Arithmetical functions in python

Tjerand Silde

23. april 2014

For those who care

To get a better understanding of the arithmetical functions studied in chapter 2 and chapter 3, it may help to actually calculate them. I decided to program some of the functions, the sums of the functions and some Dirichlet convolutions, in python. This gave me a great insight, and the code is attached below.

The code

```
import math

#check that n is in N\{0}.
def isValid(n):
    if (n<=0):
        return 0
    elif (n-int(n) != 0):
        return 0
    return 1

#The Mobius function.
def mu(n):
    if not isValid(n):
        return 0
    elif (n==1):
        return 1
    #Important to chech this first.
    elif (isDivisibleBySquare(n)):
        return 0
    else:
        #As n is not divisible by a square,
        #every power of a prime is equal to one.
        #Hence, the number of factors is correct.
        return (-1)**numberOfFactors(n)

#Check if n is divisible by a square.
def isDivisibleBySquare(n):
    i = 2
    while (i**2<=n):
        if (n%(i**2)==0):
            return 1
        i += 1
    return 0
```

```
#Count the number of prime factors of n.
def numberOfFactors(n):
    counter = 0
    while (n>1):
        i = 2
        while (i<=n):
            if (n%i==0):
                n /= i
                counter += 1
                break
            i += 1
    return counter

#The summation of mu(n).
#mu(1)=1, 0 otherwise.
def sumOfMu(n):
    d = 1
    muSum = 0
    while (d<=n):
        if (n%d==0):
            muSum += mu(d)
        d += 1
    return muSum

#The number of numbers smaller than or
#equal to n which is relatively prime to n
def phi(n):
    if not isValid(n):
        return 0
    if (n==1):
        return 1
    else:
        for i in range(2,n+1):
            if ((n%i) == 0 and isPrime(i)):
                n *= (1 - (1.0 / i))
        return int(n)

#Check if a numer is prime or not
def isPrime(n):
    return not (n < 2 or any(n % i == 0 for i in range(2, int(n**0.5)+1)))
```

```
#The summation of phi(n).
#The sum over the divisors
#is equal to n.
def sumOfPhi(n):
    d = 1
    phiSum = 0
    while (d<=n):
        if (n%d==0):
            phiSum += phi(d)
        d += 1
    return phiSum

#The natural function
#which return the input
def N(n):
    if not isValid(n):
        return 0
    return n

#The identity function.
#I(1)=1, 0 otherwise
def I(n):
    if n==1:
        return 1
    return 0

#The unit function u(n).
#Return 1 for all n.
def u(n):
    return 1

#The convolution of phi(n).
#phi = mu * N.
def phiConvolution(n):
    d = 1
    convolution = 0
    while (d<=n):
        if (n%d==0):
            convolution += mu(d) * N(n/d)
        d += 1
    return int(convolution)
```

```
#The convolution of N(n).
#N = phi * u.
def naturalConvolution(n):
    d = 1
    convolution = 0
    while (d<=n):
        if (n%d==0):
            convolution += phi(d) * u(n/d)
        d += 1
    return int(convolution)

#Return the logarithm to base e.
def log(n):
    return (math.log(n))

#The Mangoldt function
#Return log p if n is
#a prime power, i.e.
#n = p**m for some prime
#p and integer m>=1.
def Mangoldt(n):
    if not isValid(n):
        return 0
    d = 2
    while (d<=n):
        #Find a divisor
        if (n%d==0):
            #Chech if n is d to some power
            if (isPrimePower(n,d)):
                return log(d)
            else:
                #If not, it must be a product
                #of different divisors.
                return 0
        d += 1
    return 0

#Calculation of logarithms.
def isPrimePower(n,d):
    return (math.log(n,d)-int(math.log(n,d))==0)
```

```
#The summation of Mangoldt.
#log(n) = divisorsum Mangoldt.
def sumOfMangoldt(n):
    d = 1
    mangoldtSum = 0
    while (d<=n):
        if (n%d==0):
            mangoldtSum += Mangoldt(d)
        d += 1
    return mangoldtSum

#The Mangoldt convolution.
#Mangoldt = mu * log
def MangoldtConvolution(n):
    d = 1
    convolution = 0
    while (d<=n):
        if (n%d==0):
            convolution += mu(d) * log(n/d)
        d += 1
    return convolution

#The Liouville function.
#Liouville(1) = 1.
#Liouville(n) = (-1) to the
#power of the sum of the divisors.
def Liouville(n):
    if not isValid(n):
        return 0
    if (n==1):
        return 1
    else:
        return (-1)**numberOfFactors(n)

#The summation of Liouville.
#Divisorsum Liouville = 1 if
#n is a square, 0 otherwise.
def sumOfLiouville(n):
    d = 1
    mangoldtSum = 0
    while (d<=n):
        if (n%d==0):
            mangoldtSum += Liouville(d)
        d += 1
    return mangoldtSum
```

```
#The Chebyshev varphi function.
#Sum of Mangoldt.
def Liouville(n):
    if not isValid(n):
        return 0
    chebyshev = 0
    for d in range(1,n):
        if (n%d==0):
            chebyshev += Mangoldt(d)
    return chebyshev

#The Chebyshev vartheta function.
#Sum of prime logarithms.
def Liouville(n):
    if not isValid(n):
        return 0
    chebyshev = 0
    for p in range(1,n):
        if (isPrime(p)):
            chebyshev += log(p)
    return chebyshev
```