Code & doc style

Ali Farnudi, Nov 2023

Why document code?

Pros

- **Accelerates team member communication**
- Short on-board time
- Organise big projects
- **High development speed**

Cons

- Time (and money) consuming
- Quickly gets out of date
- Developers don’t like it

Comments

- Explain intentions, not what code does
- Deviations from standard
- Unexpected choices of implementation

```
def func1(x):
    # assign x squared to a
    a = x**2
    # double a
    a *= 2
    # return root of a
    return math.sqrt(a)
```

```
def calc_hypotenuse_of_isosceles_right_triangle(edge):
    return math.sqrt(2*edge**2)
```

```
def calc_hypotenuse_of_isosceles_right_triangle(edge):
    """return hypotenuse using the Pythagorean theorem"""
    return math.sqrt(2*edge**2)
```

Docstrings and PEP 257

one-line

```
def add(x,y):
    """Return sum of two objects."""
    return x+y
```

```
def function(a, b):
    """Do X and return a list."""
```

- String literal: The first statement in a module, function, etc
- All modules, functions, and classes should normally have docstrings
- “””triple double quotes”””

- It should be a command ("Do this", "Return that"),
- Don’t describe: "Returns the pathname ..."
- Nature of return value should be mentioned

Docstrings and PEP 257

Multi-line

- A summary line (like a one-line) + a blank line,
- More description
- Everything is indented the same as the quotes
- Numpy Style
- Pandas docstrings
- Google Style

```
def complex_number(real=0.0, imag=0.0):
    """Form a complex number.

    Keyword arguments:
    real -- the real part (default 0.0)
    imag -- the imaginary part (default 0.0)
    """
    if imag == 0.0 and real == 0.0:
        return complex_zero
    ...
```

PEP 257 website

Hands on
PEP 8 - pycodestyle and black
PEP 257 - pydocstyle