Lecture 1: Introduction to Python

Chirag Sakhuja

About Me

- Chirag Sakhuja
- chirag.sakhuja@utexas.edu

About Me

- Graduated from UT in May 2017
  - B.S. ECE, B.S. CS, M.S. ECE
- GPU Architect at NVIDIA
- Also a lecturer!

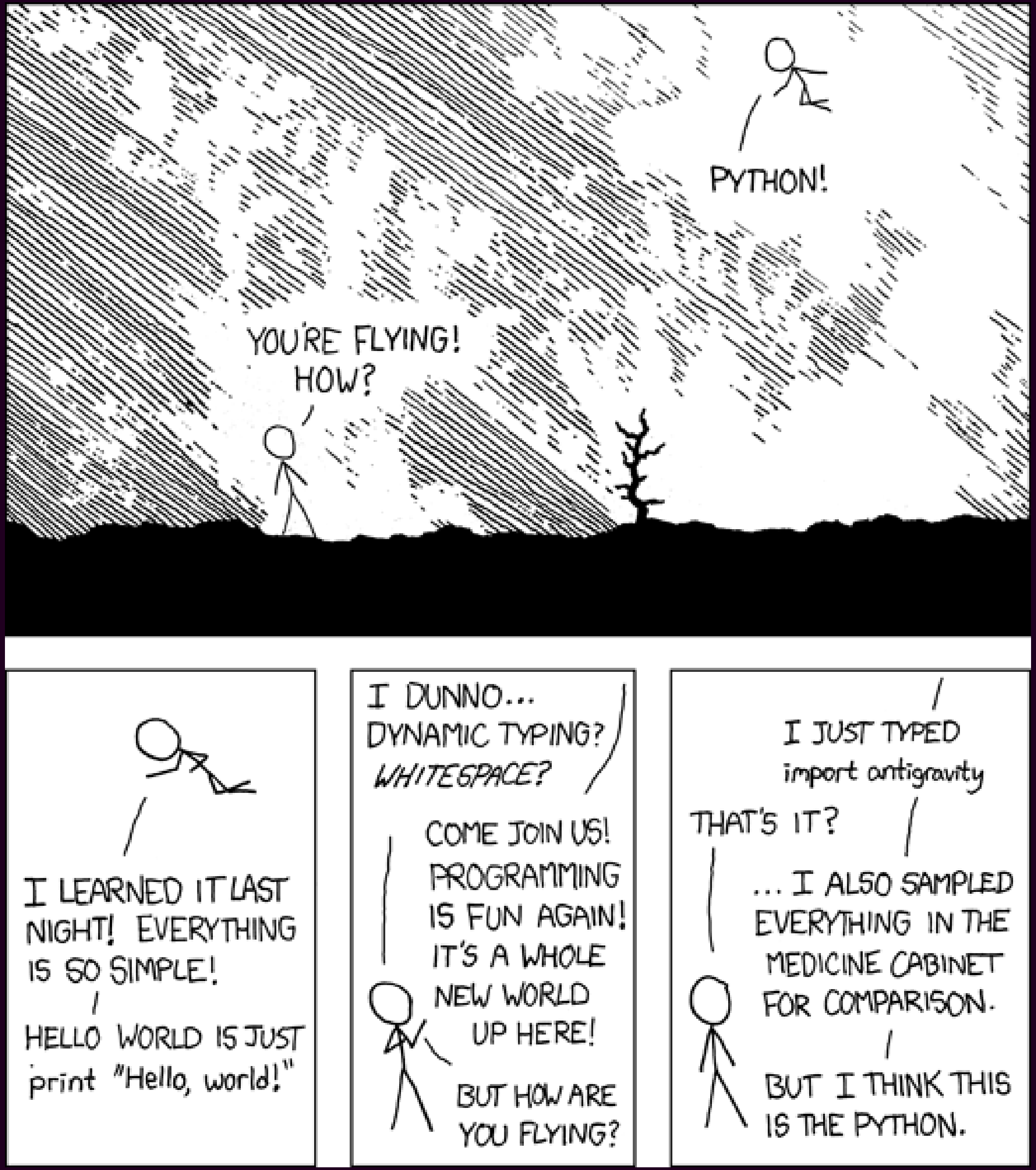
Syllabus – In a Nutshell

- Lectures are Tuesdays from 6:00 – 7:30 in EER 1.516
- Lab sections are at 6:00 – 7:30 on Wednesdays in EER 0.810 or Thursdays in EER 0.810
- We will meet for a total of seven weeks
- Attendance is optional
- To pass, you must receive an average >= 60 on the four assignments
- Assignments should take 2-4 hours
- Assignments do not have due dates, but anything turned in after the Friday of the last week of class will be penalized by 15 points

Quick Poll

- What year are you in?
- How many of you have used a high-level language other than C/C++?
- Has anyone used C++11 or beyond features?
- How many of you have used Python?
- How many of you aren't officially registered?

https://xkcd.com/353/


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Alt text: I wrote 20 short programs in Python yesterday. It was wonderful. Perl, I'm leaving you.

Python vs. C++

- Interpreted vs. compiled
- Duck typing vs. explicit types
- Focus on the Pythonic way vs. giving you powerful language primitives
- Automatically managed memory vs. insidious memory bugs

However…

- A language is just a tool to describe an algorithm
- Python is almost always simpler than C++
- The language you use should depend on the task

Hello world in C++

```cpp
#include <iostream>

int main(int argc, char ** argv) {
    std::cout << "Hello, world!\n";
    return 0;
}
```

Hello world in Python

```python
print("Hello, world!")
```

No semicolon

Arithmetic in C++

```
float x = 1.0f;
float y = 2.0f;
float z = 3.0f;
// (x + y) ^ (y + z)
float a = std::pow(x + y, y + z);
// a / y (integer division)
int b = (int) a / (int) y;
```

Arithmetic in Python

```
x = 1.0                                 # => 1.0
y = 2.0                                 # => 2.0
z = 3.0                                 # => 3.0
# (x + y) ^ (y + z)
a = (x + y) ** (y + z)                  # => 243.0
# a / y (integer division)
b = a // y                              # => 121.0
```

Python arithmetic operators

`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division (floating point)
`//`	Division (integer)
`%`	Modulus
`**`	Exponentiation

The interpreter

```
Python 3.6.4 (default, Dec 23 2017, 19:07:07)
[GCC 7.2.1 20171128] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 1
# => 1
>>> print("Hello, world!")
Hello, world!
>>>
```

Comments

```
# single line comment

"""
multi
line
comment
"""
```

Booleans

```
True                                   # => True
False                                  # => False
not False                              # => True
False and True                         # => False
True or False                          # => True
1 == 1                                 # => True
# (1 <= 3) and (3 > 2)
1 <= 3 > 2                             # => True
```

Capitalized booleans

Full word logical operators

Boolean operators

`==`	Equal
`!=`	Not equal
`>`	Greater than
`>=`	Greater than or equal
`<`	Less than
`<=`	Less than or equal
`not`	Logical NOT
`and`	Logical AND
`or`	Logical OR

Strings

```
str1 = "Hello"     # => "Hello"
str2 = 'Hello'     # => "Hello"
str1 == str2       # => True
str1 + ', world!' # => "Hello, world!"
```

No difference between ' and "

The in operator

```
x = "Chirag"
"a" in x                                # => True
"b" in x                                # => False
# equivalent to x == "Chirag"
("Chirag" in x) and (x in "Chirag")     # => True
```

Querying types

```python
x = 1
type(x)          # => <class 'int'>
x = 1.0
type(x)          # => <class 'float'>
x = "Chirag"
type(x)          # => <class 'str'>
type(int)        # => <class 'type'>
```

Converting between types

```
int(1.5)        # => 1
str(1)          # => "1"
str(1.5)        # => "1.5"
int("1")        # => 1
float("1.5")    # => 1.5
int("1.5")      # => ValueError!
```

Console I/O

```
x = input("Num: ")                # => Num: ; <= 10
print(x + 1)                      # => TypeError!
print("+1:", int(x) + 1)          # => +1: 11
print(x + str(1))                 # => 101
```

If statements

```
if cond1:
    print("cond1 was True")
```

Start with a colon

Whitespace is significant?!

If statements

```
if cond1:
    print("cond1 was True")
else:
    print("cond1 was False")
```

If statements

```python
if cond1:
    print("cond1 was True")
elif cond2:
    print("cond1 was False, but cond2 was True")
else:
    print("cond1 and cond2 were False")
```

For loops

```
for item in iterable:
    # code to handle item

l = "123"
for x in l:
    print(x)    # => 1
                     2
                     3
```

While loops

```
while cond:
    # execute this until cond is False

# loop until user types in 'q'
while input() != "q":
    # do something
```

Key insights

- Python is intuitive!
  - Don't think too hard yet
  - If you think it may work, it probably will
- You learn a language by speaking it; you learn a programming language by using it
- Google, Stack Overflow, Python documentation; Google, Stack Overflow, Python documentation; Google, Stack Overflow, Python documentation

Credits

You may notice a striking similarity between my slides and the Stanford Python course…that's not a coincidence

http://stanfordpython.com