room (4 boilers). (24) First class dining room. (25) Captain's table, at which Princess Elizabeth and the Duke of Edinburgh may occasionally dine. (26) Main kitchens serving both dining rooms. (27) Geared turbine engine rooms. (28) Diesel-electricity generator. (29) Port propeller shaft tunnel. (30) Cold store. (31) Tourist class entrance hall. (32) Tourist lounge. (33) Tourist cabins. (34) Tourist dining room. (35) Tourist cabins. (36) Port and starboard propellers. (37) Stern anchor capstan. (38) Docking bridge.

FLT_ROUNDS (float.h macro)

sys.float_info.rounds

integer representing the rounding mode for floating-point arithmetic. This reflects the value of the system FLT_ROUNDS macro at interpreter startup time:

- -1: indeterminable
- 0: toward zero
- 1: to nearest ("unbiased") - default
- 2: toward positive infinity
- 3: toward negative infinity

IEEE 754

https://docs.python.org/3/library/sys.html#sys.float_info

```
from ctypes.util import find_library

libm = ctypes.cdll.LoadLibrary(find_library('m'))
FE_TONEAREST = 0x0000
FE_DOWNWARD = 0x0400
FE_UPWARD = 0x0800
FE_TOWARDZERO = 0x0c00
```

/usr/include/x86_64-linux-gnu/bits/fenv.h

```
libm.fesetround(FE_TONEAREST)
(1/10).hex()      → '0x1.999999999999ap-4'

libm.fesetround(FE_TOWARDZERO)
(1/10).hex()      → '0x1.9999999999999p-4'
```

```
libm.fesetround(FE_TONEAREST)

9007199254740992.0 + 1
9007199254740992.0

libm.fesetround(FE_UPWARD)

9007199254740992.0 + 1
9007199254740994.0
```

round() - processor-independent

round() - x rounded to n digits, rounding halfs **to even**.

```
round(0.5)
0

round(1.5)
2

round(2.5)
2

round(3.5)
4
```

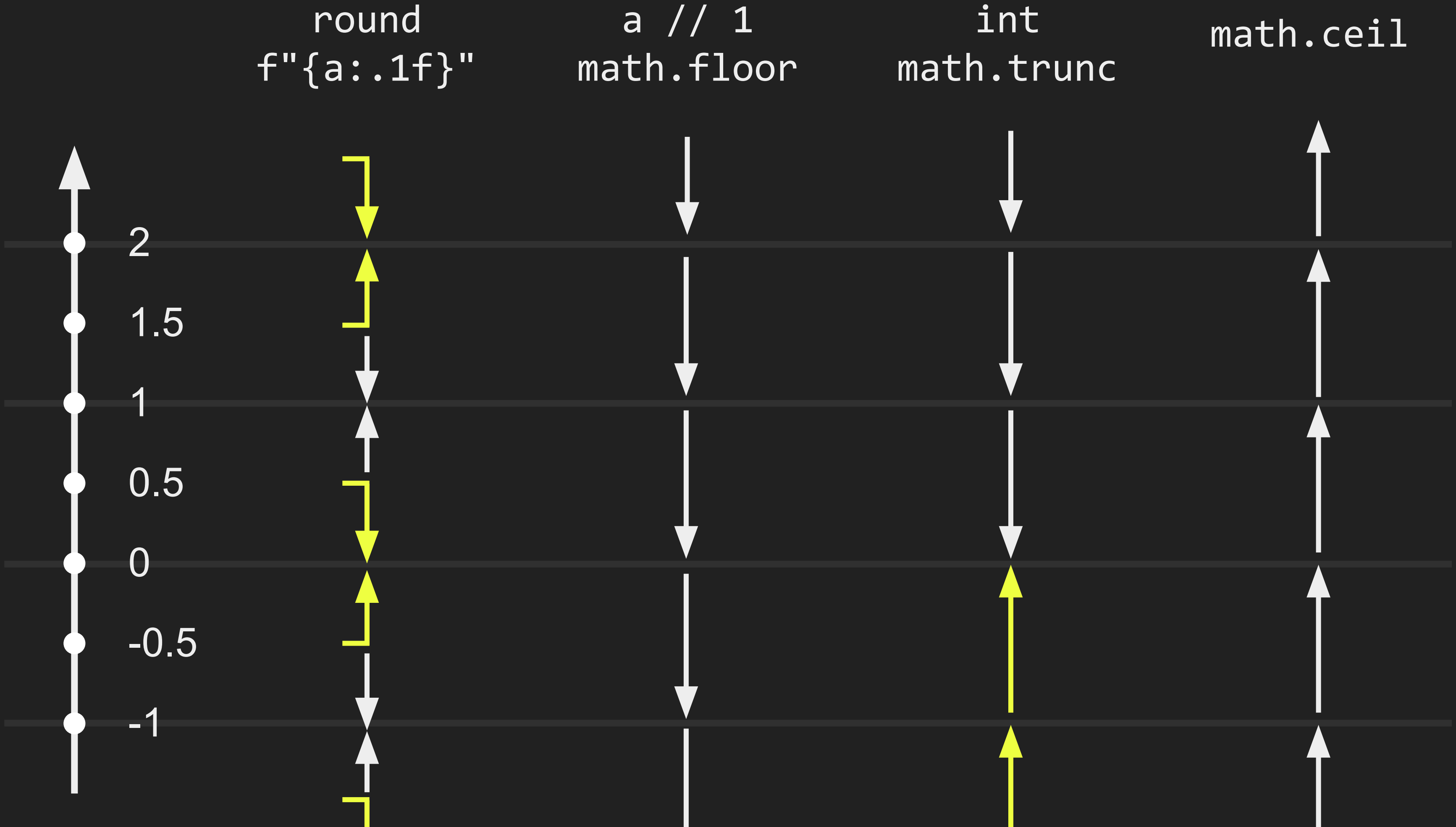
`a // b`

Int division, rounded tonwards -*Infinity*.
Result can be float

`int(), math.trunc()`

truncates (towards zero)

`math.floor(), x.__floor__`
`math.ceil(), x.__ceil__`


round
f"{a:.1f}"
a // 1
math.floor
int
math.trunc
math.ceil
2
1.5
1
0.5
0
-0.5
-1

Traps

Star Wars Episode VI: Return of the Jedi

Trap

```
f = 0
while f != 1:
    f += 0.1
```

Do you really need "equal"?

```
f = 0
while f < 1:
    f += 0.1
```

Trap

```
0.1 + 0.1 + 0.1 == 0.3
False
```

```
math.fsum([0.1, 0.1, 0.1]) == 0.3
True
```

Trap - ?

```
#include <stdio.h>
int main() {
    float meters = 0;
    int iterations = 100000000;
    for (int i = 0; i < iterations; i++) {
        meters += 0.01;
    }
    printf("Expected: %f km\n", 0.01 * iterations / 1000 );
    printf("Got: %f km \n", meters / 1000);
}
```

```
Expected: 10000.000000 km
Got: 262.144012 km
```

https://jvns.ca/blog/2023/01/13/examples-of-floating-point-problems/

Trap

```
13.716 / 4.572 → 3.0

13.716 % 4.572 → 4.571999999999999
```

```
13.716 // 4.572 → 2.0
```

or…

```
divmod(13.716, 4.572)
(2.0, 4.571999999999999)
```

`divmod(a, b)` - *For integers, the result is the same as (`a // b, a % b`). For floating point numbers the result is (`q, a % b`), where q is usually `math.floor(a / b)` but may be 1 less than that.*

Real world examples

"Fancy strawberries in April, and in mid ocean. The whole thing is positively uncanny. Why you would think you were at the Ritz."
—LADY LUCILE DUFF GORDON, FIRST CLASS PASSENGER

Premier Exhibitions

Patriot, 1991

Dhahran, Saudi Arabia. Patriot did not shoot at Scud rocket.

Internal clock gives ticks each 0.1 s.

Increments by 0.1 in 24 bit *fixed point* register. This gives error of 0.000000095.

With 100h of standby this gives error of 0.34 s.

Other Scud rocket, that *was* shot down

https://www-users.cse.umn.edu/~arnold/disasters/patriot.html

Ariane 5, 1996

First Ariane 5 flight.

Cast error (64-bit float -> 16-bit signed int). Ada language.

```
L_M_BV_32 := TBD.T_ENTIER_32S ((1.0/C_M_LSB_BV) * G_M_INFO_DERIVE(T_ALG.E_BV));

if L_M_BV_32 > 32767 then
    P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
elsif L_M_BV_32 < -32768 then
    P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
else
    P_M_DERIVE(T_ALG.E_BV) := UC_16S_EN_16NS(TDB.T_ENTIER_16S(L_M_BV_32));
end if;
```

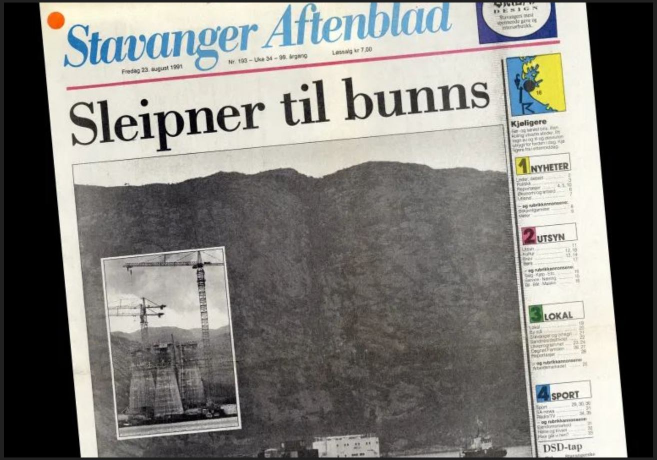
```
P_M_DERIVE(T_ALG.E_BH) :=
  UC_16S_EN_16NS (
    TDB.T_ENTIER_16S ((1.0/C_M_LSB_BH) *
    G_M_INFO_DERIVE(T_ALG.E_BH))
);
```

Sleipner A, 1991

The concrete foundations for an oil rig collapsed during the test.

The incorrect calculations attributed to the NASTRAN program (in Fortran) - no details.

https://github.com/nasa/NASTRAN-95



Stavanger Aftenblad

Fredag 23. august 1991 Nr. 193 – Uke 34 – 99. årgang Løssalg kr 7,00

DESIGN

Sleipner til bunns

Kjøligere

[illegible]

1 NYHETER

[illegible]

2 UTSYN

[illegible]

3 LOKAL

[illegible]

4 SPORT

[illegible]

DSD-tap

Schleswig-Holstein parliamentary elections, 1992

The electoral threshold was 5%.

The Green Party received 4.97%.

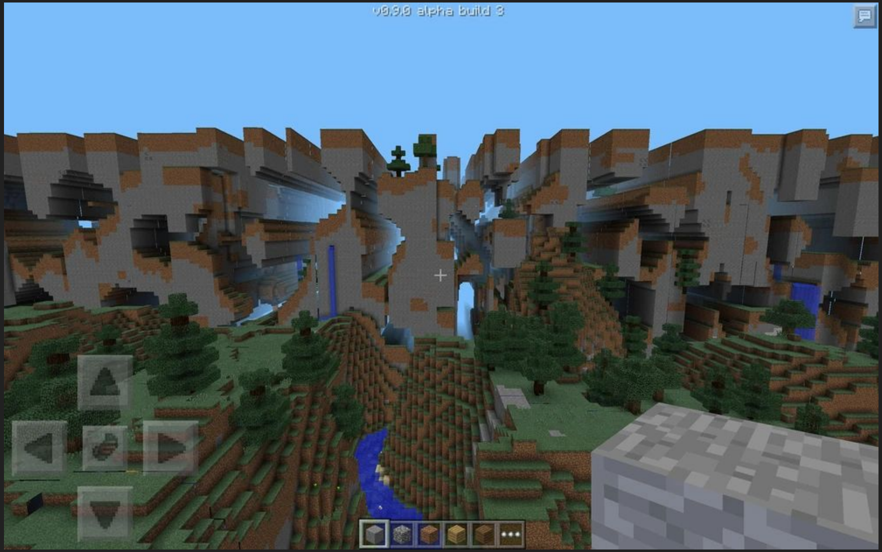
The result was rounded to 5.0% on printout.

The program has been used for years.

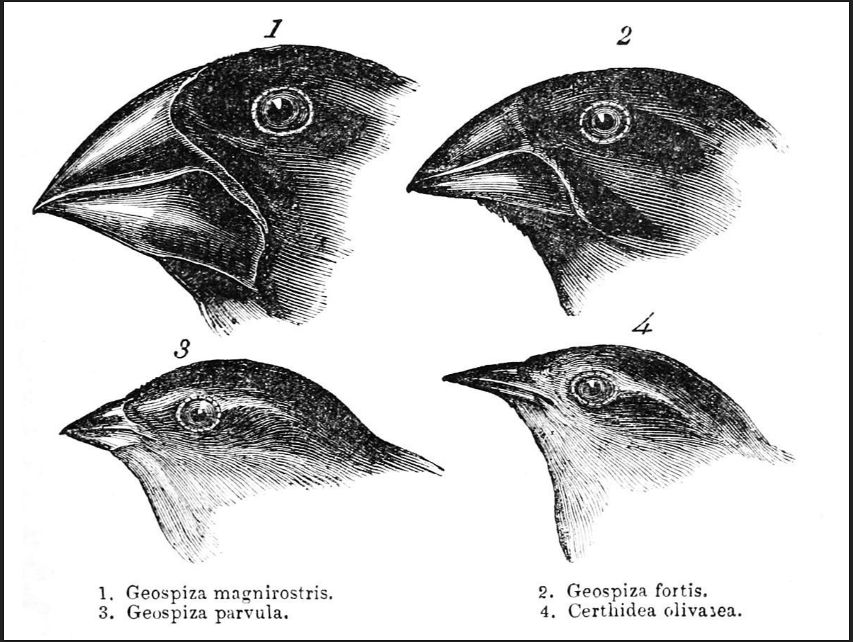
Reisemagazin Online

Kerbal Space Program: Deep Space Kraken

Minecraft: Far Lands



Similar species



Ravell

John Gould (1804 - 1881)

Decimal

```
from decimal import Decimal

Decimal('7.325').quantize(Decimal('0.01'), rounding=ROUND_DOWN)

json.loads('0.1', parse_float=Decimal)
```

from fractions import Fraction

```
Fraction.limit_denominator(max_denominator=1000000)

Fraction(1.1).limit_denominator()
Fraction(11, 10)
```

```
class Fraction(numbers.Rational):
    def __new__(cls, numerator=0, denominator=None, *, _normalize=True):
        self = super(Fraction, cls).__new__(cls)
        …
        self._numerator = numerator
        self._denominator = denominator
        return self
```

https://github.com/python/cpython/blob/main/Lib/fractions.py

mpmath

```
>>> from mpmath import mp
>>> mp.dps = 50
>>> print(mp.quad(lambda x: mp.exp(-x**2), [-mp.inf, mp.inf]) ** 2)
  3.1415926535897932384626433832795028841971693993751
```

PySim

```
>>> from sympy import *
>>> x, t, z, nu = symbols('x t')
>>> integrate(sin(x**2), (x, -oo, oo))
```

$$\frac{\sqrt{2} \cdot \sqrt{\pi}}{2}$$

uses mpmath :)

Thanks for your attention

Konrad Gawda

linkedin: konradgawda

orange™