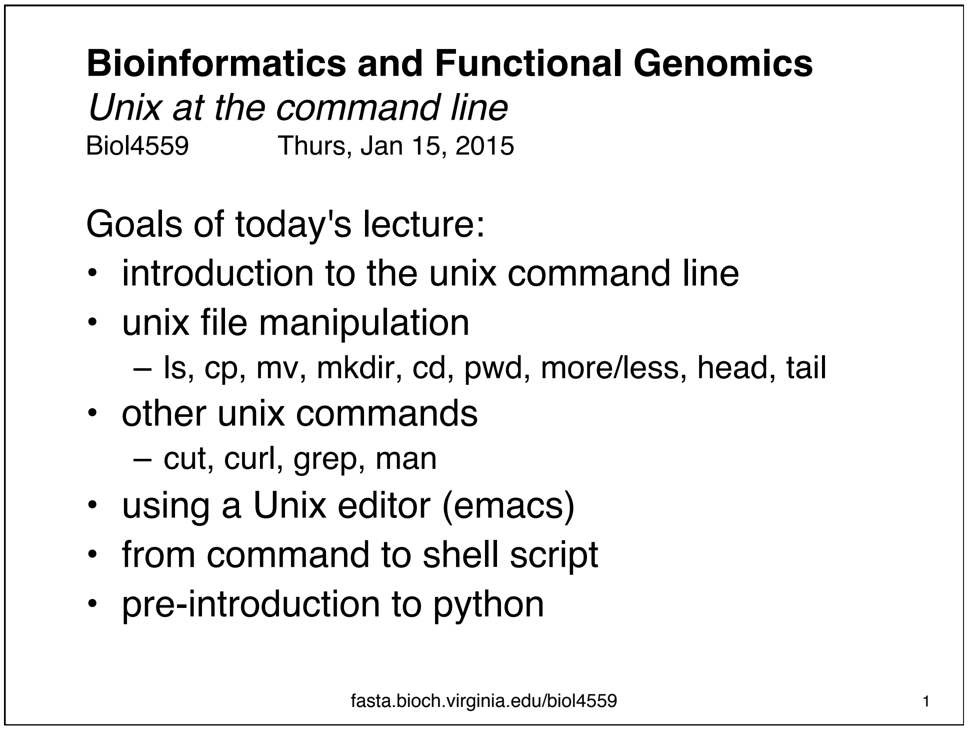


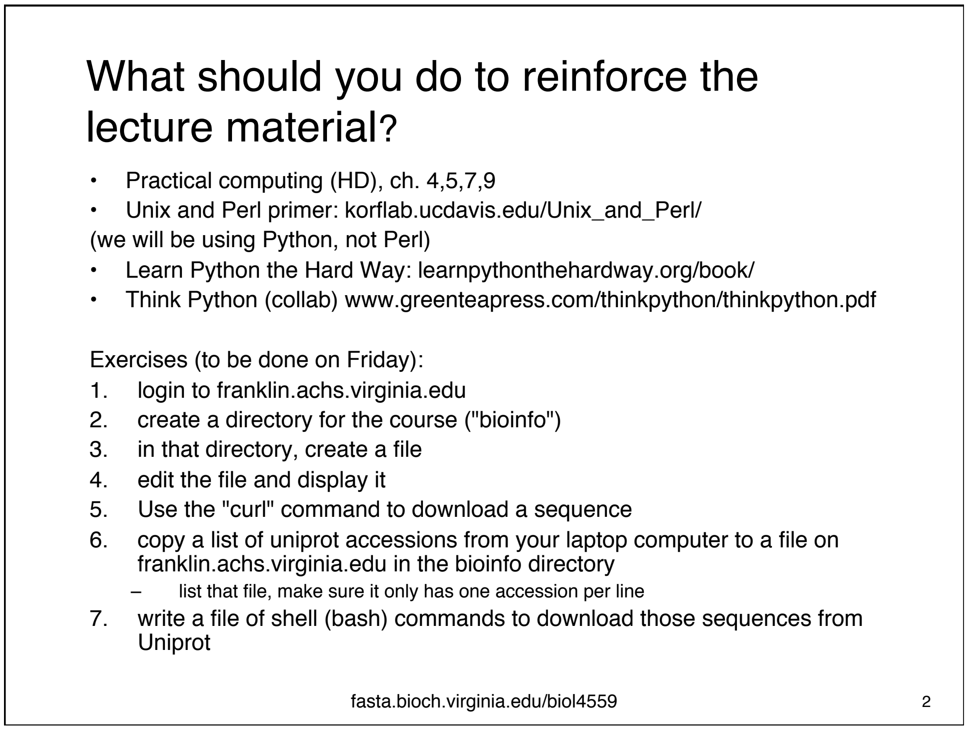
Bioinformatics and Functional Genomics

Unix at the command line

Biol4559 Thurs, Jan 15, 2015

Goals of today's lecture:
- introduction to the unix command line
- unix file manipulation
  - ls, cp, mv, mkdir, cd, pwd, more/less, head, tail
- other unix commands
  - cut, curl, grep, man
- using a Unix editor (emacs)
- from command to shell script
- pre-introduction to python

fasta.bioch.virginia.edu/biol4559 1



What should you do to reinforce the lecture material?

- Practical computing (HD), ch. 4,5,7,9
- Unix and Perl primer: korflab.ucdavis.edu/Unix_and_Perl/

(we will be using Python, not Perl)
- Learn Python the Hard Way: learnpythonthehardway.org/book/
- Think Python (collab) www.greenteapress.com/thinkpython/thinkpython.pdf

Exercises (to be done on Friday):
1. login to franklin.achs.virginia.edu
2. create a directory for the course ("bioinfo")
3. in that directory, create a file
4. edit the file and display it
5. Use the "curl" command to download a sequence
6. copy a list of uniprot accessions from your laptop computer to a file on franklin.achs.virginia.edu in the bioinfo directory
   - list that file, make sure it only has one accession per line
7. write a file of shell (bash) commands to download those sequences from Uniprot

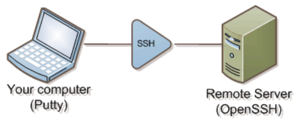
fasta.bioch.virginia.edu/biol4559 2

Computing environments

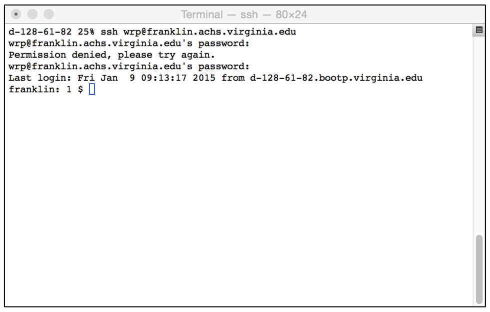
- UNIX computing: the command line
  - "shell" environment, built-in tools
  - infinitely extensible: download/install tools
    - most bioinformatics algorithms/tools are implemented as UNIX command line utilities or libraries
    - or, write your own algorithms/tools from scratch
  - highly automatable by scripting (python, Python, etc.)
  - interoperation between tools only limited by your ability to glue together input/output formats
  - almost entirely free access to tools
- demo

fasta.bioch.virginia.edu/biol4559 3

Using `franklin.achs.virginia.edu`



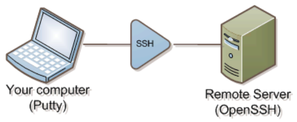
On MacOS:
1. Open "terminal"
2. `ssh your_its_id@franklin.achs.virginia.edu`



```
d-128-61-82 25% ssh wrp@franklin.achs.virginia.edu
wrp@franklin.achs.virginia.edu's password:
Permission denied, please try again.
wrp@franklin.achs.virginia.edu's password:
Last login: Fri Jan  9 09:13:17 2015 from d-128-61-82.bootp.virginia.edu
franklin: 1 $
```

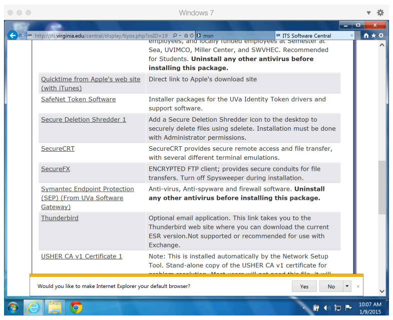
fasta.bioch.virginia.edu/biol4559 4

Using `franklin.achs.virginia.edu`



On Windows:

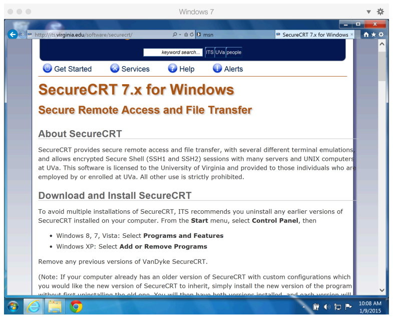
1. Download `"SecureCRT"` from `www.its.virginia.edu/central;` install
2. connect to `franklin.achs.virginia.edu`



	employees, and locally funded employees at Semester at Sea, UVIMCO, Miller Center, and SWVHEC. Recommended for Students. **Uninstall any other antivirus before installing this package.**
Quicktime from Apple's web site (with iTunes)	Direct link to Apple's download site
SafeNet Token Software	Installer packages for the UVa Identity Token drivers and support software.
Secure Deletion Shredder 1	Add a Secure Deletion Shredder icon to the desktop to securely delete files using sdelete. Installation must be done with Administrator permissions.
SecureCRT	SecureCRT provides secure remote access and file transfer, with several different terminal emulations.
SecureFX	ENCRYPTED FTP client; provides secure conduits for file transfers. Turn off Spysweeper during installation.
Symantec Endpoint Protection (SEP) (From UVa Software Gateway)	Anti-virus, Anti-spyware and firewall software. **Uninstall any other antivirus before installing this package.**
Thunderbird	Optional email application. This link takes you to the Thunderbird web site where you can download the current ESR version.Not supported or recommended for use with Exchange.
USHER CA v1 Certificate 1	Note: This is installed automatically by the Network Setup Tool. Stand-alone copy of the USHER CA v1 certificate for

Would you like to make Internet Explorer your default browser? Yes No

10:07 AM 1/9/2015



SecureCRT 7.x for Windows

Secure Remote Access and File Transfer

About SecureCRT

SecureCRT provides secure remote access and file transfer, with several different terminal emulations, and allows encrypted Secure Shell (SSH1 and SSH2) sessions with many servers and UNIX computers at UVa. This software is licensed to the University of Virginia and provided to those individuals who are employed by or enrolled at UVa. All other use is strictly prohibited.

Download and Install SecureCRT

To avoid multiple installations of SecureCRT, ITS recommends you uninstall any earlier versions of SecureCRT installed on your computer. From the **Start** menu, select **Control Panel**, then

- Windows 8, 7, Vista: Select **Programs and Features**
- Windows XP: Select **Add or Remove Programs**

Remove any previous versions of VanDyke SecureCRT.

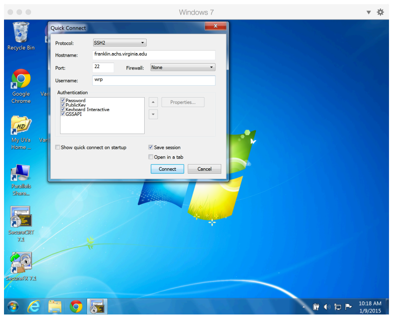
(Note: If your computer already has an older version of SecureCRT with custom configurations which you would like the new version of SecureCRT to inherit, simply install the new version of the program

10:08 AM 1/9/2015

Using `franklin.achs.virginia.edu`

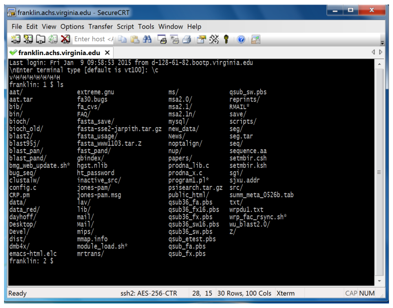
On Windows:

1. Download `"SecureCRT"` from `www.its.virginia.edu/central;` install
2. connect to `franklin.achs.virginia.edu`
   (in options/session, terminal=xterm, no delete/backspace mapping)



Protocol: SSH2
Hostname: franklin.achs.virginia.edu
Port: 22 Firewall: None
Username: wrp
Authentication
✓ Password
✓ PublicKey
✓ Keyboard Interactive
✓ GSSAPI
☐ Show quick connect on startup
✓ Save session
☐ Open in a tab

10:18 AM 1/9/2015


franklin.achs.virginia.edu - SecureCRT
File Edit View Options Transfer Script Tools Window Help
Enter host
franklin.achs.virginia.edu

```
Last login: Fri Jan  9 09:58:53 2015 from d-128-61-82.bootp.virginia.edu
\nEnter terminal type [default is vt100]: \c
v^H^H^H^H^H^H^H
franklin: 1 $ ls
aat/               extreme.gnu                ms/               qsub_sw.pbs
aat.tar            fa30.bugs                  msa2.0/           reprints/
bib/               fa_cvs/                    msa2.1/           RMAIL*
bin/               FAQ/                       msa2.1n/          save/
bioch/             fasta_save/                mysql/            scripts/
bioch_old/         fasta-sse2-jarpith.tar.gz  new_data/         seg/
blast2/            fasta_usage/               News/             seg.tar
blast95j/          fasta_www1103.tar.Z        noptalign/        seq/
blast_pan/         fast_pand/                 nup/              sequence.aa
blast_pand/        gbindex/                   papers/           setmbir.csh
bmg_web_update.sh* hgst.nlib                  prodna_lib.c      setmbir.ksh
bug_seq/           ht_password                prodna_x.c        sgi/
clustalw/          inactive_src/              program1.pl*      sjxu.addr
config.c           jones-pam/                 psisearch.tar.gz  src/
CRP.pm             jones-pam.msg              public_html/      summ_meta_0526b.tab
data/              lav/                       qsub36_fa.pbs     txt/
data_red/          lib/                       qsub36_fx16.pbs   wrpdu1.txt
dayhoff/           mail/                      qsub36_fx.pbs     wrp_fac_rsync.sh*
Desktop/           Mail/                      qsub36_sw16.pbs   wu_blast2.0/
Devel/             mips/                      qsub36_sw.pbs     Z/
dist/              mmap.info                  qsub_etest.pbs
dmb4x/             module_load.sh*            qsub_fa.pbs
emacs-html.elc     mrtrans/                   qsub_fx.pbs
franklin: 2 $
```

Ready   ssh2: AES-256-CTR   28, 15   30 Rows, 100 Cols   Xterm   CAP NUM

Using `franklin.achs.virginia.edu`

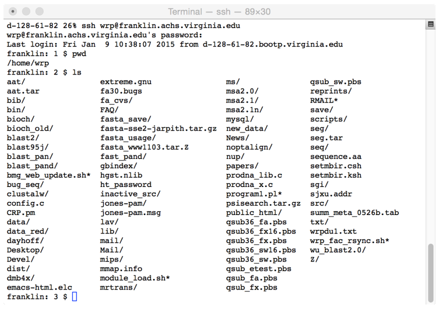
To reset password: its.virginia.edu/accounts/changelost.htm

After logging in:

1. to see current location: `pwd`
2. to list files in directory: `ls`
3. logout: `^D` (ctrl-D) or "`exit`"

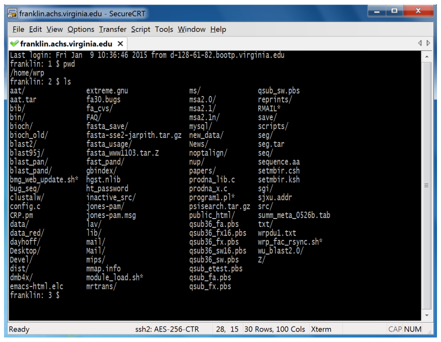
   ^C (ctrl-C) for emergencies

MacOS terminal



```
d-128-61-82 26% ssh wrp@franklin.achs.virginia.edu
wrp@franklin.achs.virginia.edu's password:
Last login: Fri Jan  9 10:38:07 2015 from d-128-61-82.bootp.virginia.edu
franklin: 1 $ pwd
/home/wrp
franklin: 2 $ ls
aat/                 extreme.gnu                ms/                qsub_sw.pbs
aat.tar              fa30.bugs                  msa2.0/            reprints/
bib/                 fa_cvs/                    msa2.1/            RMAIL*
bin/                 FAQ/                       msa2.1n/           save/
bioch/               fasta_save/                mysql/             scripts/
bioch_old/           fasta-sse2-jarpith.tar.gz  new_data/          seg/
blast2/              fasta_usage/               News/              seg.tar
blast95j/            fasta_www1103.tar.Z        noptalign/         seq/
blast_pan/           fast_pand/                 nup/               sequence.aa
blast_pand/          gbindex/                   papers/            setmbir.csh
bmg_web_update.sh*   hgst.nlib                  prodna_lib.c       setmbir.ksh
bug_seq/             ht_password                prodna_x.c         sgi/
clustalw/            inactive_src/              program1.pl*       sjxu.addr
config.c             jones-pam/                 psisearch.tar.gz   src/
CRP.pm               jones-pam.msg              public_html/       summ_meta_0526b.tab
data/                lav/                       qsub36_fa.pbs      txt/
data_red/            lib/                       qsub36_fx16.pbs    wrpdu1.txt
dayhoff/             mail/                      qsub36_fx.pbs      wrp_fac_rsync.sh*
Desktop/             Mail/                      qsub36_sw16.pbs    wu_blast2.0/
Devel/               mips/                      qsub36_sw.pbs      Z/
dist/                mmap.info                  qsub_etest.pbs
dmb4x/               module_load.sh*            qsub_fa.pbs
emacs-html.elc       mrtrans/                   qsub_fx.pbs
franklin: 3 $ 
```

Windows SecureCRT


franklin.achs.virginia.edu - SecureCRT
File Edit View Options Transfer Script Tools Window Help
franklin.achs.virginia.edu
Ready
ssh2: AES-256-CTR
28, 15
30 Rows, 100 Cols
Xterm
CAP NUM

```
Last login: Fri Jan  9 10:36:46 2015 from d-128-61-82.bootp.virginia.edu
franklin: 1 $ pwd
/home/wrp
franklin: 2 $ ls
aat/                 extreme.gnu                ms/                qsub_sw.pbs
aat.tar              fa30.bugs                  msa2.0/            reprints/
bib/                 fa_cvs/                    msa2.1/            RMAIL*
bin/                 FAQ/                       msa2.1n/           save/
bioch/               fasta_save/                mysql/             scripts/
bioch_old/           fasta-sse2-jarpith.tar.gz  new_data/          seg/
blast2/              fasta_usage/               News/              seg.tar
blast95j/            fasta_www1103.tar.Z        noptalign/         seq/
blast_pan/           fast_pand/                 nup/               sequence.aa
blast_pand/          gbindex/                   papers/            setmbir.csh
bmg_web_update.sh*   hgst.nlib                  prodna_lib.c       setmbir.ksh
bug_seq/             ht_password                prodna_x.c         sgi/
clustalw/            inactive_src/              program1.pl*       sjxu.addr
config.c             jones-pam/                 psisearch.tar.gz   src/
CRP.pm               jones-pam.msg              public_html/       summ_meta_0526b.tab
data/                lav/                       qsub36_fa.pbs      txt/
data_red/            lib/                       qsub36_fx16.pbs    wrpdu1.txt
dayhoff/             mail/                      qsub36_fx.pbs      wrp_fac_rsync.sh*
Desktop/             Mail/                      qsub36_sw16.pbs    wu_blast2.0/
Devel/               mips/                      qsub36_sw.pbs      Z/
dist/                mmap.info                  qsub_etest.pbs
dmb4x/               module_load.sh*            qsub_fa.pbs
emacs-html.elc       mrtrans/                   qsub_fx.pbs
franklin: 3 $
```

UNIX file editors

- UNIX newlines are "`\n`"
  - PC is "`\r\n`"; Mac is "`\r`" (sometimes);
- Use a UNIX editor on UNIX files:
  - `nano`
  - `emacs` vs. `vi`/`vim`
  - do not use: Word, NotePad/WordPad, TextEdit, etc.
- every editor has pros and cons (focus on nano and emacs if starting out)

filesystem navigation

- UNIX filenames are "case-sensitive"

  `seq.file != Seq.file`

  – lower case only, only "a-z_0-9" (avoid '/', '[]')
- `cd` – change directory
- `pwd` – print working directory (current dir.)
- `ls` – list files
- `pushd`/`popd` – `cd`, but remember stack
- `find` – search through filesystem
- `basename`/`dirname` – extract filename pieces

filesystem manipulation

- `cp` – copy files
- `mv` – move files
- `rm` – remove files
- `rmdir` – remove directories
- `touch` – make a new, empty file
- `mkdir` – make a new, empty directory

file inspection

- `more` – read/browse through a file/stdin
- `cat` – dump file contents to stdout
- `head`/`tail` – print first/last N lines
- `od` – look at the raw data
- `sort` – sort the lines in the file
- `uniq` – report unique lines
- `cut` – extract specific columns
- `grep` – search for matching lines
- `wc` – count words/lines/characters

UNIX permissions

- `chmod` – change the permissions on a file/dir
- `chown` – change the ownership of a file/dir
- `chgrp` – change the group of a file/dir

the UNIX `$PATH`

Unix uses the $PATH variable to find programs.
Programs in the $PATH can be found by name:

- `blastp -help`
- On `franklin.achs` your path should include `/seqprg/bin`
- `echo $PATH`

```
.:/home/wrp/bin:/seqprg/bin:/usr/NX/bin:/usr/
kerberos/bin:/usr/local/bin:/bin:/usr/bin
```

UNIX host status

- `top/ps` – what processes/apps are running
- `kill` – force-quit running processes/apps

- `df -h` – available disk resources
- `du` – disk space usage

other UNIX commands

- `builtins` – list available shell commands
- `which/where` – find path of commands
- `time` – measure how long something take
- `echo/tee` – print/report text
- `wget/curl` – download files
- `gzip/gunzip/bunzip/zcat` – compressed files
- `ssh/scp` – login/copy to/from remote hosts
- `history` – what have I done previously
- `man` – get help

redirection, pipes, replacements

- `>` - redirect `stdout` into file, replace existing
- `>>` - redirect `stdout` into file, appending
- `|` - redirect/pipe stdout to stdin of next command
- `` `backticks` ``- replace with captured `stdout`

globs

- `*` wildcard matching
- `{a,b,c}` multiple choice
- `[a-c],[1-5,9]` range/set choice
- `^` negation

- `ls -l *.bam`
- `ls chr[1-23,X,Y].bed`

environment variables

- `$USER` – who you are
- `$SHELL` – what shell you are running
- `$PWD` – your current working dir
- `$PATH` – where the shell will go to look for commands
- `$EDITOR` – your default editor

- set in your `.cshrc`/`.bashrc`, `set`/`setenv`

UNIX editors: learn (at least) one

- `nano`
  - simple, easy
  - no mouse, use arrow keys
  - how to quit: ctrl-X (all commands at screen bottom)
- `emacs`
  - not so simple to use
  - incredibly versatile, customizable, programmable
  - how to quit: ctrl-X ctrl-C
- `vi`
  - not so simple to use
  - guaranteed to be on any UNIX machine
  - often the default `$EDITOR`
  - how to quit: [colon]q![enter]

Beginning emacs

```
sh> emacs
^x^c exit
sh> emacs filename
type some stuff
^f,^b,^p,^n forwd,back char, prev, next
  line
^x^s      save it
^x^c exit
sh>
```

Intermediate emacs

```
sh> emacs myscript.py
^s, ^r search forward, reverse
^a, ^e start, end of line
 esc = M-
M-<, M-> start, end of buffer
M-%   query-replace
^k kill-line (and put in kill buffer)
^k^k  delete line and linefeed (EOL)
^y (yank - insert kill buffer)
^x 2, ^x 1, ^x o (multiple windows)
^u    (repeat number)
^h (help,^h-t tutorial, ^h-a apropos)
```

Transferring Files

- Always initiate transfer from desktop machine (franklin.achs.virginia.edu has a "known" name and address, your laptop does not)
- MacOS:
  - open terminal
  - cd to directory with data file

```
scp file.data your_id@franklin.achs.virginia.edu:~/bioinfo/
```

- Windows:
  - download and use "SecureFX" (menu driven)

Download a set of accessions from www.uniprot.org (one per line) and transfer to franklin.achs

(bash) shell scripts

- files ending with .sh suffix
- shebang: `#!/bin/bash` or `#!/bin/sh`
- useful to capture (potentially long) history of UNIX commands into a reproducible analysis
  - you will always need to repeat your analysis
  - you will never remember all the necessary steps
- with some modification, your script can be made generic, and reusable for other data

Downloading sequences (from the command line)

```
Uniprot – use accession: P09488  (not GSTM1_HUMAN)

curl http://www.uniprot.org/uniprot/P09488.fasta

>sp|P09488|GSTM1_HUMAN Glutathione S-transferase Mu 1 OS=Homo sapiens
GN=GSTM1 PE=1 SV=3
MPMILGYWDIRGLAHAIRLLLEYTDSSYEEKKYTMGDAPDYDRSQWLNEKFKLGLDFPNL
PYLIDGAHKITQSNAILCYIARKHNLCGETEEEKIRVDILENQTMDNHMQLGMICYNPEF
EKLKPKYLEELPEKLKLYSEFLGKRPWFAGNKITFVDFLVYDVLDLHRIFEPKCLDAFPN
LKDFISRFEGLEKISAYMKSSRFLPRPVFSKMAVWGNK
```

shell scripts are commands

- shell scripts can simply be copies of commands you have run:

```
curl http://www.uniprot.org/uniprot/P09488.fasta > gstm1_hs.fa
curl http://www.uniprot.org/uniprot/P28161.fasta > gstm2_hs.fa
curl http://www.uniprot.org/uniprot/P21266.fasta > gstm3_hs.fa
curl http://www.uniprot.org/uniprot/Q03013.fasta > gstm4_hs.fa
curl http://www.uniprot.org/uniprot/P46439.fasta > gstm5_hs.fa
```

- if `download_uniprot_gstm.sh` contains those five lines, you get the same result:

```
sh download_uniprot_gstm.sh
```

  - what would happen if you did not send the "`curl`" output to a specific file name?
  - how would you put all these sequences in one file?

control flow statements

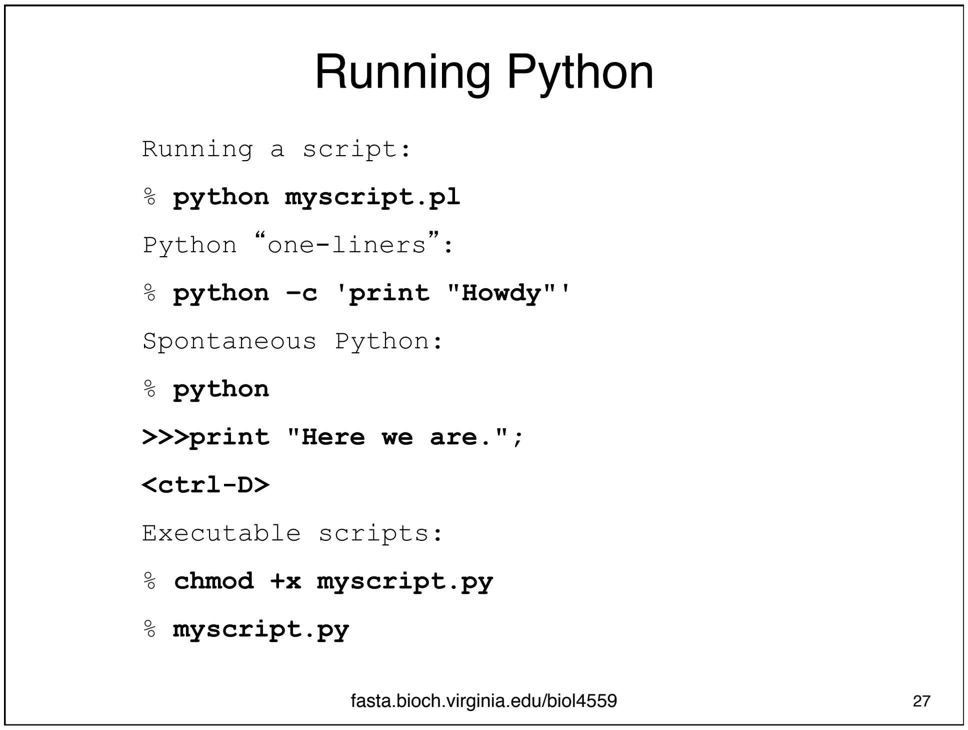
- `for name in […] ; do […] ; done`
  - do something for each item in a list
- `if […] ; then […] ;`
  `elif […]; then […];`
  `else […] fi`
  - specify behavior depending on conditions
- variables (name) and loops (for) reduce typing:

```
sh $ for acc in P09488 P28161 P21266 Q03013 P46439 ;
> do curl http://www.uniprot.org/uniprot/${acc}.fasta;
> done
sh $ for acc in `cat gstm.accs` ;
> do curl http://www.uniprot.org/uniprot/${acc}.fasta;
> done
```

backtics

alternative scripting languages

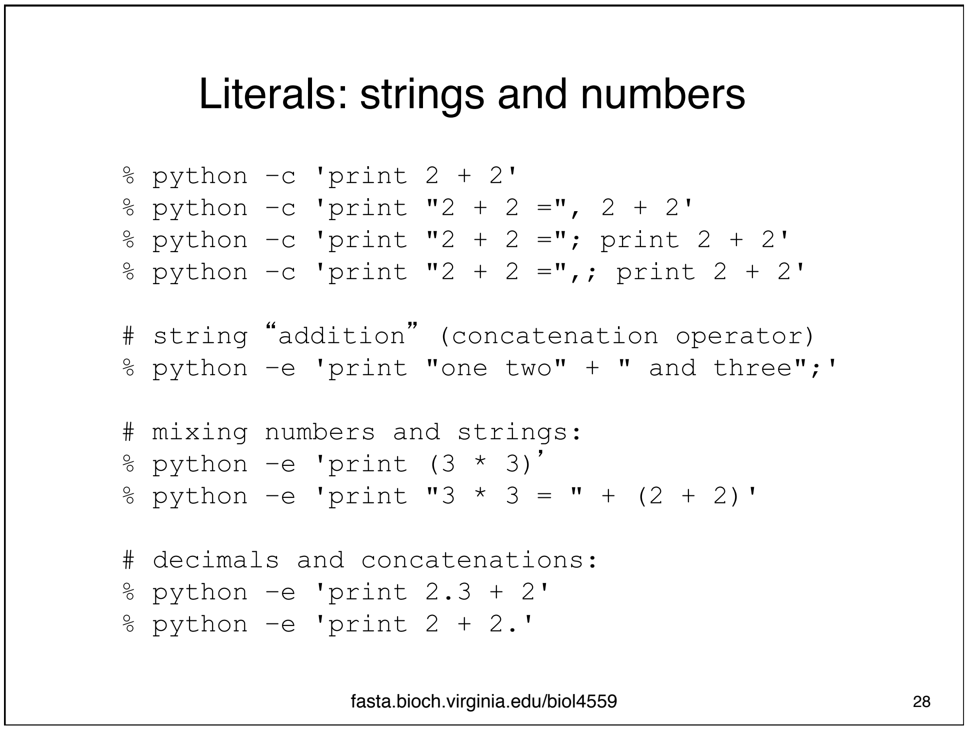
- Perl
  - once the mainstay of WWW/CGI programming
  - long history == lots of reusable packages
- PHP
  - mainly limited to dynamic WWW pages
- Python
  - extremely popular (used in this class, ?easier? to learn)
- Ruby
  - compact, expressive
- …



Running Python

```
Running a script:
% python myscript.pl
Python “one-liners”:
% python -c 'print "Howdy"'
Spontaneous Python:
% python
>>>print "Here we are.";
<ctrl-D>
Executable scripts:
% chmod +x myscript.py
% myscript.py
```

fasta.bioch.virginia.edu/biol4559 27



Literals: strings and numbers

```
% python -c 'print 2 + 2'
% python -c 'print "2 + 2 =", 2 + 2'
% python -c 'print "2 + 2 ="; print 2 + 2'
% python -c 'print "2 + 2 =",; print 2 + 2'

# string “addition” (concatenation operator)
% python -e 'print "one two" + " and three";'

# mixing numbers and strings:
% python -e 'print (3 * 3)’
% python -e 'print "3 * 3 = " + (2 + 2)'

# decimals and concatenations:
% python -e 'print 2.3 + 2'
% python -e 'print 2 + 2.'
```

fasta.bioch.virginia.edu/biol4559 28

Python vs. bash scripts

- "`.py`" file extension, e.g. "`myScript.py`"
- begins with a "shebang"

```
#!/usr/bin/python
#!/seqprg/bin/python (on franklin.achs)
```

- "`.py`" scripts need `chmod +x` to be executable:

  invoked with python: `python myScript.py`

  or directly: `./myScript.py`

Python variables

- Like many scripted languages, python has several data types (numeric, sequence, set, class, etc). We will be using three in this class:
  - numeric (integers and floats) `four=4; pi=3.14`
  - sequences (strings, arrays), indexed starting at 0

    ```
    seq="ACGT"; print seq[1]
    arr=(1,4,9,16,25); print arr[2]
    ```
  - dicts (key, value pairs, aka "hashes")

    ```
    seq_entry = {"acc":"P09488",
            "seq":"MPMILGYWDIRGLAHAIRLL"}
    print seq_entry["acc"]; print seq_entry["seq"][0:3]
    ```
- Variables are not declared in advance; scalars (numerics), sequences (strings, arrays), and dict {} variables all look the same. Consider using naming conventions to distinguish them.

our first Python script

```
#!/seqprg/bin/python
# or #!/usr/bin/python
# or #!/usr/bin/python26

import sys

print sys.version

name="Bill"

print "my name is: "+name
```

Tell the shell this is a python script

use sys functions

print the python version

assign the string "Bill" to the variable "name"

print out the label and variable "name"

our first Python script

```
franklin: $ python myscript.py
2.4.3 (#1, Jan  9 2013, 06:47:03)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-54)]
my name is: Bill

franklin: $ which python
/usr/bin/python

franklin: $ chmod +x myscript.py

franklin: $ myscript.py
2.7.9 (default, Jan  8 2015, 10:54:28)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-54)]
my name is: Bill
```

Why are the python versions different?

Python can act like bash

```
#!/seqprg/bin/python
# or !/usr/bin/python

import subprocess
accs=('P09488', 'P28161', 'P21266', 'Q03013', 'P46439')

for acc in accs:
  subprocess.call("curl --silent http://www.uniprot.org/uniprot/" +
                   acc + ".fasta", shell=True)
```

why "acc", not "accs"?

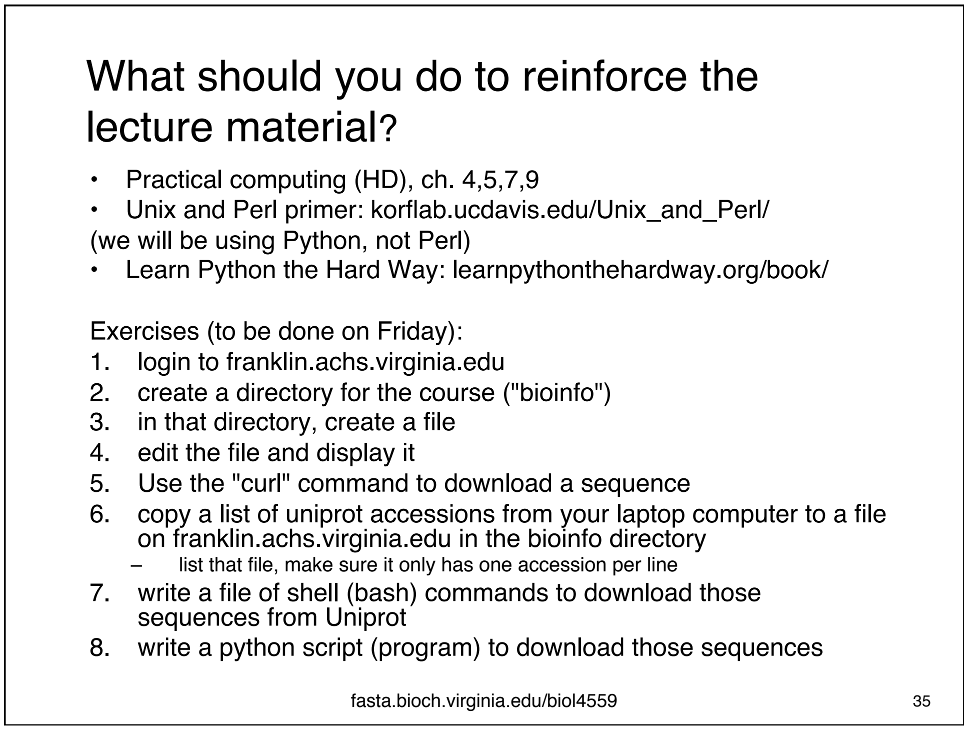
Python can be a web-browser

```
#!/seqprg/bin/python

from urllib import urlopen
base_url = 'http://www.uniprot.org/uniprot'
accs=('P09488', 'P28161', 'P21266', 'Q03013', 'P46439')

for acc in accs:
  print urlopen(base_url + acc + '.fasta').read(),
```

why use "base_url"?

Before Unix Lab (Friday)

1. Make certain your laptop can use the "Cavalier" wireless
2. Windows: download and install SecureCRT
3. Know/reset your "its" eservices password
   `its.virginia.edu/accounts/createacct.html`
4. (For work outside UVA) Install UVA Anywhere VPN
5. (optional) Try to connect (ssh) to `franklin.achs.virginia.edu`



What should you do to reinforce the lecture material?

- Practical computing (HD), ch. 4,5,7,9
- Unix and Perl primer: korflab.ucdavis.edu/Unix_and_Perl/

(we will be using Python, not Perl)

- Learn Python the Hard Way: learnpythonthehardway.org/book/

Exercises (to be done on Friday):

1. login to franklin.achs.virginia.edu
2. create a directory for the course ("bioinfo")
3. in that directory, create a file
4. edit the file and display it
5. Use the "curl" command to download a sequence
6. copy a list of uniprot accessions from your laptop computer to a file on franklin.achs.virginia.edu in the bioinfo directory
   - list that file, make sure it only has one accession per line
7. write a file of shell (bash) commands to download those sequences from Uniprot
8. write a python script (program) to download those sequences

fasta.bioch.virginia.edu/biol4559 35