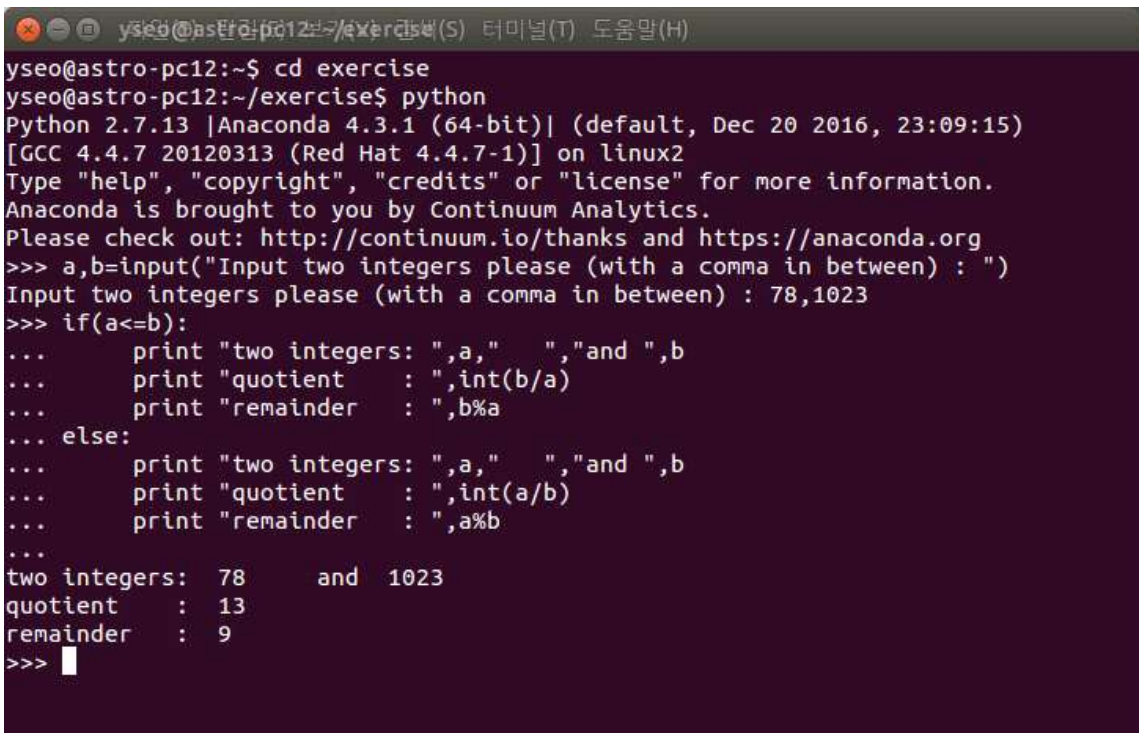
전산천문학 hw1.

2013-12239 서유경
제출일 : 17.03.20

1번. 몫과 나머지 구하기

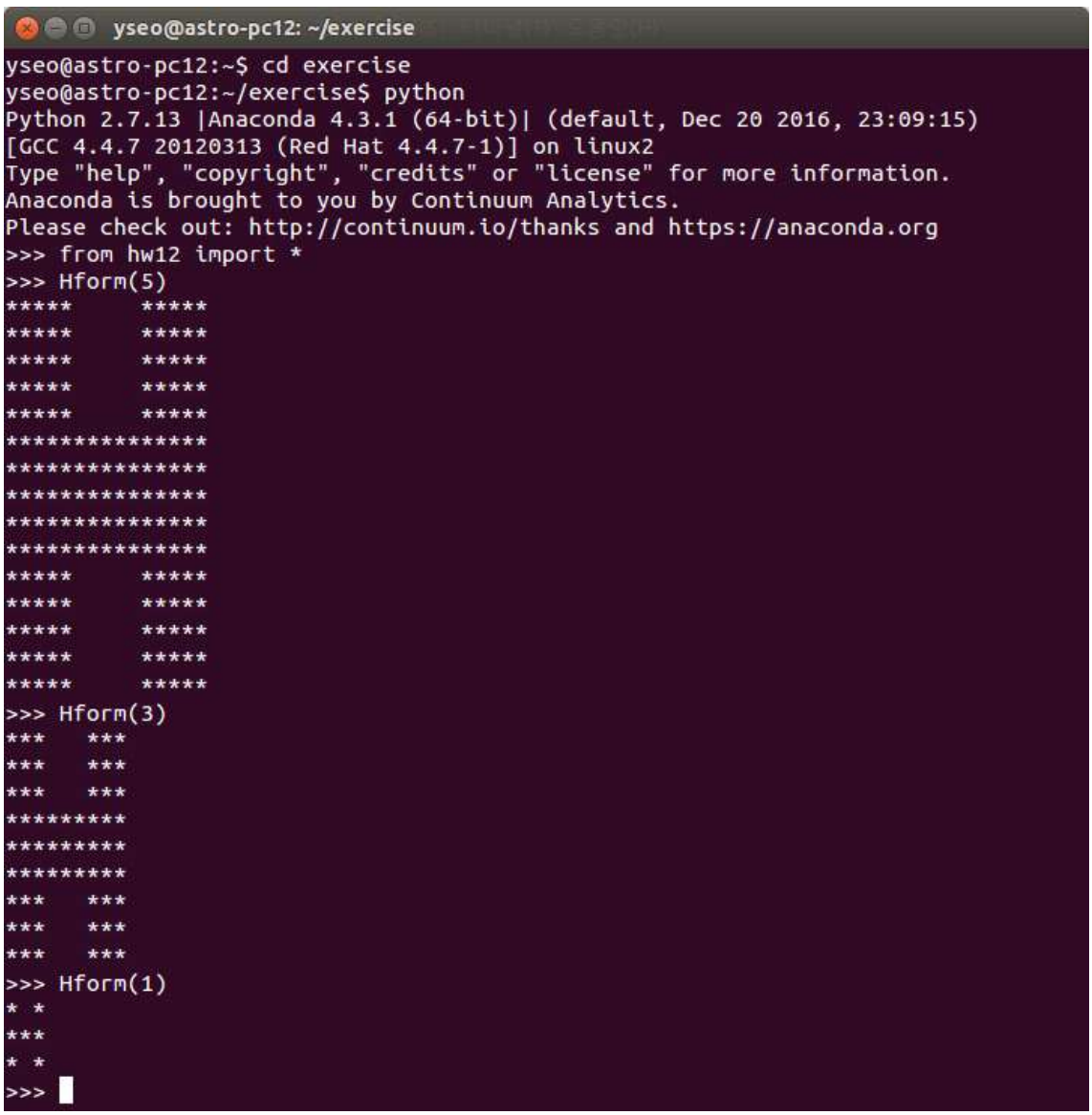
```
a,b=input("Input two integers please (with a comma in between) : ")

if(a<=b):
   print "two integers: ",a,"    ","and ",b
   print "quotient     : ",int(b/a)
   print "remainder    : ",b%a
else:
   print "two integers: ",a,"    ","and ",b
   print "quotient     : ",int(a/b)
   print "remainder    : ",a%b
```



2번. H만들기

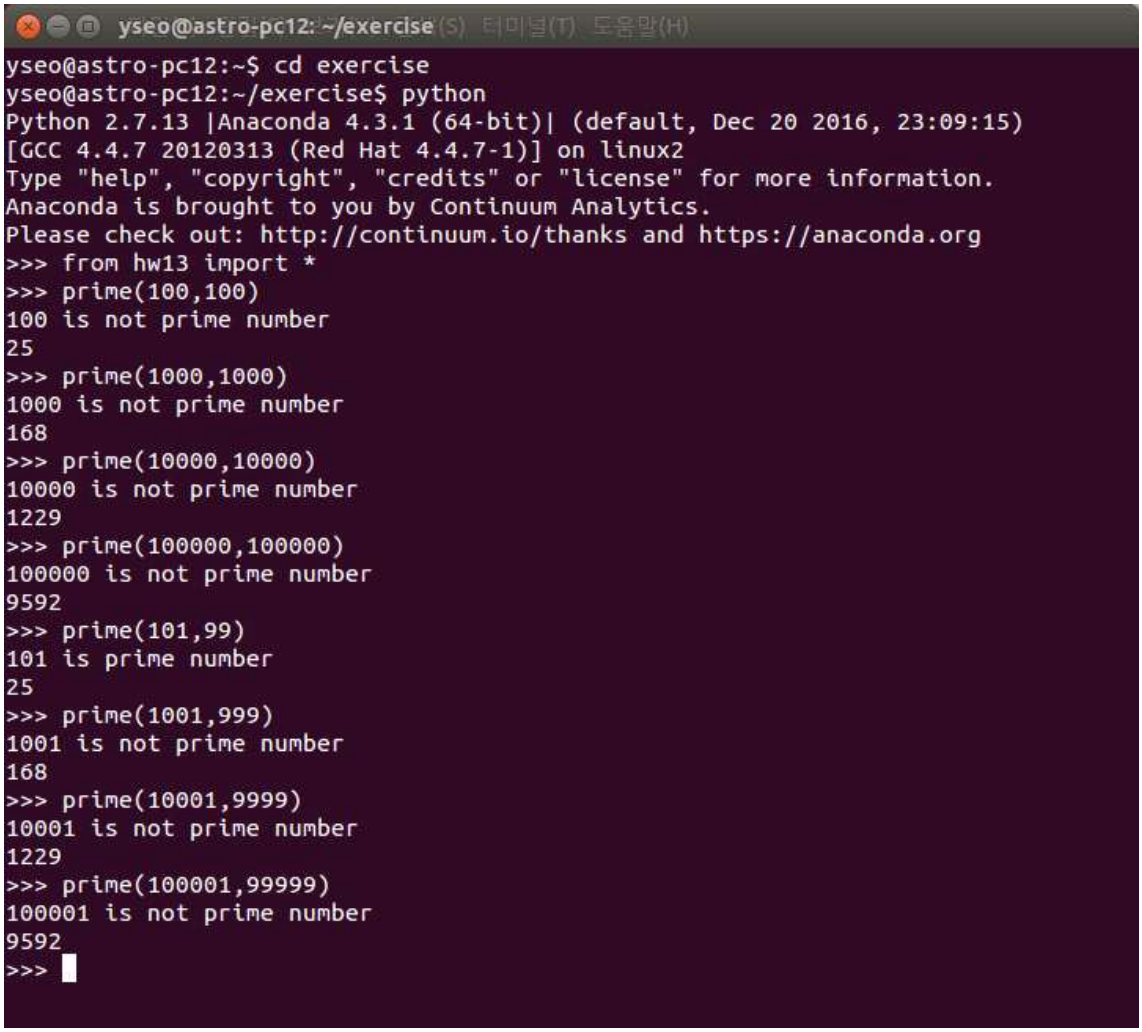
```
def Hform(x):
        a="*"*x+" "*x+"*"*x
        b="*"*x+"*"*x+"*"*x
        for n in range(1,3*x+1):
                if(n<=x):
                        print a
                elif(x<n<=2*x):
                        print b
                else:
                        print a
```



3번. 소수 판별과 개수 세기.
2 이상의 변수 n과 nmax를 입력받습니다. 그리고 n이 소수인지 아닌지, 그리고 2부터 nmax까지의 수 중 소수가 몇 개인지 세는 코드입니다. (2와 nmax를 포함하여 카운트 합니다)

```
def prime(n,nmax):
        p=n
        for x in range(2,int(n/2)+1):
                if(n%x==0):
                        p=p-1
                        break
                else:
                        p=p
        if(n==p):
                print n, 'is prime number'
        else:
                print n, 'is not prime number'
        for x in range(2,nmax+1):
                for y in range(2,int(x/2)+1):
                        if (x%y==0):
                                nmax=nmax-1
                                break
                        else :
                                nmax=nmax
        print nmax-1
```

결과 :
1~100까지 소수 개수 : 25개
1~1,000까지 소수 개수 : 168개
1~10,000까지 소수 개수 : 1229개
1~100,000까지 소수 개수 : 9592개
(캡쳐화면 아래 첨부)


yseo@astro-pc12: ~/exercise
yseo@astro-pc12:~$ cd exercise
yseo@astro-pc12:~/exercise$ python
Python 2.7.13 |Anaconda 4.3.1 (64-bit)| (default, Dec 20 2016, 23:09:15)
[GCC 4.4.7 20120313 (Red Hat 4.4.7-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Anaconda is brought to you by Continuum Analytics.
Please check out: http://continuum.io/thanks and https://anaconda.org
>>> from hw13 import *
>>> prime(100,100)
100 is not prime number
25
>>> prime(1000,1000)
1000 is not prime number
168
>>> prime(10000,10000)
10000 is not prime number
1229
>>> prime(100000,100000)
100000 is not prime number
9592
>>> prime(101,99)
101 is prime number
25
>>> prime(1001,999)
1001 is not prime number
168
>>> prime(10001,9999)
10001 is not prime number
1229
>>> prime(100001,99999)
100001 is not prime number
9592
>>>

4번. hw1_p4.dat파일을 불러내서 4번째 행만을 저장하여 따로 파일을 만든 후, 첫 번째 열을 x, 두 번째 열을 a, 세 번째 열을 b로 지정하여, 그래프를 그리기.

```
import numpy as np
import matplotlib.pyplot as plt

hw=np.loadtxt('hw1_p4.dat')
x=hw[:,0]
a=hw[:,1]
b=hw[:,2]

file=open('hw1_p4_fourth.dat','w')

for i in range(1,len(hw)):
        if ((i+1)%4==0):
                txt=str(x[i])+'\t'+str(a[i])+'\t'+str(b[i])+'\n'
                file.write(txt)
file.close()

plt.plot(x,a*x,'k-',label='a*x')
plt.plot(x,b*x,'b:',label='b*x')
plt.plot(x,a*x*b,'r--',label='a*b*x')
plt.legend(loc=2, numpoints=1)
plt.xlabel('time')
plt.ylabel('value')

plt.savefig('hw1-4.pdf')

plt.show()
```

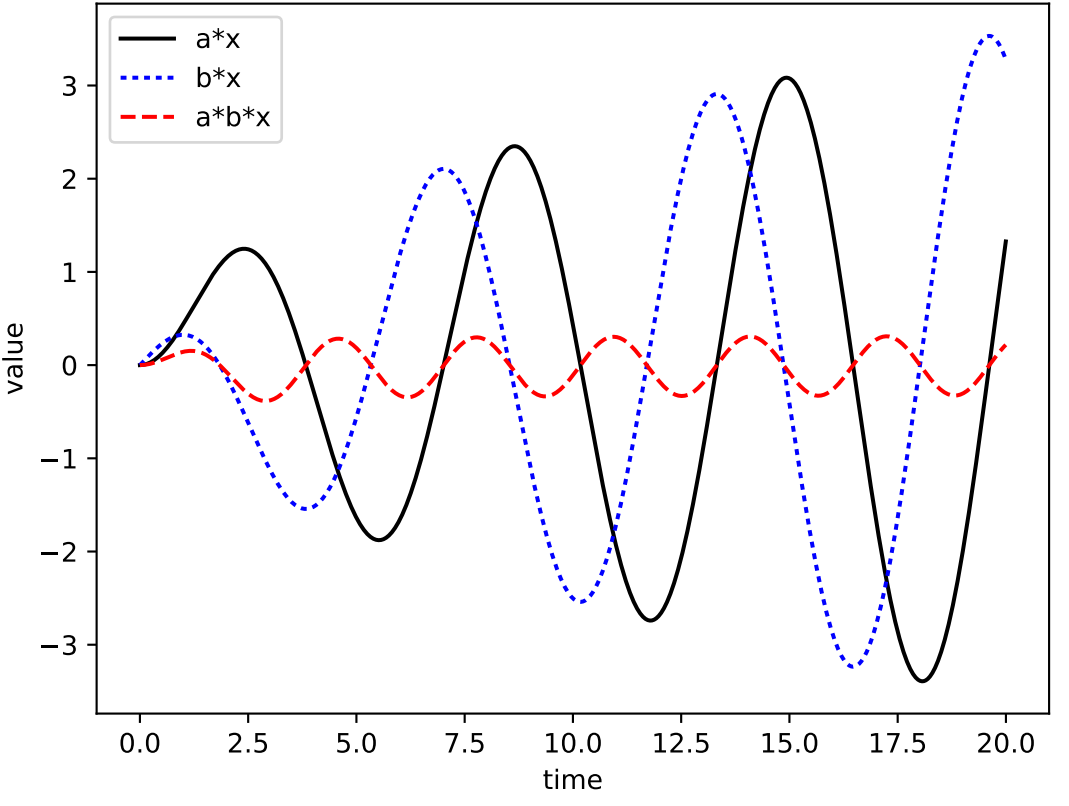
<hw1_p4_fourth.dat 파일 따로 저장한 결과>

0.005	0.002499992	0.4999953
0.013	0.006499863	0.4999683
0.021	0.01049942	0.4999173
0.029	0.01449848	0.4998423
0.037	0.01849683	0.4997433
0.045	0.02249431	0.4996204
0.053	0.0264907	0.4994734
0.061	0.03048582	0.4993025
0.069	0.03447947	0.4991076
0.077	0.03847147	0.4988888
0.085	0.04246163	0.498646
0.093	0.04644975	0.4983793
0.101	0.05043563	0.4980887
0.109	0.0544191	0.4977742
0.117	0.05839996	0.4974358
0.125	0.06237801	0.4970735
0.133	0.06635307	0.4966874
0.141	0.07032494	0.4962775
0.149	0.07429344	0.4958437
0.157	0.07825838	0.4953862
0.165	0.08221956	0.494905
0.173	0.0861768	0.4944
0.181	0.0901299	0.4938713
0.189	0.09407867	0.4933189
0.197	0.09802294	0.4927429
0.205	0.1019625	0.4921433
0.213	0.1058972	0.4915201
0.221	0.1098268	0.4908733
0.229	0.1137511	0.4902031
0.237	0.1176699	0.4895093
0.245	0.1215832	0.4887921
0.253	0.1254906	0.4880516
0.261	0.1293919	0.4872876
0.269	0.1332871	0.4865003
0.277	0.1371759	0.4856898
0.285	0.1410581	0.484856
0.293	0.1449335	0.483999
0.301	0.148802	0.4831189
0.309	0.1526633	0.4822157
0.317	0.1565174	0.4812894
0.325	0.1603639	0.4803401
0.333	0.1642028	0.4793679
0.341	0.1680337	0.4783728
0.349	0.1718567	0.4773548
0.357	0.1756714	0.476314
0.365	0.1794776	0.4752505
0.373	0.1832753	0.4741643
0.381	0.1870642	0.4730555
0.389	0.1908441	0.4719242
0.397	0.1946149	0.4707703
0.405	0.1983764	0.469594
0.413	0.2021284	0.4683953
0.421	0.2058707	0.4671742
0.429	0.2096031	0.465931
0.437	0.2133255	0.4646655
0.445	0.2170377	0.463378
0.453	0.2207395	0.4620684
0.461	0.2244307	0.4607368
0.469	0.2281112	0.4593833
0.477	0.2317808	0.4580079
0.485	0.2354393	0.4566108

0.493	0.2390865	0.4551921
0.501	0.2427223	0.4537517

...(중략)
....

19.629	0.002365441	0.1799273
19.637	0.003804475	0.1798294
19.645	0.005242681	0.1797201
19.653	0.006679966	0.1795994
19.661	0.008116241	0.1794673
19.669	0.009551412	0.1793238
19.677	0.01098539	0.179169
19.685	0.01241809	0.1790028
19.693	0.01384941	0.1788253
19.701	0.01527926	0.1786365
19.709	0.01670756	0.1784364
19.717	0.01813421	0.178225
19.725	0.01955913	0.1780024
19.733	0.02098222	0.1777685
19.741	0.0224034	0.1775234
19.749	0.02382257	0.1772671
19.757	0.02523964	0.1769996
19.765	0.02665453	0.176721
19.773	0.02806715	0.1764312
19.781	0.0294774	0.1761303
19.789	0.0308852	0.1758183
19.797	0.03229046	0.1754952
19.805	0.03369309	0.1751611
19.813	0.03509301	0.174816
19.821	0.03649012	0.1744599
19.829	0.03788434	0.1740928
19.837	0.03927558	0.1737148
19.845	0.04066375	0.1733259
19.853	0.04204876	0.1729261
19.861	0.04343054	0.1725155
19.869	0.04480898	0.172094
19.877	0.04618401	0.1716618
19.885	0.04755554	0.1712188
19.893	0.04892348	0.1707651
19.901	0.05028775	0.1703007
19.909	0.05164827	0.1698256
19.917	0.05300494	0.1693399
19.925	0.05435768	0.1688436
19.933	0.05570641	0.1683368
19.941	0.05705104	0.1678195
19.949	0.05839149	0.1672916
19.957	0.05972768	0.1667534
19.965	0.06105952	0.1662047
19.973	0.06238692	0.1656457
19.981	0.06370982	0.1650763
19.989	0.06502812	0.1644967
19.997	0.06634174	0.1639068


a*x
b*x
a*b*x
value
time
3
2
1
0
−1
−2
−3
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0