Lab 02 – Mini-Lecture
Scripting with Bash and Python

Prof. Brendan Kochunas

NERS/ENGR 570 - Methods and Practice of Scientific Computing (F20)

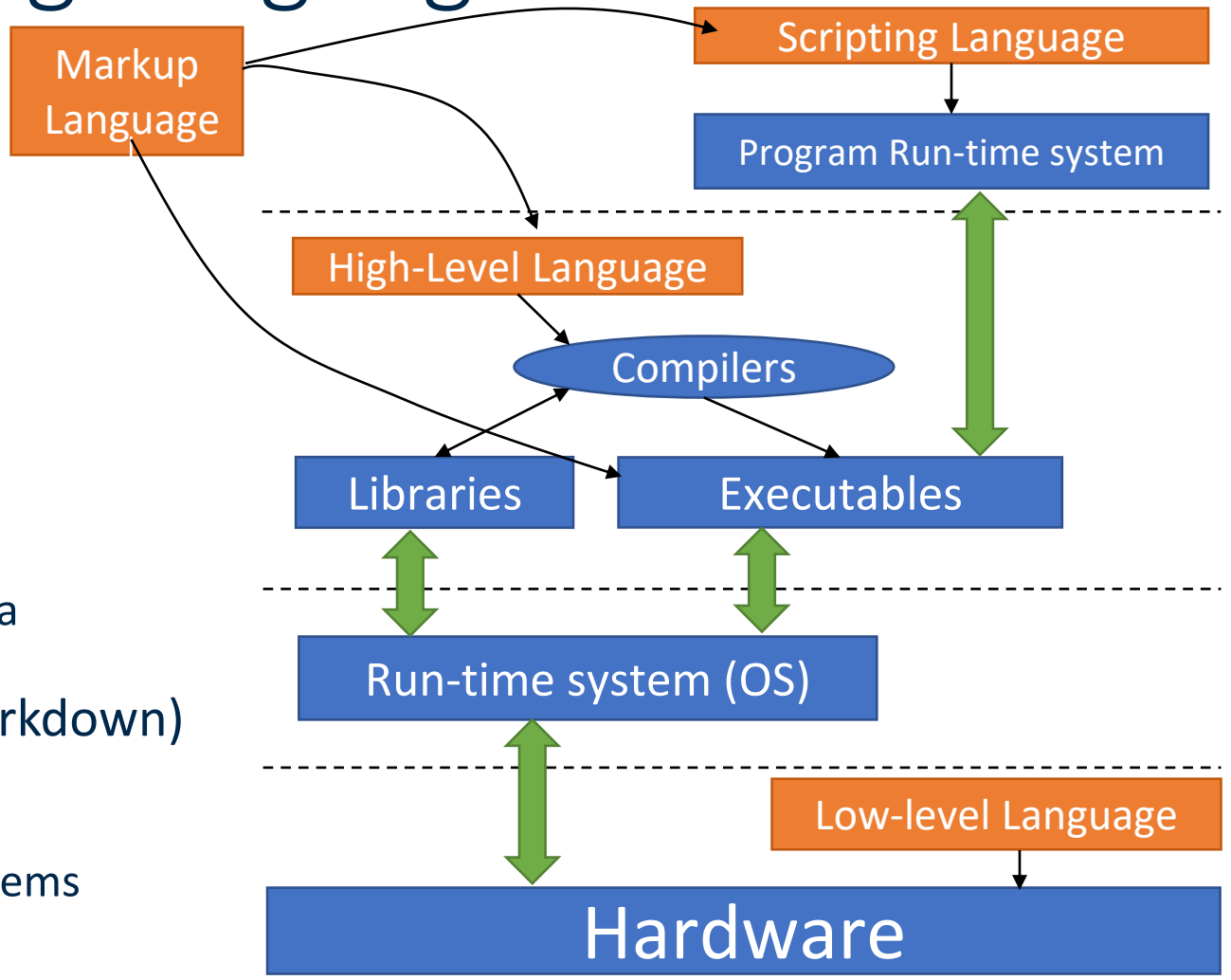
COLLEGE OF ENGINEERING
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES
UNIVERSITY OF MICHIGAN

Outline

- Introduction to Bash Scripting (in 10 minutes)
- Introduction to Python (in 10 minutes)

Types of Programming Languages

- Low Level language (Assembly)
  - Defined by hardware (less portable)
- High Level language (C/C++, Fortran, Java)
  - Defined by run-time system (e.g. Operating System)
  - Portable, depends on compilers
- Scripting language (MATLAB, Python, Bash)
  - Defined by portable run-time system of a program
- Markup language (e.g. XML, YAML, Markdown)
  - Used for annotation
  - Data transfer
  - Input to multiple types of programs/systems



Shell Scripts in Linux

- Scripts are just text files
- First line is special

Bash Hello World

```
#!/bin/bash

echo "Hello World!"
```

Python Hello World

```
#!/usr/bin/python

print("Hello World!")
```

Perl Hello World

```
#!/usr/bin/perl

print("Hello World!\n");
```

Awk Hello World

```
#!/usr/bin/awk -f
BEGIN {
  print "Hello World!"
}
```

Symbol is sometimes called “bang”

Symbol is sometimes called “sha”

Program for interpreting the script

Together it is called “Shebang”

```
#!/usr/bin/python

print("Hello World!")
```

Scripting with Bash

More documentation at Too Long Didn't Program (tldp)

http://tldp.org/LDP/abs/html/index.html

Bash: The Basics

- The contents of your bash file will also work if entered correctly on the command line*

```
#!/bin/bash

# A comment

#print something
echo "Hello World"

#set a variable
myvar=1

#access a variable
echo $myvar
```

```
#!/bin/bash

# For loop
for i in 1 2 3 4; do
  echo $i
done
```

```
#!/bin/bash

# if else
touch newfile
if [ -e  newfile ]; then
  echo "Found newfile"
else
  echo "Did not find newfile"
fi
```

*The exception here is script variables.

Pitfalls

```
#!/bin/bash

# A comment

#print something
echo "Hello World"

#set a variable
myvar=1

#access a variable
echo $myvar
echo "$myvar"
echo '$myvar'
```

No whitespace around assignment operator!

Single quote and double quote matter!

Variables defined in script do not persist after script!

Must have whitespace!

Test command "[" is very confusing

```
#!/bin/bash

# if else
touch newfile
if [ -e  newfile ]; then
  echo "Found newfile"
else
  echo "Did not find newfile"
fi
```

```
$ var=''
$ [ $var = '' ] && echo True
-bash: [: =: unary operator expected
$ [ "$var" = '' ] && echo True
True
$ [[ $var = '' ]] && echo True
True
```

Summary advice on Bash Scripts

When to use

- Simplifying complex commands
- Modifying files and directories
- Simple parsing and modification of file contents
- Working with operating system
- Generating inputs and jobs for parametric studies
- Some software project infrastructure tasks

When not to use

- Arithmetic
  - Especially floating point
- Data processing
  - e.g. parsing CSV files, finding max values, min values, averages
- Graphics (should be obvious)

Introduction to Python

It is the number 1 used programming language!

It is FREE!

It can do most anything other languages can do.

It is comparatively easier to use than high level languages.

There's a package for just about everything!

Executing python

- Interactively
  - Start a python “shell”: `$ python`
  - To exit the python “shell” enter: Ctrl+d
- Run a script
  - `$ python myscript.py`
- Run a script interactively
  - `$ python –i myscript.py`

Basic Syntax and Operators

Arithmetic Operators

```
print('Hello world')

x = 10.

print(x + 5.)
print(x - 4.)
print(x * 2.)
print(x / 3.)

print(x ** 2.)
# (x^2 in MATLAB)
```

Boolean Operators

```
x = 10.
# comparison
print(x == 15.)
print(x <= 12.)
print(x != 5.)

# boolean logic
print(not True)
print(True and False)
print(True or False)
```

Intrinsic types

- Python uses “dynamic typing”
  - No need to explicitly declare `int`, `float`, `bool`, etc.
  - Types are *implied*

```
x = 10.              #implied float
i = 1                #implied int
l = True             #implied bool
c = “Hello World”    #implied string
```

- Declare variables anywhere!

```
#Python 2
2/3 == 0
2./3. == 0.66667

#Python 3
2/3 == 2./3. == 0.66667
2//3 == 0
```

Execution Control Constructs

If-Else

```
# indentation required:
if condition1:
    statements
elif condition2:
    other statements
elif condition3:
    other statements
else:
    alt statements
```

Looping

```
# i = 0, 1, ..., n-1
for i in range(n):
    print(i)

# iterate through a
# sequence
x_list = [1, 2, 4, 8]
for x in x_list:
    print(x)
```

Python Lists

- Effectively the implementation of arrays
  - Can be mixed “types”
- But these arrays are fancy
  - Can function like the classical data structure definitions of stacks, queues, or decks
- Operations on lists are very easy compared to other languages.
- Not very fast...
  - For speed you want numpy.

```
VY0 = [2., 3., 4.]
print(VY0)
print(VY0[0],VY0[-1])
print(len(VY0))

VY0.append(5.)
VY0[2] = "cat"
print(VY0)

VY0.pop(0)
print(VY0)
```

Further Reading

- Numpy and Scipy
  - https://docs.scipy.org/doc/
- Gorelick and Ozsvald, “High Performance Python,” O’Reilly Media, 2014.
  - https://search.lib.umich.edu/catalog/record/018003128
- MOOC: https://www.sololearn.com/Course/Python/
- A great new tool for productivity is the Jupyter Notebook
  - https://jupyter-notebook-beginner-guide.readthedocs.io