Lecture 4

For loops

Most common kind of loop that we will use

Bring up 1[illegible]-simple-for.py

program 1st makes list
for each value in list, executes
& goes to next value in list

Indentation is important here

can use "break" & "continue" in a for loop to break out or continue to next iteration for long lists w/ obvious values to loop through, you can use the range function. (built-in function)

Bring up 2- range-for.py

different options for the range function

range(5) → [0,1,2,3,4]

range(2,8) → [2,3,4,5,6,7]

range(2,20,3) → [2,5,8,11,14,17]

range(20,2,-3) → [20,17,14,11,8,5]

The following program prints out the 1st ten powers of two:

```
for n in range(1,11)
    print(2**n)
```

upper limit must be given as 11

- All arguments to range should be integers

the following program won't work

```
p=10
q = 2
for n in range(p/q)
        print(n)
```

doesn't work because p/q returns float

use p//q instead.

Function similar to range from numpy package is arange

arange returns an array
It does accept floats as input & then returns float

linspace is also like range
and from numpy package

e.g., linspace(2.0, 2.8, 5)

divides interval from 2.0 to 2.8
into 5 values and returns

[2.0, 2.2, 2.4, 2.6, 2.8]

including last value.

Using for loops to compute ~~a~~ sum

Suppose we want

$$s = \sum_{k=1}^{100} \frac{1}{k}$$

Bring up 3-sum-for.py

could also do this by loading
from a file

Bring up 4-sum-file.py

However, if data loaded into an array, usually faster to use array operations (such as "sum") rather than for loops.

Return to the emission lines of hydrogen program
can write a simpler version w/ range
Bring up 5-emission-lines.py

User-defined functions

We can write & define our own functions. Very useful if you're writing the same code over & over again

Example: Factorial function

$$n! = \prod_{k=1}^{n} k$$

Python code for this

```
f=1.0
 for k in range(1,n+1)
       f*=k
```

can write this as a function

Bring up 6-factorial-function.py

Variables created inside the function are local variables, meaning that they are only defined + used inside function + disappear when we leave.

if you try to print local variables outside the function, you get an error.

User-defined functions can take more than one argument

For example, function to convert cylindrical coordinates to distance from the origin.

Bring up 7-cylindrical.py

arguments to functions can be ints, floats, complex, lists or arrays.

can also return any type as in

```
def cartesian(r, theta):
    x = r*cos(theta)
    y = r*sin(theta)
    position = [x,y]
    return position
```

could also abbreviate last 2 lines as

```
return [x,y]
```

can return multiple values w/ the multiple assignment feature of Python.

Recall `x,y = a,b`

w/ a user-defined function, we can write

```
def f(z):
    ...
    return a,b
```

To call the function we would need

```
x,y = f(1)
```

user-defined functions need not return any value

function will end by respecting indentation

Why would you do this?

Suppose you have a program handling 3-D vectors + you're printing them out a lot. Then you could do

```
def print_vector(r)
    print ("(",r[0],r[1],r[2],")")
```

then call print_vector(r) later

definition of a user-defined function can occur anywhere in program, but before it is used

can use user-defined functions inside of other ones + can even use them w/ map command

Bring up 8-user-map.py

you can even collect together your own functions in a file called, e.g., "mydefs.py" + at the beginning of your python program write

from mydefs import myfunc

+ this will become available

this is what is happening in
many cases when importing
(but sometimes calling (code)

Bring 9-prime-factors.py

Good programming style:

1. use comments
2. use meaningful variable names
3. use the right type of variables
4. import functions first
5. give physical constants names
6. employ user-defined functions when necessary
7. print out partial results throughout program

e.g. could have something like

```
for n in range(1000000)
    if n % 1000 == 0:
        print("Step", n)
```

8. Lay out programs clearly

e.g., split long program lines into multiple ones using backslash \

(tells Python that next line is part of current one)

9. don't make programs unnecessarily complicated.

Graphics (Chapter 3)

1st focus on making simple graphs

We will use the pylab package which is part of larger package called matplotlib

first use the plot function in pylab

Bringup 10-simple-plot.py

plot causes the figure to be stored in memory.

show() causes it to be displayed.

Bring up 11-simple-x-y-plot.py

Why two functions plot & show?

You might want to
plot multiple curves on the same figure (several calls to plot function)
& then show all at once

Bring up 12-plot-sin.py

using linspace to get sample points. 100 sample points makes the curve look like sin, even though the points are connected by straight line segments.

importing data from a file &
plotting

Bring up 13-plot-from-file.py

can calculate values to go into
a graph one-by-one.
Will often do this as part of
a calculation.

Bring up 14-plot-calculate.py

can change features of the
graph. axis limits,
add labels,
different colors,
tick marks instead of
lines

Bring up 15-plot-styles.py

---

data is often not such that y is a function of x, so we might instead need a scatter plot to observe the data.

There is a scatter function to do so.

Bring up
16 - scatter.py

file shows temperature vs. brightness of several stars
(Hertz-sprung-Russell diagram)

Many of the features ~~of~~ plot work w/ scatter.

Another useful plot method is
the density plot. show data
values as a color on a
or brightness
2D plot

Bring up 17-density.py

file circular.txt ~~contains~~
2D values like

```
0.005     0.0233   - - -
0.0233    0.0516   - - -
  ⋮          ⋮       ⋱
```

use origin = "lower"
to flip convention for plot
numbering

can change to grayscale
for display

other options besides gray such as
jet (default)

hot (black-red-yellow-white)

spectral (full spectrum)

bone (gray scale w/ some blue)

hsv (rainbow starting & ending
w/ red)

you can also define your own
color scheme

other options for imshow such as

extent (change scale of data) &

aspect (change aspect ratio)

can limit range of data w/ xlim + ylim