# Pydon'ts

## Write elegant Python code (v1.1)

by Rodrigo Girão Serrão

DjangoCon US 2021

## About me

Rodrigo Girão Serrão

- Formal education: maths
- Writing Python for 9 years
- Training/teaching:
    - Python, maths, etc (mathspp.com)
    - APL (Dyalog Ltd.)

@mathsppblog

## Task

## Code...

## Refactoring recap

## Starting point

```python
def myfunc(a):
    empty=[]
    for i in range(len(a)):
        if i%2==0:
            empty.append(a[i].upper())
        else:
            empty.append(a[i].lower())
    return "".join(empty)
```

## Code style matters

```python
def myfunc(a):
    empty = []
    for i in range(len(a)):
        if i % 2 == 0:
            empty.append(a[i].upper())
        else:
            empty.append(a[i].lower())

    return "".join(empty)
```

## Naming matters

```python
def alternate_casing(text):
    chars = []
    for idx in range(len(text)):
        if idx % 2 == 0:
            chars.append(text[idx].upper())
        else:
            chars.append(text[idx].lower())

    return "".join(chars)
```

## Enumerate me

```python
def alternate_casing(text):
    chars = []
    for idx, char in enumerate(text):
        if idx % 2 == 0:
            chars.append(char.upper())
        else:
            chars.append(char.lower())

    return "".join(chars)
```

## Nest sparingly

```python
def alternate_casing(text):
    chars = []
    for idx, char in enumerate(text):
        if idx % 2 == 0:
            capitalised = char.upper()
        else:
            capitalised = char.lower()
        chars.append(capitalised)

    return "".join(chars)
```

## Leverage the PSL

```python
from itertools import cycle

def alternate_casing(text):
    chars = []
    funcs = cycle((str.upper, str.lower))
    for char, func in zip(text, funcs):
        chars.append(func(char))

    return "".join(chars)
```

## References

- Pydon'ts:
  - Bite-sized refactoring, https://mathspp.com/blog/pydonts/bite-sized-refactoring
  - Does elegance matter, https://mathspp.com/blog/pydonts/does-elegance-matter
  - Code style matters, https://mathspp.com/blog/pydonts/code-style-matters
  - Naming matters, https://mathspp.com/blog/pydonts/naming-matters
  - Enumerate me, https://mathspp.com/blog/pydonts/enumerate-me
  - Zip up, https://mathspp.com/blog/pydonts/zip-up

gum.co/pydonts

Rodrigo Girão Serrão | rodrigo@mathspp.com

@mathsppblog

mathspp.com/subscribe

/mathspp/talks

$$\overbrace{\underbrace{\text{rodrigo}}_{\text{name}}\text{@}\underbrace{\text{mathspp.com}}_{\text{site}}}^{\text{email}}$$