Algorithms & Programming

gcd(m, n)

- Exhaustive calculation of factors of m, n
- Proportional to min(m, n)
- Reducing the problem to a smaller one

Inductive defns

$$gcd(m, n) = gcd(n, m-n) \qquad m > n$$

$$gcd(m, n) = m \quad \text{if } m = n$$

$\gcd(2, \text{large odd } n) \xrightarrow[n/2 \text{ steps}]{} \gcd(2,1) \rightsquigarrow \gcd(1,0) = 1$

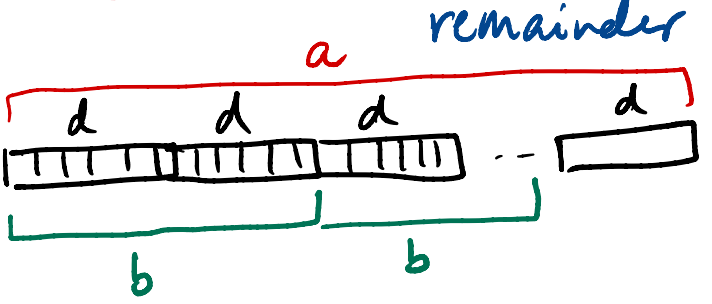
Euclid's Algorithm

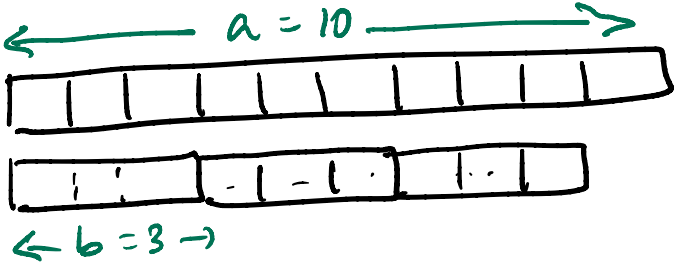
$m > n \qquad \gcd(m,n) = \gcd(n, \underbrace{m \bmod n}_{\text{remainder}})$

Assume $m = ad$

$n = bd$

$(m \bmod n) = cd$





$gcd(m,n) = gcd(n, m\%n)$ $\quad m > n$

$m\%n < n$

$gcd(2, 999) \longrightarrow gcd(2,1)$

Want solution proportional to # digits

$gcd(m,n) \rightarrow gcd(n, m\%n)$ values halve every two steps

$n \to n/2 \to n/4 \ldots \qquad n/2^k = 1$

$k = \log n$ = # of digits in base 2

7684 /10 — 768 — 76 — 7 — 0

4

$\log_2 n = \frac{\log_{10} n}{\log}$

Lecture 2, 23 Sep 2021 (in class)

Values and *name*

- Equality symbol assigns a value
- RHS can be an expression
- Expression can use previously assigned names
- Using uninitialized names on RHS generates error
- Names are not "declared" or "announced" in advance

```
z = 8
y = (7+3)*50
x = y*z
w = a*b
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-8-b2009ae7e915> in <module>()
      2 y = (7+3)*50
      3 x = y*z
----> 4 w = a*b

NameError: name 'a' is not defined
```

SEARCH STACK OVERFLOW

```
z,y,x
```

```
(8, 500, 4000)
```

Mistyping a name can raise problems that are difficult to debug

```
width = 5
witdh = 10
```

```
width
```

```
5
```

Python keeps track of names dynamically

- Values are assigned as they come
- Values have types, names inherit their type from the value

```
x = 5.0
type(x)
```

```
float
```

Numbers can be `int` or `float`

- `int` stands for integers
- `float` stands for "floating point" = reals
    - The decimal point "floats"
- Representation, and hence manipulation, of these is different
- 18/3 + 72/8

Other types?

- `if m < n` ...
- `True` and `False` are values of type `bool`
- Expressions are made up of `and, or, not`

Type specifies

- Domain of values that can be used
- Set of valid operators that are available in expressions

Collections of values

- List
- [1,2,3,4] is a list of integers
- Python allows mixed lists [1,3.5,True]
- Extract an individual item from the list, by position
- Positions are numbered from 0
- mylist = [1,3.5,True]
- mylist[1] is the value at position 1, second valued, which is 3.5

```
mylist = [1,3.5,True]
```

```
mylist[0],mylist[1],mylist[2]
```

```
(1, 3.5, True)
```

```
mylist[3]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-18-dc8cb6d85d59> in <module>()
----> 1 mylist[3]

IndexError: list index out of range
```

SEARCH STACK OVERFLOW

Determining the length of a list?

- `len(l)`

- Valid positions are 0,1,2,..., len(l)-1

```
len(mylist)
```

```
3
```

```
x = [5.0]
len(x)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-35-6058626d8b15> in <module>()
      1 x = [5.0]
----> 2 len(x), x[1.0]

TypeError: list indices must be integers or slices, not float
```

SEARCH STACK OVERFLOW

```
7 and y
```

```
500
```

```
True + False
```

```
1
```

Arithmetic operators

- Arithmetic: + , - , *
- Division?: written / -- what is 7/3?
- Normal division / **always** produces a `float`
- "integer division" // which (for int arguments) produces an int = quotient
- remainder: %
- Given two integers m,n, m = (m//n)*n + (m%n)
- Exponentiation: m ** n (in some languages m^n)
- In general, if the answer requires a float, the type will be automatically "upgraded"

```
7 // 3, 7/3, 7%3, 7.1//2.01, 8/4, 8//4, 2**3, 2**0.5
```

```
(2, 2.3333333333333335, 1, 3.0, 2.0, 2, 8, 1.4142135623730951)
```

```
type(2**3), type(2.0**3), type(4**0.5)
```

```
(int, float, float)
```

```
sqrt(2), sin(pi/2)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-41-930b1c0f65f7> in <module>()
```

```
import math
math.sqrt(2), math.sin(math.pi/2)
```

```
(1.4142135623730951, 1.0)
```

```
from math import *
```

```
log(2.71818)
```

```
0.9999625387017254
```

Logical operations => comparisons

- Comparison: m < n, a > b
- With equality: m <= n, b >= a
- Not equal? m != b
- Equality? Cannot be m = n, so m == n
- All return a value of type `bool`

```
a = 7
b = 8
c = 9
```

```
a < 7, a < b, c >= b, a != b, b == 8
```

```
(False, True, True, True, True)
```

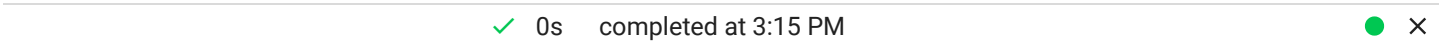
Comparisons are useful to "control the flow" of instructions

- A program is a sequence of instructions
- Most basic instruction is to assign a value to a name
- Alter the sequence by conditional expressions, repetition -- **control flow**

```
import math as mt
```

```
mt.pi
```

```
3.141592653589793
```


0s
completed at 3:15 PM