Giving and receiving feedback, Modular code development and Python package structure

Sarah Alidoost, RSE

netherlands eScience center

netherlands eScience center

Giving and receiving feedback

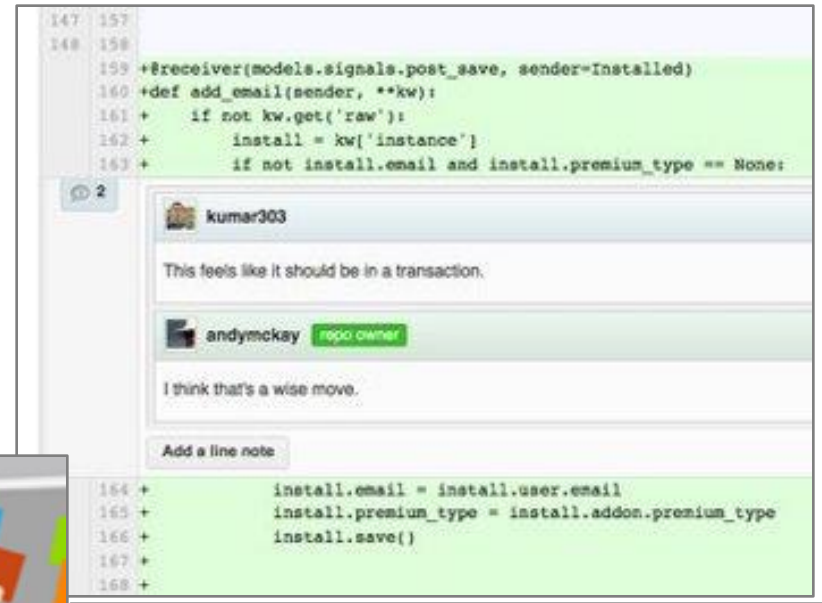
- Why feedback
- Giving feedback
  - I-I-You
  - Open questions vs closed comments
  - Directive vs non-directive comments
- Receiving feedback
- Other tips!

netherlands eScience center

Why feedback?

by getting comments we can learn and improve:

- Stimulate positive behavior
- Correct behavior
- Clarify relationships between people



Giving feedback

1. Ask if the other wants feedback
2. Keep the feedback specific and ‘to the point’
3. Use I-I-You with open and non-directive messages
4. Give positive as well as constructive feedback; in other words, move from strengths to weaknesses
5. Choose a proper place to give the feedback

I-I-You

I: Describing which behavior I have observed

- if I understood …
- I notice …
- I see ...

I: Indicating which effect this behavior had on me

- What feeling did this cause?
- How did it affect me?
- What was my response?

You: Taking a step toward the other

- Do you recognize that?
- What do you think?
- You can fix this ...

Open questions vs closed comments

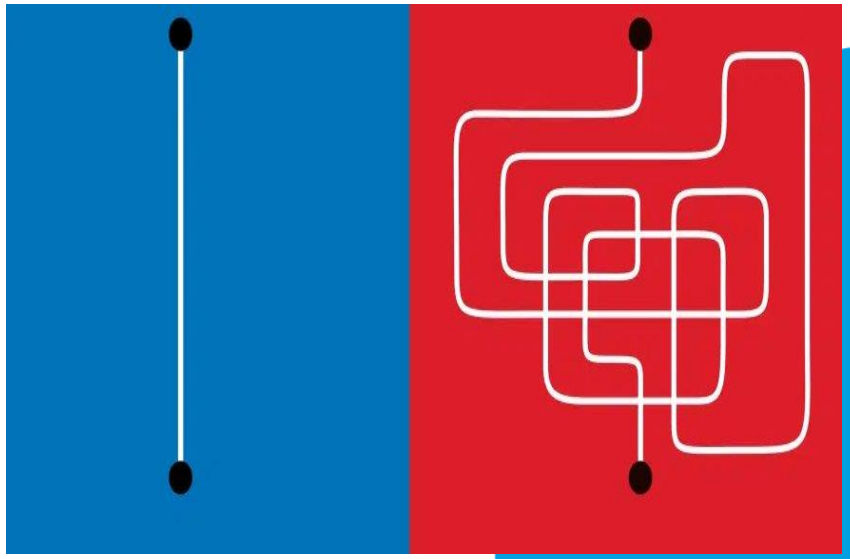
- Instead of saying: "This is unclear",
  - ask: "What do you mean by this?”
- Instead of saying “You don't have a hypothesis",
  - ask: "Can you show me your hypothesis?”
- Instead of saying "X should come before Y"
  - ask: “Why did you put Y before X?"

Receiving feedback

1. Try to understand the feedback
2. Let the feedback giver know you appreciate the feedback
3. Evaluate the feedback
   - Feedback is not a personal attack
   - Don’t let your emotions get the best of you
   - Be open to compliments
4. Act on the feedback

Other tips

- Make the message sound natural.
- Don't give only compliment, and don't use this template: "this is nice, but…".
- Notice cultural differences in expressing opinions

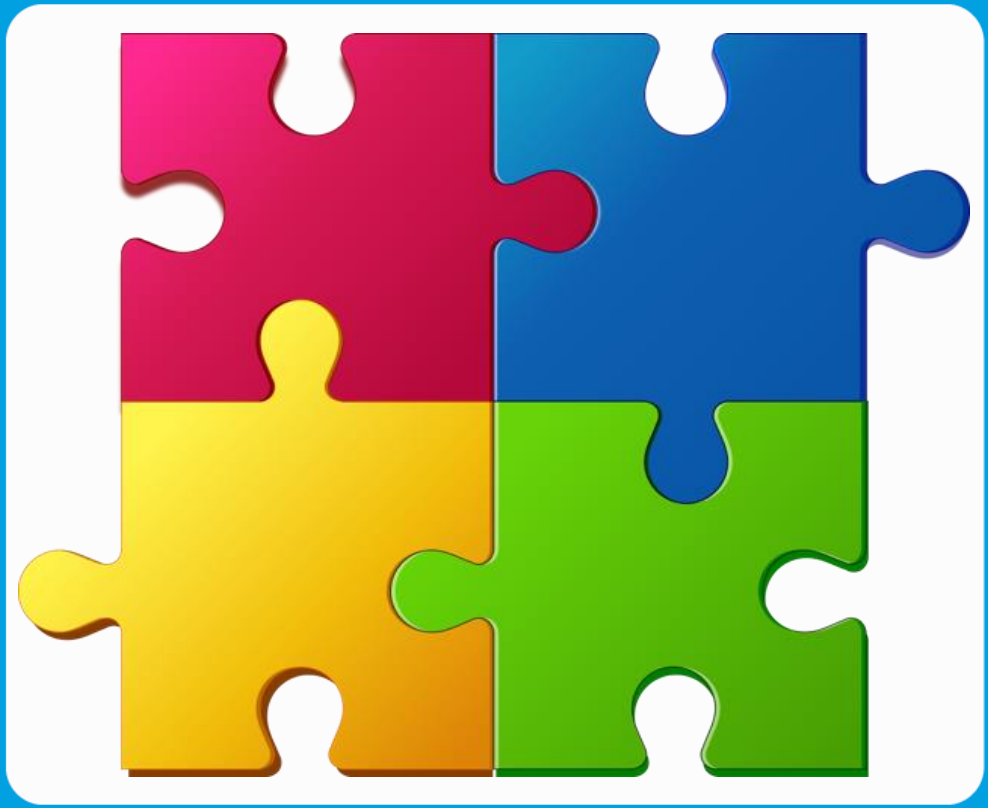


https://geerthofstede.com/culture-geert-hofstede-gert-jan-hofstede/6d-model-of-national-culture/

netherlands eScience center

Modular code development

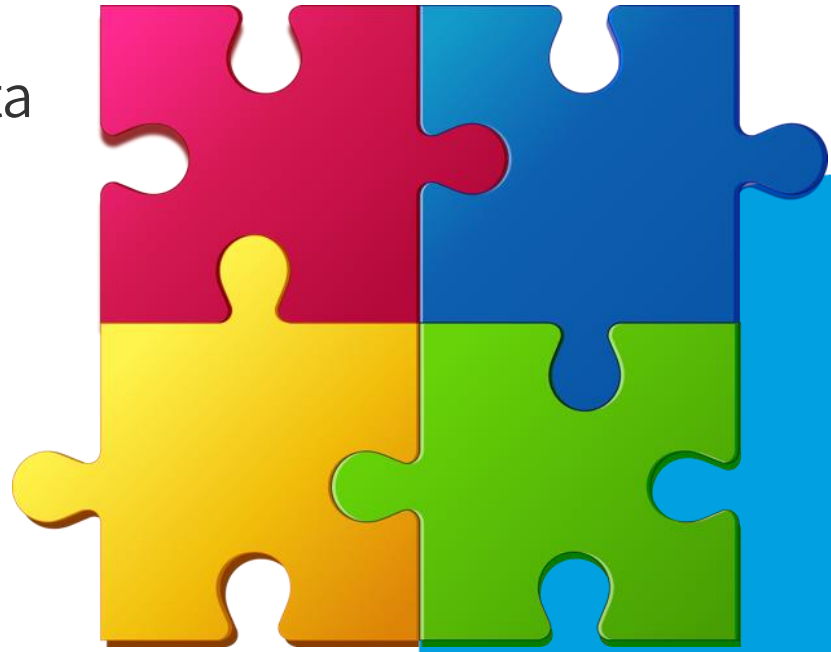
- Modularity and composition
- When to modularize



Modularity and composition

Build complex behavior from simple components:

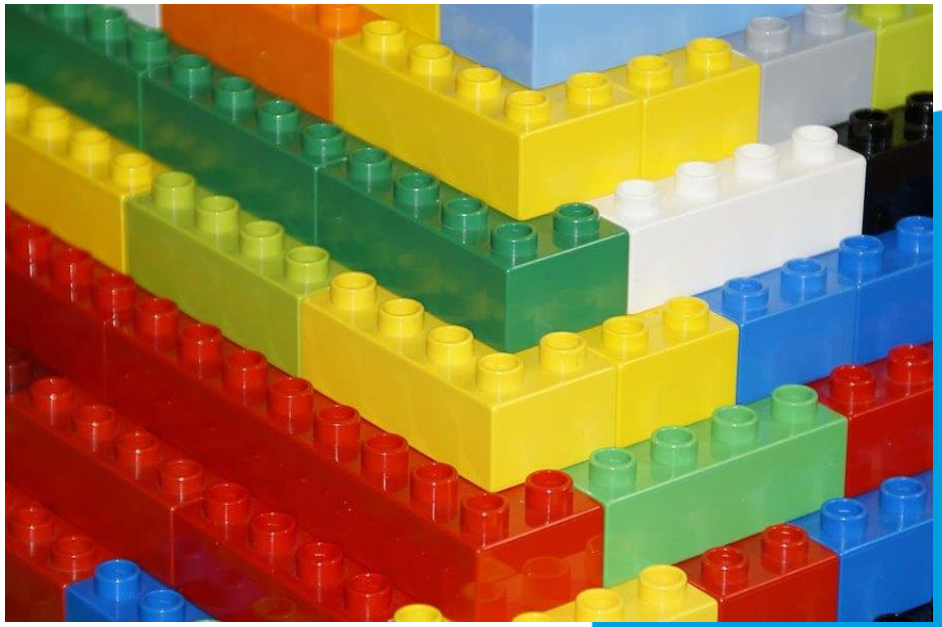
- Elements are self-contained and independent.
- Each element handles a specific (set of) task(s).
  - Example: load data, process data, plot data, save data



Modularity and composition

Break your code down to more units:

- functions
- classes
- modules
- libraries/packages
- programs



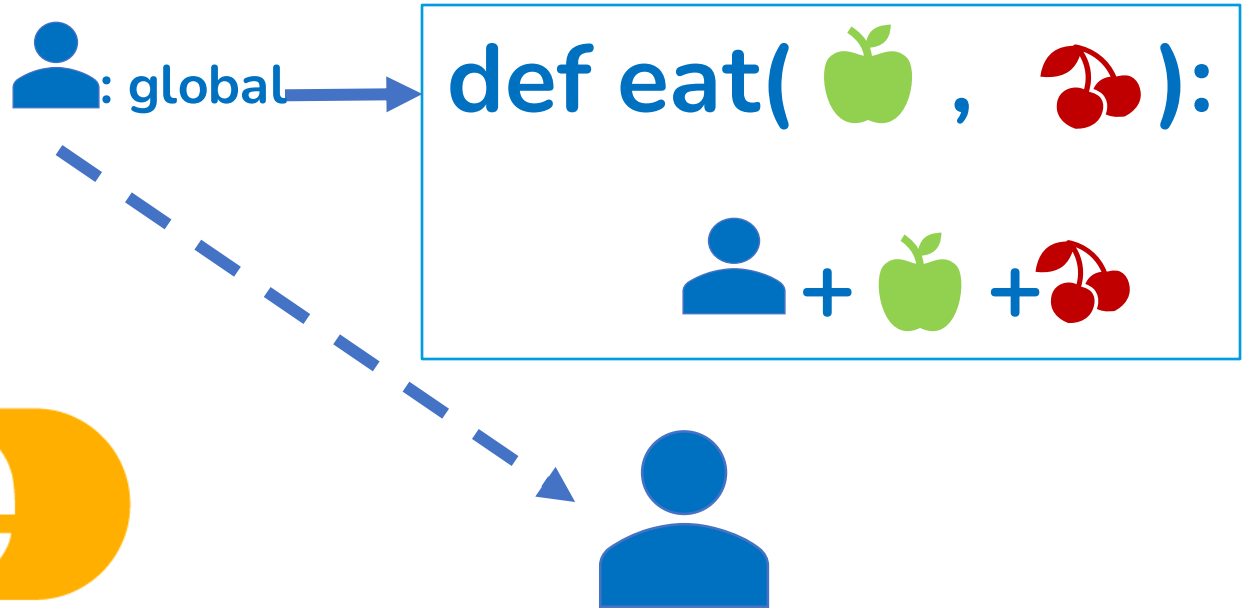
netherlands eScience center

Modularity and composition

Pure functions:

- They take input values and return values
- They are easy to understand, test, and reuse

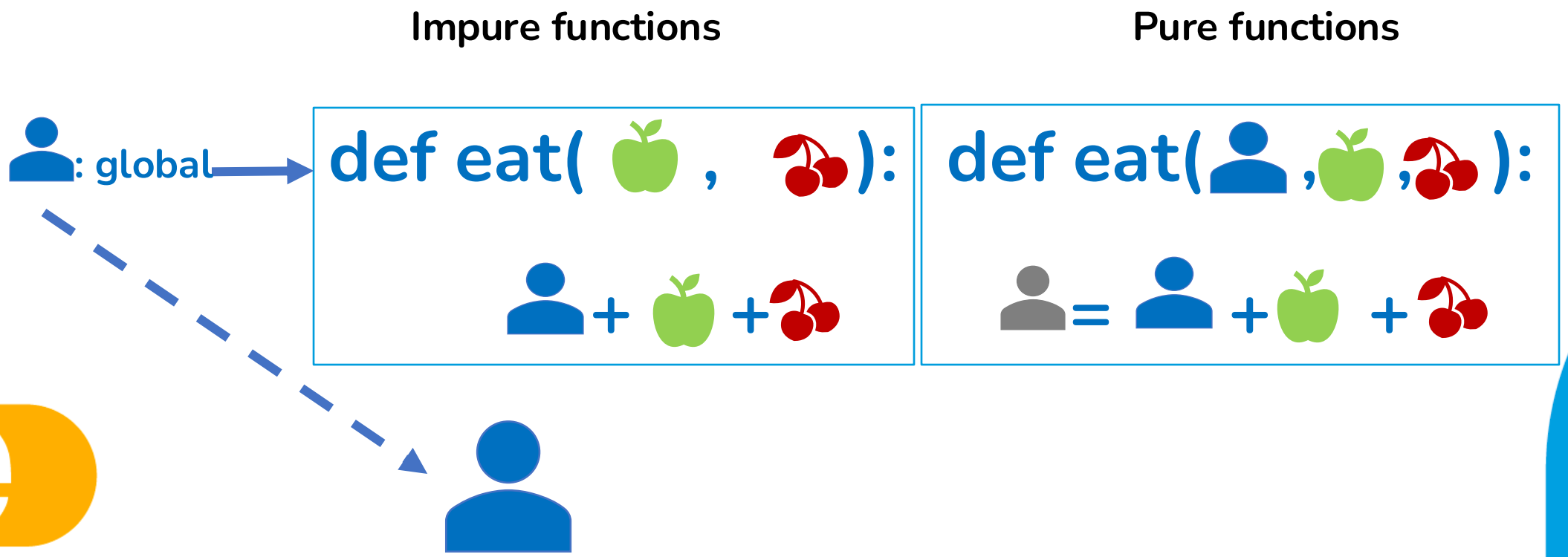
Impure functions



Modularity and composition

Pure functions:

- They take input values and return values
- They are easy to understand, test, and reuse



netherlands eScience center

When to modularize

Poor readability

python

```python
x = [1, 2, 3, 4, 5]
y = 0
for i in x:
    y += i**2
print(y)
```

python

```python
def sum_of_squares(numbers):
    total = 0
    for number in numbers:
        total += number ** 2
    return total
```

netherlands eScience center

When to modularize

Repetition

```python
print("Hello, Alice!")
print("Hello, Bob!")
print("Hello, Charlie!")
print("Hello, Diana!")
```

```python
def greet(name):
    print(f"Hello, {name}!")

names = ["Alice", "Bob", "Charlie", "Diana"]

for name in names:
    greet(name)
```

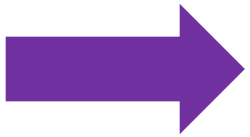
netherlands eScience center

When to modularize

Nested code

```python
numbers = [5, 12, 7, 20, 3]

for num in numbers:
    if num > 10:
        if num % 2 == 0:
            print(f"{num} is greater than 10 and even")
```



```python
def is_greater_than_ten(number):
    return number > 10

def is_even(number):
    return number % 2 == 0

def main():
    numbers = [5, 12, 7, 20, 3]

    for num in numbers:
        if is_greater_than_ten(num) and is_even(num):
            print(f"{num} is greater than 10 and even")

if __name__ == "__main__":
    main()
```

netherlands eScience center

Python package structure

- Package structure
- Project.toml and installation

netherlands eScience center

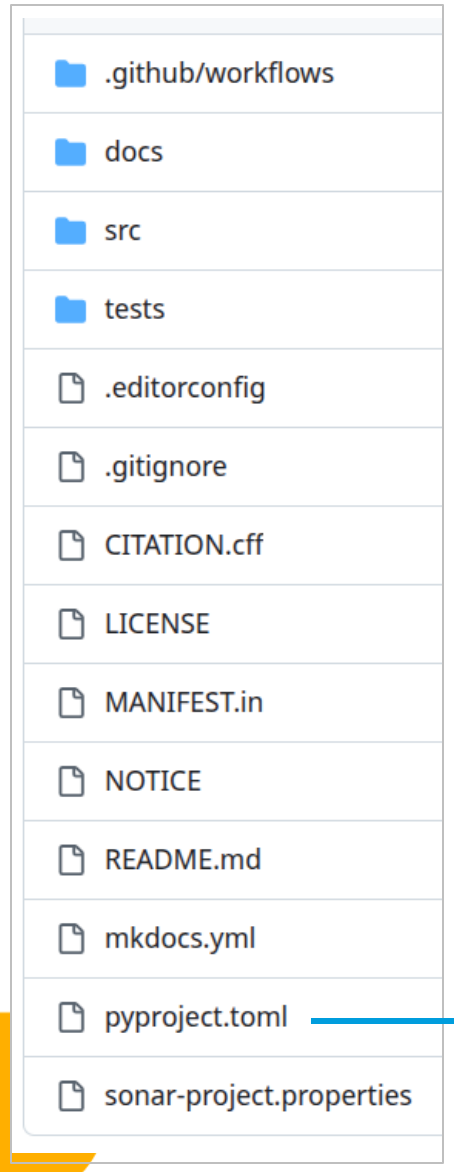
Python package structure


```
├── src/
│   └── awesome_package/
│       ├── __init__.py
│       └── module.py
```

https://packaging.python.org/en/latest/discussions/src-layout-vs-flat-layout/#src-layout-vs-flat-layout

netherlands eScience center

Python package structure




```toml
[build-system]
requires = ["hatchling"]
build-backend = "hatchling.build"

[project]
name = "spam-eggs"
version = "2020.0.0"
dependencies = [
  "httpx",
  "gidgethub[httpx]>4.0.0",
  "django>2.1; os_name != 'nt'",
  "django>2.0; os_name == 'nt'",
]
requires-python = ">=3.8"
authors = [
  {name = "Pradyun Gedam", email = "pradyun@example.com"},
  {name = "Tzu-Ping Chung", email = "tzu-ping@example.com"},
  {name = "Another person"},
  {email = "different.person@example.com"},
]
maintainers = [
  {name = "Brett Cannon", email = "brett@example.com"}
]
description = "Lovely Spam! Wonderful Spam!"
readme = "README.rst"
license = "MIT"
license-files = ["LICEN[CS]E.*"]
keywords = ["egg", "bacon", "sausage", "tomatoes", "Lobster Thermidor"]
classifiers = [
  "Development Status :: 4 - Beta",
  "Programming Language :: Python"
]

[project.optional-dependencies]
gui = ["PyQt5"]
cli = [
  "rich",
  "click",
]
```

Recommendation

```
python -m pip install .

python -m pip install --editable .

python -m build
```

https://packaging.python.org/en/latest/guides/writing-pyproject-toml/#a-full-example

Python package structure


```
└── tests
    └── test_module.py
```

https://carpentries-incubator.github.io/good-practices-lesson/4-testing.html

netherlands eScience center

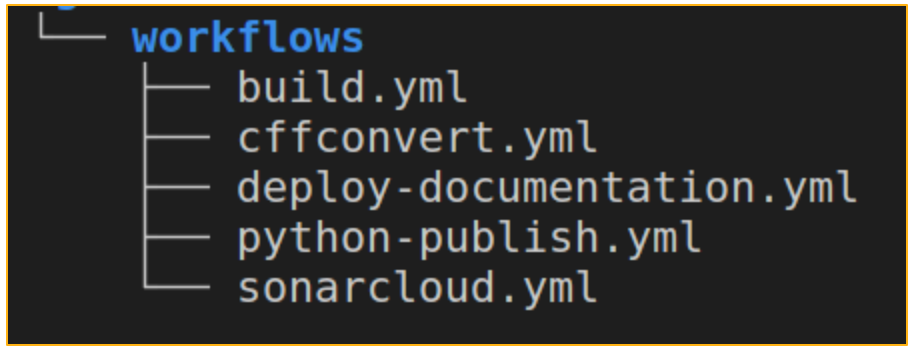
Python package structure


```
docs
    guides.md
    my_module.md
```

https://carpentries-incubator.github.io/good-practices-lesson/3-documentation.html

netherlands eScience center

Python package structure



https://docs.github.com/en/actions/writing-workflows/quickstart

netherlands eScience center

Check out the references at :

https://github.com/SarahAlidoost/mentorship-material/blob/main/Reference.md