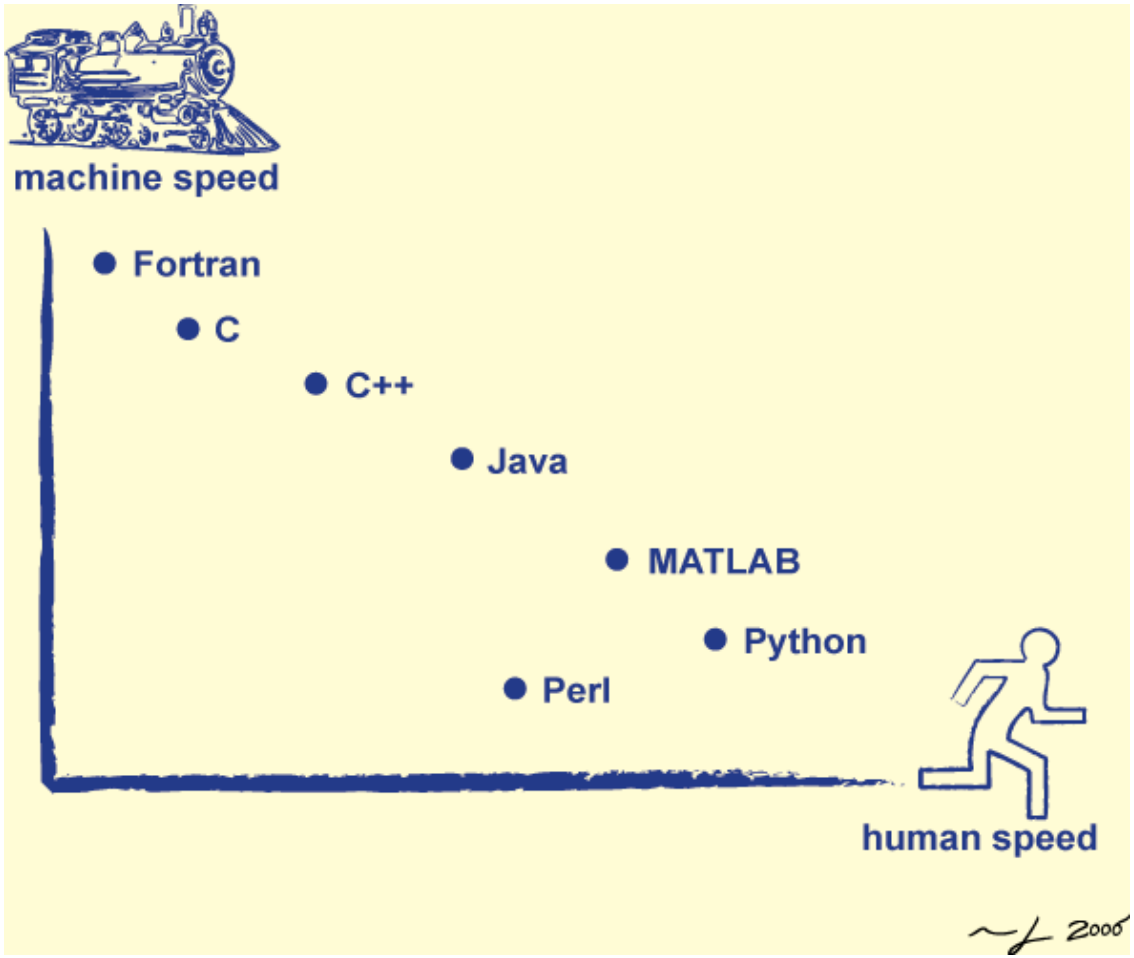
Python

Overview

Introduction

- Two things determine time to solution:
  - How long it takes to write a program (human time)
  - How long it takes that program to run (machine time)
- Different languages make different tradeoffs between these
  - High-level languages let programmers express their thoughts more quickly…
  - …but the more abstract the language, the more slowly it runs

Where Python Stands?



Still why Python?

- Scripting languages are increasingly popular

because they optimize human time

  - Which is now more expensive than machine time in all but a handful of cases
- Object Oriented. Design Patterns.
- As flexible as the shell, but with real data structures.
  - A world is not made of just lists
    - Sets (Implemented using Hash Table) , Dictionaries (Key-Value), tuples, etc. are also there to achieve required task in easy manner.

Unsuccessful attempt with Shell Script

```
#Removing all numbers from file names to deceipt secret message from sorted names of directories/files.
for original_name in `ls`
do
echo $original_name
  new_name=`echo $original_name | sed 's/[0-9]*//g'`
  mv $original_name $new_name
done;
```

** Here, shell script is failing for names having spaces in directory names. i.e. “New York2100”
Note that Python Udacity course shows very accurate python script to achieve the same quickly. **

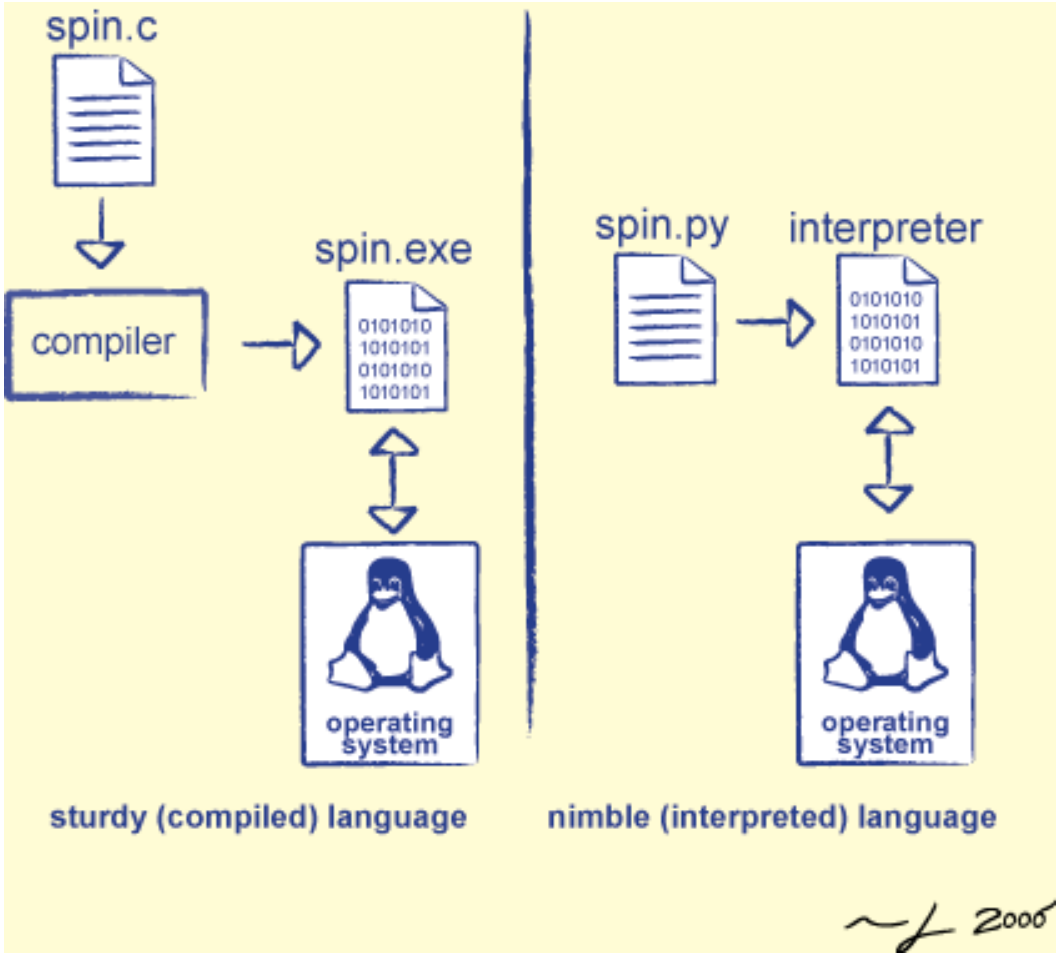
https://www.udacity.com/course/programming-foundations-with-python--ud036

```
[jigar.pandya@localhost shellscripts]$ ls
123Alibi
New York2100
renscript.sh
[jigar.pandya@localhost shellscripts]$ ./renscript.sh
123Alibi
New
mv: cannot stat ‘New’: No such file or directory
York2100
mv: cannot stat ‘York2100’: No such file or directory
renscript.sh
mv: ‘renscript.sh’ and ‘renscript.sh’ are the same file
[jigar.pandya@localhost shellscripts]$ ls
Alibi
New York2100
renscript.sh
[jigar.pandya@localhost shellscripts]$
```

Execution Cycle

- Sturdy languages use a two-step execution cycle
  - Compile source code, putting machine-oriented form in file
  - Run the contents of that file on top of an operating system or virtual machine
- Nimble languages combine these two steps
  - Compiler and virtual machine are the same program
  - Load source code, translate into more compact form if necessary, and execute

Sturdy vs. Nimble Execution



…

- Compiling gives the computer a chance to optimize that is why sturdy programs run faster
- But nimble programs are faster to write as Load-and-go makes human turnaround faster

Highlights

- Python doesn't use begin/end or {…}
  - Not really needed because indentation is what everyone actually pays attention to.
  - use colon and indentation to show nesting

```
num_moons = 3
while num_moons > 0:
        print num_moons
        num_moons -=1
```

- Python borrows C's comparison operators, too
  - But allows you to chain comparisons together, just as in mathematics "3 < 5 <= 7 is True"
  - Immediate string comparison "'abc' < 'Abc' is False"

Highlights…

- The built-in function str converts things to strings

  print "Diameter: " + str(1280) + "-" + str(1760) + " km"

  Diameter: 1280-1760 km

- Use int, float, etc. to convert values to other types

  print int(12.3)

  print float(4)

  12

  4.0

- Expression ‘Py’ * 2 produces ‘PyPy’

Highlights…

- There is no separate data type for characters
  - A character is simply a string of length 1
- Negative indices count backward from the end of the string
  - x[-1] is the last character
  - x[-2] is the second-to-last character
- Python always does an out-of-bounds check when you
- index a single item
  - But it truncates out-of-range indices when you take a slice

```
$ python
>>> element = 'helium'
>>> print element[1:22]
elium
>>> x = element[22]
IndexError: string index out of range
```

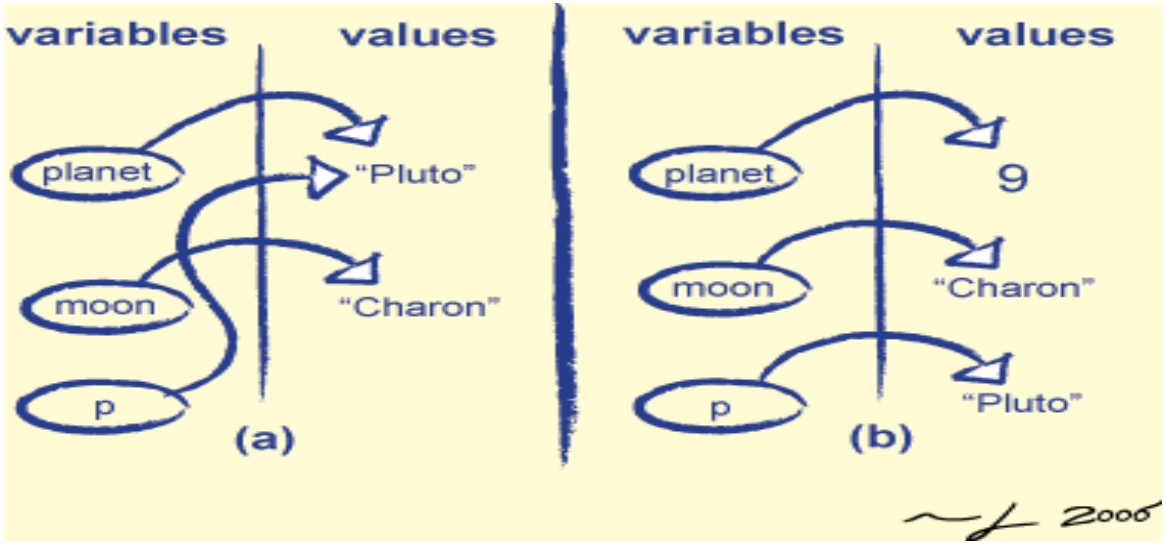
- Allows multi-valued assignment

  left, right = "gold", "lead" assigns "gold" to left, and "lead" to right

No variable types

- No types: a variable is just a name, and can refer to different types of values at different times

```
planet = "Pluto"
moon = "Charon"
p = planet
planet = 9
```



No variable types! What's the trick?

- Variables don't have types, *but values do*
  - Python complains if you try to operate on incompatible *values*

```
x = "two" # "two" is a string
y = 2 # 2 is an integer
print x * y # multiplying a string concatenates it repeatedly
twotwo
print x + y # but you can't add an integer and a string
```

Experience(die) hard!

- I just wanted to experience it hard!

Which all packages do use python in my Linux???

$>Yum remove python
$>More than 855 packages
$>Are you sure (y/n)? Y
Doooooooom.

My Linux is now almost having nothing. I will have to reinstall everything. Even the entry from grub is removed. See the heights of dependency on Python.

- Google brands python from years then decide yourself which is the brand.
- Many of mundane operations are automated using python by many engineers who know it. i.e. File/Text processing.

Broader Idea by an application developed using Python

Introduction

- What if your server program listening at LOOPBACK?
  - You are serving.....JUST YOURSELF!
- What if your server program listening at interface, which, has NOT got any worldwide IP?
  - You are serving..... JUST INTRANET.
- Can your server program still serve THE INTERNET, without having a worldwide IP?

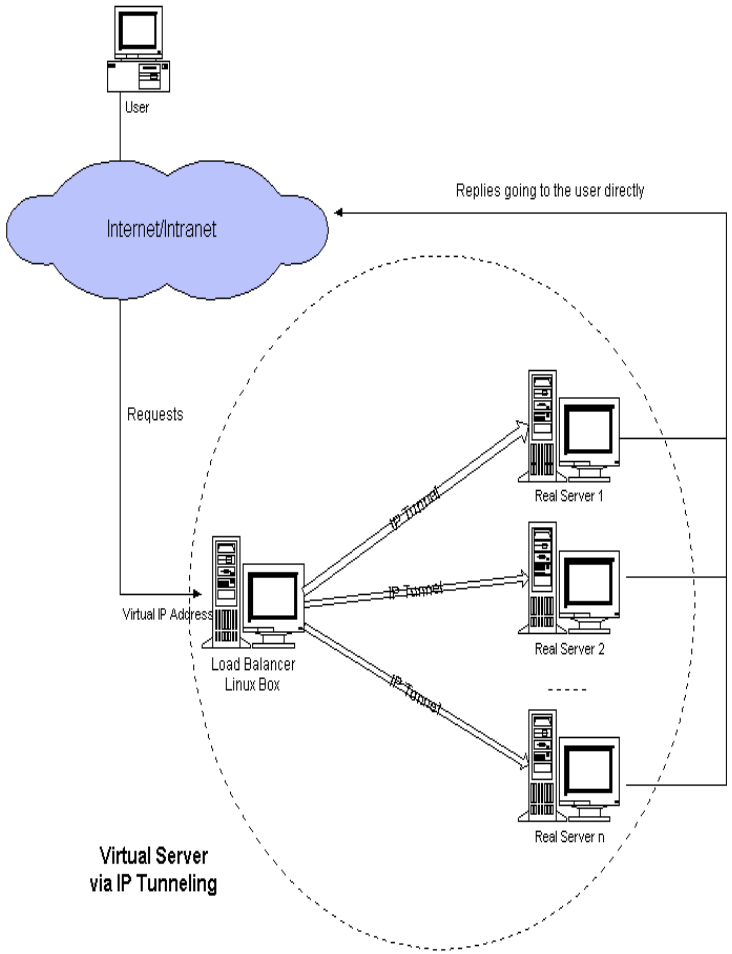
The answer is, YES!

Use Port Forwarding (IP Tunneling related)

- Demo Application
- Developed to understand/use port forwarding

SCENARIO

- Networks in a college campus.
- Local hosts of one network need to provide services to external world, can be local hosts of another network.
- BUT, not all local hosts are having worldwide IP.
- Only Head Nodes of networks having worldwide IP.



• • •

... Applications

1. Internet Connection Sharing
2. Traffic balancing to multiple servers
3. Helping to save IP as limited IPV4 addresses worldwide. Talking about when IPV6 was just a theory.

Summary

- Python is one of the cleanest scripting languages around
- A good tool in its own right
- An excellent way to build other tools

*Courtesy *

Prof. Derek Harter, TAMU-C. U.S.A

Attendee: Jigar M. Pandya

Spring 2008