Shell / Python Tutorial

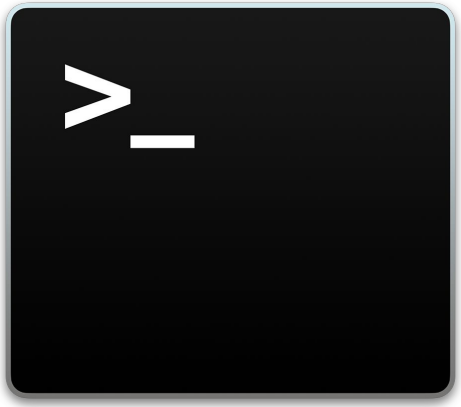
CS279 – Autumn 2017

Rishi Bedi

Shell (== console, == terminal, == command prompt)

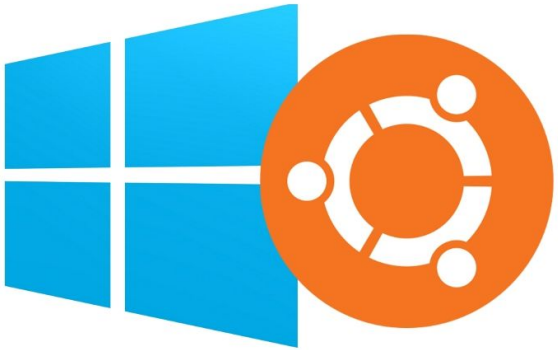
You might also hear it called “bash,” which is the most widely used shell program

macOS



Launch **Terminal**

Windows **10+**



[Windows Subsystem for Linux]

Linux

Launch **Terminal**

Getting around in Bash

cd	Change working directory	**cd** ~/some/path
ls	List all files in working directory	**ls** -ltrh
pwd	Print current working directory	**pwd**
mkdir	Create a new directory	**mkdir** /new/dir
cat	Dump out the contents of a file	**cat** file.txt
less	Preview file contents	**less** file.txt
cp	Copy a file	**cp** /path/to/old/file /path/to/new/file
mv	Move a file	**mv** /path/to/old/file /path/to/new/file

Use **man** pages to find more information about a command:
man ls

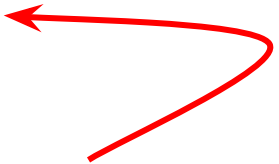
You can do *a lot* in bash

- In principle, the bash scripting language is a complete programming language
- It’s especially useful for things like...
  - Plumbing (connecting the inputs & outputs of different console programs)
  - System administration
  - Automating simple command line tasks
  - Quickly examining and editing text files
- I wouldn’t use it for…
  - Anything else

Running Python from the Shell

- There are many ways to run Python
- The most “bare-bones” is to run the `python` command in your shell

```
bash-3.2$ python
Python 2.7.5 (v2.7.5:ab05e7dd2788, May 13 2013, 13:18:45)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```



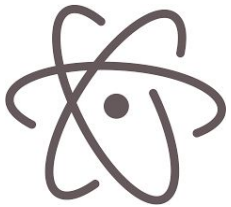
Note that we are running Python 2.7, **not** Python 3

This is a new kind of shell!
To avoid confusion, we’ll call it the **Python interpreter**
We can’t run bash commands in it, but we can execute Python statements

Running a Python Script

- The interpreter is neat, especially when you're learning Python, but we also want to be able to save programs and run them in their entirety
- Many ways to write Python programs – the easiest is to use any **text editor**

Sublime Text



Atom

Vim**

- Save your Python program, we use the ".py" extension by convention
- In your bash shell, run this command, filling in the path to your program:
  `python /path/to/your/program.py`

Python Fundamentals

Python slides adapted from Sam Redmond's CS41

- Comments
- Variables
- Output
- Strings
- Lists
- Control Flow
- Functions

Comments

In Python

```
# Single line comments start with a '#'

"""
    Multiline strings can be written
    using three "s, and are often used
    as function and module comments
"""
```

In Java / C++

```
// single line comment

/* multi line
   comment
*/
```

adapted from Sam Redmond's CS41

Variables

In Python

```
x = 5
y = x + 7
z = 3.14

name = “Rishi”


1 == 1 # => True
5 > 10 # => False

True and False # => False
not False # => True
```

In Java / C++

```
int x = 5;
int y = x + 7;
double z = 3.14;

String name = “Rishi”;   // Java
string name(“Rishi”);    // C++

1 == 1 # => true
5 > 10 # => false

true && false # => false
!(false) # => true
```

adapted from Sam Redmond’s CS41

Output

In Python

```
x = 5
print x

name = ‘Rishi’
print name + str(x)
```

In Java / C++

```
// Java:
int x = 5;
System.out.println(x);

String name = “Rishi”;
System.out.println(name + x);

// C++:
int x = 5;
cout << x << endl;

string name(“Rishi”);
cout << name << x << endl;
```

adapted from Sam Redmond’s CS41

Strings

```
greeting = 'Hello'
group = "world"

greeting + ' ' + group + '!' # => 'Hello world!'
```

```
  0 1 2 3 4 5 6
s = 'protein'
```

```
s[0]   = ‘p’
s[4]   = ‘e’
s[7]   = BAD!
s[0:3] = ‘pro’
s[4:]  = ‘ein’
```

adapted from Sam Redmond’s CS41

Lists

```
# Create a new list
empty = []
letters = ['a', 'b', 'c', 'd']
numbers = [2, 3, 5]

# Lists can contain elements of different types
mixed = [4, 5, "seconds"]

# Append elements to the end of a list
numbers.append(7)   # numbers == [2, 3, 5, 7]
numbers.append(11)  # numbers == [2, 3, 5, 7, 11]
```

adapted from Sam Redmond's CS41

Lists

```
# Access elements at a particular index
numbers[0]  # => 2
numbers[-1]  # => 11

# You can also slice lists - the same rules apply
letters[:3]  # => ['a', 'b', 'c']
numbers[1:-1]  # => [3, 5, 7]

# Lists really can contain anything - even other lists!
x = [letters, numbers]
x  # => [['a', 'b', 'c', 'd'], [2, 3, 5, 7, 11]]
x[0]  # => ['a', 'b', 'c', 'd']
x[0][1]  # => 'b'
x[1][2:]  # => [5, 7, 11]
```

There are many more data structures!

dicts are like Maps/HashMaps
sets are like Sets
tuples are immutable lists

adapted from Sam Redmond's CS41

`if` Statements

```
if some_condition:
    print 'Some condition holds'
elif other_condition:
    print 'Other condition holds'
else:
    print 'Neither condition holds'
```

or

- Each condition should evaluate to a boolean
- Zero or more elifs
- else is optional
- Python has no switch statement!

Whitespace matters, unlike in C++ or Java!

adapted from Sam Redmond's CS41

for loops

Strings and lists, amongst other things, are iterables

```
for item in iterable:
    do_something(item)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
for idx in range(0,10):
    do_something(item)
```

```
for idx in range(0,10):
    for item in iterable:
        do_something(idx, item)
```

You can nest loops like this however you'd like

adapted from Sam Redmond's CS41

Functions

```
def fn_name(param1, param2):
    value = do_something()
    return value
```

- “def” starts a function definition
- return is optional
  - if either return or its value are omitted, implicitly returns None
- Parameters have no explicit types

```
def isEven(num):
    if num % 2 == 0:
        return True
    else:
        return False

myNum = 100
if isEven(myNum):
    print str(myNum) + “ is even”
```

adapted from Sam Redmond’s CS41

Calling Library Functions

- Many functions are built-in to Python
- Some are available in the standard installation, but their **modules** need to be imported
- In general, look for built-ins / library functions before writing your own
- Example: square root function

```
import math
math.sqrt(25)
```

More Python Resources

- Stanford Python (CS41)
- Codecademy Python
- Official Documentation
- LearnXinYminutes