Giving a Pythonic Talk (No Talent Required)

Also Presenting: The Pythonista's Star Wars Theory of Everything!

Xan and Catherine Vongsathorn

Chicago Python User Group, March 12, 2015

Who we are

Xan: **Ch**ewbacca + **Han** Solo; Pythonomist; Grad Student at UChicago

Catherine: Narrative Scientist; PyLady; Former Math Instructor

Source: Twentieth Century Fox

Who we aren't

- We are not: experts, naturally talented speakers
- Where to go for wisdom?

Yoda van Rossum!

- (Wise. Sees future. Talks funny.)
- Dagoba (noun): a dome-shaped memorial containing relics of Buddha.
  - (Dagobah, Yoda) == (Zen Planet, Zen Master)

Source: Twentieth Century Fox/Wikipedia

Zen Planet: Home of Friendly Pythons!

No rubies, no pearls...

- The Zen of Python is coming...

Source: Twentieth Century Fox

About This Talk

- High-level discussion plus specific guidelines (*not* rules) you can follow.
- Focus: effectively developing the **content** of your talk (not the **performance**).
- Many of the principles we will cover apply to communication more generally, not just giving technical talks.
- Why would you give a talk?
  - To communicate knowledge or ideas to others
  - To practice communicating knowledge or ideas to others

We don't expect you to remember everything we talk about, but we hope you remember some of it.

The Zen of Python

```
>>>import this

      Beautiful is better than ugly.
      Explicit is better than implicit.
      Simple is better than complex.
      Complex is better than complicated.
      Flat is better than nested.
      Sparse is better than dense.
      Readability counts.
      ...
```

"Explicit is better than implicit"

- The **Narrative Structure** of Django vs Rails
  - Explicit is better than implicit

```
from django.shortcuts import render
from blog.models import Post
def index(request):
    all_posts = Post.objects.order_by('-created_at')
    context = {'all_posts': all_posts}
    return render(request, 'polls/index.html', context)
```

  - Convention over Configuration

```
class BlogController < ApplicationController
  def index
    @all_posts = Post.order('created_at DESC')
  end
end
```

Narrative: What are you talking about? Why?

Developing the Narrative for Your Talk

- The narrative of your talk should:
  - Exist
  - Be explicit and coherent
  - Connect everything you talk about
  - Have broad appeal to your audienceaudience
    - Who is your audience?
    - What do they want to know?
- There comes a point in any technical talk when the audience is going to have to *do some work*. **Give them a reason.**

> *Your audience does not care about your topic. You have 1-2 slides to change their minds. –Jesse Shapiro*

"Simple is better than complex"

> *There are two kinds of teachers. One goes up to the blackboard and says, "Look how smart I am!" The other says, "Look how easy this is!" --Brian Bi quotes a teacher on teaching*

- "two kinds of speakers..." $\diamond$ "two kinds of coders..."
- "Look how smart I am" leads to the *dark side*.

  Luke: Is the dark side stronger?
  Yoda: No, no, no. Quicker, easier, more seductive.

- Python: minimize the **thinking:understanding** ratio for code
- Even more important in a talk – thinking on the fly is *very hard*, yet understanding is *entire purpose*.

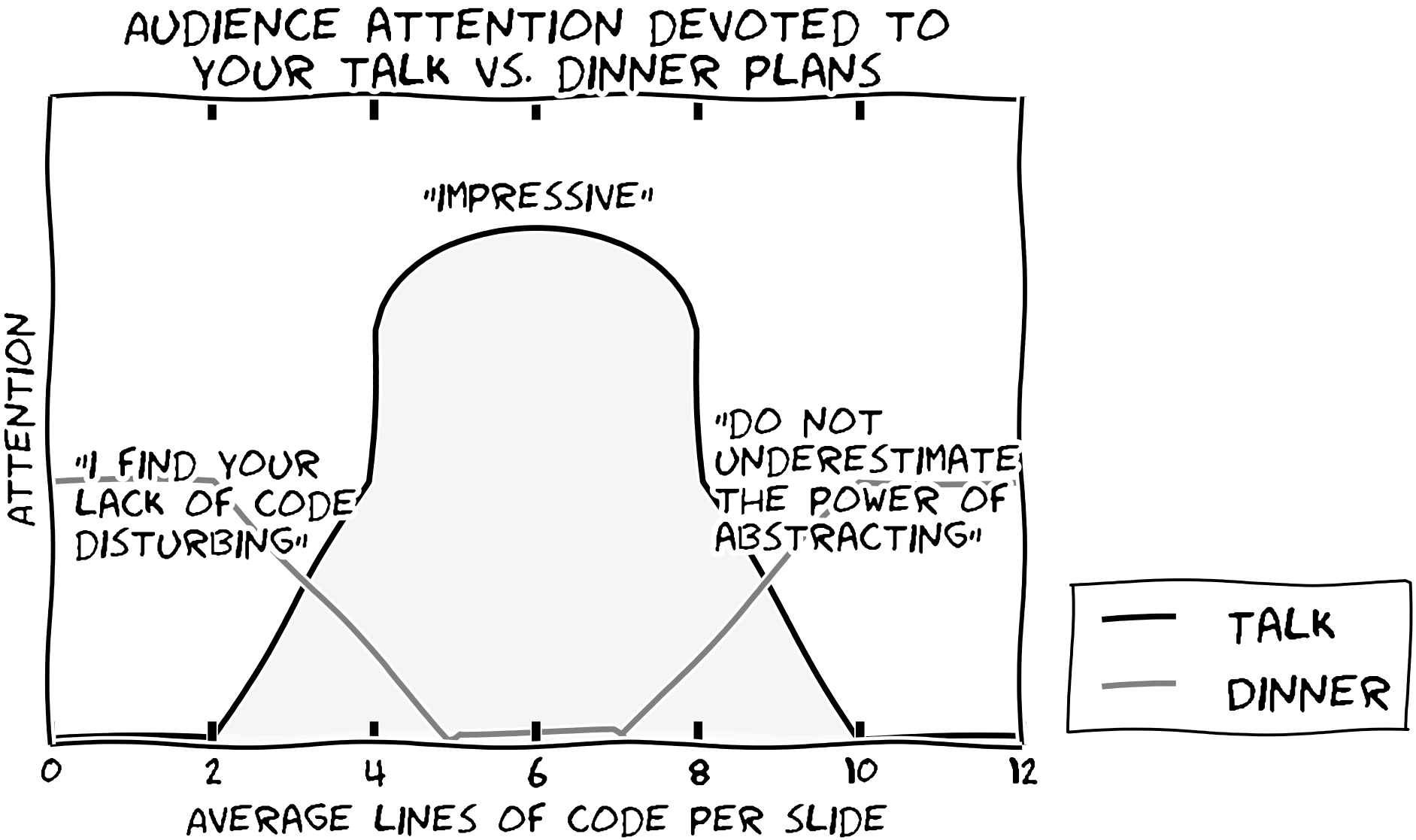
Playing Grandma's Advocate

- By default, we **vastly overestimate** audience comprehension.
  - Intelligence/experience = poor substitute for the *time* you have spent thinking about the topic.
  - Rule of thumb for estimating percentage of audience who will follow your technical talk: **Divide naive estimate by 10.**
- **Grandma's Advocate** *always* argues that your talk is not clear enough yet.
  - "If you can't explain it to your grandma..."
- Play grandma's advocate with yourself when designing a talk: Is it clear enough yet? How about now?

"Complex is better than complicated"

- Technical subjects can be complex. Your job is to *manage* the complexity for your audience, not eliminate it.
- Focus on advancing your narrative.
  - Technical details exist to **support the narrative**.
  - Cut extraneous details that don't.
- Good News: We are programmers!
  - *No one* is in a better position to abstract away from irrelevant technical details.
  - Encapsulation, abstraction...

Example: Managing Complexity



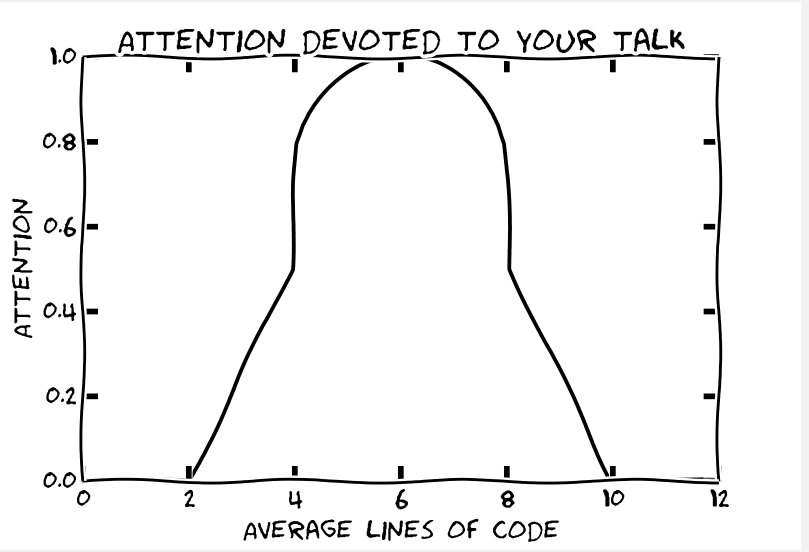
Example: Managing Complexity (Continued)

```
# A simpler, minimal example
import numpy as np
import matplotlib.pyplot as plt

plt.xkcd()
fig = plt.figure()

x = np.linspace(0, 12, 120)
y = calc_y_helmet(x)

plt.xlabel("Average lines of code")
plt.ylabel("Attention")
plt.title("Attention devoted to your talk")
```



Zen Planet of Pythons (Redux)

...no Jawas, no asps...

Source: Twentieth Century Fox

"Sparse is better than dense"

In the beginning Guido created the Python. Now darkness was over the surface of the code. And Guido said, **"Let there be White Space,"** and He saw that it was good. ◇ (Han shoots Guido (*first*)) "Oops, no white space today. **Sorry about the mess.**" ◇ "You will never find a **more wretched slide** of scum and villainy. We must be cautious." ◇ "I find your lack of white space disturbing." ◇ "**STAY ON TARGET.** A dense slide lacks focus, sacrifices your ability to **control your audience's attention**, and probably means you're trying to tell them **too much at once**." ◇ "That's no slide, it's a **waste station**. They're just spewing a bunch of made up quotes and **talking about something else entirely**." ◇ "**IT'S A TRAP!** Psst, if your slide is too dense, many people in the audience are **not listening to you**. Instead, they are sucked into **reading your slide**. They are storing the sound of your voice in a **limited mental buffer** designed to hold auditory stimuli for a short time before **processing for meaning**. If what you just said does not connect to what they're reading in the next 5 seconds, it's **gone forever**.

"Readability counts"

No reward is worth this. No reward is worth this.

- The urge to cram too much onto one slide often leads to font that is too small to read.
  - What's the "point"?
- **Rule of thumb:** Font should be large enough that it is *impossible* to fit more than about 20 lines on your slide.
  - Of course, try not to *fill* all 20 lines (Sparse is better than dense)

Zen Planet of Pythons (Redux)

...just python after python after python!

- Conclusion: Zen Planet literally covered with pythons.
- Conclusion: George Lucas really into **Zen of Python**.

Source: Twentieth Century Fox

"Errors should never pass silently"

Strike me down, and I will become more powerful than you could possibly imagine – Obi-Wan

- **Solicit feedback.** And be receptive to it.
  - *Luke*: I don't, I don't believe it.
  - *Yoda*: That...is why you fail.
- No shame in making mistakes. Learn from them.
- Give actionable feedback

```
raise Exception("I have a bad feeling about this.")
```

vs.

```
raise NameError("I didn't understand what you meant
    by 'Grandma's Advocate'.")
```

"Now is better than never"

```
try:
    from __future__ import speakingtalent
except ImportError:
    print "I cannot train him. The boy has no patience."
```

- **Practice. Starting now.**

  "Awww! But I was going into Tosche Station to pick up some power converters!"

- Brains get programmed by *doing things repeatedly*.
- Natural talent not necessary. *You* decide to prioritize communication.

```
try:
    actually_care_about(communicating)
except YodaError:
    print "Do, or do not. There is no try."
```

- You work hard to make your code clear. Work hard to make your talks clear, too.

What Did We Talk About?

- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Sparse is better than dense
- Readability counts
- Errors should never pass silently
- Now is better than never

Luke's Zen Moment

"Import *this!*"

Conclusion: Zen of Python is a source of deep wisdom and power that applies far more broadly than just to your Python code!

Source: Twentieth Century Fox

"Namespaces are one honking great idea – let's do more of those!"

xanvong.com
xan.vong@gmail.com

catherinev.com
cvongsathorn@narrativescience.com