PHY307- Revision Sheet

You should look over carefully all posted lecture notes. Additionally I would recommend looking at your lab writeups including a brief review of the main elements in your final lab codes. The questions will be mostly short reponse questions - write out your answer in the format requested (eg. up to 4 lines of code, a few short sentences or a single name). They will be designed to test basic understanding of both programing and physics issues – there will be no long calculations or difficult code to interpret. To try and clarify the kinds of things I will expect you to be able to handle please study the examples below - if you don't understand something I suggest you go back to the appropriate lecture.

1 Python

1. Know how to assign and do arithmetic operations on simple variables in Python eg.

```
x=1.0
y=2.0
z=x/y
print z
```

2. Know how to create a list
   - Explicitly eg. `x=[1.0,2.0,3.0]`
   - Using the `arange()` function eg `x=arange(1.0,4.0,1.0)` Notice the latter yields (real) floating point numbers (1.0,2.0,3.0) but **not** 4.0. For an integer list use the simple `range()` function.

3. Know how to access an element of a list eg `x[2]` which would yield the result 3.0 on the above list. Remember list elements start with index 0 not 1.

4. Know how to loop over all elements of a list

```
for i in range(0,4):
            x[i]=x[i]+3.0
```

Again, notice that range(0,4) produces the list (0,1,3)

5. Know how to create an array. Arrays are like lists but are more efficient in numerical work. A 1D array can be defined by passing it a list eg `y=array([1.0,2.0,3.0])`, or `y=array(arange(1.0,4.0,1.0))`.

6. If we want to define a one dimensional array of say 100 floats which are initially zero we can say `y=zeros((100),Float)`. A two dimensional array of size (100,100) is defined similarly `y=zeros((100,100),Float)`. Elements may be accessed using the command `y[i][j]` or `y[i,j]`

7. Example of using an array in conjunction with a for loop

```
for i in range (0,100):
    for j in range (0,100):
        y[i][j]=y[i][j]+1.0
```

8. `while` loops. Eg the main loop of a simulation might read

```
while(1):
    t=t+dt
    update(y)
    if(t>10.0):
        finishup()
```

9. Note tabbing *crucial* in Python. In above code everything at same tab level is executed each time around the `while` loop. But only if the `if` statement is true does control pass to the function `finishup()`. The latter is said to be under control of the `if` statement.

10. `if` statements allow flow through program to depend on logical tests (see above)

11. Functions. Take arguments. Compute one or more values. Must be defined either in a separate `module` or earlier in the code before being used. Simpifies reading of long/complex code.

```
def update(y):
    for i in range (0,100):
        for j in range (0,100):
            y[i][j]=y[i][j]+1.0
```

12. Modules eg `mymodule.py` containing Python code and functions may be included using the line `from mymodule import *` at beginning of main module. Most important of these is the `visual` module which allows us to use code for manipulating 3D objects in a 3D visual environment.

13. Know how to create eg `sphere()` objects and give them certain attributes eg.

```
ball=sphere(pos=(0,0,0),color=color.red,
radius=0.1,velocity=vector(0,1,2))
```

14. Accessing these features is easy eg.
    `ball.velocity=ball.velocity+force*dt`
    (In above `force` is a vector object)

15. Syntax to create and plot pairs of values to a simple 2D graph eg.

```
mygraph=gdisplay(width=400,height=400,xmin=0.0,xmax=1.0,...)
myplot=gcurve(mygraph,color=color.blue)
....
myplot.plot(x,y)
```

16. Random numbers: the `random()` and `randint(a,b)` functions in the `random` module.

2 Algorithms

1. Euler. An equation of form $\frac{dy}{dt} = f(y,t)$ is solved with the iteration

   ```
   y[n+1]=y[n]+dt*f(y[n],t)
   ```

   Notice this uses a Python array/list to hold the values of `y` at different discrete times `t=ndt`.

2. Know how to write Newton's 2nd law as a pair of such equations.

3. Leapfrog algorithm:

   ```
   x[n+1]=x[n]+p[n]*dt+a[n]*dt*dt*0.5
   p[n+1]=p[n]+dt*0.5*(a[n]+a[n+1])
   ```

4. Metropolis algorithm. For simulating system with many d.o.f undergoing thermal fluctuations. Update system according to probability $e^{-\Delta H}$ where H is energy.

5. Box counting method for computing fractal dimensions. Find the number of "boxes" $N(s)$ of side length s needed to cover all points of a fractal.
   $d_F = \frac{\ln N(s)}{\ln s}$

3 Physics

1. Planetary motion. Simple orbits for 2 bodies using inverse square law – circles and ellipses. For three or more bodies more complicated motions – precession of elliptical orbit. True for Earth and inner planets due to Jupiter. Also correction due to General Relativity.

2. Chaos. Nonlinear system can exhibit apparently random behavior. Example: nonlinear driven pendulum. Features:

   - Motion never repeats
   - Extreme sensitivity to initial conditions
   - Motion at large times confined to *strange attractor*
   - Period doubling one common approach to chaos. Period of motion keeps doubling as some control parameter is tuned to a finite limit.
   - What is a Poincare section ?

3. Strange attractor is a *fractal* object. Non-integer dimension. Know how to compute dimension by "box counting".

4. Properties of fractals:

   - Self-similar
   - Non-integer fractal dimension
   - Structure at all length scales
   - Examples: simple non-linear systems. Julia sets, Henon map. Sierpinski triangle (know how to construct by recursive drawing)

5. Know defining map of Julia set

   ```
   z[n+1]=z[n]*z[n]+a
   ```

   Check for $|z| \to \infty$ as $n \to \infty$.

6. Self-organized critical system. Know properties: power law distributions. No need to tune external parameters – dynamics drives system there. Be able to describe models for following

- One dimensional sandpile
- Two dimensional earthquakes (simple model)

7. Critical phenomena. Tune eg temperature. System undergoes phase transition. Characterized by power law behavior for thermodynamic quantities. Critical exponents. Structure at all length scales.

8. 2d Ising model. Basics of model. Energy function. Phase diagram.