

NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 1, Lecture 1

Madhavan Mukund, Chennai Mathematical Institute

<http://www.cmi.ac.in/~madhavan>

Algorithms, programming

- * Algorithm: how to systematically perform a task
- * Write down as a sequence of steps
 - * “Recipe”, or program
- * Programming language describes the steps
 - * What is a step? Degrees of detail
 - * “Arrange the chairs” vs “Make 8 rows with 10 chairs in each row”

Our focus

- * Algorithms that manipulate information
 - * Compute numerical functions — $f(x,y) = x^y$
 - * Reorganize data — arrange in ascending order
 - * Optimization — find the shortest route
- * And more ...
 - * Solve Sudoku, play chess, correct spelling ...

Greatest common divisor

- * $\gcd(m, n)$
 - * Largest k such that k divides m and k divides n
 - * $\gcd(8, 12) = 4$
 - * $\gcd(18, 25) = 1$
- * 1 divides every number
- * At least one common divisor for every m, n

Computing $\gcd(m, n)$

- * List out factors of m
- * List out factors of n
- * Report the largest number that appears on both lists
- * Is this a valid algorithm?
 - * Finite presentation of the “recipe”
 - * Terminates after a finite number of steps

Computing $\gcd(m, n)$

- * Factors of m must be between 1 and m
- * Test each number in this range
- * If it divides m without a remainder, add it to list of factors
- * Example: $\gcd(14, 63)$
- * Factors of 14

1	2	3	4	5	6	7	8	9	10	11	12	13	14
---	---	---	---	---	---	---	---	---	----	----	----	----	----

Computing $\gcd(14, 63)$

- * Factors of 14

1	2	7	14
---	---	---	----

- * Factors of 63

1	3	3	9	21	63	9	...	21	...	63
---	---	---	---	----	----	---	-----	----	-----	----

- * Construct list of common factors

- * For each factor of 14, check if it is a factor of 63

1	7
---	---

- * Return largest factor in this list:

7

An algorithm for $\text{gcd}(m, n)$

- * Use f_m , f_n for list of factors of m , n , respectively
- * For each i from 1 to m , add i to f_m if i divides m
- * For each j from 1 to n , add j to f_n if j divides n
- * Use cf for list of common factors
- * For each f in f_m , add f to cf if f also appears in f_n
- * Return largest (rightmost) value in cf

Our first Python program

```
def gcd(m,n):  
    fm = []  
    for i in range(1,m+1):  
        if (m%i) == 0:  
            fm.append(i)  
  
    fn = []  
    for j in range(1,n+1):  
        if (n%j) == 0:  
            fn.append(j)  
  
    cf = []  
    for f in fm:  
        if f in fn:  
            cf.append(f)  
  
    return(cf[-1])
```



Some points to note

- * Use names to remember intermediate values
 - * $m, n, f_m, f_n, c_f, i, j, f$
- * Values can be single items or collections
 - * m, n, i, j, f are single numbers
 - * f_m, f_n, c_f are lists of numbers
- * Assign values to names
 - * Explicitly, $f_n = []$, and implicitly, `for f in c_f :`
- * Update them, `$f_n.append(i)$`

Some points to note ...

- * Program is a sequence of steps
- * Some steps are repeated
 - * Do the same thing for each item in a list
- * Some steps are executed conditionally
 - * Do something if a value meets some requirement