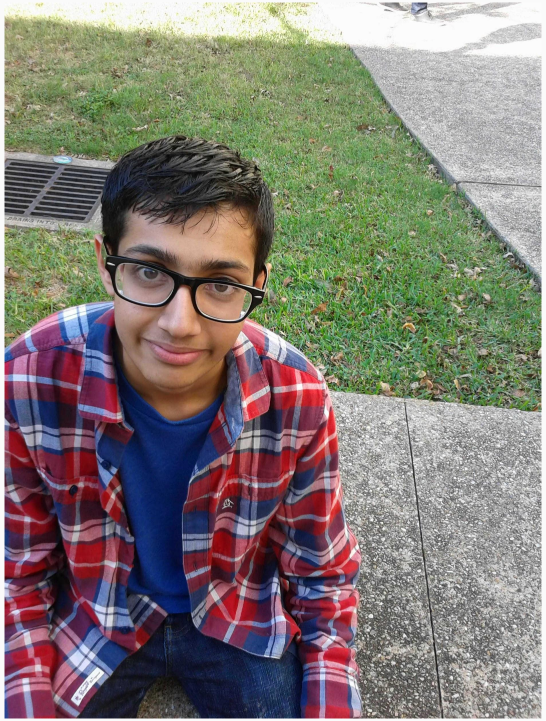
Python Workshop

IEEE Computer Society @ The University of Texas at Austin

Sean Kirmani



Cassidy Burden

We can't teach you everything, but we can help you get started!

Why Python Sucks

Python is slower than Java.

Python is a LOT slower than C++.

It doesn’t compile, so a typo can break a program in runtime.

It’s not at all optimized for performance.

But Python is easy!

Why You Should Use Python

You don’t have to think about memory!

You can do a lot with a few lines of code!

You can make more complicated things more quickly!

It reads like English (basically)!

Fundamentally, it’s a fast and simple way to get things done.

Why did Guido van Rossum write Python?

“Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).”

Story Time

A student interviewed at Dropbox, and said he knew Python pretty well.

He gets interviewed and it’s the hardest interview of his life.

Turns out the inventor of Python works at Dropbox and interviewed him.

It ends sadly.

Moral of Story

Lots of companies use Python as their primary language!

- Dropbox
- Quora
- Pinterest
- Reddit
- Venmo

So Python is useful to know!

Special Python Features!

Python Features Overview

- Interpreter
- Generators
- List Comprehension
- Built-in Functions
- Built-in Data Structures
- Inferred Typing

Interpreter

Open up a UNIX shell and type ‘`python`’.

If you don’t have it installed go to: http://repl.it

It’s easy to use. Open it now and follow along!

Generators

```
>>> doubles = []
>>> for n in range(10):
...   doubles.append(2 * n)
...
>>> print doubles
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

```
>>> doubles = [2 * n for n in range(10)]
>>> print doubles
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

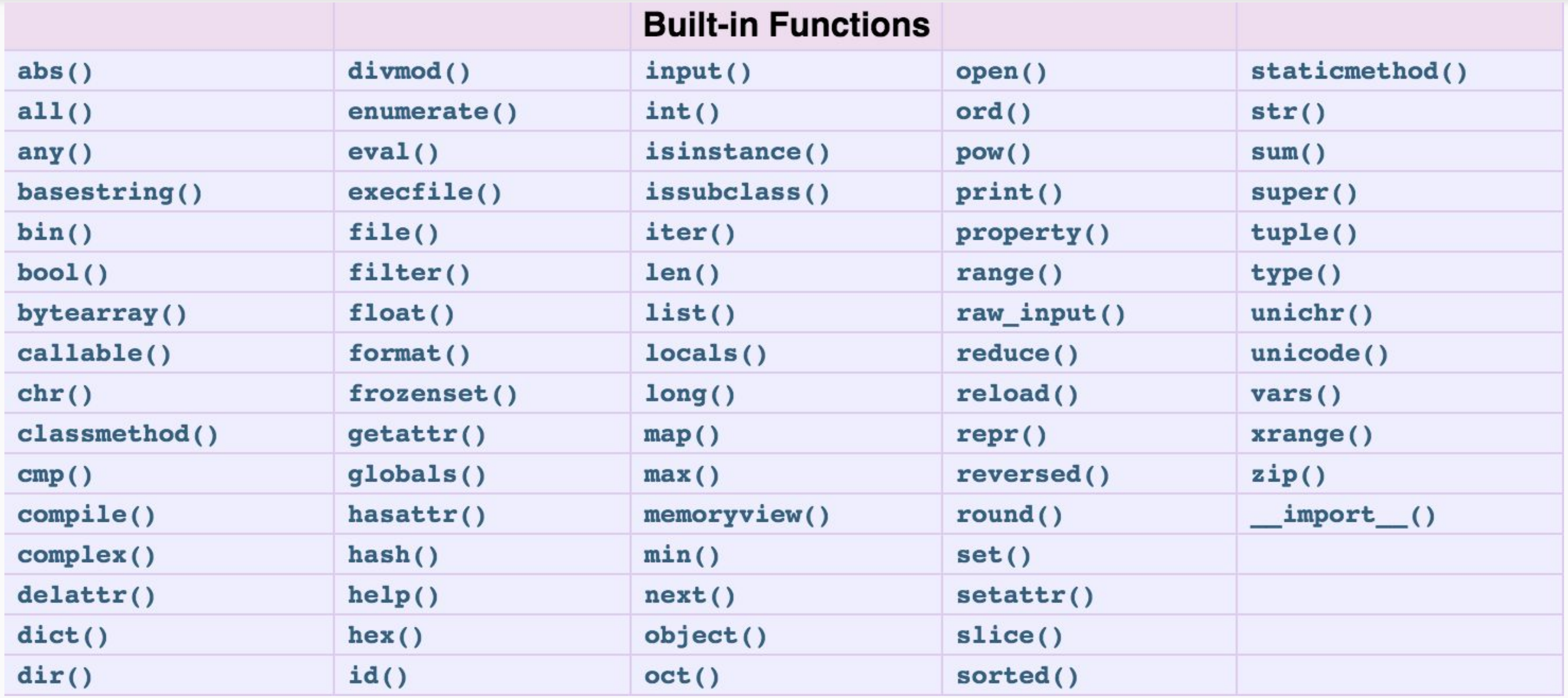
List Comprehension

```
>>> S = [x**2 for x in range(10)]
>>> V = [2**i for i in range(13)]
>>> M = [x for x in S if x % 2 == 0]
>>>
>>> print S; print V; print M
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]
[0, 4, 16, 36, 64]
>>> words = 'The quick brown fox jumps over the lazy dog'.split()
>>> print words
['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
```

Built-in Functions

```
>>> import random
>>> arr = [random.randint(-50, 50) for x in range(20)]
>>> print arr
[-19, 46, -25, 10, -35, 32, 49, -13, -22, -16, 41, -35, 3, -32, 34, 42, -30, 43,
27, 36]
>>> print sorted(arr)
[-35, -35, -32, -30, -25, -22, -19, -16, -13, 3, 10, 27, 32, 34, 36, 41, 42, 43,
46, 49]
>>> print sorted(set(arr))
[-35, -32, -30, -25, -22, -19, -16, -13, 3, 10, 27, 32, 34, 36, 41, 42, 43, 46,
49]
```

List of built-in Python functions



		Built-in Functions		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	

Built-in Data Structures

```
list: []
  -   Acts as stack
  -   Acts as queue
  -   Acts as set
  -   Can be sorted
dictionary: {}
  -   Essentially a HashMap
  -   Most commonly used data structure in
      computing
  -   lookup, add, and remove in O(1)
```

```
# same as the list comprehension above
doubles = list(2 * n for n in range(50))
```

Inferred Types

```
>>> x = 1   # notices that there is no type telling it to be an int
>>> print x
1
>>> x += 0.5  # it’s now a double because it infers the type
>>> print x
1.5
>>> x = 2 ** 33   # it doesn’t need to worry about integer overflow, python is smart
>>> print x
8589934592
```

Syntax!

Syntax Overview

- Writing a Python Script
- Slicing
- How to define a function
- How to use a loop
- Everything is a reference (no pointers!)

Writing a Python Script

```
def main():
    print “Hello World!”

if __name__ == '__main__':
    main()
```

Slicing (with String)

```
def main():
    greeting = “Hello”
    print greeting + “ World!”
    british_greeting = ‘\’’ + greeting[1:]
    print british_greeting
    greeting += “ World!”
    words = greeting.split()
    print “Number of words: ” + len(words)
    print “First word: ” + words[0]
          + “, last word: ” + words[len(words) - 1]
```

Defining a function

```
def main():
    for i in range(50):
        print fib(i)
    # we can call functions in generators
    fibs = [fib(x) for x in range(20)]
    print fibs[:10]  # you can slice lists too!

def fib(num):
    return fib(num - 1) + fib(num - 2) if num > 1 else 1
```

Let’s code!

Follow along and code with us!

Examples

- Making a large list with generators
- Find missing number in a large list of integers
- Quicksort in one (long) line of code
- Matrix multiplication
- Functional Programming

Here are some practice problems!

Download the ZIP file here:

goto.kirmani.io/python

We'll walk around and help with any questions!