Lecture 9

Writing Programs
Debugging
GUI Programming Intro

Homework Review

Write a program that finds all of the .jpg files in the current directory, reads each one, performs some sort of image processing with PIL on each, and writes each back to disk with "_proc" in its name, ie - image.jpg -> image_proc.jpg

Homework Review

- How do we represent the data ?
    - A list of filenames (strings)
    - 1 PIL image at a time
- Break the task into small pieces
    - Read the list of files, and keep only those with .jpg
    - Loop over filenames
        - Read image
        - Modify image
        - Write image with new name
- Code each of the pieces

Homework Review

```
from PIL import Image,ImageChops
import os

filenames=[fsp for fsp in os.listdir(".") if fsp[-4:]
in (".jpg",".JPG")]

for fsp in filenames:
    img=Image.open(fsp)
    ImageChops.invert(img)
    img.save(fsp[:-4]+"_proc.jpg")
```

Debugging

- print statements ?
- traceback module: print_exc()

Debuggers

- Spyder in Anaconda, built in editor/debugger!

- Built in IDLE (broken on Mac 10.6 system python)
- http://wiki.python.org/moin/PythonDebuggers
- http://www-pcmdi.llnl.gov/svn/repository/cdat/trunk/Packages/pydebug/Lib/pydebug.py

Programming

Find all prime numbers <100000

- How do we represent the data ?
    - Build a list containing primes
- Break the task into small pieces
    - Loop over all possible primes
        - Check if prime
        - If so, add to list
- Code each of the pieces

- Check to see if each number is divisible by every other number

```
for i in range(100000):
    for j in range(2,i):
        if i%j==0 : break
    else: print(i)
```

```
from time import time
t0=time()
primes=[]

for i in range(100000):
    for j in range(2,i):
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)

61.9 sec
```

```
from time import time
t0=time()
primes=[]

for i in range(100000):
    for j in range(2,i):
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)

61.9 sec
```

Scales as $O(n^2)$

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

```
from time import time
t0=time()
primes=[1,2,3]

for i in range(5,100000,2):
    for j in range(3,i//2,2):
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

```
15.4 sec
```

- 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...

```
from time import time
t0=time()
primes=[1,2,3]

for i in range(5,100000,2):
    for j in range(3,int(i**.5)+1,2):
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

Scales as $O(n^{1.5})$

```
0.22 sec
```

- The only factors we need to check are themselves prime numbers.

```
from time import time
t0=time()
primes=[1,2,3]

for i in range(5,100000,2):
    for j in primes[2:]:
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

```
9.3 sec
```

- Oops, forgot about the sqrt(i) limit

```
from time import time
t0=time()
primes=[1,2,3]

m=1
for i in range(5,100000,2):
    while primes[m-1]**2<i : m+=1
    for j in primes[2:m]:
        if i%j==0 : break
    else: primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

```
0.13 sec
```

Sieve of Eratosthenes

- remove non-primes:

```
from time import time
t0=time()

primes=set(range(100001))
for i in range(2,317):
    bad=range(i*2,100001,i)
    primes.difference_update(bad)

t1=time()
primes=list(sorted(primes))
print(primes[:10],primes[-10:])
print(t1-t0)
```

Profiling

- Spyder also has a profiler!
- 3 built-in profilers
  - profile, cProfile, hotshot
  - python -mcProfile script.py
  - line_profiler
- ‘C’ level profilers
  - valgrind/cachegrind (linux)
  - dtrace (mac)

```
from time import time
t0=time()
primes=[1,2,3]

def check(i,primes,m):
    for j in primes[2:m]:
        if i%j==0: return False
    return True

m=1
for i in range(5,100000):
    while primes[m-1]**2<i : m+=1
    if check(i,primes,m) : primes.append(i)

t1=time()
print(primes[:10],primes[-10:])
print(t1-t0)
```

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

```
[i for i in range(3,100000,2) if len([j for j in
                     range(3,int(i**.5)+1,2) if i%j==0])==0]
```

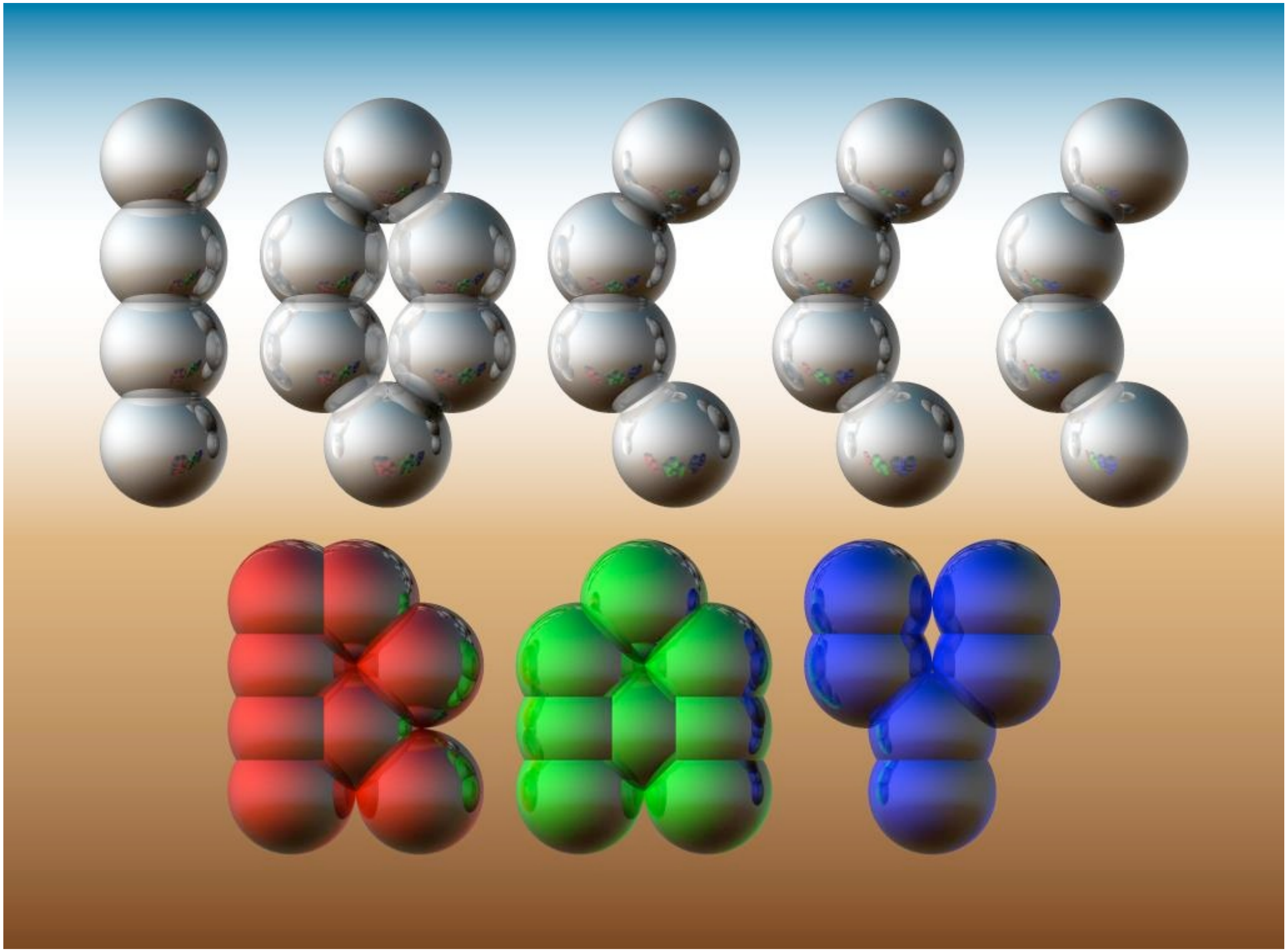
Obfuscated C Contest

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

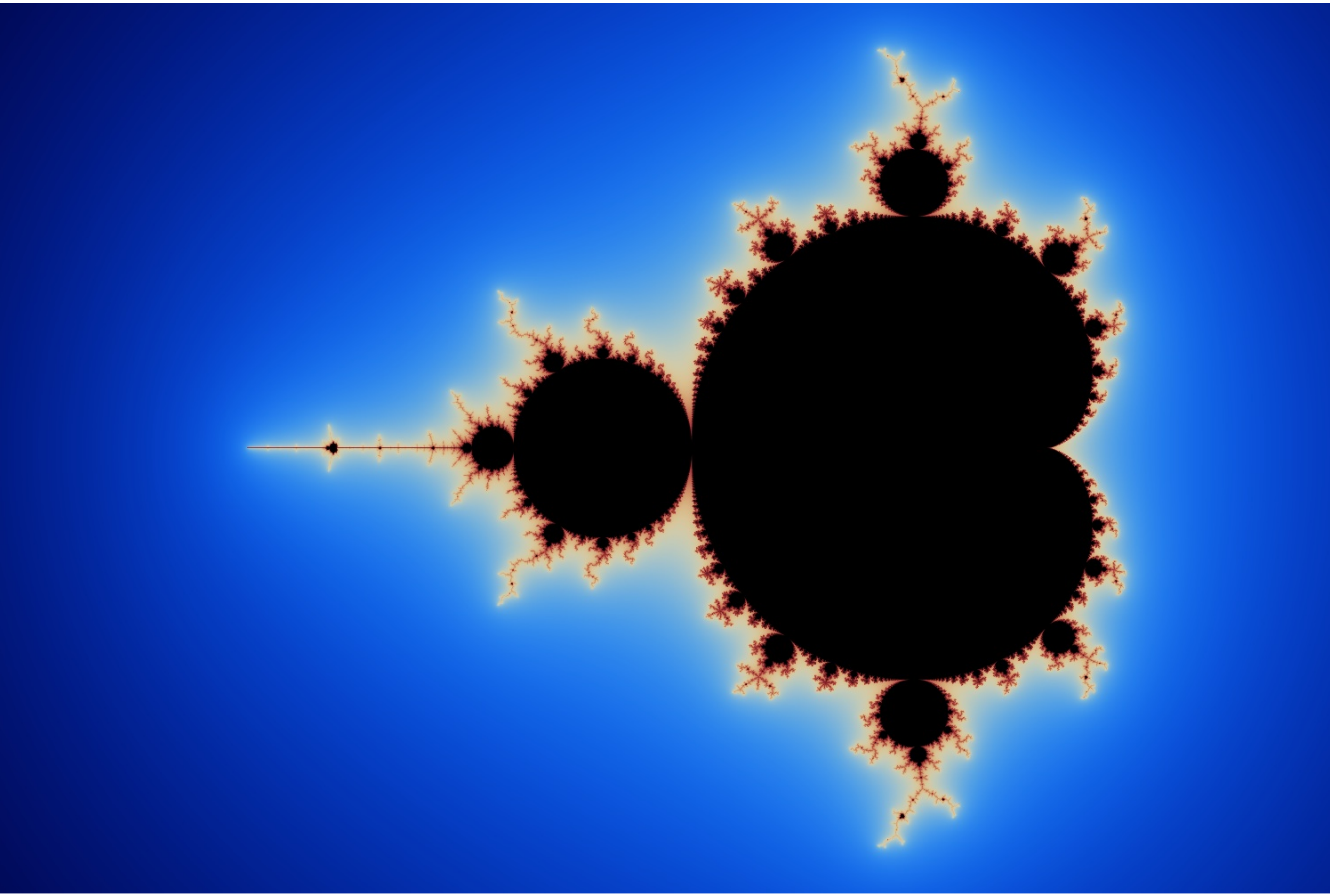
```
                              X=1024; Y=768; A=3;

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



Obfuscated Python ?

```
_                                      =   (
                                        255,
                                      lambda
                               V       ,B,c
                             :c   and Y(V*V+B,B,  c
                               -1)if(abs(V)<6)else
               (              2+c-4*abs(V)**-0.4)/i
                 )  ;v,      x=1500,1000;C=range(v*x
                  );import  struct;P=struct.pack;M,\
            j  ='<QIIHHHH',open('M.bmp','wb').write
for X in j('BM'+P(M,v*x*3+26,26,12,v,x,1,24))or C:
            i  ,Y=_;j(P('BBB',*(lambda T:(T*80+T**9
                  *i-950*T  **99,T*70-880*T**18+701*
                 T  **9     ,T*i**(1-T**45*2)))(sum(
               [              Y(0,(A%3/3.+X%v+(X/v+
                               A/3/3.-x/2)/1j)*2.5
                             /x   -2.7,i)**2 for  \
                               A       in C
                                      [:9]])
                                        /9)
                                       )   )
```



pickle

- 'Serialization' - converting a complex object to a stream of data
- from pickle import dump,load,dumps,loads
- dump(obj,file)   # stores ‘obj’ in ‘file’
- obj=load(file)   # restores ‘obj’ from file
- str=dumps(obj)   # pickled representation of obj
- obj=loads(str)   # restore representation of obj

shelve

- import shelve   # dictionary-like object on disk
- dct=shelve.open(filename)
- dct=shelve.open(filename,writeback=True)
- dct.close()

GUI Programming

Standard ↘ Best (IMHO) ↙

- Tkinter, PyQt, PyGTK, wxPython, FXPy, PyOpenGL*

- widget - A graphical object, like a button or a slider
- callback - a function which is called when the user interacts with a widget
- geometry or layout manager - controls where widgets are displayed

* - just 3-D graphics, no widgets

Tkinter

- ‘standard’ Python GUI toolkit
- Python interface elegant, but built on top of Tcl/Tk
- A bit clunky and slow, but has been used to build some very large applications (eg - Chimera)
- If you have a choice, for larger projects,
  use PyQt4 (just my suggestion)
- http://www.pythonware.com/library/tkinter/introduction/index.htm
- Tkinter extended by PMW and Tix

Modal Widgets

- Get specific info from the user without writing a full GUI for the program.
- or can be used as part of a full GUI.

- tkFileDialog
- tkMessageBox
- tkColorChooser

tkinter.filedialog

- import tkinter.filedialog
  - askdirectory(**options)
  - askopenfile(mode='r', **options)
  - askopenfilename(**options)
  - askopenfilenames(**options)
  - askopenfiles(mode='r', **options)
  - asksaveasfile(mode='w', **options)
  - asksaveasfilename(**options)

tkinter.messagebox

- import tkinter.messagebox
  - askokcancel(title=None, message=None, **options)
  - askquestion(title=None, message=None, **options)
  - askretrycancel(title=None, message=None, **options)
  - askyesno(title=None, message=None, **options)
  - showerror(title=None, message=None, **options)
  - showinfo(title=None, message=None, **options)
  - showwarning(title=None, message=None, **options)

tkColorChooser

- import tkColorChooser
    - askcolor(color=None, **options)

Tkinter

- Event driven programming
  - Set up all of your widgets
    - Create widget
    - Set callbacks
    - Place widget in window
  - Call the event loop
  - Cleanup

```
from tkinter import *
root = Tk()          # Initializes Tkinter

###setup widgets

root.mainloop()      # Runs the GUI until the user triggers an exit
root.destroy()       #  Cleanup
```

Simple Tkinter

```
from tkinter import *

root = Tk()

w = Label(root, text="Hello, world!")
w.pack()

root.mainloop()
```

Tkinter Widgets

- BitmapImage
- Button
- Canvas
  - Arc, Bitmap, Image, Line, Oval, Polygon, Rectangle, Text
- Checkbutton
- Entry
- Font
- Frame (window)
- Label
- Listbox
- Menu/Menubutton
- Message
- PhotoImage
- Radiobutton
- Scale
- Scrollbar
- Text
- Toplevel Widget

Tkinter Misc

- DoubleVar
- IntVar
- StringVar

- SimpleDialog

- tkFont

- For Callbacks, use:
- command, after, bind

Geometry Managers

- Grid Geometry Manager
  - Arrange widgets like a table
- Pack Geometry Manager
  - Arrange widgets sequentially into the available space
- Place Geometry Manager
  - Explicitly position widgets - tricky

Button Callback Example

```
from tkinter import *
root = Tk()

def pushed(): print("You pushed me too far!")

w = Button(root, text="Push Me",command=pushed)
w.pack()

root.mainloop()
```

Homework 9

NO MORE HOMEWORK !
WORK ON YOUR CLASS PROJECTS !