Advanced Programming

Textbook? Not really

Evaluation

Assignment	50%
Quizzes	20%
Final Exam	30%

TA? Yes.

Zero Tolerance for copying

Haskell

Translate inductive fn definitions

Declarative "What is to be computed"

$$\begin{array}{r} {}^{1} \\ 65 \\ +\ 36 \\ \hline 101 \end{array}$$

Procedural

"How to compute"

Assume +1 is "known"

$$x + 0 = x$$

$$x + (y + \underline{\underline{1}}) = (x + y) \underline{\underline{+ 1}}$$

$$\begin{array}{r} {}^{1}65 \\ 36 \\ \hline 101 \end{array}$$

Add units, carry if needed

⋮

Need to manipulate named values

gcd(m,n)

Naive approach

Find all divisors of m] Between 1 & m

$x | y$

Find all divisors of n

x divides y

Pick largest common one

Some place to collect all factors

```
factors(b)                $\swarrow$ assign a value
    factorlist = []
    for each k in 1, 2, .. , b
        if $k|m$, add k to
            factorlist   [initially empty]

    return factorlist
```

mfactors = factors(m)

Python

special word, fn defn

punctuation

```
def factors(b):

    factorlist = []
    for k in range(1,b+1):
        if b%k == 0:
            factorlist.append(k)
    return(factorlist)
```

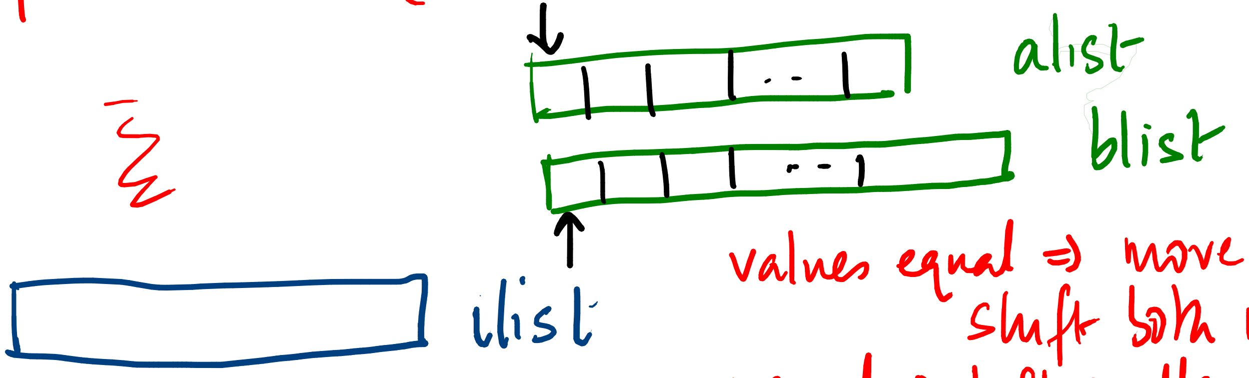
1,2,..,b

remainder or modulus

"real" equality

```
def gcd(m,n):
    mlist = factors(m)
    nlist = factors(n)
    commonfactors = intersect(mlist,nlist).

    return (last element of commonfactors)
def intersect(alist, blist):
```



values equal ⇒ move to ilist,
shift both indices
unequal ⇒ shift smaller value index

```
def intersect(alist, blist):
    ilist = []
    a = 0   } Indices into alist, blist
    b = 0   }

    While there are still elements to process in
                                            alist, blist
        if alist[a] == blist[b]:
            ilist.append(alist[a])
            a = a+1  } increment a & b
            b = b+1  }
        else:
            Increment a if alist[a] < blist[b]
                      b if blist[b] < alist[a]
```

different type of loop

Optimize the computation

$\gcd(m,n) \leq \min(m,n)$

```
for k from min(m,n) down to 1
    if k|m & k|n , k is gcd
```

Gap between specification & optimized code can be large

Good strategy: build a simplistic prototype then optimize

$$\text{Python}\left[\begin{array}{ll}\text{Version 2} & 2.7\ldots\\ \text{Version 3} & 3.4\end{array}\right.$$