Lecture 4

'Programs'

Standard Libraries

File Manipulation

Homework Review

Is programming an art or a science ?

2) Let's find primes between 1000000 and 1010000:

```
for i in range(1000000,1010000):
    for j in range(2,i/2):
        if i%j==0 : break
    else: print i
```

time = 417.6 sec

Homework Review

2) Let's find primes between 1000000 and 1010000:

What if we skip even numbers, which are divisible by 2, as well as even factors, which cannot be prime

```
for i in range(1000001,1010000,2):
    for j in range(3,i/2,2):
        if i%j==0 : break
    else: print i
```

time = 135.3 sec

Homework Review

2) Let's find primes between 1000000 and 1010000:

Actually, we just need to find the FIRST prime factor to show that it isn't prime. The first prime factor, if it exists, is guaranteed to be <sqrt(number). Think about it...

```
for i in range(1000001,1010000,2):
    for j in range(3,int(i**.5)+1,2):
        if i%j==0 : break
    else: print i
```

time = 0.226 sec (0.205 sec without print)

Homework Review

2) Let's find primes between 1000000 and 1010000:

The only factors we need to check are themselves prime numbers. So, first, find all of the primes <sqrt(number), then only use these when checking for primeness.

```
# first find all primes < sqrt(1010000)
primes=[3]
for i in range(5,1005,2):
        for j in primes:
                if i%j==0: break
        else: primes.append(i)

# now find the primes we're after
for i in range(1000001,1010000,2):
        for j in primes:
                if i%j==0 : break
        else: print i
```

time = 0.0812 sec (5000x speedup)

Homework Review

2) Let's find primes between 1000000 and 1010000:

A 1-line program which does the same thing. Not the fastest, and certainly not easy to follow.

```
[i for i in range(1000001,1010000,2) if len([j for j in range(3,int(i**.5)+1,2) if i%j==0])==0]
```

0.864 seconds

Obfuscated C Contest

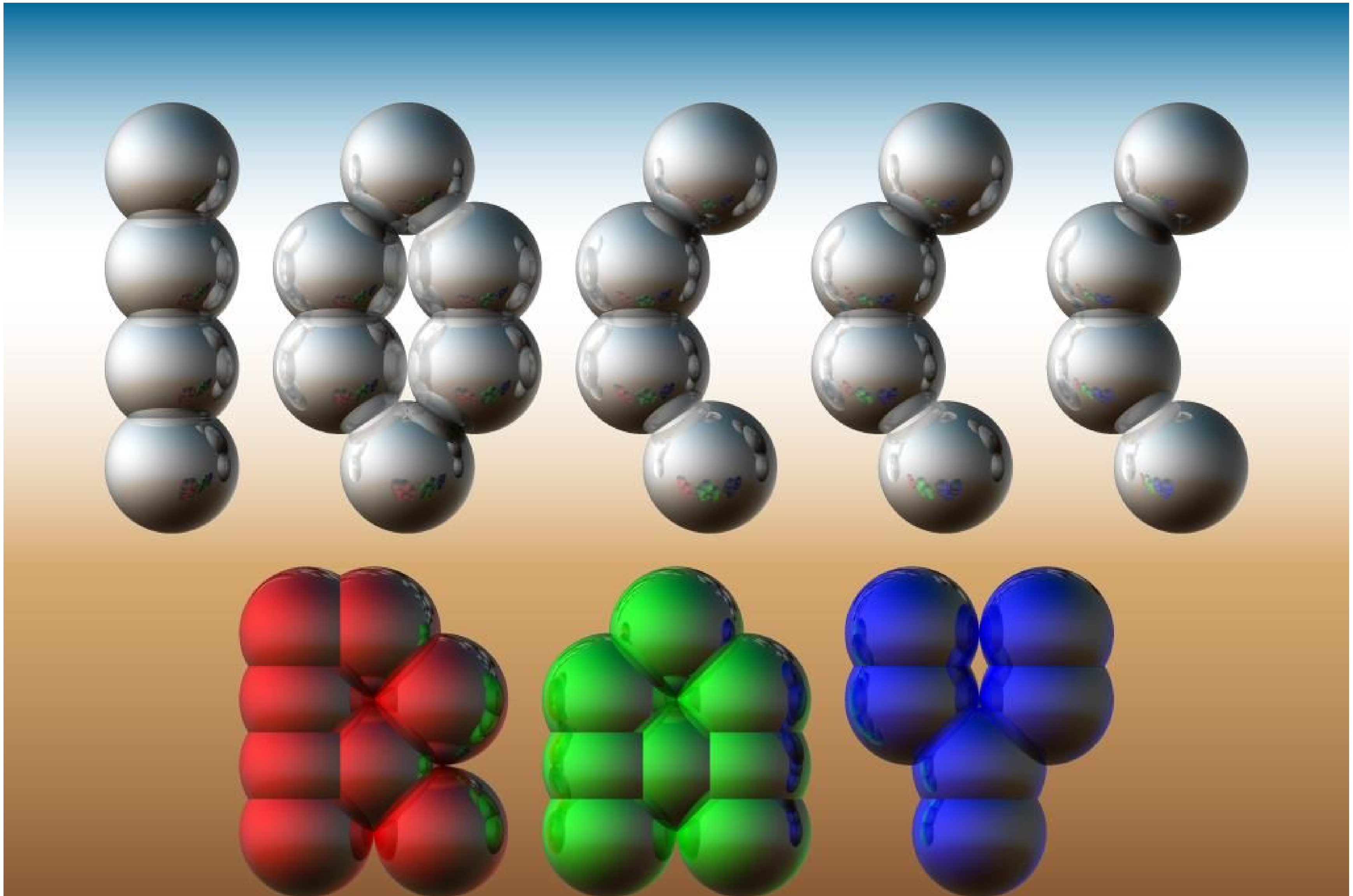
Goal of the competition is to produce a C program which does something interesting but is as difficult to read as possible, and may be internally complicated. This is much like the poetry of programming. Here is an example of a recent winner:

Obfuscated C Contest

```
X=1024; Y=768; A=3;

J=0;K=-10;L=-7;M=1296;N=36;O=255;P=9;_=1<<15;E;S;C;D;F(b){E="1""111886:6:??AAF"
"FHHMMOO55557799@@>>>BBBGGIIKK"[b]-64;C="C@=::C@@==@=:C@=:C@=:C5""31/513/5131/"
"31/531/53"[b ]-64;S=b<22?9:0;D=2;}I(x,Y,X){Y?(X^=Y,X*X>x?(X^=Y):0,  I (x,Y/2,X
)):(E=X);      }H(x){I(x,    _,0);}p;q(        c,x,y,z,k,l,m,a,          b){F(c
);x-=E*M     ;y-=S*M           ;z-=C*M         ;b=x*       x/M+         y*y/M+z
*z/M-D*D    *M;a=-x              *k/M     -y*l/M-z        *m/M;    p=((b=a*a/M-
b)>=0?(I    (b*M,_      ,0),b    =E,      a+(a>b      ?-b:b)):     -1.0);}Z;W;o
(c,x,y,     z,k,l,    m,a){Z=!    c?      -1:Z;c     <44?(q(c,x         ,y,z,k,
l,m,0,0     ),(p>      0&&c!=     a&&        (p<W         ||Z<0)          )?(W=
p,Z=c):     0,o(c+         1,    x,y,z,        k,l,          m,a)):0     ;}Q;T;
U;u;v;w    ;n(e,f,g,            h,i,j,d,a,    b,V){o(0      ,e,f,g,h,i,j,a);d>0
&&Z>=0? (e+=h*W/M,f+=i*W/M,g+=j*W/M,F(Z),u=e-E*M,v=f-S*M,w=g-C*M,b=(-2*u-2*v+w)
/3,H(u*u+v*v+w*w),b/=D,b*=b,b*=200,b/=(M*M),V=Z,E!=0?(u=-u*M/E,v=-v*M/E,w=-w*M/
E):0,E=(h*u+i*v+j*w)/M,h-=u*E/(M/2),i-=v*E/(M/2),j-=w*E/(M/2),n(e,f,g,h,i,j,d-1
,Z,0,0),Q/=2,T/=2,       U/=2,V=V<22?7:  (V<30?1:(V<38?2:(V<44?4:(V==44?6:3))))
,Q+=V&1?b:0,T               +=V&2?b        :0,U+=V    &4?b:0)     :(d==P?(g+=2
,j=g>0?g/8:g/     20):0,j    >0?(U=     j    *j/M,Q      =255-    250*U/M,T=255
-150*U/M,U=255    -100    *U/M):(U    =j*j     /M,U<M           /5?(Q=255-210*U
/M,T=255-435*U           /M,U=255    -720*     U/M):(U       -=M/5,Q=213-110*U
/M,T=168-113*U    /       M,U=111              -85*U/M)      ),d!=P?(Q/=2,T/=2
,U/=2):0);Q=Q<    0?0:      Q>O?     O:          Q;T=T<0?    0:T>O?O:T;U=U<0?0:
U>O?O:U;}R;G;B    ;t(x,y     ,a,    b){n(M*J+M    *40*(A*x   +a)/X/A-M*20,M*K,M
*L-M*30*(A*y+b)/Y/A+M*15,0,M,0,P,  -1,0,0);R+=Q    ;G+=T;B   +=U;++a<A?t(x,y,a,
b):(++b<A?t(x,y,0,b):0);}r(x,y){R=G=B=0;t(x,y,0,0);x<X?(printf("%c%c%c",R/A/A,G
/A/A,B/A/A),r(x+1,y)):0;}s(y){r(0,--y?s(y),y:y);}main(){printf("P6\n%i %i\n255"
"\n",X,Y);s(Y);}
```

Obfuscated C Contest



Programs

- Run Idle (for windows, on linux just use a text editor)

- File -> New Window

  This opens a text editor. Nothing magic about it, you can use 'notepad' if you prefer, but this is a nice editor for python.

- Enter the code from one of the Homework Review problems

- File->Save As

  Save the program to 'test.py' on your desktop. (linux users, just make a file in the current directory)

- Snake icon called 'test' should appear on the desktop. Double-click it.
  (linux users, just type 'python test.py' at the prompt, won't have the problem below)

- Note that the results are displayed in a new window on the screen, but it vanishes as soon as the program completes...

Programs

Solution 1:

- At the end of the program put:

raw_input(“Press <Enter>”)

- Run the program again

- Note that the program doesn't close the window until you press return, but you can only scroll back to see a few of the answers.

So... Solution 2:

Writing to Files

Add the following to your program:

At the beginning:

outfile=file(“myprimes.txt”,”w”)

Replace 'print i' with:

outfile.write(str(i)+”\n”)

get rid of 'raw_input' and put in:

outfile.close()

time()

How did I measure how long each answer took to run ?

```
>>> from time import time
>>> time()
1151332315.299
```

This is the absolute system time in seconds at the time that time() was called.

```
>>> a=time()
(wait a couple of seconds)

>>> print time()-a
7.93799996376
```

This is the amount of wall-clock time between the two calls to time().

Now you can use this concept to measure how long the program takes to run

String Formatting

Another use for “%”:

```
>>> print “The number is %d.”%10
The number is 10.
```

When applied to strings, the '%' operator lets you build formatted strings easily.

Basic operators:

%d	integer
%f	floating point number
%s	string

```
>>> print “The %s is %f.”%(“number”,1.23)
The number is 1.230000.

>>> print “The %s is %5.1f.”%(“number”,1.23)
The number is   1.2.
```

The 5.1 in the %5.1f means print the number with at least 5 characters and only include one digit past the decimal point. See section 2.3.6.2 (String Formatting Operations) in the manual for more details.

Reading from files

```
>>> infile=file(“myprimes.txt”,”r”)
>>> primes=infile.readlines()
>>> infile.close()
>>> primes
...
```

prints all of the lines in the file as a list of strings

```
>>> primes=[int(i) for i in primes]
>>> primes
...
```

prints all of the lines in the file as a list of integers

Homework

1) Follow the descriptions in the first few pages of the lecture notes so you get your own 'myprimes.txt' file.

Write a new program called 'test2.py' which reads in myprimes.txt, multiplies each prime by 2, and writes the results to a new file called myprimes2.txt.

Email me 'test2.py' before class thursday.

Self-test

Predict what each will do, then run it in python and see if you were right:

1)
```
for i in range(10):
        if i>8 : break
else: print "complete"
print i
```

2)
```
for i in range(10):
        if i>10 : break
else: print
"complete"
print i
```

3)
```
for i in range(5):
        for j in range(5):
                print
"%4d"%(i*j),
        print " "
```

4)
```
for i in range(5):
        for j in range(5):
                if i*j>4: break
                print
"%4d"%(i*j),
        print " "
```

5)
```
for i in range(5):
        for j in range(5):
                if i*j>4: break
                print
"%4d"%(i*j),
        else: print " *",
        print " "
```