Lesson 8

Python Essentials

Overview

- 1. Clean code
- 2. Linting

This is valid, but it is clean?

```
x = 12
if x== 24:
  print('Is valid')
else:
    print("Not valid")

def helper(name='sample'):
    pass

def convertTouc( name = 'sample'):
        return name.upper()
```

Why clean code?

- Makes code readable
- Easier to debug
- $\bullet\,$ Easier to maintain by you and others

Spacing

- Spaces, not tabs
- VSCode automatically converts tabs to spaces

• Avoid extraneous whitespaces

```
{'good': 200}
{ 'bad': 100}
```

Naming

- Spaces, not tabs
- VSCode automatically converts tabs to spaces
- Avoid extraneous whitespaces

```
Bad: - d - DaysSinceCreation - daysSinceCreation - c = dta_rcrd_102()
Good: - days_since_creation - customer_address = Address()
Notes: https://www.python.org/dev/peps/pep-0008/
```

Functions

- Think small
- Should do one thing
 - They should do it well.
 - They should do it only.
- Descriptive names. Don't fear long names
- Reduce number of arguments.
 - wrap arguments inside meaningful objects

```
makeCircle(double x, double y, double radius)
makeCircle(Point center, double radius)
```

Notes:

Reduce number of arguments 0: ideal 1: ok 2: ok 3: justify 4 or more: avoid

Comments

- "Don't comment bad code—rewrite it." —Brian W. Kernighan and P. J. Plaugher
- Can be helpful or damaging
- Inaccurate comments are worse than no comments at all
- Used to compensate failure expressing with code

- They can lie
- Must have them, but minimize them
- Express yourself in code

Bad:

```
# Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) && (employee.age > 65))
Good:
if (employee.isEligibleForFullBenefits())
```

Documentation

- String literal in the first statement in a module, function, class, or method definition.
- Used for documentation.

```
def print_hello(user_name: str) -> str:
    """
    Generates a greeting to the user by name

Parameters:
    user_name (str): The name of the user
Returns:
    str: The greeting
"""

return 'Hello, ' + name
```

Classes

- Should be small
- Single responsibility principle
- Automatically run by Visual Studio Code

Linting

- Identify formatting issues
- Pylint for Python

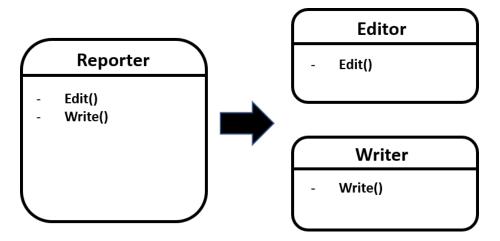


Figure 1: image

Windows:

```
pip install pylint macOS or Linux: pip3 install pylint
```

Type hints

Error!

- Tell the editor and linter what data types to expect
- $\bullet\,$ DOES NOT cause "compiler" errors

```
def get_user_greeting(user_name):
    return 'Hello, ' + user_name

greeting = get_user_greeting(42)

print(greeting)
```

TypeError: must be str, not int

```
def get_user_greeting(user_name: str) -> str:
    return 'Hello, ' + user_name

greeting = get_user_greeting(42)

print(greeting)

Notes: - parameter type - return type
```