Introduction to Linux/Emacs/Sage

January 25, 2011

Logging In

Login server: `login.math.wisc.edu`

Mac:

This is easy. Just pull up a terminal and go for it!

Windows:

1. `putty` - my preferred option, takes seconds to downloads and start using
2. cygwin - incredibly large program for windows that emulates linux. It has a built in ssh command and you can run X applications natively in windows.
3. VmWare+Ubuntu - not for the faint of heart or those with a slow computer!

As a comment for those with a Mac or either of the two latter solutions, `ssh -X username@login.math.wisc.edu` is a way to use programs that have a graphical interface. There is also something called Sage Notebook. I don't know very much about but it does seem slow compared to using a terminal. The initial investment of learning linux is a little higher but it will pay off in the long run.

Linux

ls	listing of the elements in the current directory
cd	change directory
cp	copy files
logout/exit	to exit the system
less	command to read the contents of a file
pwd	the name of the current directory, it is easy to get lost
mkdir	make a new directory, for those who like to stay organized
top	see what processes are running
mv	move or rename files
>	pipe output to a special file, more on this later
man	a help file on any command!
C-z	if you need to stop a command

Emacs

Emacs is an incredibly powerful text editor with anything you want built in. For our purposes we will only work on basic commands.

C-h t	a nice (but long) tutorial on emacs
C-x C-s	save the current "buffer"
C-x C-c	save the buffer and exit
C-x u	undo
C-a	move the cursor to the beginning of the line
C-e	move the cursor to the end of the line
C-v	move the cursor forward one page
A-v	move the cursor backward one page
C-space	set mark
C-w	cut region
A-w	copy region
C-y	paste

Sage

Finally the meat. This part is tricky, so I just tried to go through all the sage help files and disseminate information from the various sections in one place. I hope this is useful.

Remember you can get information on anything in python at any time by putting a `?` after it. Also if you type any part of a string and hit `tab`, it will show you your options/autocomplete.

Useful Commands

quit	quit sage
%hist	shows you past commands
!cmd	you can use the unix command cmd in sage!
logstart file	log input to file
load file	load a log file, if you defined things in the past, this will preserve the definitions
%time	print the amount of time it took to run a command
str()	make something into a string
o.open('file', 'w')	opens a file for writing
o.write(str(5))	writes the string representation of 5 to a file
o.close()	close the stream o
save_session('file')	saves variable definitions
load_session('file')	loads these definitions

Assignment, Equality, Arithmetic

a=5	assignment of 5 to a
`==, <=, >=, <,>`	comparison operators
`**, ^`	exponentiation
% m	mod m
n(v,digits = k), v.n(digits = k)	gives the number v with precision k

Functions, Iteration

```
def plusk(n,k):
    return n+k
```

Don't forget the colon or indentation. You can make a default value for k by:

```
def plusk(n,k=7):
    return n+k
```

but this can be overridden.

To define a `for` loop, use the following,

```
for i in range(3):
    print i
```

The range function is extremely powerful. It gives a list of numbers from 0 to 2 in the previous example. You can give it two values and even negative values. By giving a this parameter you can step in increments. For example, `range(1,20,2)`, steps from 1 to 20 in 2's.

Lists and Dictionaries

l = [1,2,3, "hello world"]	create a list
l[0]	the first element in the list
l.append(5)	add the element 5 to the list
len(l)	length of l
del l[i]	delete the i-th element of the list
`[a^2 for a in range(20)]`	makes a list of the squares up to 20
a[i:k]	gives the sublist of a between indices i,k
d = {3:2, 6:7, 2/3:"hiya"}	create a dictionary
d['a'] = 'b'	appends a new rule to the dictionary
d.clear()	empty a dictionary
d.keys()	a list of keys

Algebra in Sage You have to specify variables in sage.

```
y = var('y')
```

specifies a variable y. From playing around I think there is a default variable x you don't need to specify.

```
solve([y^2+2*y+1 == 3], y)
```

This solves the above equation. In general solve takes multiple equations and multiple variables and solves them simultaneously! If you need a numerical solution in the interval (a, b) use

```
find_root(eq, a,b)
```

Sage has something called a symbolic expression. These are things that you do algebra/calculus on. It is a good practice to use def when you want a function that is not mathematical and to use symbolic ones otherwise. For example

```
g(x) = x^4+x
```

defines a function that does what you think

There are a host of issues around this. MUST READ: Some Common Issues With Functions. This will save you many headaches later.

Rings in Sage

ZZ	integers
QQ	rationals
QQbar	algebraic closure of Q
RR	real numbers
CC	complex numbers
K.<a> = GF(k,'a')	a finite field of order k with generator a
IntegerModRing(n)	the ring of integers mod n
K.<a> = NumberField(poly)	a number field with a prescribed root of a polynomial
L.<b> = K.extension(poly)	extend a field
L.galois_groups()	gives you the galois group
I	square root of minus 1
reset('i')	in case you lose I
`z = 4+7*I`	define a complex number
z.imag(), z.real()	imaginary and real parts respectively
R.<t> = QQ[], R.<t> = PolynomialRing(QQ)	specify a polynomial ring over the rationals in variable t
factor(poly)	factors the polynomial over the ring it was defined in!
poly in R	checks if poly is in the ring R
list(poly)	gives a list of the coefficients of the polynomial, can be used in other ways too!
f.gcd(g)	gcd of f and g
`I = (f,g)*R`	ideal generated by f and g

Number Theory

R = IntegerModRing(n)	the ring of integers mod n
R(a)	the element a in the ring m
is_square	is it a square?

`gcd, next_prime,previous_prime`
 	
`divisors, factor, factorial,`
 	
`prime_divisors, euler_phi, ...`
	all self explanatory

sum(a)	given a list a, sums the elements in it
EllipticCurve(R,[p,q])	gives the elliptic curve $y^2 = x^3 + px + q$ over the ring R
E.rank(), E.conductor()	self explanatory

Programming Files need to have the `.sage` extension. If you want to pipe output to the screen you have two options:

1. Use .open/.write statements
2. sage file.sage > output.txt

Don't forget that indentation is key!