FORMAT | LINT | PRE-COMMIT

✨🐍✨

Speaker notes

Today we are going to talk about tools that will help you have a consistent python codebase across projects and teams.
And how to keep it that way by automating the process with pre-commit.

linkedin.com/in/arthurio

notivize.com

github.com/arthurio/pre-commit

Speaker notes

I'm Arthur Rio, CTO at Notivize. We have built a notification system plateform so you don't have to build your own. If you need notications with complicated rules, multiple channels, templates, make sure to check us out!

And if you or someone you know loves code quality as much as I do and wants to join an early seed-funded startup, please reach out to me.

The slides, configuration samples and code examples are available on the github repo you see here.

TOOLS

🧰

FORMATTERS 🧼✨

isort: isort your imports, so you don't have to.

black: The Uncompromising Code Formatter.

flynt: String formatting converter.

Speaker notes

Formatters will make modifications to your code in place. So, if you want to run them manually, I would suggest you to stage your changes first (`git add`) so that you can see the changes easily with `git diff`.

Why are formatters important?

- Keep the code reviews focused on the logic and structure of the code
- And remove all the "nit" comments about styling issues
- Consistency increases code readability

Tools:

- isort: Helps you keep your imports ordered and grouped properly: Standard library, third party, local/application.
- black: Max line length, line breaks, quotes, spacing between function and class definitions, trailing commas, etc.
- flynt: Transform as many string formatting as possible into f strings.

LINTERS 👮‍♀️👮‍♂️

bandit: Find common security issues.

flake8: Your Tool For Style Guide Enforcement.

mypy: Static type checker.

Speaker notes

- bandit:
    1. api key, password, token leaks.
    2. usage of "unsafe" packages (random, pickle) or algorithms (md5)
    3. bad file permissions
    4. mark safe
    5. etc.
- flake8:
    1. Unused/undefined variable
    2. Unused/missing import
    3. wildcard import
    4. missing formatting argument
    5. Function redefinition (copy/paste test function)
    6. etc.
- flake8 plugins: flake8-docstrings, flake8-fastapi, flake8-eradicate, etc. Search for `awesome flake8 extentions`.

Linters help you identify problems before they become one and most of all can help with code completion and hints if you use an IDE or editor that supports it.

AUTOMATION 👨‍🔧🤖

pre-commit: A framework for managing and maintaining multi-language pre-commit hooks.

Speaker notes

All sorts of hooks are supported, not just pre-commit, but that's the only one I personally have a use for.

CONFIGURATION

Keep it to a minimum

Speaker notes

If you can, try not to change any of the defaults, especially if you have multiple projects. They'll get out of sync quickly. And what makes it somewhat worse is that there is not yet a consensus on using pyproject.toml for configuration, only black and isort are using it at the moment.

BLACK

```toml
# pyproject.toml

[tool.black]
line-length = 120
```

Speaker notes

Black is opinionated for a reason and I agree with most of their decisions so I would recommend to stick to the defaults, especially if you don't want to justify yourself, you can just blame it on Łukasz Langa and cut the conversation short. Here you can see that I changed the line length but it's based on my code style.

ISORT

```toml
# pyproject.toml

[tool.isort]
profile = "black"
line_length = 120
skip_gitignore = true
```

Speaker notes

- Skiping gitignore is critical if you place your virtualenv in your repo or have a node_modules folder lying around for example.
- profile = "black" is important to make it compatible otherwise they'll fight each other for the formatting of the imports.
- line_length is just there to keep things consistent across tools.

FLYNT

Nothing to see here 👀 ...

Speaker notes

Flynt just magically works all the time. I haven't had a single issue with it after 2 years of use.

BANDIT

```
# .bandit

[bandit]
targets: src,tests
skips: B101
```

B101: assert used

Suppressing individual lines: `# nosec: B101`

Speaker notes

The configuration for bandit is a bit painful... But it might catch big mistakes like commiting a token, api key or password. So worth having IMO.

- ⚠️ Don't mixup the "ini" file **AND/OR** the "yaml" config file. ⚠️
- Assert is removed with compiling to optimized byte code.
- Bandit likes to complain about random, use the secrets module instead if you can or silent individual lines with `# nosec: B311`

FLAKE8 AND PLUGINS

```
# .flake8

[flake8]
max_line_length = 120
extend-ignore =
    # Whitespace before ":"
    E203
    # Missing docstring
    D1
docstring-convention = google
```

Ignore errors: `# noqa: E201`

Speaker notes

Gives you great insights like unused imports, undefined variables, etc.

- E203: Incompatible with black for things like ranges.
- D1: Used with flake8-docstrings. I don't care about missing docstrings as I try to keep them to a minimum. Instead I prefer to rely on explicit variable and function names, as well as type hints.

Use `extend-ignore` instead of `ignore` to keep defaults.

Doesn't support pyproject.toml because of legacy code around configuration that needs to be cleaned up first.

MYPY

```toml
# pyproject.toml

[tool.mypy]
mypy_path = ["src"]
show_error_codes = true

[[tool.mypy.overrides]]
module = [
    "package.*",
    "otherpackage.*",
]
ignore_missing_imports = true
```

Silencing errors: `# type: ignore[arg-type]`

Speaker notes

- Mypy supports pyproject.toml only since the latest release couple weeks ago (0.901)
- `mypy_path`: May not be required if you work on a library or package but you might need it for an application.

PRE-COMMIT

```yaml
repos:
  - repo: local
    hooks:
      - id: isort
        name: isort
        entry: isort
        types: [python]
        language: python
      - id: flynt
        name: flynt
        entry: flynt
        args: [--fail-on-change]
        types: [python]
        language: python
      - id: black
        name: black
        entry: black
        language: python
        types: [python]
        exclude: node_modules
      - id: flake8
        name: flake8
        entry: flake8
        language: python
        types: [python]
        exclude: .serverless,node_modules
      - id: bandit
        name: bandit
        entry: bandit
        language: python
        types: [python]
        args:  [--ini, .bandit, --recursive]
      - id: mypy
        name: mypy
        entry: mypy
        types: [python]
        language: python
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v4.0.1
    hooks:
      - id: end-of-file-fixer
        exclude: .*(min.js|min.css|html|svg|css.map|js.map)
      - id: trailing-whitespace
        exclude: .*(md)
```

Speaker notes

Notice that everything is setup as "local" because I keep track of the packages (black, flake8, etc.) in my dev requirements.

Benefits are:

- Tools are installed for IDEs instead of separate virtual environment just for pre-commit.
- You can have dependabot keep your tools up to date.
- Keeps things in sync between dev environment and pre-commit.

DEV REQUIREMENTS

```
# dev-requirements.in

bandit==1.7.0
black==21.4b2
flake8-docstrings==1.6.0
flake8==3.9.1
flynt==0.64
isort==5.8.0
mypy==0.901
pre-commit==2.12.1
```

Speaker notes

We use pip-tools to compile our requirements.txt files with pinned libraries for dependencies.

USAGE

SETUP YOUR IDE/EDITOR

Speaker notes

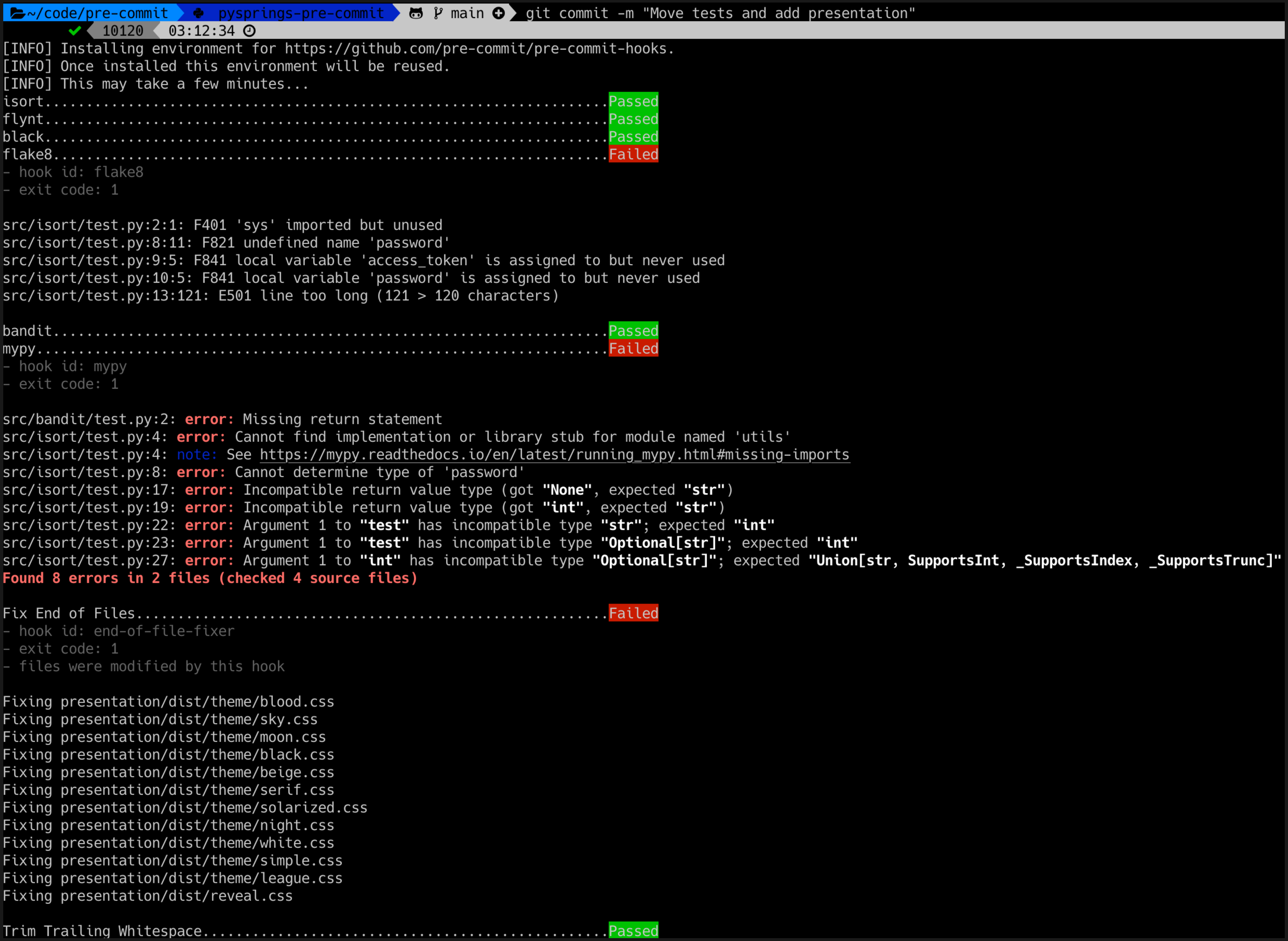
This is the first place where having those tools running in the background helps tremendously. Take the time to setup your IDE or editor, this is your most important environment.

INSTALL PRE-COMMIT HOOK

```
# Optional if installed via dev-requirements.txt
pip install pre-commit

pre-commit install
git add .
git commit -m "Some bad code"
```

Speaker notes

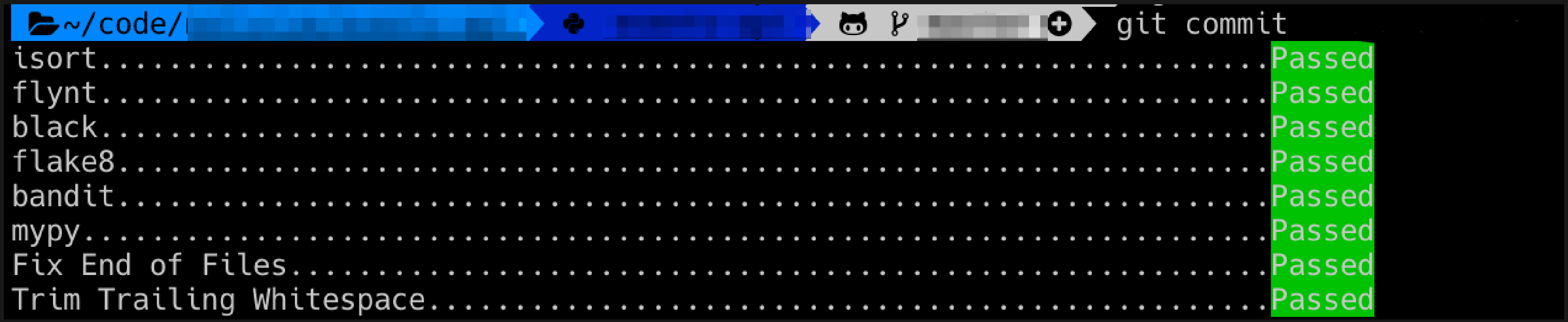
- pre-commit runs only for the staged files, not the entire project. And only relevant linters/formatters are applied.
- Also, because pre-commit will run everytime you commit, make sure you keep it fast otherwise nobody will want to use it.



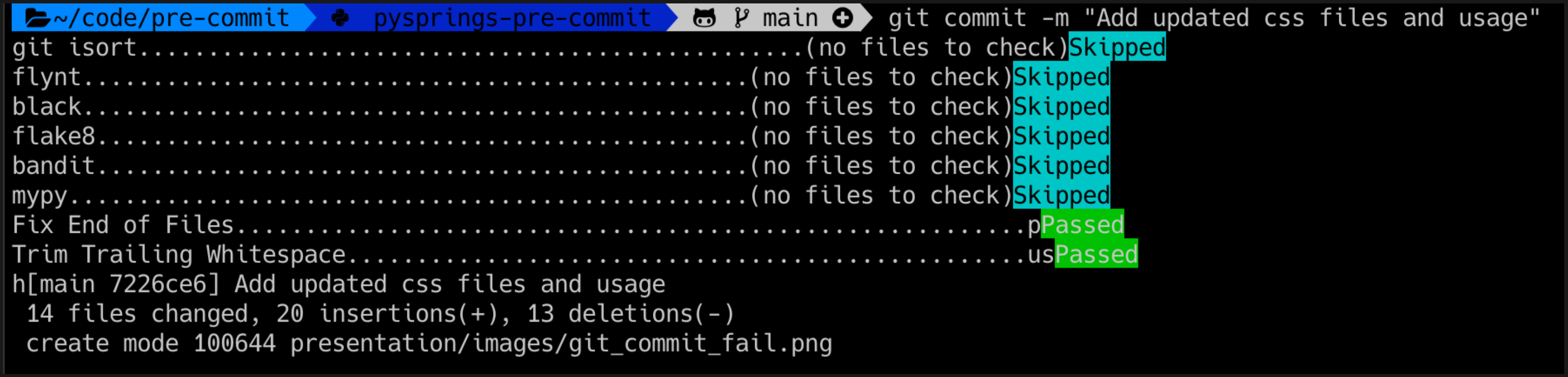
```
~/code/pre-commit   pysprings-pre-commit    main   git commit -m "Move tests and add presentation"
    ✔  10120  03:12:34
[INFO] Installing environment for https://github.com/pre-commit/pre-commit-hooks.
[INFO] Once installed this environment will be reused.
[INFO] This may take a few minutes...
isort...................................................................Passed
flynt...................................................................Passed
black...................................................................Passed
flake8..................................................................Failed
- hook id: flake8
- exit code: 1

src/isort/test.py:2:1: F401 'sys' imported but unused
src/isort/test.py:8:11: F821 undefined name 'password'
src/isort/test.py:9:5: F841 local variable 'access_token' is assigned to but never used
src/isort/test.py:10:5: F841 local variable 'password' is assigned to but never used
src/isort/test.py:13:121: E501 line too long (121 > 120 characters)

bandit..................................................................Passed
mypy....................................................................Failed
- hook id: mypy
- exit code: 1

src/bandit/test.py:2: error: Missing return statement
src/isort/test.py:4: error: Cannot find implementation or library stub for module named 'utils'
src/isort/test.py:4: note: See https://mypy.readthedocs.io/en/latest/running_mypy.html#missing-imports
src/isort/test.py:8: error: Cannot determine type of 'password'
src/isort/test.py:17: error: Incompatible return value type (got "None", expected "str")
src/isort/test.py:19: error: Incompatible return value type (got "int", expected "str")
src/isort/test.py:22: error: Argument 1 to "test" has incompatible type "str"; expected "int"
src/isort/test.py:23: error: Argument 1 to "test" has incompatible type "Optional[str]"; expected "int"
src/isort/test.py:27: error: Argument 1 to "int" has incompatible type "Optional[str]"; expected "Union[str, SupportsInt, _SupportsIndex, _SupportsTrunc]"
Found 8 errors in 2 files (checked 4 source files)

Fix End of Files........................................................Failed
- hook id: end-of-file-fixer
- exit code: 1
- files were modified by this hook

Fixing presentation/dist/theme/blood.css
Fixing presentation/dist/theme/sky.css
Fixing presentation/dist/theme/moon.css
Fixing presentation/dist/theme/black.css
Fixing presentation/dist/theme/beige.css
Fixing presentation/dist/theme/serif.css
Fixing presentation/dist/theme/solarized.css
Fixing presentation/dist/theme/night.css
Fixing presentation/dist/theme/white.css
Fixing presentation/dist/theme/simple.css
Fixing presentation/dist/theme/league.css
Fixing presentation/dist/reveal.css

Trim Trailing Whitespace................................................Passed
```

Speaker notes

- Linters only show errors.
- Formatters automatically fix the code so you might have unstaged changes.



```
~/code/[redacted]  [redacted]  [redacted]  git commit
isort...................................................................Passed
flynt...................................................................Passed
black...................................................................Passed
flake8..................................................................Passed
bandit..................................................................Passed
mypy....................................................................Passed
Fix End of Files........................................................Passed
Trim Trailing Whitespace................................................Passed
```



```
~/code/pre-commit  pysprings-pre-commit  main  git commit -m "Add updated css files and usage"
git isort.............................................(no files to check)Skipped
flynt.............................................(no files to check)Skipped
black.............................................(no files to check)Skipped
flake8............................................(no files to check)Skipped
bandit............................................(no files to check)Skipped
mypy..............................................(no files to check)Skipped
Fix End of Files.........................................................pPassed
Trim Trailing Whitespace.................................................usPassed
h[main 7226ce6] Add updated css files and usage
 14 files changed, 20 insertions(+), 13 deletions(-)
 create mode 100644 presentation/images/git_commit_fail.png
```

CONTINUOUS INTEGRATION

```
pre-commit run --all-files
```

Speaker notes

Devs can skip hooks with `git commit --no-verify`.

RECAP

1. Discuss it with your team and plan your rollout.
2. Run the formatters and linters.
3. Add pre-commit to your dev requirements and CI pipeline.
4. Do a brown bag with your team to get their Editor/IDE setup as well as pre-commit

Speaker notes

Based on the size of your codebase you have different rollout strategies, but I would recommend doing one formatter at a time, then add the linters and fix remaining issues.

You can ignore the code formatting commit for git blame by adding the rev to a .git-blame-ignore-revs file and do: `git config blame.ignoreRevsFile .git-blame-ignore-revs`.

ADDITIONAL RESOURCES

Ignore bulk change with git blame

Flake 8 plugins list

Pre-commit ci

Test and code podcast (episodes 156 and 157)

Python bytes podcast (episode 237)

Speaker notes

Anthony Sottile Brian Okken Michael Kennedy

Q&A

linkedin.com/in/arthurio

notivize.com

github.com/arthurio/pre-commit