Variables

C

Need to be very explicit about data being stored.

→ Need to specify variable type

eg. `uint8_t a = 10;`

uint8_t → variable type

Python

```
a: 10
a = a + 20
a = a + 0.165
a = -a        → a = -30.165
a: "lec1"     → a = "lec1"
```

C - Integer types

[unsigned] char, short, int, long → Not portable

instead: stdint.h → ex. uint8_t

- uint → "unsigned" cannot go negative
- 8 → # of bits used
- _t → indicates as a "type"
- integer

int8_t → signed version can go negative

uint16_t
int32_t
uint64_t

bits always 8×2^n

C - floating point types

float → 32 bit fractional #

double → 64 bit fractional #

Int # storage, consider uint8_t

256 128 64 32 16 8 4 2 1

2^8 2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

X X X X X X X X

min: 0 0 0 0 0 0 0 0 = 0

0 0 0 0 0 0 0 1 = 1

0 1 0 0 0 1 0 0 = 68

max: 1 1 1 1 1 1 1 1 = 255 $= 2^N - 1$

# of bits ↓ N = 8

total # of unique values: 2^N

$N = 8 \Rightarrow 2^8 = 256$

if N = 16, then max: $2^{16}-1$ = 65535

Consider int8_t N = 8

-128 64 32 16 1

-2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

X X X X X X X X

0 0 0 0 0 0 0 0 = 0

0 0 0 0 0 0 0 1 = 1

max: 0 1 1 1 1 1 1 1 = 127 $2^{N-1}-1$

min: 1 0 0 0 0 0 0 0 = -128 -2^{N-1}

still 256 unique values!

Limitations:

```
uint8_t a = 0;
a = a+1;   -> a = 1

a = 255;
a = a+1;   -> a = 0

uint16_t b = a+1;
           ↑
          255
   -> b = 256
```

1 1 1 1 1 1 1 1

+ 0 0 0 0 0 0 0 1

1 0 0 0 0 0 0 0 0

thrown away — stored back into a

0000000100000000

```
int8_t c = 127;
c = c+1;   -> c = -128
```