Core Python: Numeric Types, Dates, and Times

REVIEW

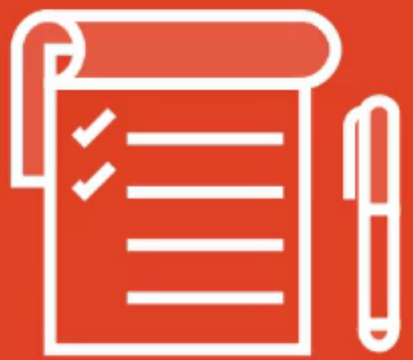
Austin Bingham
COFOUNDER - SIXTY NORTH

@austin_bingham

Robert Smallshire
COFOUNDER - SIXTY NORTH

@robsmallshire

Overview



Review `int` and `float`

Look at their inherent limitations

Further references

Review of `int` and `float`

`int`

Integers

`int`s represent integers, a.k.a. whole numbers

Arbitrary magnitude

The magnitude of an `int` is only limited by memory and the time for computation

Not fixed size

Integers in many languages are limited to a specific number of bits

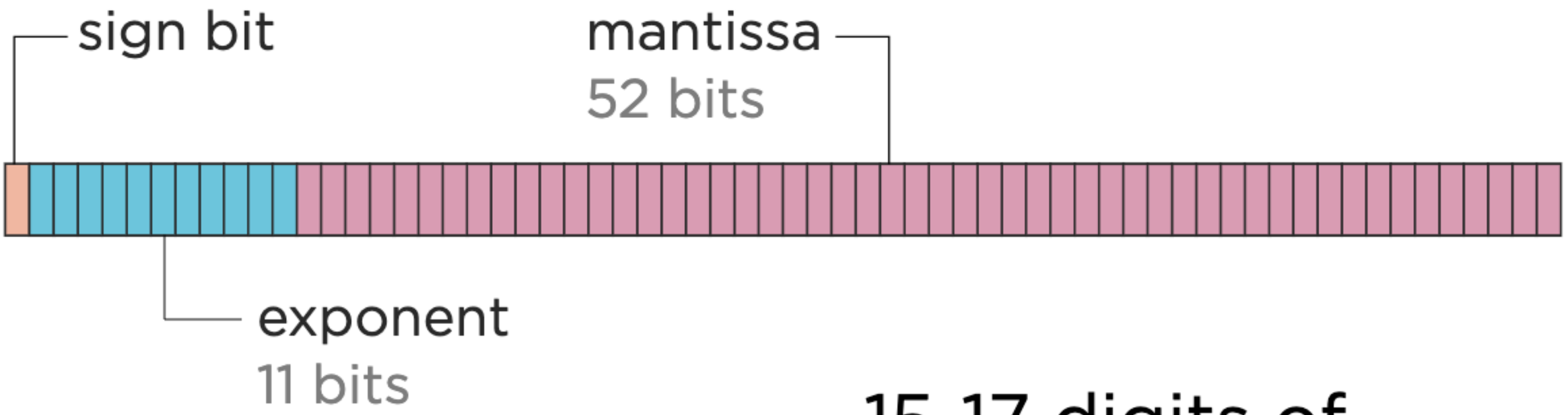
int

```
>>> from math import factorial as fac
>>> fac(100)
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000
>>>
```

`float`

Floating point	IEEE-754	C/C++
64-bit floating point number	Using a standard format known as *binary64*	Known as `double` in C-derived languages

64-bit Floating Point Representation



15-17 digits of
decimal precision

You can convert decimals with 15 figures into Python floats and back without loss of information.

float

```
>>> import sys
>>> 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, epsi
lon=2.220446049250313e-16, radix=2, rounds=1)
>>> most_negative_float = -sys.float_info.max
>>> most_negative_float
-1.7976931348623157e+308
>>> greatest_negative_float = -sys.float_info.min
>>> greatest_negative_float
-2.2250738585072014e-308
>>>
```

Don't assume that any Python `int` can be converted to `float` without loss of information.

Loss of Precision

```
>>> float(10)
10.0
>>> 2**53
9007199254740992
>>> float(2**53)
9007199254740992.0
>>> float(2**53 + 1)
9007199254740992.0
>>> float(2**53 + 2)
9007199254740994.0
>>> float(2**53 + 3)
9007199254740996.0
>>> float(2**53 + 4)
9007199254740996.0
>>>
```

Some fractional values can't be accurately represented with `float`.

Loss of Precision

```
>>> 0.8 - 0.7
0.10000000000000009
>>> 2 / 3
0.6666666666666666
>>>
```

Floating Point Arithmetic

A full treatment of floating-point arithmetic is beyond the scope of this course

Understanding its limitations motivates the introduction of other numeric types

To learn more, see David Goldberg's classic "What Every Computer Scientist Should Know About Floating-Point Arithmetic"

Summary

`int` represents whole numbers of arbitrary magnitude

`float` represents floating-point numbers

`float` uses 64 bits to store its value

`float` is the same as `double` from C-derived languages

`float` can't exactly represent every floating-point value in its range