

Four years of Python

Tales of an (un)experienced Pythonista

PyData Copenhagen 25/05/2022

Duarte O.Carmo

duarteocarmo.com - @duarteocarmo



Who the *hell* am I?

- */du-art/*
- From Lisbon, based in Copenhagen
- I like running a lot
- Jeg snakker ikke dansk (bit ashamed)
- *Past:* Strategy, Product Management, New Ventures, Management Consulting
- *Now:* ML Engineer @ Amplemarket / Contractor
- *Always:* Python (and Computers in general)



This is not a technical talk.

```
if years_of_experience <= 4:
```

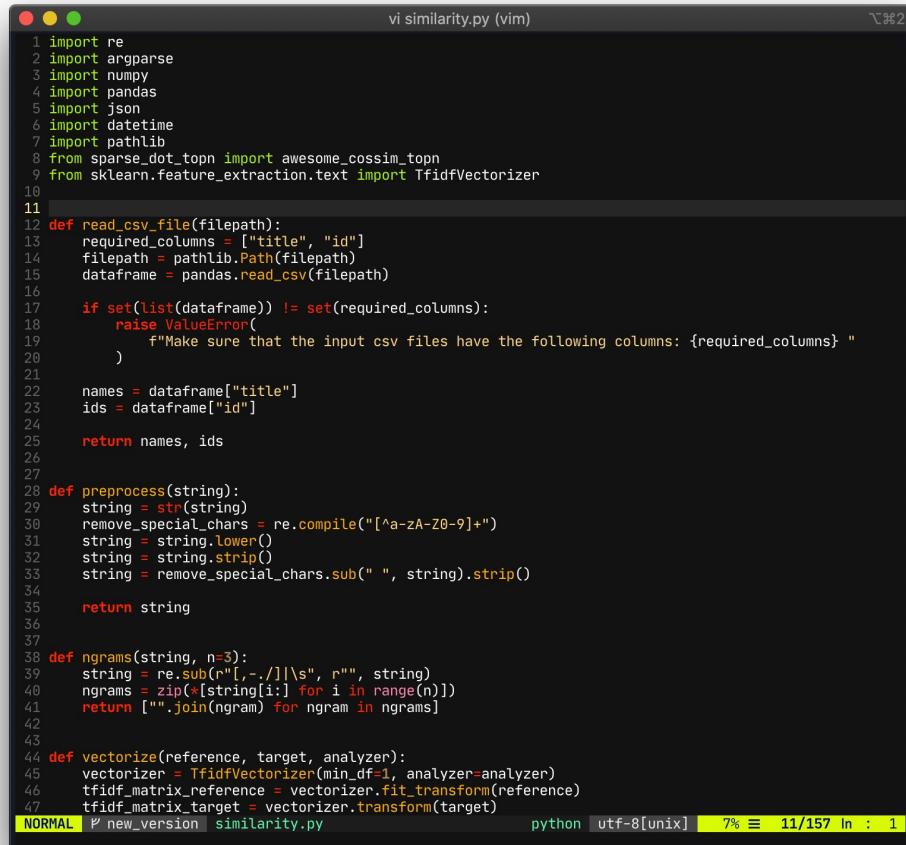
- How to be a better dev
- Tips and tricks
- Opinionated advice

```
else:
```

- Reminisce
- Celebrate
- Stay intentional

1 | Reading is better than googling

When we start, we have *superpowers*

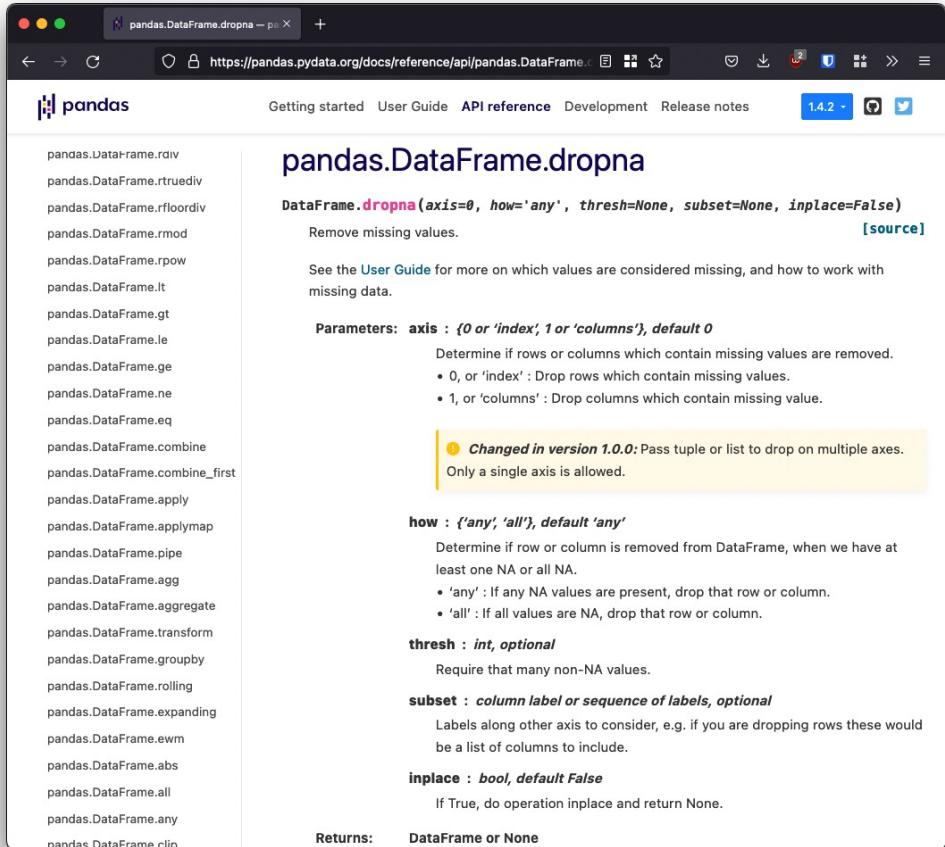


A screenshot of a terminal window titled "vi similarity.py (vim)". The code is a Python script named "similarity.py". The script imports various libraries like re, argparse, numpy, pandas, json, datetime, and pathlib. It defines functions for reading CSV files, preprocessing strings (removing special characters and converting to lowercase), generating n-grams, and vectorizing text using a TfidfVectorizer. The code uses f-strings for error messages. The terminal status bar at the bottom shows "NORMAL P new_version similarity.py" and "python Utf-8[unix] 7% ≡ 11/157 ln : 1".

```
1 import re
2 import argparse
3 import numpy
4 import pandas
5 import json
6 import datetime
7 import pathlib
8 from sparse_dot_topn import awesome_cossim_topn
9 from sklearn.feature_extraction.text import TfidfVectorizer
10
11
12 def read_csv_file(filepath):
13     required_columns = ["title", "id"]
14     filepath = pathlib.Path(filepath)
15     dataframe = pandas.read_csv(filepath)
16
17     if set(list(dataframe)) != set(required_columns):
18         raise ValueError(
19             f"Make sure that the input csv files have the following columns: {required_columns} "
20         )
21
22     names = dataframe["title"]
23     ids = dataframe["id"]
24
25     return names, ids
26
27
28 def preprocess(string):
29     string = str(string)
30     remove_special_chars = re.compile("[^a-zA-Z0-9]+")
31     string = string.lower()
32     string = string.strip()
33     string = remove_special_chars.sub(" ", string).strip()
34
35     return string
36
37
38 def ngrams(string, n=3):
39     string = re.sub("[.-/]|\s", "", string)
40     ngrams = zip([string[i:] for i in range(n)])
41     return ["".join(ngram) for ngram in ngrams]
42
43
44 def vectorize(reference, target, analyzer):
45     vectorizer = TfidfVectorizer(min_df=1, analyzer=analyzer)
46     tfidf_matrix_reference = vectorizer.fit_transform(reference)
47     tfidf_matrix_target = vectorizer.transform(target)
```

- Autocomplete
- Google
- Stack overflow
- Nails everywhere
- Pip install the world
- But.. We forget quickly

But there's quite nothing like reading



- What does it do?
- Options?
- Default behaviors
- Maybe I can re-use this
- **It *actually* sticks**

2 | **Explicit** is better than **implicit**

“I can make this program shorter”

```
print len((lambda lookandsay: (lambda func1, in1, current_depth1, target_depth1:  
func1(func1, in1, current_depth1, target_depth1))(lambda self, _in, current_depth,  
target_depth: _in if current_depth == target_depth else self(self, ''.join('%d%s'  
% (len(seq), seq[0])) for seq, trash in re.findall(r'((\d)\w*)', _in)),  
current_depth + 1, target_depth), lookandsay, 0, 40))(_input))
```

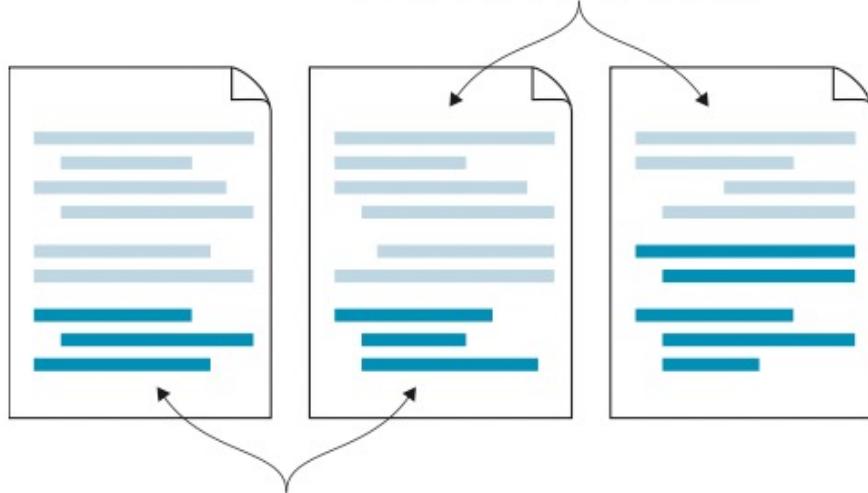
```
print len((lambda lookandsay: (lambda func1, in1, current_depth1, target_depth1:  
func1(func1, in1, current_depth1, target_depth1))(lambda self, _in, current_depth,  
target_depth: _in if current_depth == target_depth else self(self, ''.join('%d%s'  
% (len(seq), seq[0])) for seq, trash in re.findall(r'((\d)\w*)', _in)),  
current_depth + 1, target_depth), lookandsay, 0, 40))(_input))
```

Less lines = better code

```
strings = [x if isinstance(x, str) else pass for index, x in enumerate(list_)]
```

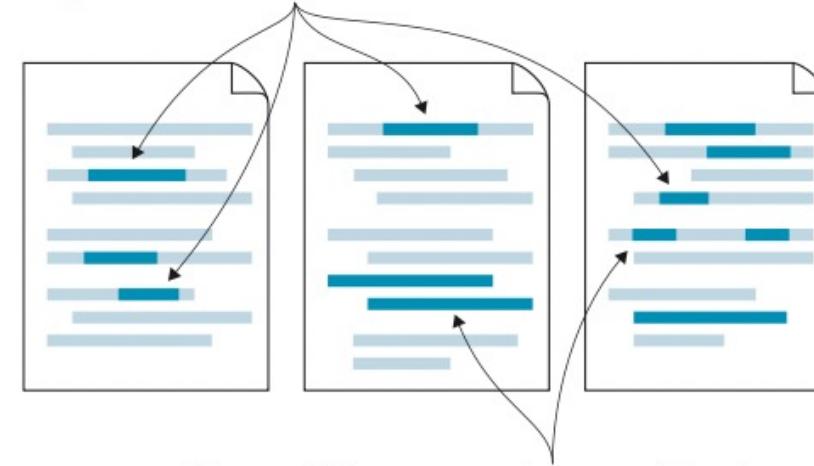
Scheduler, State Machine, Abstract, Controller, Operator...

**Extensible code doesn't require
the editing of existing code.**



**Extensible code allows you to add
a new feature by adding new code.**

**Code that isn't extensible requires many edits
throughout the code to add a new feature.**



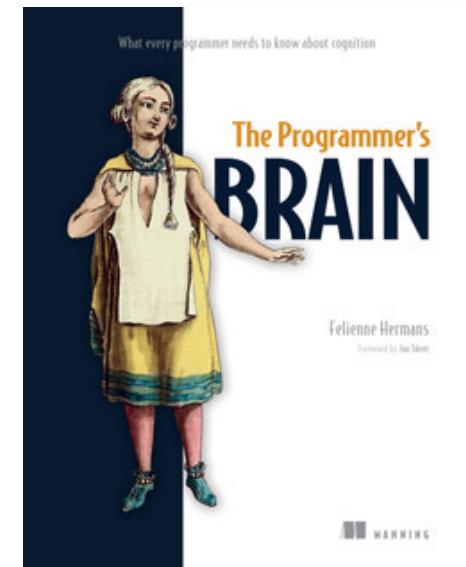
**Often, additions are made to conditional expressions
or by adding new `else` cases, making the code harder
to understand over time.**

[Reference: Practices of the Python Pro]

duarteocarmo.com - @duarteocarmo

Less lines != better code

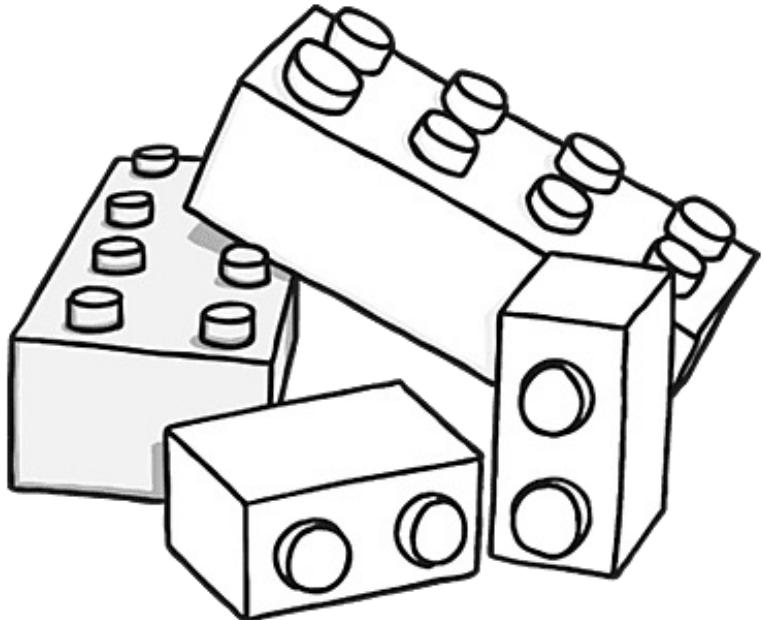
- Readable
- Understandable
- Changeable
- Shareable
- Enjoyable



3 | First make it work, then make it pretty



First make it work, then make it pretty



- The bare minimum
- Catching all exceptions
- 100% code coverage
- That weird edge case
- Do users care?
- What NOT to write

More ranting: duarteocarmo.com/blog/simple-software

duarteocarmo.com - @duarteocarmo

4 | **Test early, test often**

Tests are a mirrage....

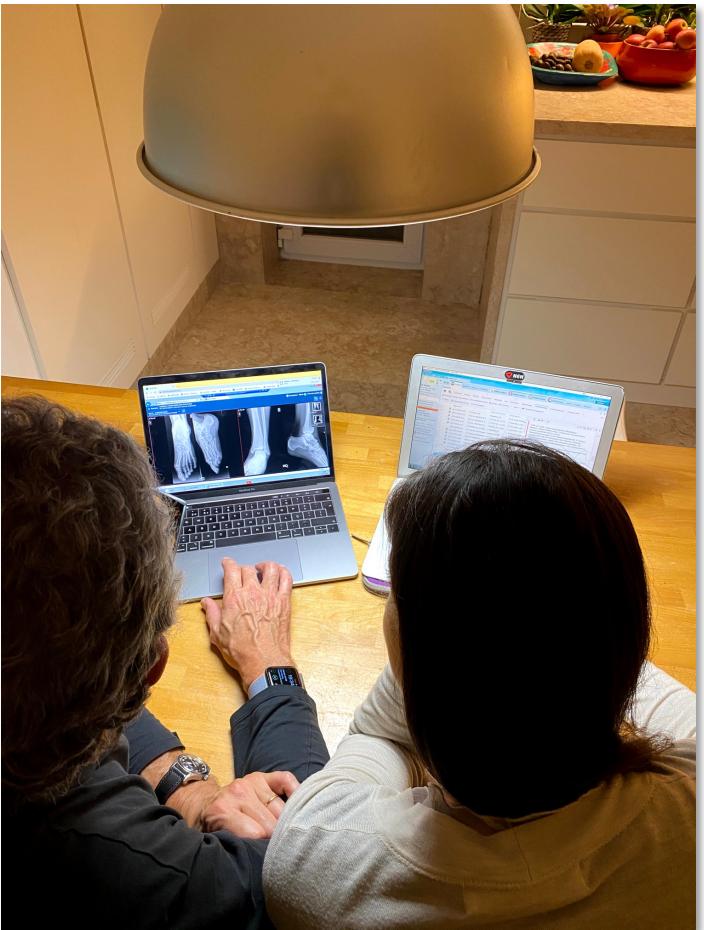
- 80% projects don't
- Value is not obvious
- Users don't see them
- Bugs can still happen

Tests are a mirrage....But they matter!

- 80% projects don't
- Value is not obvious
- Users don't see them
- Bugs can still happen
- Yours should
- Deploy with confidence
- What if the app goes down?
- Minimizing vs. Eliminating

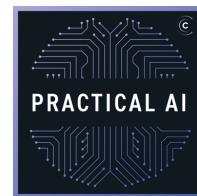
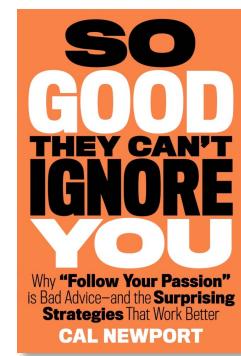
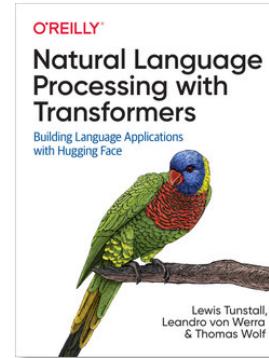
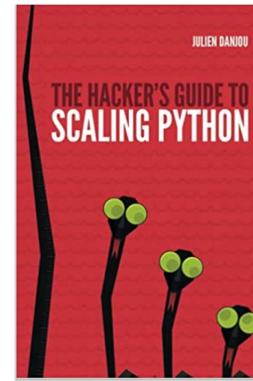
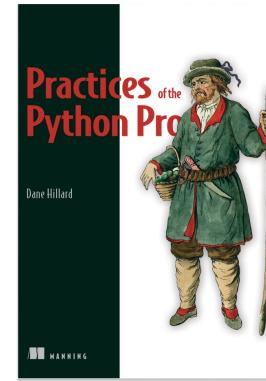
5 | Continuously learn

Python is our craft



We should be masters of our craft

- Study
- Stay up-to-date
- Learn regularly
- Build things
- Give back and write



An OCD list of resources I use

Books

Practices of the Python Pro
Hacker's guide to scaling Python
Designing Data-Intensive Applications
Serious Python

Tutorials

Flask Megatutorial
RealPython
Stack Abuse
Kaggle + GitHub

YouTube

CodingTech
Sentdex
Theo
MLOPs Community

Podcasts

Talk Python to Me
Python Bytes
Podcast.__init__
Practical AI

News

PyCoder's Weekly
Medium
Awesome Python Weekly
Reddit RSS

...

Thank you, questions?