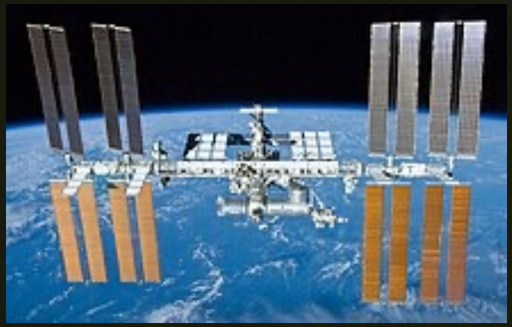
Slides/Code at github.com/ste-pool/europython2024 →

DESIGNING CONFIG FILES

The Conflicting Needs of Programmers and Users

WHO AM I?

WHERE DO I WORK?



What’s in this talk?

- (Brief) Introduction to configs/tomls
- Configuration driven programs
- Separation of concerns
- Why to not use library files
- How to deal with lots of config files
- Pushing complexity into config files
- Tips/tricks for debugging and testing

Introduction to configs

```
port = 8080
index = “homepage.html”
```

Introduction to toml

```toml
a = 1
[hello]
x = "yes"
[hello.there]
y = "no"

[[yes]]
five = 5
[[yes]]
four = 4
```

```python
import tomllib
with open(“config.toml”, “rb”) as cfg:
  data = tomllib.load(cfg)
```

```json
{
  "a": 1,
  "hello": {
    "x": "yes"
    "there": {
      "y": "no"
    }
  },
  "yes": [
    {"five": 5},
    {"four": 4},
  ]
}
```

Separation of concerns

```
living_room_heater_id = “living_room_heater”
living_room_heater_name = “Living Room Heater”
living_room_heater_temperature_target = 34
living_room_heater_temperature_tolerance = 2
living_room_heater_temperature_minimum_shutdown_time = 2
...
```

Separation of concerns

```
[living_room]
heater_id = “living_room_heater”
heater_name = “Living Room Heater”
[living_room.temperature]
target = 34
tolerance = 2
minimum_shutdown_time = 2
```

Separation of concerns

```python
def set_temperature(temp_config):
  temperature.set_temperature(temp_config["target"])
  ...
def main(config):
  ...
  living_room = config["living_room"]
  set_temperature(living_room["temperature"]) ...
```

Engineers love abstraction

- Bakers do not

```
[oven]
import = “./appliance_settings/fan_oven.toml”
...
[mixer]
import = “./appliance_settings/mixer_high_speed.toml”
...
[recipe]
import = “./recipe_book/brownies.toml”
...
# import isn’t an inbuilt keyword but easy to implement
```

Engineers love abstraction

- Bakers do not

- Okay we'll store them as .toml.template
- Okay we'll store both .toml.template and .toml
- So… We're stuck with one file?

```toml
[oven]
import = “./appliance_settings/fan_oven.toml”
[mixer]
import = “./appliance_settings/mixer_high_speed.toml”
[recipe]
import = “./recipe_book/brownies.toml”
```

Engineers love abstraction

```
[oven]
type = "fan"
preheat = true
duration = 600
[oven.temperature]
target = 230
tolerance = 5
...
[mixer]
speed = "medium"
stability = 6
[mixer.quality_check]
consistency = 7
...
[recipe]
ingredients = ["flour", "eggs", "sugar", "chocolate"]
...
```

```
[oven]
type = "fan"
preheat = true
duration = 600
[oven.temperature]
target = 230
tolerance = 5
...
[mixer]
speed = "low"
stability = 6
[mixer.quality_check]
consistency = 5
...
[recipe]
ingredients = ["flour", "eggs", "sugar"]
...
```

Engineers love abstraction

```
[oven]
type = “fan”
preheat = true
duration = 600
[oven.temperature]
target = 230
tolerance = 5
…
[mixer]
speed = “medium”
stability = 6
[mixer.quality_check]
consistency = 7
…
[recipe]
ingredients = [“flour”, “eggs”, “sugar”, “chocolate”]
…
```

```
[oven]
type = “fan”
preheat = true
duration = 600
[oven.temperature]
target = 230
tolerance = 5
…
[mixer]
speed = "ow”
stability = 6
[mixer.quality_check]
consistency = 5
…
[recipe]
ingredients = [“flour”, “eggs”, “sugar”]
…
```

Engineers love abstraction

- Bakers do not

```
assert_differences("browines.toml", "cookies.toml", [
    ("mixer.speed", "medium", "low"),
    ("mixer.quality_check.consistency", 7, 5),
    ("recipe.ingredients",
          ["flour", "eggs", "sugar", "chocolate"],
          ["flour", "eggs", "sugar"],
    )
])
```

Explosion of config files

Cookies_electric.toml

```toml
[oven]
temperature = 200
preheat = true
duration = 600
```

Cookies_fan.toml

```toml
[oven]
temperature = 180
preheat = true
duration = 600
```

Explosion of config files

Cookies_electric.toml

```
[oven]
temperature = 200
preheat = true
duration = 600
```

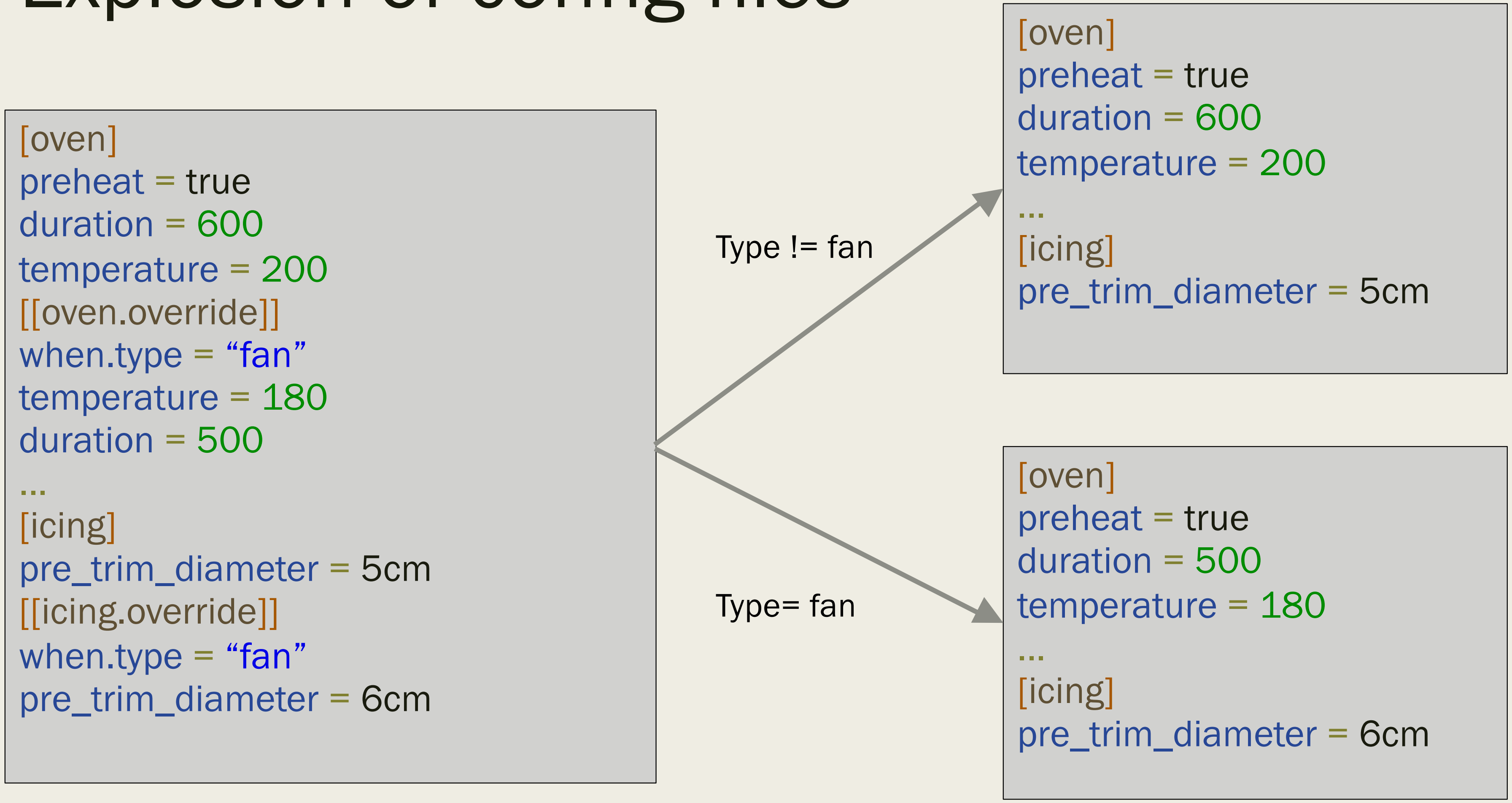
Cookies_fan.toml

```
[oven]
temperature = 180
preheat = true
duration = 600
```

All_the_cookies.toml

```
[oven]
preheat = true
duration = 600
temperature = 200
[[oven.override]]
when.type = “fan”
temperature = 180
```

Explosion of config files

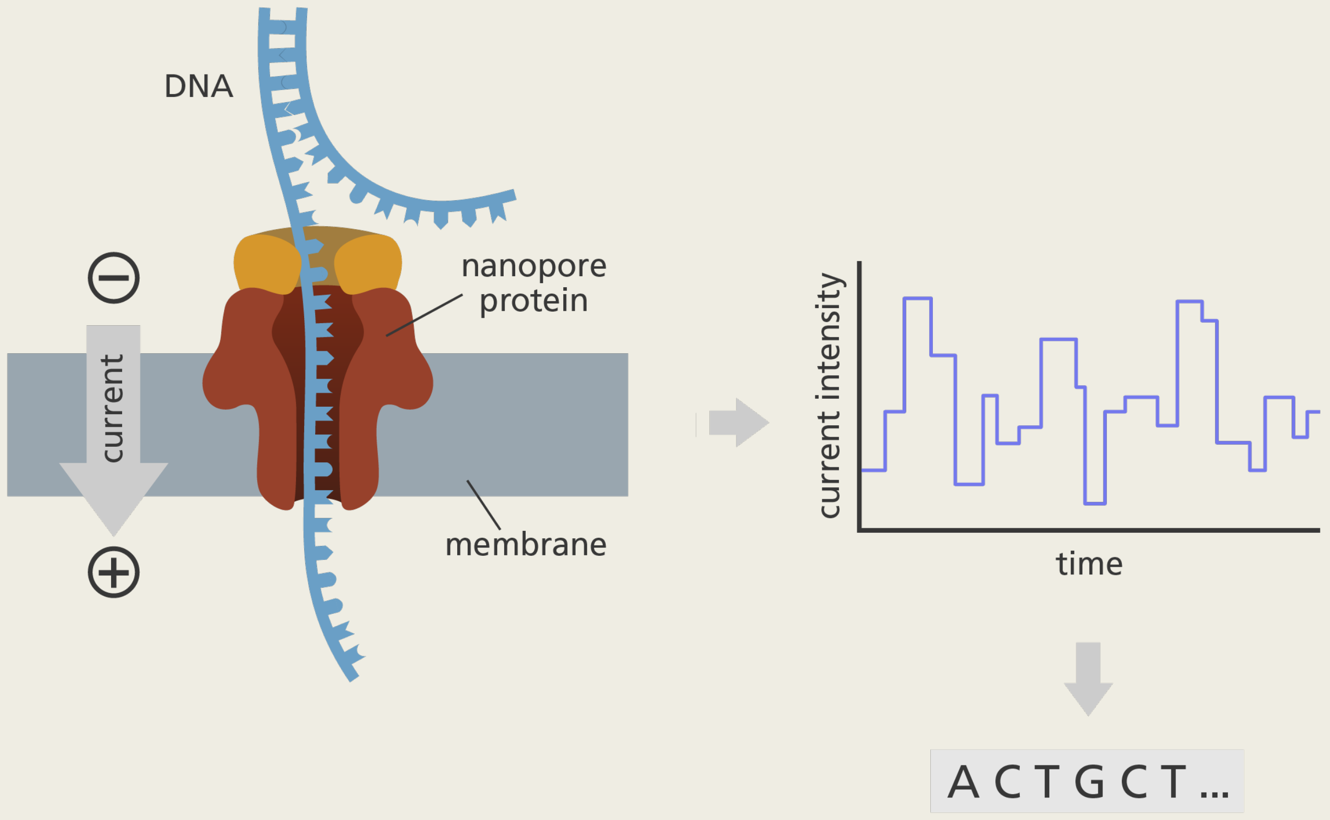


```
[oven]
preheat = true
duration = 600
temperature = 200
[[oven.override]]
when.type = “fan”
temperature = 180
duration = 500
…
[icing]
pre_trim_diameter = 5cm
[[icing.override]]
when.type = “fan”
pre_trim_diameter = 6cm
```

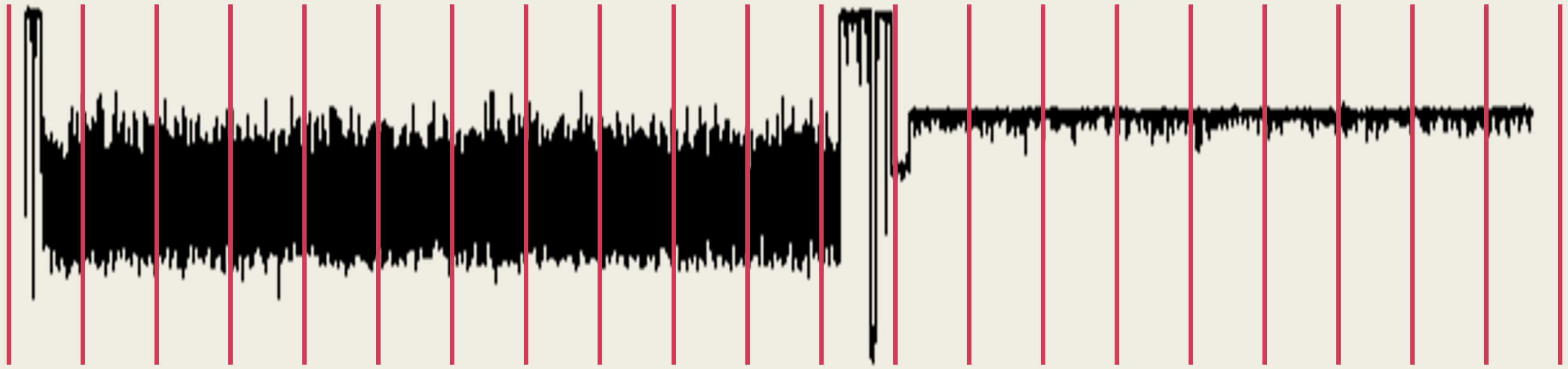
```
[oven]
preheat = true
duration = 600
temperature = 200
…
[icing]
pre_trim_diameter = 5cm
```

```
[oven]
preheat = true
duration = 500
temperature = 180
…
[icing]
pre_trim_diameter = 6cm
```

Pushing complexity into configs



Pushing complexity into configs



```
[analysis_configuration.read_classification.parameters]
strand = “(150 > median > 100) & (sd > 5) | ...”
pore   = “(250 > median > 200) & (sd < 1) & ...”
block  = “(180 > median > 150) & (sd < 1) & ...”
```

Pushing complexity into configs

```
[oven]
preheat = true
duration = 900
temperature = 180
```

```
[oven_1]
preheat = true
duration = 900
temperature = 180

[oven_2]
preheat = true
duration = 900
temperature = 180
```

Pushing complexity into configs

```
[mix_crust]
appliance = “mixer”
…
[mix_filling]
appliance = “mixer”
…
[oven_prebake]
appliance = “oven”
…
[oven_bake]
appliance = “oven”
…
recipe_order = [“mix_crust”, “oven_prebake”, “mix_filling”, “oven_bake”]
```

```
for section in config["recipe_order"]:
    section_config = config[section]
    match section_config["application"]:
        case “oven”:
            oven.execute(section_config)
        …
```

Tips for debugging/testing

- Output full *effective* config
- Tool to diff configs
  - *Asserting differences between configs*
- Debug config logging in functions
- Pydantic models for validation
- Pointless config options

Summary

1. Group config options to increase clarity
2. Try to not refactor into library configs to increase portability
3. Add runtime overrides to mitigate lots of similar configs
4. Push complexity into config files to increase flexibility
5. Output effective config in logs

- It is more work on developers side to maintain
  - *Making it easier for non technical people to contribute and check*