Lecture 2

Continuing w/ basics of python
(chapter 2 of book)

Basic Arithmetic operations

x+y	addition
x-y	subtraction
x/y	division
x*y	multiplication
x**y	raise x to y

- the type of a result depends upon the type of the inputs
  - output is more general of the types
  division does not give an integer result (unique to Python 3)

Other operations

x//y	integer part of x divided by y (rounded down)

x % y - modulo, remainder after x is divided by y.

other bitwise operations that we likely won't use.

Can combine several operations together

x+2*y ⇔ $x + 2y$

3*x**2 ⇔ $3x^2$

x/2*y ⇔ $1/2(xy)$

operations carried out left to right, but calculated before others.

The following won't work

2*x=y

Python cannot solve the equation for you in this form. should write x=y/2

Statement like

$x = x + 1$

is an assignment. It does not make sense mathematically. Python takes current value of x, adds one, & stores in variable x.

So consider

```
x = 0
print(x)
x = x**2 - 2
print(x)
```

This won't solve equation $x = x^2 - 2$ but instead assigns $0^2 - 2 = -2$ to x

Some shortcuts in Python (also common in other languages)

`x += 1` add 1 to x

`x -= 4` subtract 4 from x

`x *= -2.6`

`x /= 5*y`

`x //= 3.4` divide x by 3.4 & round down

Feature of Python:
can assign multiple variables in a single line

`x,y = 1, 2.5` is the same as

`x = 1`
`y = 2.5`

More complicated example:

`x,y = 2*z+1, (x+y)/3`

All of the right side is evaluated before assigning to the left.

Nice way of swapping variables

```
x, y = y, x
```

Example program of calculating height of ball after being dropped from tower of height h.

Initial velocity is zero

ball falls amount $s = \frac{1}{2} g t^2$ in time t where

$g = 9.81\ m\ s^{-2}$ is gravitational acceleration at Earth's surface

Bring up "dropped.py"

Better to write

```
g = 9.81
s = g * t ** 2 / 2
```

Python comes w/ packages which can be imported

- One important package is "math"

If you want to use the logarithm function, then write

```
from math import log
```

at the top of your program (not necessary but makes for better programs)

After doing this run

```
x = log(2.5)
```

which calculates natural logarithm

commonly used math functions:

log
log10
exp
sin, cos, tan
asin, acos, atan
sinh, cosh, tanh
sqrt

You could use x**0.5
for square root but calling function
is quicker + more accurate

can import constants like π w/

```
from math import pi
```

+ then run, e.g.

```
print (pi**2)
```

can import multiple functions w/

```
from math import log,exp
```

can import a very long list of functions or constants

could also do

```
from math import *
```

to import all, but generally advised not to so that you're aware of everything that you're using

Some packages have modules, which are like smaller subpackages. E.g., numpy is one we will use w/ a linear algebra module. import as

```
from numpy.linalg import inv
```

Example: Bring up

polar.py

Write a program to convert polar to Cartesian coordinates

1. User enters values of

r & θ

2. Calculate

$x = r\cos\theta$

$y = r\sin\theta$

3. Print out results

discuss polar.py

Built-in functions are always available (no need to import)

Examples are

float, int, complex, abs, input, print

Another important feature of Python (& any other language) is comments

Any line starting w/ #

is ignored. For example,

this is a comment

would be ignored.

very helpful & necessary for describing how a program works (for remembering &

comments don't have to start at the beginning of a line

---

controlling program flow w/ "if"

- take action when some condition is true
- need to indent everything that is part of the if statement
  (here you need to be careful w/ spacing at beginning of line)

Bring up 3-if-... .py

Other possibilities:

```
if   x==1:        # check if x=1
if   x>1:
if   x>=1:
if   x<1:
```

```
if x<=1:
if x!=1:          # check if x ≠ 1
```

can combine conditions as in

```
if x>10 or x<1:
```

```
if 2<=10 and x>=1:
```

other constructs use

else +

elif

to give different actions based on conditions

Bring up 4-....py +

5-....py

the while statement

keep running a loop while a condition is true

Bring up 6-while.py

Breaking out of a while loop w/ break

Bring up 7-while-break.py

program illustrates an if statement nested inside of a while loop.

Bring up 8-even-odd.py & discuss

Bring up 9-fibonacci.py & discuss