Guidoさんに学ぶPythonが愛される理由

みんなのPython勉強会 #100

1/25, 2024

辻 真吾 (TSUJI, Shingo)

自己紹介

- 辻真吾（つじしんご）
  - 大学の研究所に勤めています
  - Pythonを使ったデータ分析が得意です
  - www.tsjshg.info
- 略歴
  - 工学部計数工学→大学院修士→創業したてのITベンチャー（入社時社員7人）
    - 2年半で退職
  - 大学院博士課程（バイオインフォマティクス）
    - 2005年博士（工学）
  - バイオインフォマティクス、エネルギーシステム、教育などにデータサイエンスを応用する研究活動
  - 2005年頃JavaからPythonに乗り換え
  - 2024年は良いことしか起こらないような気がしている

おもな著書

プログラミング未経験者向けに書いたPythonの入門書。いちばん売れたPythonの入門書として有名（2010, 2018改訂）

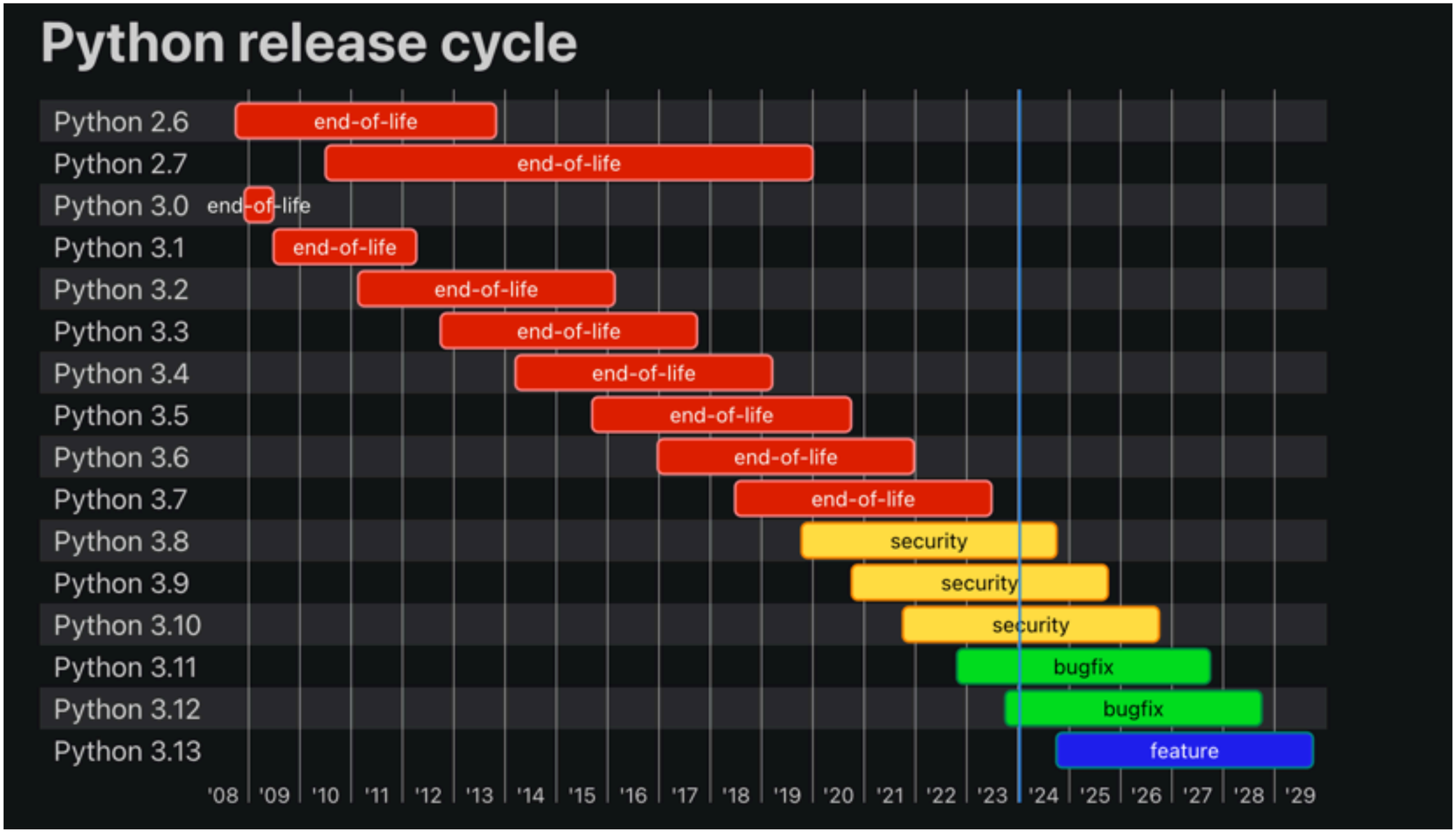
Pythonでデータサイエンスを実践するために必要な基礎ツールの解説。数学の章を執筆（2018, 2022改訂）

講談社データサイエンス入門シリーズの1冊。大学初等レベルのアルゴリズムとデータ解析の基礎をPythonで学ぶ本（2019）

データサイエンスをRとPythonの両方で学べる本。小学校の同級生との共著（2021）

『Pythonデータ分析 実践ハンドブック』が昨年9月にでました！

リリースサイクル（自己紹介スライドの3枚目）



毎年10月に新バージョン

5年でEOL（End of Life）を迎えるサイクル

https://devguide.python.org/versions/

まずはじめに ～Pythonの基本～

- プロジェクトを始めた人
  - Guido van Rossum（グイドヴァンロッサム）氏

- 最初のバージョン
  - 1991年に公開

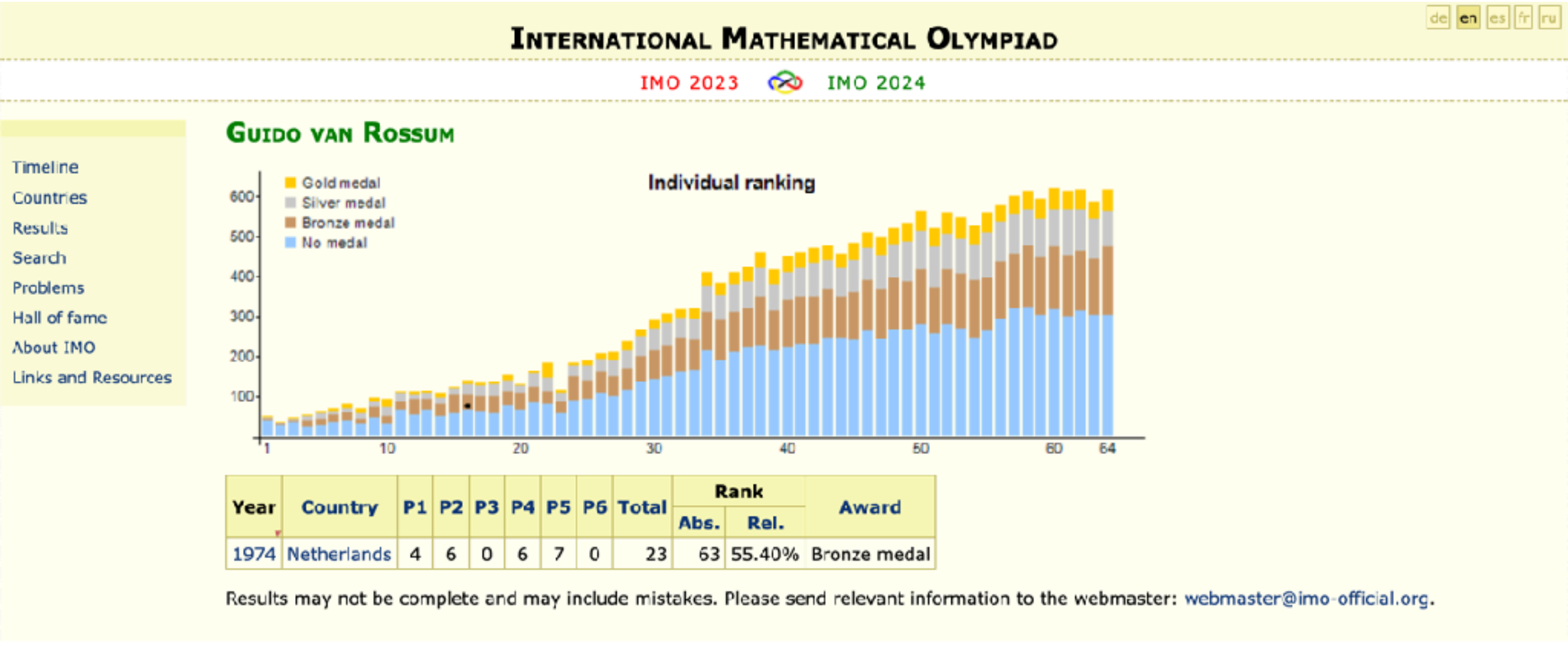
- 何でつくられているか？
  - C言語で作られたCPythonが広く利用されている

- どうしてこんなに人気なのか？
  - 今日のテーマ

Guido van Rossum
1956年生まれ（67歳）のオランダ出身プログラマ

https://gvanrossum.github.io/

ただの天才



de en es fr ru

INTERNATIONAL MATHEMATICAL OLYMPIAD

IMO 2023 IMO 2024

- Timeline
- Countries
- Results
- Search
- Problems
- Hall of fame
- About IMO
- Links and Resources

GUIDO VAN ROSSUM

Year	Country	P1	P2	P3	P4	P5	P6	Total	Rank		Award
									Abs.	Rel.	
1974	Netherlands	4	6	0	6	7	0	23	63	55.40%	Bronze medal

Results may not be complete and may include mistakes. Please send relevant information to the webmaster: webmaster@imo-official.org.

1974年開催の国際数学オリンピックで銅メダルを獲得している！

https://www.imo-official.org/participant_r.aspx?id=10303

NEC C&C賞を受賞！

公益財団法人
NEC C&C財団

ホーム | C&C賞 | 研究助成 | 財団概要

2023年度C&C賞受賞者

グループB

グイド　ヴァンロッサム　氏
Mr. Guido van Rossum

マイクロソフト　ディスティングイッシュド　エンジニア

業績記

プログラミング言語 Python の開発とオープン化への多大な貢献

2023年11月の受賞式のため来日
JDLA（日本ディープラーニング協会）とPyCon JP Associationがイベントを開催

https://www.candc.or.jp/kensyo/2023/group_b.html

まとめ記事と当日の動画あります

https://gihyo.jp/article/2023/12/guido-talkshow

https://www.youtube.com/watch?v=aaZuEPmHiQY

個人的にはまず、Guidoさんに謝りたい・・・

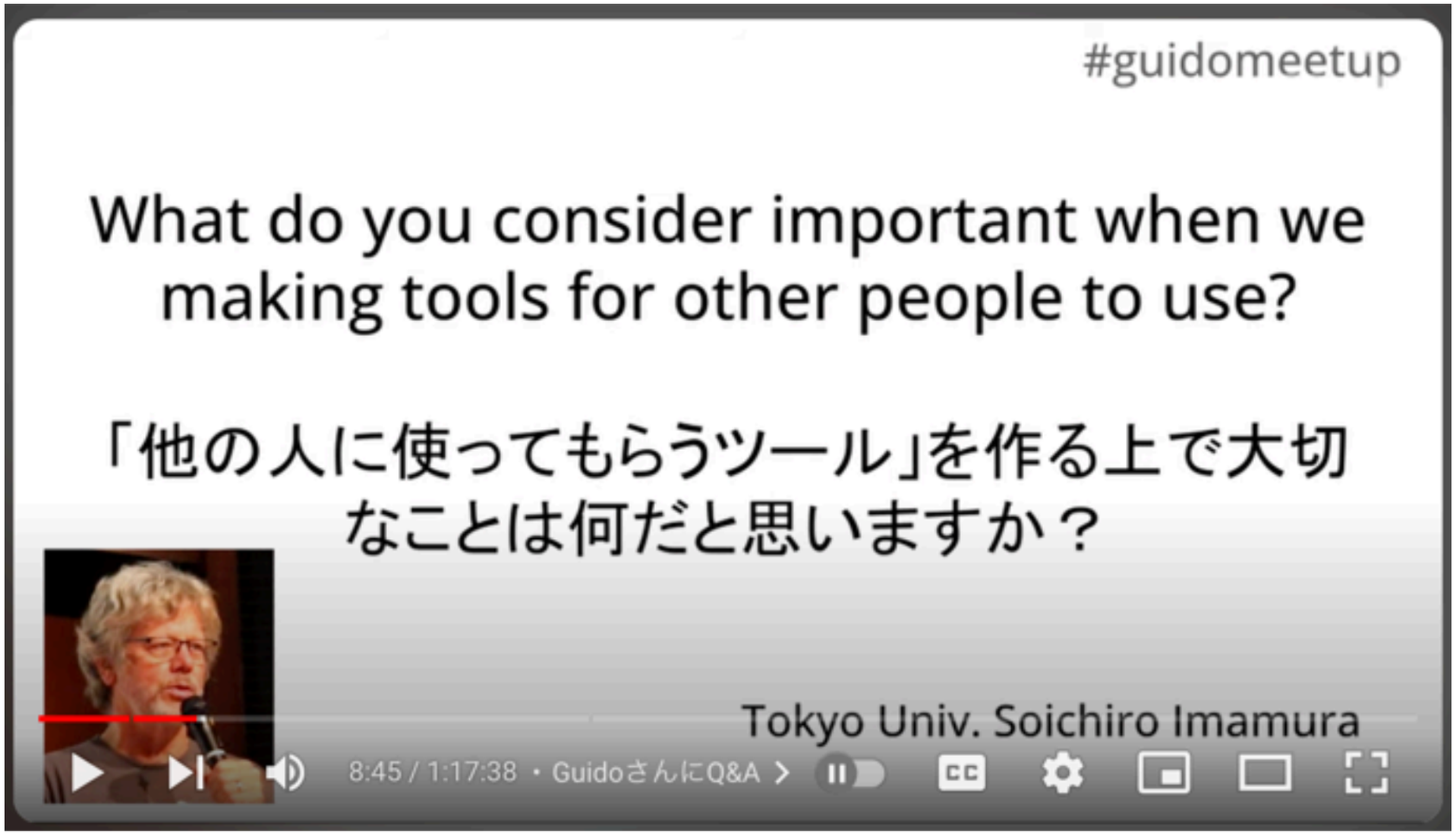
なぜ謝りたいか？

- 時々「なぜPythonはこんなに流行っているの？」と聞かれたとき、「偶然の産物じゃないですかねー」と答えていた

だいぶ前に読んだこの本からの影響

https://www.hayakawa-online.co.jp/shopdetail/0000000006787

徹底的なユーザ目線



ユーザが何を求めているかを知ることは難しいので
まず作る→使ってもらう→フィードバックをもらう→改良する
これが30年以上ずっとPythonがやってきたこと

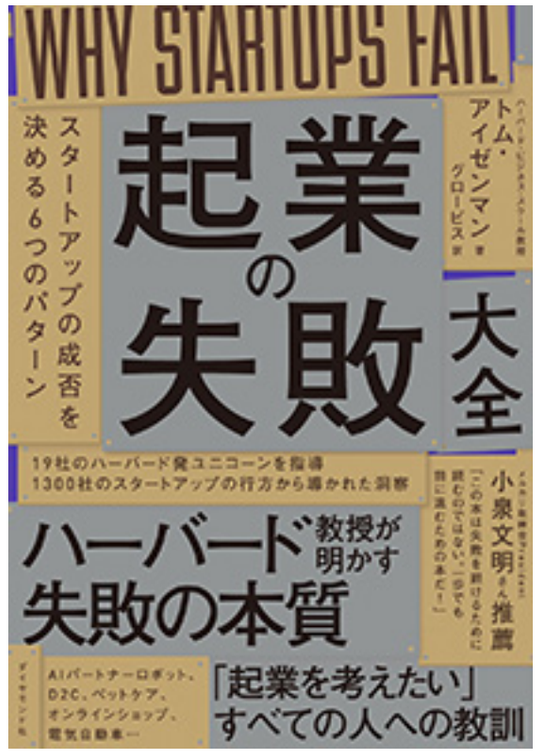
フィードバックは重要だけど、

- ユーザが求めるものをそのまま新機能として実装してはいけない
- なぜユーザがそれを求めるのかを掘り下げて考え、機能を追加する必要がある

起業の失敗大全に書いてある

- MVP（Minimum Viable Product）を作って顧客のフィードバックをもらう
- 求められるがままに新機能を追加
- 一部の顧客のためだけ、さらに悪いときは誰の役にも立たない機能を追加してしまう

- これは「フライング」というカテゴリーに分類される
- フライングを防ぐには、”顧客の真のニーズを理解することが重要”とある



https://www.diamond.co.jp/book/9784478113851.html

ABCの失敗からこの教訓を得たようです

Wednesday, April 27, 2016

King's Day Speech　からの抜粋

Sadly, for a variety of reasons, our marketing (or perhaps our timing) sucked, and after four years, ABC was abandoned. Since then I've spent many hours trying to understand why the project failed, despite its heart being so clearly in the right place. Apart from being somewhat over-engineered, my best answer is that ABC died because there was no internet in those days, and as a result there could not be a healthy feedback loop between the makers of the language and its users. ABC's design was essentially a one-way street.

About Me

Guido van Rossum

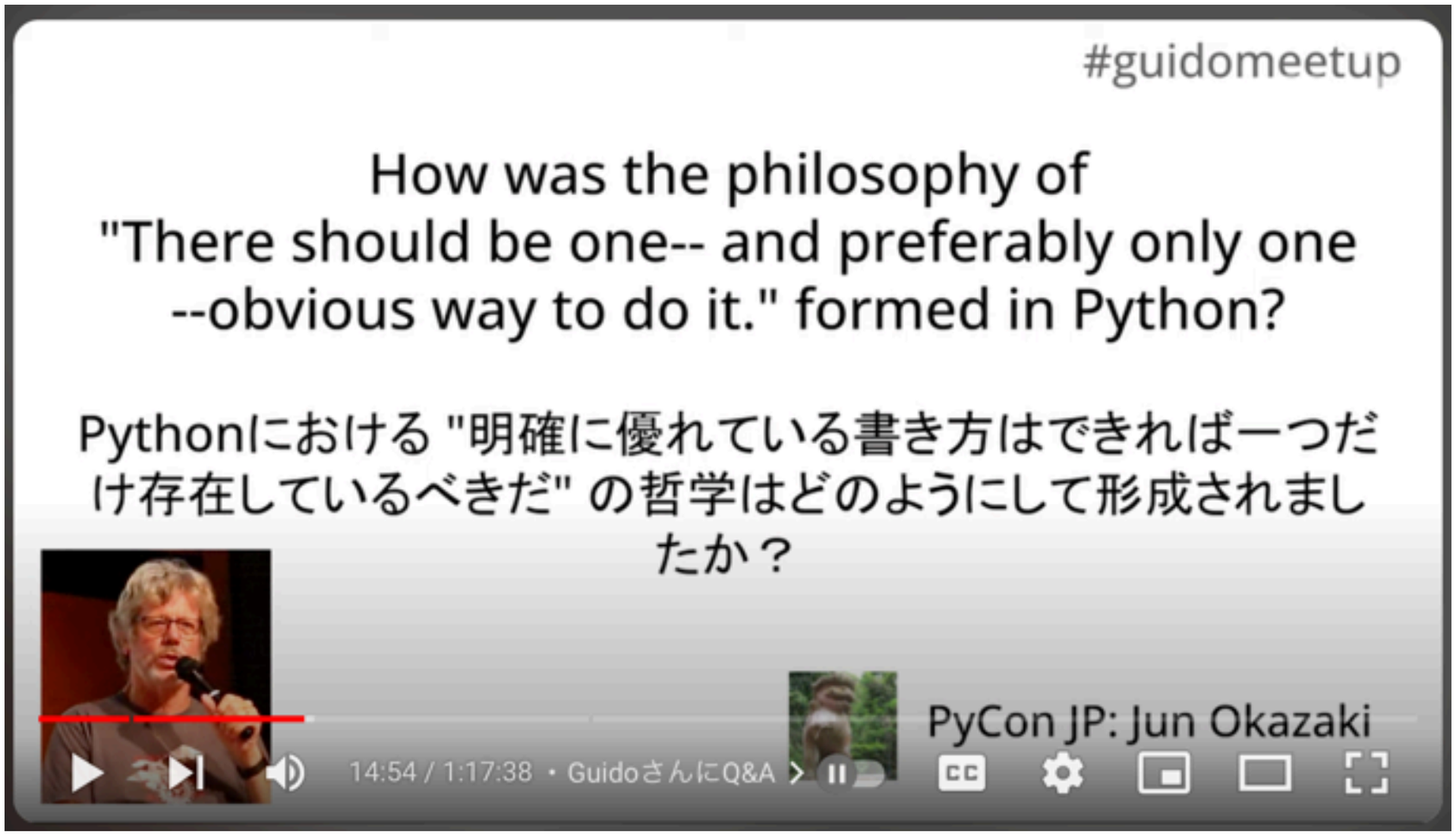
Python's BDFL

View my complete profile

Basicに代わる初級者向け言語としてABCを開発するも流行らず。
時間をかけて失敗を考察
インターネットが無い時代だったのでフィードバックを得ることが困難で
ユーザへの一方通行になってしまった

https://neopythonic.blogspot.com/2016/04/kings-day-speech.html

Pythonの哲学



この質問の背景とGuidoさんの回答を理解するために、まず周辺知識を紹介

The Zen of Python

```
>>> import this
The Zen of Python, by Tim Peters

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.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Pythonインタープリタを起動してimport thisとするとPythonの哲学が出てくる

作者はTim Peters（Guidoさんではない）

Tim Peters

0.9.1（90年代初期）からPythonの開発に参加

2006年米国ダラスで開催されたPyConでのインタビュー動画より

https://www.youtube.com/watch?v=1wAOy88WxmY

TimSortの開発者

- ソートとは、数字や文字を順番に並べ替えること

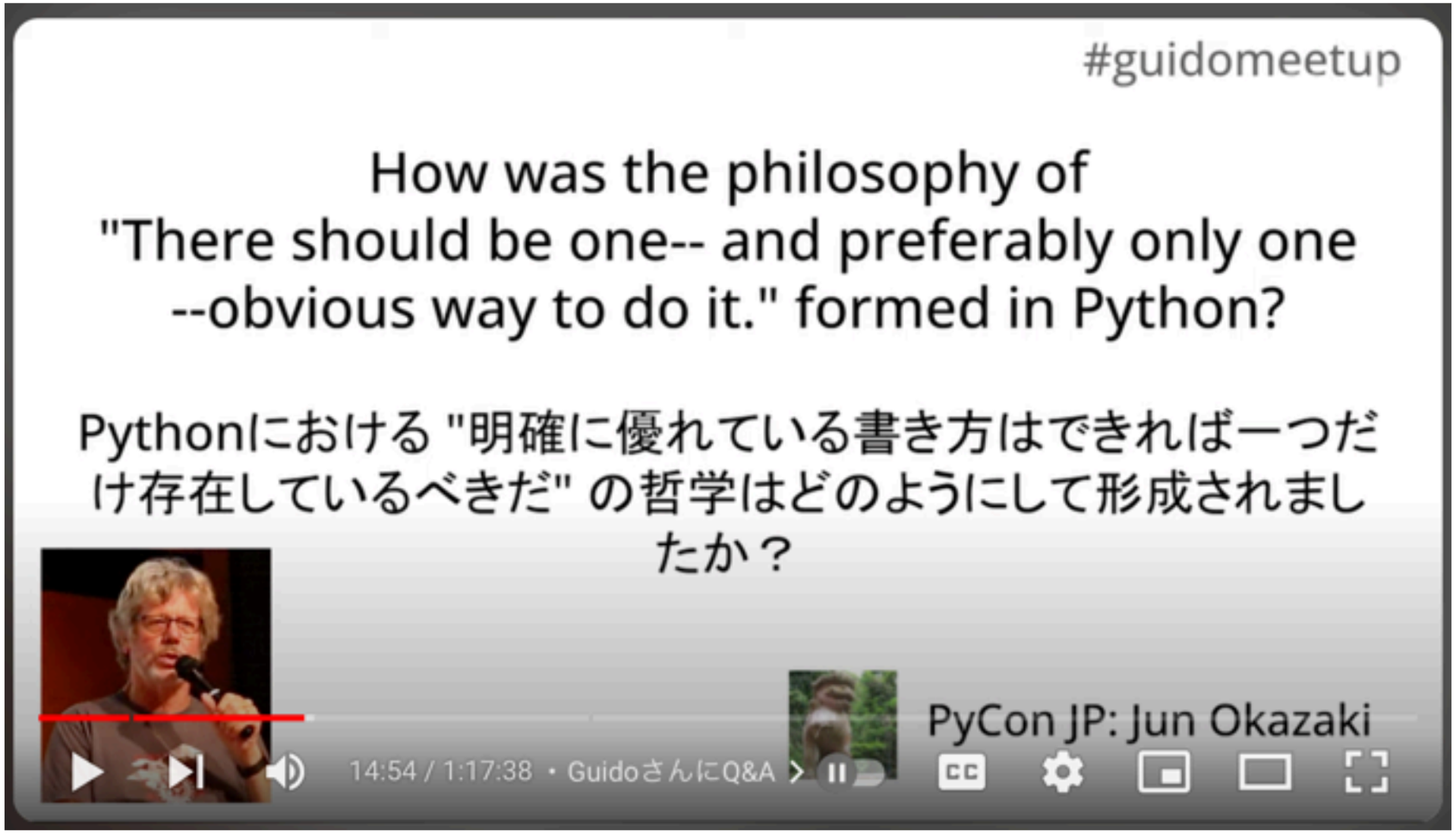
- これまでのコンピュータサイエンスの研究の成果で、いろいろなやり方（アルゴリズム）が開発されている
  - ソートについて詳しく知りたい方は拙著『Pythonで学ぶアルゴリズムとデータ構造』など

- 2001年、TimがPythonのためにTimSortを開発

- 性能が良い（実世界のデータで速い）ため、その他の言語でも使われている

https://bookclub.kodansha.co.jp/product?item=0000276050

Pythonの哲学



Pythonと似た言語にPerlがある

Perlはいろいろな書き方ができるので、同じ機能を実現するコードがプログラマによって変化する

Pythonはユーザ目線で分かりやすさ（読みやすさ）を重視

詳しく解説したブログもあります

https://python-history.blogspot.com/2009/01/pythons-design-philosophy.html

日本語訳もあります　https://python-history-jp.blogspot.com/2009/04/python.html

誰が書いても同じようなコードになることの意味

Wednesday, April 27, 2016

King's Day Speech　からの抜粋

Think of the incredible success of companies like Google or Facebook. At the core of these are ideas — ideas about what computers can do for people. To be effective, an idea must be expressed as a computer program, using a programming language. The language that is best to express an idea will give the team using that language a key advantage, because it gives the team members — people! — clarity about that idea. The ideas underlying Google and Facebook couldn't be more different, and indeed these companies' favorite programming languages are at opposite ends of the spectrum of programming language design. And that's exactly my point.

GoogleやFacebookのコアはアイディア

（うまく設計された）プログラミング言語を使えば、アイディアを簡潔に伝わりやすい形で表現できる！

実際、Googleの共同創業者Larry PageとSergey Brinは最初の検索エンジンのアイディアをPythonで表現した

https://neopythonic.blogspot.com/2016/04/kings-day-speech.html

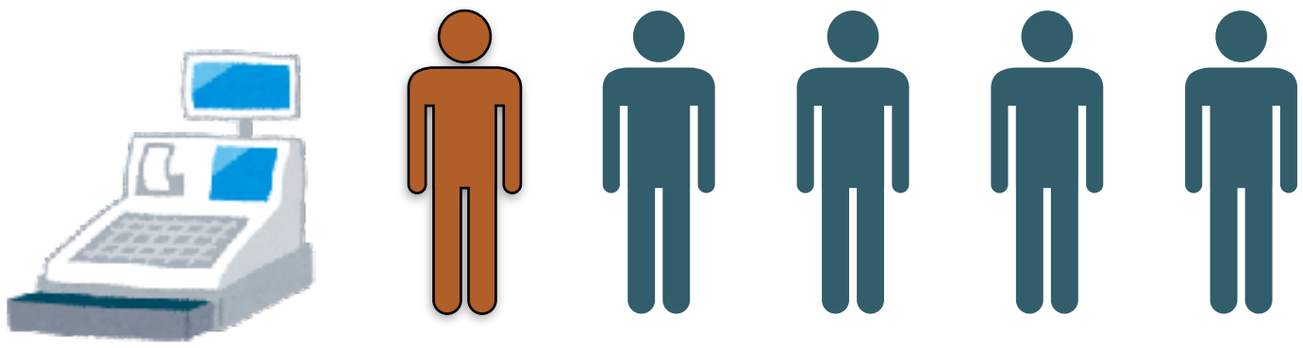
GIL（ぎる）の解消とPythonの高速化

まずは、用語の解説から

Global Interpreter Lockとは？（1/2）

インタープリタ（プロセス）が実行するスレッド
はある瞬間には1つのCPUしか使えないという制限

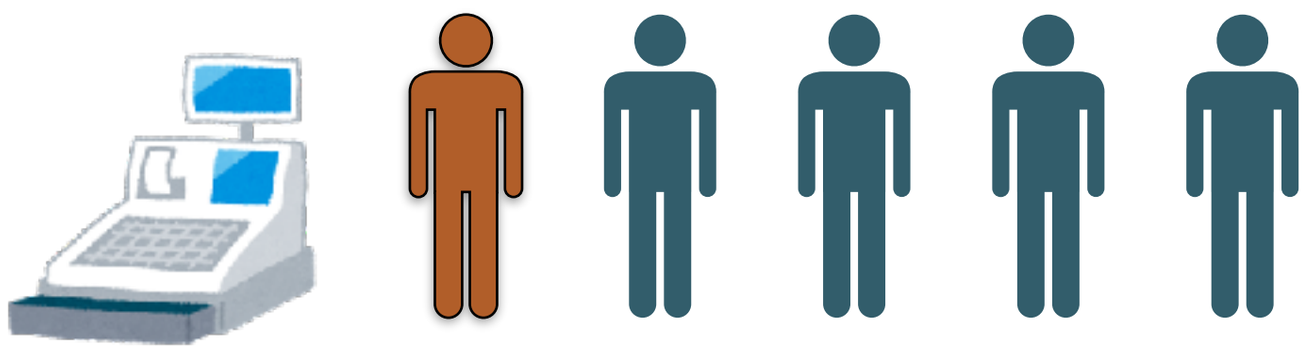
CPUをレジ、スレッドをお客さんの列とすると



CPU　　スレッド

1CPU（レジ）に1列（スレッド）なら何の問題もない

Global Interpreter Lockとは？　(2/2)



CPU（レジ）が2つあって、スレッドが2つ（お客さんが2列）あるのにレジを同時には使えない（ある瞬間はどちらか1つ）

（注）multiprocessingモジュールなど、いまでもこれを解決する手段はある

GILの制限を解放する

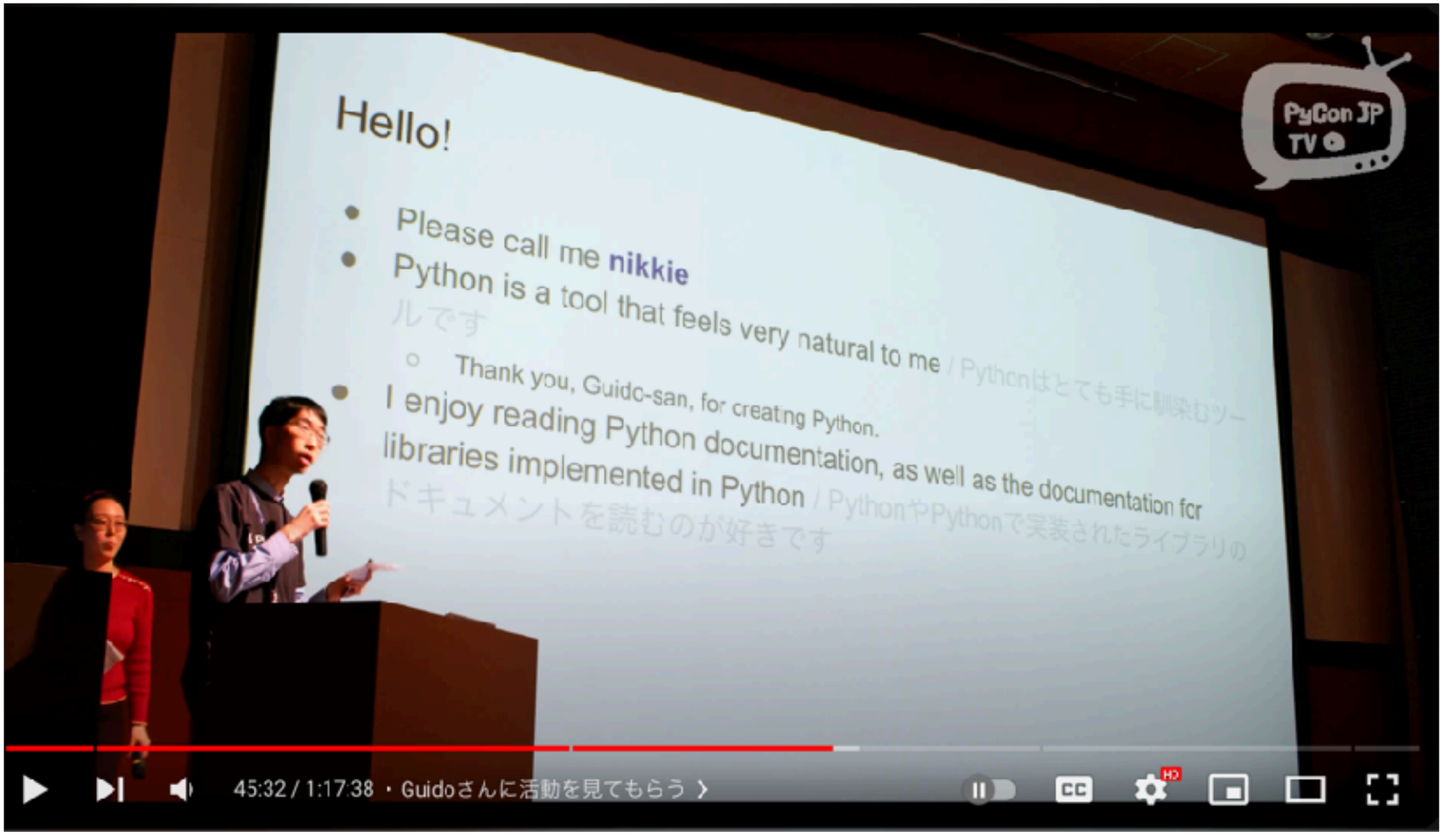
- 実装がシンプルになるなどの利点も多いが、議論の的になってきたのも事実
- Sam GrossさんによるGIL無しPythonのプロトタイプがすでにある
- 3.13（2024年10月のリリース予定）からGIL無しCPythonを選べる
- ただし、まだ道半ば。この先5年くらいかけてじっくり取り組む
- その他JIT（Just In Time）コンパイラの導入も検討されている
- ただ、高速化は思ったより難しいので、簡単にはいかないと思う

https://www.linkedin.com/in/samgross/

流行の大規模言語モデルを使った生成AIについて

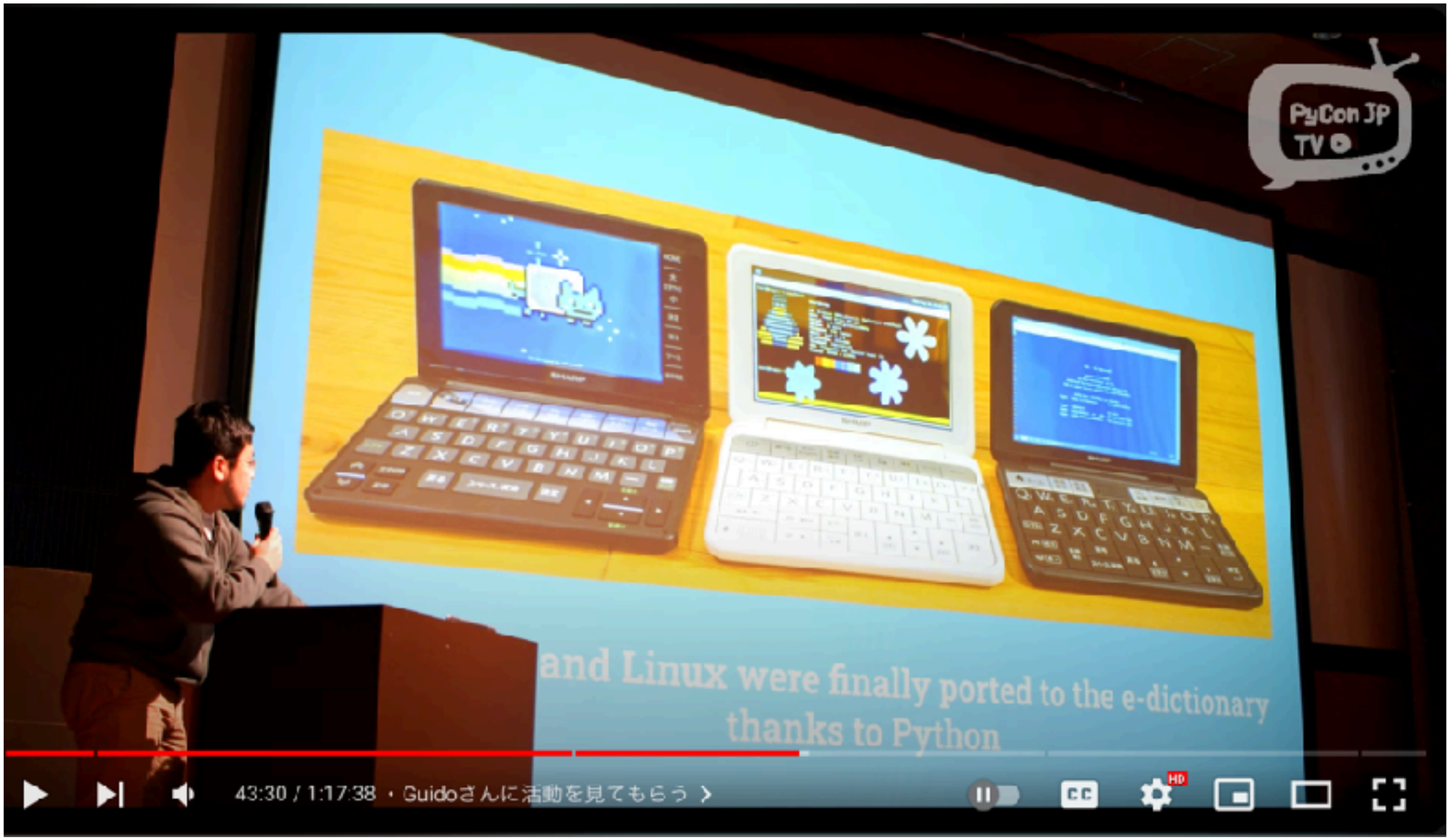
- 単純な作業をさせるのはいい
- プログラミング初心者の手助けにはなるだろう
- ただ、クリエイティブではない
- 「OSを設計しろ」とかいっても無理
- 「Pythonを高速化するには？」とか聞くと通り一遍の回答しか返ってこないしね
- ただ、コンピュータサイエンス分野が面白いことになってきているのは事実

Guidoさんに活動を見てもらうLT



我らがnikkie登壇！

個人的にはこれが1番



市販の電子辞書に無理矢理Linuxを入れる@puhitakuさん

Guidoさんも感想で「Super Cool」と絶賛

Follow your heart, Be curious!

- 高校卒業から大学入学の時期についての質問
- 数学と物理は得意だったので、数学を専攻（副専攻に物理を選択）
- 2年経ち、数学は難しい、物理はもっと厳しいことが分かった
- でも大学ではじめて出会ったコンピュータは楽しかった！
  - 当時はパンチカードでプログラミングだけどね

- 「会場にいるであろう18歳の質問者へ一言」という司会への返答がタイトルの言葉

Pythonが愛される理由

- **徹底的なユーザ目線にたった開発**
  - 失敗から学ぶ事は重要
  - GuidoさんによるとTimもユーザ目線の人
  - 外部ライブラリの開発者もユーザの1人

- **Pythonの根底にある強固な哲学**
  - 誰が書いても似たコードになる
  - アイディアの表現、交換に使える

- 結局、Guidoさんの人柄か？
  - 例）司会者からGIL（ぎる）の質問を受けたとき、第一声でglobal interpreter lockと言い直す

ご静聴ありがとうございました

今年もよろしくお願いいたします