CLI (command line interface)	**GUI (graphical user interface)**
● Steep learning curve ● Pure control (eg scripting) ● Cumbersome multitasking ● Speed: Hack away at keys ● Convenient remote access ● Eg: grep foo file.txt|wc -l	● Intuitive ● Limited control ● Easy multitasking ● Limited by pointing ● Bulky remote access

LINUX (HAS TREE HIERARCHY) COMMANDS

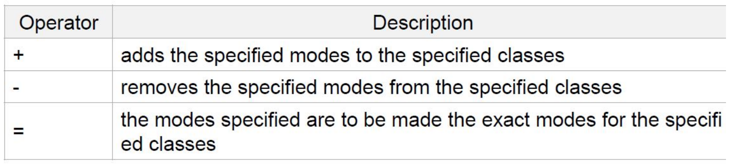
- *uname*
    - Linux
- *uname* -a
    - Linux lnxsrv07.seas.ucla.edu 3.10.0-693.2.2.el7.x86_64 #1 SMP Sat Sep 9 03:55:24 EDT 2017 x86_64 x86_64 x86_64 GNU/Linux
- *uname* -r
    - 3.10.0-693.2.2.el7.x86_64 (operating system release)
- who
    - classrel pts/0   2017-10-02 10:35 (wifi-natpool-131-179-59-47.host.ucla.edu)
    - wangmi   pts/1   2017-10-02 09:31 (cardiff.seas.ucla.edu)
    - mallett pts/2   2017-10-02 09:14 (s-169-232-245-113.resnet.ucla.edu)
- *whoami*
    - Akshara
- *man*   //q to exit, / to search
    - What manual page do you want?
- *man* whoami
    - **man -k** printf
    - Search the short descriptions and manual page names for the keyword printf as regular expression. Print out any matches. = to **apropos -r** printf**.**
- *clear*
- *echo:* -n to prevent new line at the end of the line.
    - echo $SHELL
    - /bin/bash
    - -e to enable interpretation of backslash escapes
    - echo “hey\n”->hey\n echo -e “hey\n”-> hey(new line here)
- *pwd*
- *wget:* Downloads a file from a URL (a web server)
- Find linux distribution: lsb_release -a
- *ls*:
    - -l :file mode, number of links, owner name, group name, number of bytes in the file, abbreviated month, day-of-month file was last modified, hour file last modified, minute file last modified, and the pathname.
    - -d: list only directories

    - -a: list all files including hidden ones
    - -l: show long listing including permission info
    - -s: show size of each file, in blocks
- *mv*: move a file (no undos!)
- *cp*: copy a file
- *touch*: update last modified timestamp or create a new file (-t for custom time)
- *head*: Print the first 10 lines of each FILE to standard output. With more than one FILE, precede each with a header giving the file name. With no FILE, or when FILE is -, read standard input.
- *tail*: Print last 10 lines
- *du* - estimate file space usage
    - `[akshara@lnxsrv09 ~/sophomore]$ du`
    - `8       .`
    - `[akshara@lnxsrv09 ~/sophomore]$ du -a`
    - `0       ./file1.txt`
    - `0       ./file.txt`
    - `4       ./foo.txt`
    - `0       ./bar.txt`
    - `8       .`
- *ps* - report a snapshot of the current processes. Reads virtual files in /proc
    - `[akshara@lnxsrv09 ~/sophomore]$ ps`
    - `  PID TTY          TIME CMD`
    - `32080 pts/28   00:00:00 bash`
    - `39909 pts/28   00:00:00 ps`
    - For all users: ps -a
    - More info: ps -aux
    - Executable file becomes a process when loaded in memory and has execution state and is currently running
- *kill* - terminate a process
- *diff:* -u -Output NUM (default 3) lines of unified context.
    - diff -u one two
        - --- one 2017-12-12 09:01:27.000000000 -0800
        - +++ two   2017-12-12 09:01:34.000000000 -0800
        - @@ -1 +1 @@
        - -Hello
        - \ No newline at end of file
        - +hellllo
        - \ No newline at end of file
    - diff one two
        - 1c1
        - < Hello
        - \ No newline at end of file
        - ---

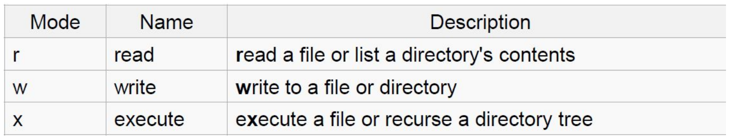
    - > hellllo
    - \ No newline at end of file
- *ln:* Links a file (defaulted to hard linking). The –s option allows for symbolic links.
  - Hard links: points to physical data, point to same inode
  - Soft links aka symbolic links (-s): points to a file
  -
- *whereis:* Will give three locations: First location –binary or executable location. Second location – source code location. Third location – man page for command
- *which* <command>: gives only where the executable file is
- *whatis* <command>:search the whatis database for complete words.
- *cmp*: bytewise comparison
- *wc*: counts number of words/bytes/lines etc
    - [akshara@lnxsrv09 ~]$ ps | wc -l
    - 4
    - [akshara@lnxsrv09 ~]$ ps -aux | wc -l
    - 531
  - **-c**, **--bytes:** print the byte counts
  - **-m**, **--chars:** print the character counts
  - **-l**, **--lines:** print the newline counts
- *sort*: sorts lines of text files
  - -u removes repeated values and gives unique lines
- *find:* Can use regex, find -name "cs*"
  - -type: type of file
  - -perm: permission of a file
  - -name: name of a file
  - -user: owner of a file
  - -maxdepth: how many levels to search
  - Eg: find . -mtime -21 -type d searches for directories last modified in the past 3 weeks from the current directory
- *grep*: Search a file, use | to search output of a command
  -
```
[akshara@lnxsrv09 ~]$ ps -aux | grep root
root         1  0.0  0.0 193360  6380 ?        Ss   Sep15   6:14
/usr/lib/systemd/systemd --switched-root --system --deserialize 21
root         2  0.0  0.0      0     0 ?        S    Sep15   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    Sep15   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   Sep15   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        S    Sep15   0:00 [kworker/u96:0]
akshara  43215  0.0  0.0 112664   976 pts/28   S+   10:36   0:00 grep
--color=auto root
```
  - Can use ^ for only beginning of file
    - [akshara@lnxsrv09 ~]$ ps -aux | grep ^root
    - **root** 1 0.0 0.0 193360 6380 ? Ss Sep15 6:14 /usr/lib/systemd/systemd --switched-root --system --deserialize 21

- **root** 2 0.0 0.0 0 0 ? S Sep15 0:00 [kthreadd]

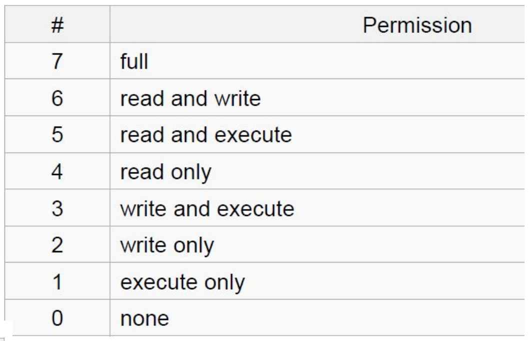
chmod: change permissions



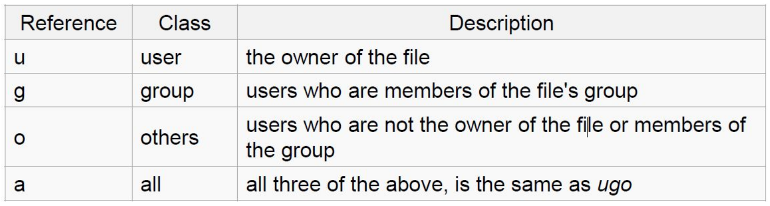
Operator	Description
+	adds the specified modes to the specified classes
-	removes the specified modes from the specified classes
=	the modes specified are to be made the exact modes for the specified classes



Mode	Name	Description
r	read	read a file or list a directory's contents
w	write	write to a file or directory
x	execute	execute a file or recurse a directory tree



#	Permission
7	full
6	read and write
5	read and execute
4	read only
3	write and execute
2	write only
1	execute only
0	none



Reference	Class	Description
u	user	the owner of the file
g	group	users who are members of the file's group
o	others	users who are not the owner of the file or members of the group
a	all	all three of the above, is the same as *ugo*

- Sticky bit:
  - (o + t) <file>
  - On shared directories, it locks files within the directory from being modified/deleted by users other than the file creator, owner of the directory or root. Even if others have write permissions to directory
  - Eg: /tmp
  - There exist special permissions such as a sticky bit (+t) to lock files from being modified within some directories to that specific reference. The (+s) when applied to executables grants access to the process that runs this file based on the owner/group of the file, rather than the user running the executable file.
- Set user id/set group id: setuid, setgid
  - (u+s, g+s) "Set user ID upon execution"
  - Run an executable with the permissions of the executable's owner or group
  - So g+s means when anyone executes this program it will run with the same permission as the file's group

Directory:
- ~: home directory
- .: current directory
- ..: parent directory

Redirection
- > file: write stdout to a file
- >> file: append stdout to a file
- < file: use contents of a file as stdin

History
- *<up arrow>*: previous command

- *<tab>*: auto-complete
- *history*: display commands history
- *!!:* replace with previous command
- *![-n]:* replace with the previous n command.
- *^old^new:* repeat last command replace old to new

WEEK 2

LOCALE

- Set of parameters that define a user's cultural preferences
  - Language
  - Country
  - Other area-specific things

locale command

Prints info about current locale environment to std output

LC_*Environment Variables

- Locale gets data from the LC_*environment variables
- Eg:
  - LC_TIME: date and time format
  - LC_NUMERIC: non-monetary numeric format
  - LC_COLLATE: Order for comparing and sorting
    - LC_COLLATE='C': sorting is in ASCII order
    - LC_COLLATE='en_US': sorting is case insensitive except when the two strings are otherwise equal and one has an uppercase letter earlier than the other.

Environment Variables

- Variables that can be accessed from any child process
- Eg:
  - HOME: path to user's home directory
  - PATH: list of directories to search in for command to execute
    - PATH will contain /usr/bin for find during hw
  - USER: the current username
  - SHELL: name of current shell
- Print value using echo: echo $VARIABLE
- Change value: export VARIABLE=...

comm:Compare 2 sorted files line by line prints to STDOUT what common lines and unique lines to each file, depends on locale

- comm [option] file1 file2
- -1 suppress first column, -2 suppress second column, -3 suppress third column

- -i case insensitive line comparison

tr: Translate or delete characters

- tr [option] SET1 [SET2]
- Character to character translation, -c for complement, -s to squeeze repeated i/p with one instance of that character, -d to delete
-
```
[akshara@lnxsrv07 ~]$ tr ab cd
ucla
uclc
haaaahaaaab
Hccccchccccd
```
- Can translate using ranges too
```
[akshara@lnxsrv07 ~]$ tr a-z A-Z
ahHHHaaaabbB
AHHHHAAAABBB
```
- To put translated output in a file and translate a preexisting file
```
[akshara@lnxsrv07 ~]$ tr a-z A-Z < example.txt >tr_output.txt
```
- You can delete characters too
```
tr -d o < example.txt
```
  - Can also delete a range of characters (a-h)
  - Can't delete a string of characters (ab)

grep: Search for word in a file, use | to search through the output of a command.Can use regex

- grep [A-Z,1-9] e.txt
  I have two cats. **I** love my cat.
  I live in **131**0 street name**,** **L**os **A**ngeles.
  CS35L is a lot of work.

- **egrep '(.)bb\1' myfile.txt**

  Find every line with 2 b's and the same character both before and after those b's.

- **who | grep -F austen** //Find where austen is logged in, -F is fixed expression

sed: sed 's/regExpr/replText/[g]' #extract and replace text
sed [options] commands [file-to-edit]

- Options: p:print, d:delete and s:substitute
- -n prevents printing of input to STDOUt
- echo $PATH | sed 's/:.*//' #remove everything after and including first colon
- echo $PWD | sed 's/^\/.*\///' #display current directory name
- **Print**
- sed 'p' test.txt =>prints every line twice, since by default sed prints input and output
- Print only line in the input file.

    - sed -n ‘3p’ test.txt
- Print line 1 through 5 in the input file.
    - sed -n ‘1,5p’ test.txt
- Print 5 lines starting from line 1.
    - sed -n ‘1,+5p’ test.txt
- Print every other line starting from line 1.
    - sed -n ‘1~2p’ test.txt
- Print lines 1,7 and 9
    - sed -n -e ‘1p’ -e ‘7p’ -e ‘9p’ test.txt

-
  ```
  sed -n '1~2p' e.txt | sed '1,2p'
  I have two cats. I love my cat.
  I have two cats. I love my cat.
  I live in 1310 street name.
  I live in 1310 street name.
  Note - e.txt contains:
  I have two cats. I love my cat.
  We are trying a new command.
  I live in 1310 street name, Los Angeles.
  CS35L is a lot of work.
  ```
- **Delete**
- sed ‘3d’ test.txt //delete line 3, prints everything else out
- sed ‘2,4d’ test.txt //delete lines 2 and 4
- sed ‘/start/,/end/d’ filename.txt //delete everything between 2 patterns
- **Substitute**
- By default replaces only first occurrence of pattern, add g to end to replace all occurrences (`sed 's/cat/dog/g' e.txt vs sed 's/cat/dog/' e.txt)`
- **-i**: edit the file in place

-
  ```
  sed 's/[0-9]*/NA/g' e.txt
  NAINA NAhNAaNAvNAeNA NAtNAwNAoNA NAcNAaNAtNAsNA.NA NAINA NAlNAoNAvNAeNA NAmNAyNA
  NAcNAaNAtNA.NA
  ```

-
  ```
  sed 's/[0-9]/NA/g' e.txt
  I have two cats. I love my cat.
  We are trying a new command.
  I live in NANANANA street name, Los Angeles.
  ```

-
  ```
  sed 's/[0-9]+/NA/g' e.txt
  I have two cats. I love my cat.
  We are trying a new command.
  I live in 1310 street name, Los Angeles.
  ```

-
  ```
  sed 's/[^A-Za-z][0-9][0-9]*/NA/g' e.txt
  I have two cats. I love my cat.
  We are trying a new command.
  I live inNA street name, Los Angeles.
  CSNAL is a lot of work.
  ```
- To include what you have extracted as part of the substitute use &:

-
  ```
  sed -E 's/([0-9][0-9]* )/(&) /g' e.txt
  I live in (1310 ) street name, Los Angeles.
  ```

Decompress: tar -zvxf filename.tar.gz

Patching: patch -pnum < patch.txt

- pnum is the number of slashes we want to remove from the path so that we apply the patch to that file respectively.
- Create a patch: diff -u orig updated
- diff Unified Format
  - diff -u file1 file2
  - --- path/to/original/file
  - +++ path/to/modified/file
  - @@-l,s+l,s @@
    - @@: beginning of a chunk
    - l: beginning line number
    - s: number of lines the change chunk applies to for each file
    - A line with a:
      - - sign was deleted from the original
      - + sign was added to original
      - stayed the same

C Program->Executable

- **Preprocessing:** Add #includes and remove comments from source files
- **Compilation:** Convert to .s with assembly
- **Assembler:** Reads the assembly code and produces assembly listing with offsets in it for the linking stage, which is all stored in a .o (obj file).
- **Linking:** Combine archive/.so(library) files and create executable

Command Line Compilation

- give g++ the names of files for compilation for C++ and gcc for C
- shop.c:#includes shoppingList.h and item.h
- shoppingList.c;#includes shoppingList.h
- item.c;#includes item.h
- To compile: **g++ shop.c shoppingList.c item.c -o shop**
  - By default output file called **a.out**
  - #include ensures that the header is added automatically to the beginning since it is a *preprocessor directive*
- If you make only a small change to item.c, don't recompile everything
  - Solution: produce a separate object code file for each source file
    - `gcc -c item.c`
    - `gcc -c shoppingList.c`
    - `gcc -c shop.c`
    - `gcc item.o shoppingList.o shop.o -o shop (combine)`
    - Less work for compiler, but more commands
  - **-c: only compile, doesn't link**
  - **-o *file*: Put o/p in file *file,* a.out by default**
  - Issues with this solution:
    - Hard to keep track when large number of input files
- If you change one of the headers or source files, rerun command to generate a new executable if large change, have to recompile everything that has a that header file

Makefile

```
#Makefile - A Basic Example, indentation is important
all: shop #usually first, all, item.o, shoppingList.o etc are labels
item.o: item.c item.h                        #rule
        gcc -c item.c                        #rule continued
shoppingList.o: shoppingList.c item.h shoppingList.h
                gcc -c shoppingList.c
shop.o: shop.c item.h shoppingList.h
        gcc -c shop.c
shop: item.o shoppingList.o shop.o #executable being made
clean:
     rm -f item.o shoppingList.o shop.o shop
```

- make -f <blah> //use <blah> as makefile

```
#comment
rulename: dependencies
tob gcc -c item.c -o item.o (commands)
shoppingList.o: item.o shoppingList.c shoppingList.h
    <tab>       gcc -c shoppingList.c item.o -o shoppingList.o
-------
-------
```

- Runs hello first by default

```
hello:
        echo "Hello World"
bye:
        echo "bye bye"
#these are dummy rules
```

-

```
hello:
        bye #run bye first
        echo "Hello World"
bye:
        echo "bye bye"
#these are dummy rules
```

-

```
item.o: item.c item.h
            gcc -c item.c -o item
bye:
         echo "bye bye"
```

- Order doesn't matter, make makes first one by default
- Dependency order matters

```
CC = gcc
CFLAGS = $(OPTIMIZE) -g3 -Wall -Wextra -march=native -mtune=native -mrdrnd

default: randall randmain randlibhw.so randlibsw.so

randall: randall.c
```

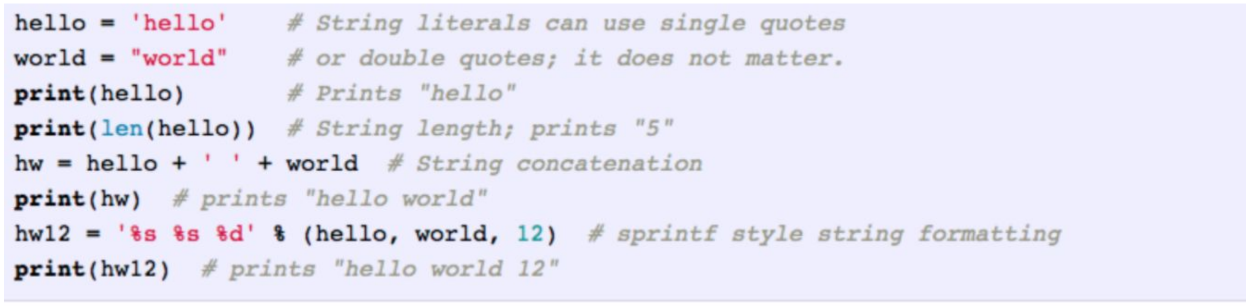
```
       $(CC) $(CFLAGS) randall.c -o $@

# randmain.mk contains instructions for building
# randmain, randlibhw.so, and randlibsw.so.
-include randmain.mk

skeleton: dlskeleton.tgz
skeleton_files = COPYING Makefile randall.c randcpuid.h randlib.h
dlskeleton.tgz: $(skeleton_files)
       tar -czf $@ --mode=a-w $(skeleton_files)

submission: dlsubmission.tgz
submission_files = lab.txt randmain.mk \
  randcpuid.c randlibhw.c randlibsw.c randmain.c \
  $(skeleton_files)
dlsubmission.tgz: $(submission_files)
       tar -czf $@ $(submission_files)

.PHONY: default clean skeleton submission

clean:
       rm -f *.o *.so *.so.* *.tgz randall randmain
```

```
randlibsw.so:randlibsw.c
       $(CC) $(CFLAGS) -c randlibsw.c -fPIC -shared -o randlibsw.o

randlibhw.so:randlibhw.c
       $(CC) $(CFLAGS) -c randlibhw.c -o -shared -fPIC randlibhw.o

randmain: randmain.o randcpuid.o
       $(CC) $(CFLAGS) randmain.o randcpuid.o -ldl -Wl,-rpath=$PWD -o randmain
```

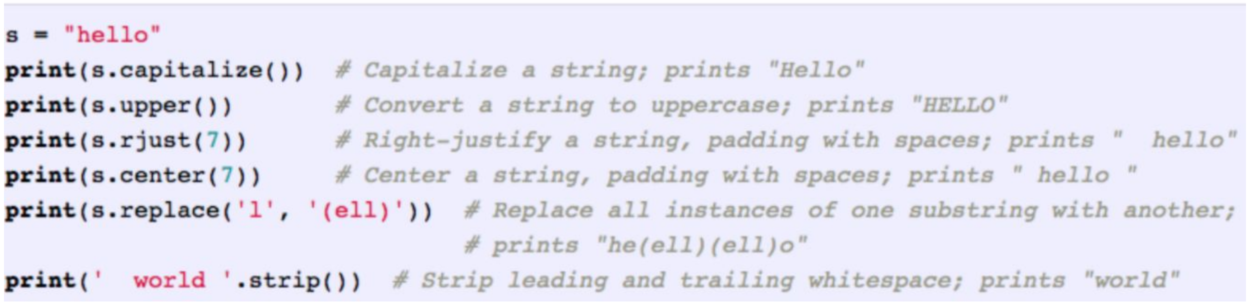
```
OBJS = MovieList.o Movie.o NameList.o Name.o Iterator.o
CC = g++
DEBUG = -g
CFLAGS = -Wall -c $(DEBUG)
LFLAGS = -Wall $(DEBUG)

p1 : $(OBJS)
    $(CC) $(LFLAGS) $(OBJS) -o p1

MovieList.o : MovieList.h MovieList.cpp Movie.h NameList.h Name.h Iterator.h
    $(CC) $(CFLAGS) MovieList.cpp

Movie.o : Movie.h Movie.cpp NameList.h Name.h
    $(CC) $(CFLAGS) Movie.cpp

NameList.o : NameList.h NameList.cpp Name.h
    $(CC) $(CFLAGS) NameList.cpp

Name.o : Name.h Name.cpp
    $(CC) $(CFLAGS) Name.cpp

Iterator.o : Iterator.h Iterator.cpp MovieList.h
    $(CC) $(CFLAGS) Iterator.cpp

clean:
	\rm *.o *~ p1

tar:
	tar cfv p1.tar Movie.h Movie.cpp Name.h Name.cpp NameList.h \
		NameList.cpp  Iterator.cpp Iterator.h
```

Python

- **Datatypes** (x=True->declare bool, y=4->declare int)
    - bool: True, False
        - operators: or,and,not, **!=(XOR)**, type(x) //<class 'bool'>
    - int,float,strings, **None is null**

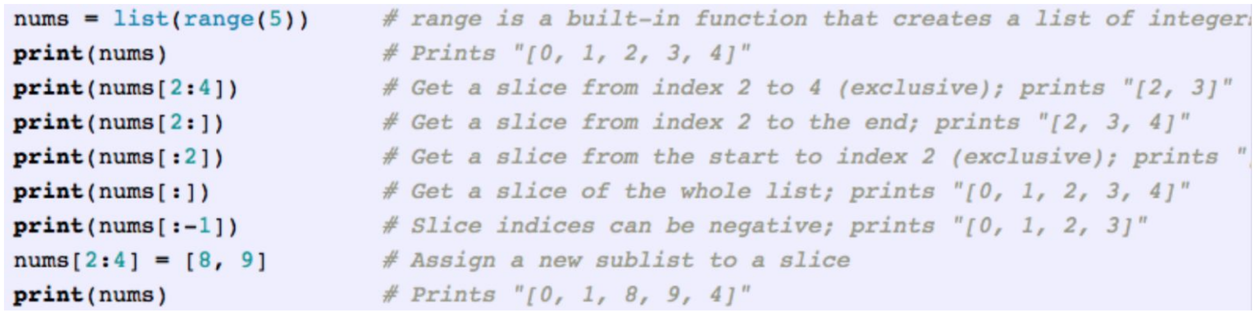


```
hello = 'hello'    # String literals can use single quotes
world = "world"    # or double quotes; it does not matter.
print(hello)       # Prints "hello"
print(len(hello))  # String length; prints "5"
hw = hello + ' ' + world  # String concatenation
print(hw)  # prints "hello world"
hw12 = '%s %s %d' % (hello, world, 12)  # sprintf style string formatting
print(hw12)  # prints "hello world 12"
```



```
s = "hello"
print(s.capitalize())  # Capitalize a string; prints "Hello"
print(s.upper())       # Convert a string to uppercase; prints "HELLO"
print(s.rjust(7))      # Right-justify a string, padding with spaces; prints "  hello"
print(s.center(7))     # Center a string, padding with spaces; prints " hello "
print(s.replace('l', '(ell)'))  # Replace all instances of one substring with another;
                                # prints "he(ell)(ell)o"
print('  world '.strip())  # Strip leading and trailing whitespace; prints "world"
```

- No ++ operator
- print -> prints stuff
- **List**: python vector, acts like a resizable array, can have different data types in it
    - **+**->merge lists,**blah[-1]**->negative indexing,**blah.append(8)**->add item
    - t = [123, 3.0, 'hello!']
    - **Slicing:Accessing sublists**

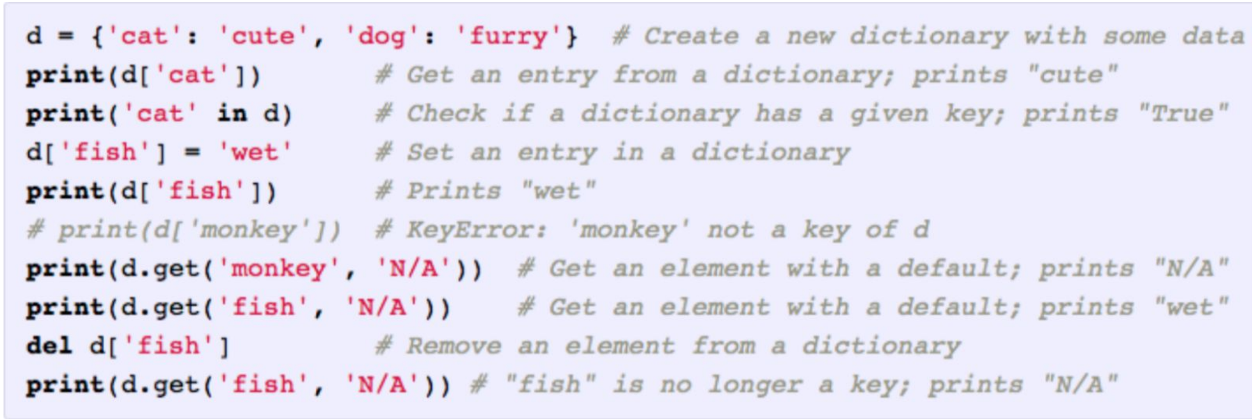


```
nums = list(range(5))     # range is a built-in function that creates a list of integer
print(nums)               # Prints "[0, 1, 2, 3, 4]"
print(nums[2:4])          # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print(nums[2:])           # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print(nums[:2])           # Get a slice from the start to index 2 (exclusive); prints "
print(nums[:])            # Get a slice of the whole list; prints "[0, 1, 2, 3, 4]"
print(nums[:-1])          # Slice indices can be negative; prints "[0, 1, 2, 3]"
nums[2:4] = [8, 9]        # Assign a new sublist to a slice
print(nums)               # Prints "[0, 1, 8, 9, 4]"
```

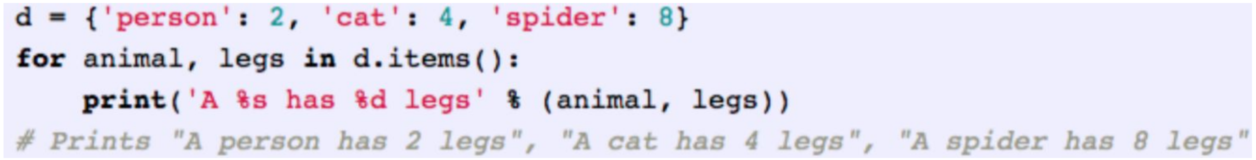
- **For loops:**
  for item in list		for i in range(len(list))		for i,pet in enumerate(pets)

print item   print i   print(‘%d: %s’%(i+1,pet))

- **List Comprehension**
  squares=[x**2 for x in nums]   even_sq=[x**2 for x in nums if x%2==0]
- **Dictionaries:** python maps



```python
d = {'cat': 'cute', 'dog': 'furry'}  # Create a new dictionary with some data
print(d['cat'])       # Get an entry from a dictionary; prints "cute"
print('cat' in d)     # Check if a dictionary has a given key; prints "True"
d['fish'] = 'wet'     # Set an entry in a dictionary
print(d['fish'])      # Prints "wet"
# print(d['monkey'])  # KeyError: 'monkey' not a key of d
print(d.get('monkey', 'N/A'))  # Get an element with a default; prints "N/A"
print(d.get('fish', 'N/A'))    # Get an element with a default; prints "wet"
del d['fish']         # Remove an element from a dictionary
print(d.get('fish', 'N/A')) # "fish" is no longer a key; prints "N/A"
```



```python
d = {'person': 2, 'cat': 4, 'spider': 8}
for animal, legs in d.items():
    print('A %s has %d legs' % (animal, legs))
# Prints "A person has 2 legs", "A cat has 4 legs", "A spider has 8 legs"
```