talk-talk-talk Documentation

Mariatta Wijaya

Jun 26, 2019

Contents

List of talk proposals I've submitted to conferences.

I hope you find these useful.

CHAPTER 1

Dial M For Mentor

This talk was accepted and given at: PyCaribbean 2017, DjangoCon Europe 2017, PyCon Italy 2017, and PyCon US 2017.

It was also accepted at two another regional Python conferences, but I declined the invitation.

One was because they didn't provide any financial aid, and I didn't have any support from my employer.

The other I declined because I was already going to another conference just one week before, and it was getting difficult for my family that I was away all the time.

1.1 Mentor

Special thanks to Ned Batchelder who helped me refine this proposal.

1.2 Proposal

The following was the proposal I submitted to PyCon US 2017. If you've seen my talk, you will realize that the proposal was not quite the same as the talk I gave. I had a life-changing event after this proposal was accepted, so I made slight modification.

1.2.1 Title

Dial M for Mentor

1.2.2 Duration

30 minutes slot

1.2.3 Who and Why

Members of Python and open source community who are looking for a mentor, and needing clues as to how to find a good mentor. Those who are hesitant to ask for help may get the encouragement to reach out. Current mentors will gain insight about what their mentees are going through.

1.2.4 Description

One of the nicest things about Python community is the availability of mentors willing to help you. Various mentors have helped me navigate the open source community and help advanced my skills. I realized finding a mentor is not as easy as it seems, and it takes a lot of courage to reach out in the first place. And then, there is impostor syndrome, where one may feel like they don't deserve the help. In this talk, I will provide advice about working with a mentor. Asking for help is not a failure.

1.2.5 Outline

1. Myths to debunk: (5-10 minutes)
    - "I don't deserve their help/I'm not worthy"
        - Impostor Syndrome. Men and women alike can experience this.
        - First, admit that this is a problem, then help yourself.
        - Provide links to other talks/resources which discuss Impostor Syndrome
    - "I'm wasting their time"
        - Mentors learn from you too. Your work may expose them to areas they haven't been before.
        - Mentors gain skills: leadership, communication, and time management.
        - Sometimes it's nice to have solvable problems to work on. "Experts" have challenging days of their own full of uncertainty and confusion. Being able to help and having answers to your questions can be a rewarding experience.
    - "I'm a failure for needing help"
    - "I know so little, I don't even know what questions to ask"
        - It's part of learning. You will reach the point where you don't need help anymore.
        - Why work with a mentor (5 minutes)
        - Maybe you've been stuck for a long time
        - Maybe you tried it on your own, and not successful.
        - Maybe you're experienced in one domain, but want to learn new skills

      Eg: You're experienced in Python, and want to start picking up Javascript/Go/C

      Mentors have gone through your problems. They can help and provide guidance But it's up to you to reach out in the first place. Don't believe in myths.

      Personal experience: Making more progress when working with mentor, compared to when trying to achieve things on my own.
2. Where to find mentors (3 minutes)
    - Face to face / in person
    - Someone within your own organization/company/school

- Someone in your local python community/meetup
- Online mentorship:
  - Pyladies, Python Core Mentorship
  - IRC or Slack channels
  - Conference speaker mentors

3. Working with your mentor (5 minutes)
   - It's two-way, requires commitment from both sides
   - Respect each other. Your mentor has other commitments and personal life
   - If it doesn't work out for any reason, find a different mentor.
   - Want to thank your mentor? Pay it forward. Mentor others.
4. Truth about mentorship (3 minutes)
   - Mentor plays a supporting role. They provide resources and advice needed to succeed. But they won't be doing your homework.
   - You are the real star of the show, you learn and do all the real work.
   - Don't forget to take credit for your own work and effort.

Thanks!

CHAPTER 2

PEP 498: The Monologue

This talk was accepted at PyCon Australia 2017. I also gave it as the closing keynote for PyCon Canada 2017.

I submitted it to PyCon US 2018, but was rejected.

It was accepted at another regional conference summer of 2017, but I ended up declining it.

I also submitted it to another regional tech conference (not a Python specific one) but rejected.

2.1 Mentor (or lack of)

This was the first time I submitted a talk proposal without help from any mentor, and I was feeling quite nervous and unsure about the entire content.

My friend Sebastian Vetter helped proofread, and he gave me the thumbs up :)

2.2 Proposal

2.2.1 Title

PEP 498: The Monologue

2.2.2 Duration

30 minutes slot

2.2.3 What

Everything you need to know about f-strings

2.2.4 Why

One of the exciting new features in Python 3.6, PEP 498 is also a case study of a successful Python enhancement proposal and implementation.

2.2.5 Who

Beginner Python users, or longtime users who have not yet upgraded to Python 3.6

2.2.6 Description

Python 3.6 was released in December 2016, and it includes 16 new PEPs! We will focus on one of them: PEP 498 - Literal String Interpolation, affectionately known as f-strings. Get a glimpse of the lifecycle of a PEP. Then, we'll learn about f-strings, see some examples, and know the gotchas. You'll want to upgrade to Python 3.6 just for this!

2.2.7 Outline

1. Introduction (4 minutes)
    - Explain who this talk is for * Beginner python users, or * Longtime python users who have not upgraded to Python 3.6 yet
    - Introduce myself * I am not the PEP author * I am not any PEP's author * I find PEP process fascinating
2. What's a PEP? (3 minutes)
    - Python Enhancement Proposal
    - Describe the usual PEP acceptance process * Discussions on python-ideas, python-dev * Champion it, draft it * More discussions * Pronouncement (either acceptance or rejection from BDFL)
    - Fun facts: * At the time of writing this proposal, there have been 349 PEPs proposed, 78 of them rejected. (22%)
3. What's PEP 498? (5 minutes)
    - the feature also known as f-strings
    - Who's the author? * Authored and implemented by Eric V Smith
    - Timeline * initial draft 2015 * pronouncement sep 2015 * Release dec 2016
    - python-dev discussions
    - Documentation https://docs.python.org/3/reference/lexical_analysis.html#f-strings
4. Why PEP 498 / Rationale (2 minutes)
    - Existing ways of formatting are either error prone, inflexible, or cumbersome
    - Similar PEPs: PEP 215, PEP 292
    - Comparison to other languages' string interpolation
5. F-strings (5 minutes)
    - Provide examples on how to use f-strings
    - Both f"" and F"" are accepted
    - Previous ways of string formatting are still accepted

    - Provide comparison of how to format strings previously, and using f-strings
6. Gotchas
    - [2 mins] Can't use it as docstrings (http://bugs.python.org/issue28739)
    - [2 mins] multi-line strings http://bugs.python.org/issue29668
    - [2 mins] IDLE syntax highlighting http://bugs.python.org/issue29071
    - [2 mins] unicode chars in format specifiers http://bugs.python.org/issue28827
    - [2 mins] Back slashes http://bugs.python.org/issue27921, http://bugs.python.org/issue27948
7. Conclusion - Inspire the audience to upgrade to Python 3.6 because of PEP 498

Thanks!

CHAPTER 3

What is a Python Core Developer?

I gave this talk at PyCon US 2018.

I didn't think this talk would have any audience outside of PyCon US, so I didn't submit it anywhere else. After I was finished with the talk, I realized that this it was too emotionally difficult to give a second time.

3.1 Mentor

I didn't seek help with the writing of this proposal. I think it was one of the reasons why I was feeling unsure about this giving this talk, even after it was accepted at PyCon.

3.2 Proposal

The following was the proposal I submitted to PyCon US 2018. Yes, I deviated from the outline in the actual talk. There was an important topic that I had left out from the proposal. I didn't actually want to touch upon it myself, but days before the conference, I decided I had to talk about it.

3.2.1 Duration

45 minutes slot

3.2.2 Description

How do you become a Python core developer? How can I become one? What is it like to be a Python core developer? These are the questions I often receive ever since I became a Python core developer a year ago. Contributing to Python is a long journey that does not end when one earns the commit privilege. There are responsibilities to bear and expectations to live up to. In the past year, I've been learning more about what it really means to be a Python core developer. Let me share all of that with you.

3.2.3 Audience

This talk is for those wanting to contribute to Python, or existing contributors to Python, so they'll gain insights from the maintainer's perspective. Other members from wider Python community will learn the challenges of maintaining an open source project with 7 million users.

3.2.4 Outline

Who am I (3 minutes)

- Parent
- Software Engineer
- Conference organizer
- PyLadies organizer
- Open source contributor
- Python core developer
- (really busy)

What is a Python Core Developer (2 minute)

- Technically, those with commit right to Python
- Since February 2017: only 30 active Python Core devs, handling 4000+ pull requests from almost 600 contributors.

What do you do once you have commit right? (3 minutes)

- As little, or as as much as you can
- Ideally, continue contributing to Python

Ways to contribute to Python, before the commit right (5 minutes)

- Make pull requests (docs, code, tests)
- Review code (but you can't merge)
- Report bugs
- Propose ideas
- Give talk or write blog posts about Python
- Be open, considerate, and respectful

Ways to contribute to Python, after the commit right (5 minutes)

- All of the above, plus
- Help other core devs and contributors to contribute
- Approve pull requests and merge them
- Make decisions (accepting/rejecting ideas)
- Owning decisions
- Follow and respond to mailing lists
- Be open, considerate, and respectful even when the person you're dealing with is not.

What I've been doing (5 minutes)

- Made 450+ pull requests to Python (not just CPython, but also to the bots, and Dev Guide)
- 200+ more to other open source projects
- Participate in several mailing lists: core-mentorship, core-workflow, occasionally python-dev
- Reviewed 600+ pull requests, focusing on pull requests from first time contributors.

What other core devs do:

- quite similar to the above, in addition to contributing to Python and CPython, they're also active in the community and maintainers of other open source projects.

Challenges (3):

- Doing all of the above for free (and justifying doing so)
- Not everyone is open, considerate, respectful
- With 7 millions users, not everyone will agree or happy with decision made

The Big Questions

How did I become a Python core developer? (2 minutes)

- I can't answer this. I just feel really lucky.
- I received a lot of support and mentorship from existing core devs.
- The actual question is really, why did the core devs before me granted me the commit access. Only my mentor and other core devs can answer that

What it's like to be a Python core developer? (1 minute)

- really busy

How can another person be a Python core developer? (3 minutes)

- perhaps my path is not one you should follow
- earn the trust from existing core devs
- earn the trust from the community

After thoughts (3):

- Becoming a core developer is not the goal. It is not the end of the journey. It is the beginning.
- The real goal is to earn the trust from the community. When they trust you, they'll stick around and continue using Python.
- Perpetual goal: how to make Python - the code and the community- better

CHAPTER 4

Build-a-GitHub-Bot Workshop

This is a tutorial, rather than a talk. I submitted this to PyCon US 2018, and it was accepted. I've submitted this to another regional conference, but was rejected.

4.1 Mentor

Thanks to Eric Holscher. Not only he gave me advice and resources on how to write and prepare a tutorial for PyCon, he also reviewed, proofread, and provided additional feedback to this proposal before I submitted to PyCon US.

Also thanks to Brett Cannon, without his gidgethub library, I wouldn't have material for this workshop. In addition, Brett reviewed, proofread, and fixed mistakes in my tutorial documentation.

4.2 Proposal

The following was the proposal I submitted to PyCon US 2018.

4.2.1 Title

Build-a-GitHub-Bot Workshop

4.2.2 Description

GitHub provides a great platform for collaborating. You can take it to the next level by creating custom GitHub bots. By delegating some of the chores to a bot, you get to spend more time developing your project and collaborating with others. Learn how to automate your workflow by building a personal GitHub assistant for your own project. We'll use libraries called gidgethub and aiohttp to write a GitHub bot that does the following:

- Greet the person who created an issue in your project.
- Say thanks when a pull request has been closed.

- Apply a label to issues or pull requests.
- Gives a thumbs up reaction to comments you made. (becoming your own personal cheer squad).

The best part is, you get to do all of the above using Python 3.6! F-strings included!

4.2.3 Audience

Intermediate/Advanced skill level.

Participants should know how to program in Python, have a GitHub account, know how to commit their work using git and to create pull requests, and understand how REST API works. They should also be familiar with the concept of Python virtual environments and installing Python libraries using pip.

Prior to attending the tutorial, participant should have Python 3.6 installed on their laptop.

This tutorial is not suitable for novices or someone new to Programming.

For this tutorial, we will deploy the code to Heroku and GitHub. Participant may want to reconsider taking this tutorial if there is anything preventing them from uploading their source code to the cloud services like Heroku or GitHub.

4.2.4 Outline

Introductions (5 minutes)

Format: presentation

- Go over the agenda (this outline)
- List the relevant resources:
- gidgethub documentation
- aiohttp documentation
- GitHub API v3 documentation
- Heroku tutorial

Intro to GitHub APIs (5 minutes)

Format: presentation

- An overview of the available API endpoints and webhook
- Examples of GitHub bots:
- Python Core Developers have 3 different GitHub bots deployed: bedevere, the-knight-who-says-ni, and miss-islington.
- Python Packaging Authority (pypa) has BrownTruck

Intro to gidgethub (5 minutes)

Format: presentation

- A python library for interfacing with GitHub APIs, written by a Python Core Developer. It requires Python 3.6.
- Used by Python Core Devs (bedevere, miss-islington) and Python Packaging Authority (BrownTruck)

Using gidgethub on the command line

Format: hands-on

Installation (15 minutes)

- Install gidgethub
- Obtain GitHub Personal Access Tokens

Write these command line scripts (30 minutes)

In the first exercise, I will provide and explain the code, thenand give attendees a chance to try it themselves. In tThe subsequent exercises ones, I will provide hints as to which API to use, and they can try it themselves. I will then reveal my solution.

- Create a new issue. API docs: https://developer.github.com/v3/issues/#create-an-issue
- Comment on an issue. API docs: https://developer.github.com/v3/issues/comments/#create-a-comment
- Close the issue. API docs: https://developer.github.com/v3/issues/#edit-an-issue

Bonus exercises

For those who finished earlier than other attendees

- React to an issue. API docs: https://developer.github.com/v3/reactions/#create-reaction-for-an-issue
- Add and remove labels to an issue. API docs: https://developer.github.com/v3/issues/labels/#add-labels-to-an-issue

Create a web service (10 minutes)

- Create an aiohttp server
- Deploy it to heroku

GitHub bot

In the first exercise, I will provide and explain the code, and give attendees to try it themselves. The subsequent ones, I will provide hints as to which webhook event and which API to use, and they can try it themselves. I will then reveal my solution.

Exercises:

- A bot that leaves a comment whenever an issue is created (30 minutes) For example, it can say “Hello user”.
- A bot that leaves a comment whenever an pull request is closed (20 minutes) For example, it can say “Thanks for merging”, or “Thanks for the review”.
- A bot that leaves a “thumbs up” reaction whenever someone leaves an issue comment (20 minutes)
- A bot that adds a label “need review” whenever a pull request is created (20 minutes)

Q & A (10 minutes - end)

Time for questions, open discussions, or if people need more time completing their exercises.

Bonus exercise

A bot that closes a pull request if the description contains certain blacklisted/disallowed keywords.

CHAPTER 5

Hands-on Intro to aiohttp

Another tutorial proposal, co-written with Andrew Svetlov. We submitted this to PyCon US 2019, and it was accepted.

5.1 Acknowledgement

Thanks Andrew Svetlov for working with me on this proposal and tutorial. Honestly, I don't feel like I'll be able to give this tutorial on my own.

5.2 Proposal

The following was the proposal we submitted to PyCon US 2019.

5.2.1 Title

Hands-on Intro to aiohttp

5.2.2 Additional Speakers

Andrew Svetlov

5.2.3 Description

Asyncio is a relatively new feature in Python, with the async and await syntaxes only recently became proper keywords in Python 3.7. Asyncio allows you to write asynchronous programs in Python. In this tutorial, we'll introduce you to an asyncio web library called aiohttp.

aiohttp is a library for building web client and server using Python and asyncio. We'll introduce you to several key features of aiohttp; including routing, session handling, templating, using middlewares, connecting to database, and

making HTTP GET/POST requests. We'll provide best practises in building your aiohttp application, as well as how to write tests for your application.

We'll use all new Python 3.7 features to build web services with asyncio and aiohttp.

5.2.4 Audience

Intermediate/Advanced skill level.

Participants should know how to program in Python, have a GitHub account, know how to commit their work using git and to create pull requests, and understand how REST API works. They should also be familiar with the concept of Python virtual environments and installing Python libraries using pip. Prior to attending the tutorial, participant should have Python 3.7 installed on their laptop.

This tutorial is not suitable for novices or someone new to Programming. For this tutorial, we will deploy the code to Heroku and GitHub. Participant may want to reconsider taking this tutorial if there is anything preventing them from uploading their source code to the cloud services like Heroku or GitHub.

5.2.5 Outline

1. Introduction (5 minutes)

- Go over the agenda
- Who we are: Python core developers, maintainer and long time user of aiohttp

2. Intro to asyncio (10 minutes)

- Explanation and demo of concurrency in Python

3. Intro to aiohttp (10 minutes)

- Why use aiohttp
- Provide documentation links: https://aiohttp.readthedocs.io/
- Examples of web apps with aiohttp

4. Hands-on intro to aiohttp server (30 minutes)

- Write our first aiohttp web server
- Introduce how to add routes/url
- Introduce how to handle GET and POST requests (useful for serving REST API)
- Provide example of how to handle file upload to the web server

5. Hands-on intro to aiohttp client (20 minutes)

- Write our first aiohttp client
- Introduce how to make GET and POST requests (useful for making REST API calls)
- Provide example of how to submit form and file upload

- Provide example of how to add request headers

6. Hands-on intro to HTML template with aiohttp (30 minutes)

- Introduce aiohttp-jinja2 renderer, and use it in the web server
- Provide example on how to structure the project (separating Python codebase and HTML templates directory)
- Documentation: http://aiohttp_jinja2.readthedocs.org/

7. Hands-on intro to middlewares (20 minutes)

- Introduce the concept of middleware, and when to use them
- Introduce how to add middleware to aiohttp server
- Provide example of error handling using middleware
- Documentation: https://aiohttp.readthedocs.io/en/stable/web_advanced.html#aiohttp-web-middlewares

8. Hands-on intro to session handling (20 minutes)

- Introduce how to handle user sessions/logging in and out on the web server

9. Hands-on intro to writing unit tests for aiohttp app (30 minutes)

- Writing tests for aiohttp client
- Writing tests for aiohttp server
- Tests using pytest and pytest-asyncio.

10. Extra time to finish exercises and QA

CHAPTER 6

Don't Be a Robot; Build the Bot

I've given this talk as keynote at DjangoCon US 2018. Even though I didn't need to submit anything there, I still prepared the abstract and outline. (I don't know how I would start preparing the talk without any outline!)

I submitted the proposal to PyCon US 2019, and it was accepted.

6.1 Acknowledgements and Background

I've had the idea about this talk since late 2017, however I never got around completing it.

Thanks to Nicholle James and Kenneth Love for inviting me as keynote speaker at DjangoCon US 2018. It gave me opportunity to give this talk. Nicholle helped brainstormed and refined the idea of this talk.

One of Zapier (my employer)'s core values is **"Don't Be a Robot; Build the Robot"**, which means *"Invest in tools and processes that lead to outsized impact so Zapier can be more productive"*. It resonates with me the most throughout the development of miss-islington. So it felt fitting to use it as the title of the talk.

The proposal was written summer of 2018, but the talk itself was completed sometime in October 2018, during a period where I experienced mental breakdown and depression. I took the month off from work to complete this talk.

6.2 Proposal

The following was the proposal I submitted to PyCon US 2019.

6.2.1 Title

Don't Be a Robot; Build the Bot

6.2.2 Duration

30 minutes slot

6.2.3 Description

Managing a large open source project like CPython is no easy task. Learn how the Python core team automated their GitHub workflow with bots, making it easier for maintainers and contributors to collaborate together. Even if you're not managing a large project, you can still build your own bot! Hear some ideas on what you can automate on GitHub and personalize your bot based on your own workflow. All you need is Python. Don't be a robot; build the bot.

6.2.4 Audience

This talk is for intermediate Python developers. I will mention concepts such as webhooks, APIs, webservices, and coroutines. Code snippets will include Python 3.7 features (async/await) and Python 3.6's f-strings. Not only the audience will learn about how Python got made, they will also learn how to build GitHub bots. They will inspired to add more automations to their life.

6.2.5 Outline

1. Self introduction (1 minute, real quick)
2. Problem in CPython's workflow (total: 4 minutes)
    - Core developers are outnumbered
    - Hundreds of contributors vs less than 50 active core developers
    - Pull requests are created at much faster rate than they get reviewed and merged
- Core python workflow is complicated
    - Every pull requests has to be checked, vetted, and reviewed thoroughly
    - Many tasks are tedious, but important
    - One of such tasks: doing backports
3. Backporting (total: 5 minutes)
    - What is backport? (3 minutes)
        - Applying changes from newer version to older versions of software
        - For CPython's purpose, backport is how you will receive bug fix releases (like 3.7.1 and 3.6.7)
    - cherry-picker.py was created to facilitate CPython backporting
    - Problems with cherry-picker.py (2 minutes)
        - Still manual process, requires a core developer to be in front of computer,
        - Often forgotten step
        - I received bug reports, and I'm unable to maintain it
        - It is just a boring chore (but important)
4. Build the bot (total: 8 minutes)
    - We knew a bot can do the backport, so let's build it

    - Architecture: (5 minutes)
        - the bot utilizes GitHub webhooks to receive pull request events
        - it uses GitHub REST APIs to perform actions on GitHub
        - other dependencies: gidgethub, aiohttp, Python 3.6+
        - hosted in Heroku, uses celery for running tasks asynchronously
    - why celery (3 minutes)
        - celery is used to run tasks asynchronously, in Heroku's worker dyno
        - Web requests in Heroku times out after 30 seconds
        - Some tasks performed by the bot are long running (cloning CPython repo takes at least 2 minutes)
    - this bot is called: miss-islington
5. miss-islington (4 minutes)
    - Initially, it has just one job: automatically create backport pull requests
    - As we gained experience with GitHub APIs and building bots, we keep asking, what else can the bot do?
    - Eventually, miss-islington was given commit privileges
    - miss-islington can now automatically merge pull requests
6. Don't be a Robot (6 minutes)
    - "I'm not managing large projects like CPython, why would I need a bot?"
    - Some tasks are boring. Really. Let the bots do it. Save time, and do the real important things
    - Maybe you don't even realize that a bot can do it? Some bots I'm making, for inspirations:
        - black out: bot for running black on pull requests
        - OOOS bot: Out-of-Open Source Github Autoreply bot
        - pyup Automerge bot: automatically merge pull requests made by pyup-bot
        - Automatically add code_of_conduct.md to GitHub repos
7. Conclusion and thanks (2 minutes)
    - Don't Be a Robot; Build the bot. Add more automation to your life.
    - This talk uses GitHub bot as examples, that's where I spend most of my time
    - You can build other bot. Many other apps have API
        - Slack, Twitter, Facebook, etc
    - If you do want to build GitHub bots: my step-by-step tutorial https://github-bot-tutorial.readthedocs.io/en/latest/

Talk Talk Talk by Darren Hayes.