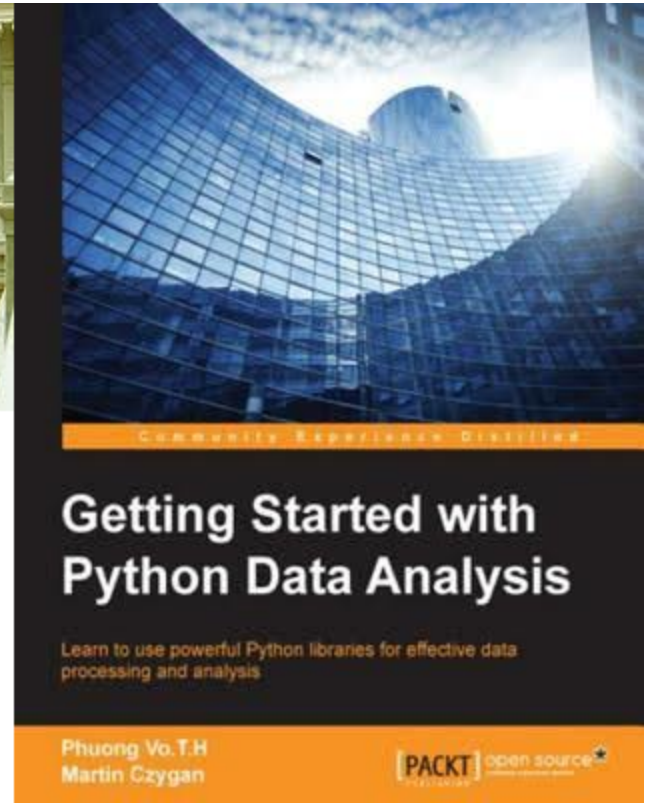
Packaging Python Applications

PyConBalkan, Belgrade, 2018-11-17 16:00-16:30

Martin Czygan / github.com/miku / @cvvfj

About me

- Software developer at Leipzig University Library
- Part-time consultant, co-author *Getting Started with Python Data Analysis* (2015)
- Maintainer of a few open source tools



About me

- interest in (build) automation: code, writing, data
- anecdata: ant, ephemeral VMs, tried to ease adoption of Python at workplace

There is some satisfaction in being able to just run *one command*.
There are also some trade-offs.

Packaging

- loosely defined as *approaches* and *tools* to create usable (installable, shippable) software
- there is this thing called: *The packaging gradient*

Packaging is moderately exiting

- not directly related to features
- many options

> ... plethora of packaging options ...
> (https://packaging.python.org/overview/)

- less formalized, project-dependent

So why care about packaging?

- as individual or team
- as company

Individuals or teams

- to share code
- collaborate or invite contributions

> It might seem strange to think about packaging before writing code, but this process does wonders for avoiding future headaches. (https://packaging.python.org/overview/)

As a company

- Aug 9, 2000: The Joel Test: 12 Steps to Better Code

There are two items related to packaging:

- #2 Can you make a build in one step?
- #3 Do you make daily builds?

On item #2

> By this I mean: how many steps does it take to make a shipping build from the latest source snapshot? On good teams, there's a single script you can run that does a full checkout from scratch, rebuilds every line of code, makes the EXEs, in all their various versions, languages, and #ifdef combinations, creates the installation package, and creates the final media — CDROM layout, download website, whatever.

On item #2

> If the process takes any more than one step, it is prone to errors. And when you get closer to shipping, you want to have a very fast cycle of fixing the "last" bug, making the final EXEs, etc. If it takes 20 steps to compile the code, run the installation builder, etc., you're going to go crazy and you're going to make silly mistakes.