

# Code Quality

# Code Quality

---

Readability

Coding Standards

Tools (pep8, pylint)

Cyclomatic Complexity (mccabe)

9 April 1941

Thursday, Jan 14  
Still have a cold. Went to work at 9:30 am. Left at 11:30 am.  
Went out at 11:30 am. Home at 12:30 pm. Had lunch at 12:30 pm.  
Gave in ph. & got a cold. And was today.

Friday, Jan. 15

Mistral N.E., Sat not alone at 7.50 per hour.  
14-0 m m m m m m m m m m m m m m m m m m  
4.80 per hour when on camp; in irregular, cold  
drain, much less than last year, 19.6 at all.

Feb 21, 1916

and many of the day, and of course, the women  
were very poor, and were not in the city.  
and the old and the new.

中法實業公司 1944. 12. 1

We administer morphine by the oral route and stomachs.

Dr. Richard L. Jones, 18,

Very much interested in this. What will you report with



Hi Reddit,  
I came across this subreddit today and since I love to hand-write, I thought I'd share my own penmanship! This is my normal, everyday handwriting I've written in this style for several years now. I like that it's quick to write, neat, and easy to read.

Sometimes I like to write in cursive. It's definitely not as neat and more difficult to write. But still, it's fun and pretty! Now my hand is starting to hurt, so I'll stop here. 😊

Sincerely,  
ellechon



CAMBRIDGE SPARK

What about code?

# How to write good code?

---

1. Follow common conventions
2. Break down into modular parts
3. Test it (next section)

# PEP8: Python Style Guide (bad)

---

```
i= 2
```

```
data=(42,55, 10, 13)
```

```
result = 20+ data[i]
```



# PEP8: Python Style Guide (bad)

```
i= 2
```

```
data=(42,55, 10, 13)
```

```
result = 20+ data[i]
```

code-style-demo/bad\_pep8.py

# PEP8: Python Style Guide (good)

---

```
i = 2
```

```
data = (42, 55, 10, 13)
```

```
result = 20 + data[i]
```

## Pylint: Python code smells (bad)

```
def find_reconciled(transactions, receipts):  
    r = []  
  
    if len(transactions) > 0:  
        for t in transactions:  
            r.append(t)  
  
    return r
```

code-style-demo/bad\_pylint.py

# Pylint: Python code smells (bad)

```
def find_reconciled(transactions, receipts):  
    r = []  
    if len(transactions) > 0:  
        for t in transactions:  
            r.append(t)  
    return r
```

# Pylint: Python code smells (good)

```
def find_reconciled(transactions, receipts):  
  
    """  
  
    :param transactions: a list of transactions objects  
  
    :param receipts: a dictionary of unique transaction id to receipt objects  
  
    :return: a list of transactions that have an associated receipt  
    """  
  
    result = []  
  
    for transaction in transactions:  
  
        if transaction.id in receipts:  
  
            result.append(transaction)  
  
    return result
```

# McCabe: Cyclomatic Complexity (bad)

```
def process(data):  
    output = ""  
    for i in range(1, 10):  
        for j in range(1, 10):  
            if i != j:  
                for a in range(1, 10):  
                    for b in range(1, 10):  
                        if a == i and j == b:  
                            output += data[i]  
    return output
```

# McCabe: Cyclomatic Complexity

---

- Outputs a software metric that gives an indication of the code “difficulty” (the smaller the better)
- Roughly based on number of conditionals and nested blocks in one function
- Suggests that a particular function should be further modularised to make it easier to reason about
- **Not to be confused *algorithmic complexity*!**

# McCabe: Cyclomatic Complexity (good)

```
def process(data):
```

```
    output = ""
```

```
    for i in range(1, 10):
```

```
        for j in range(1, 10):
```

```
            if i == j:
```

```
                output += _sub_process(data, i, j)
```

```
    return output
```

```
def _sub_process(data, i, j):
```

```
    for a in range(1, 10):
```

```
        for b in range(1, 10):
```

```
            if a == i and j == b:
```

```
                return data[i]
```





# Tools

---

```
pip install pycodestyle autopep8 pylint mccabe
```

**pycodestyle**: highlights pep8 conflicts

**autopep8**: automatically fixes pep8 problems

**pylint**: checks coding standards and potential errors

**mccabe**: reports cyclomatic complexity

## Exercise (10 min)

---

Simplify and refactor the code to clean up code warnings

```
$ pycodestyle --show-source --show-pep8 sad_code.py
```

```
$ pylint sad_code.py
```

```
$ python -m mccabe sad_code.py
```

# Takeaways

---

- Readable code helps maintainability and reduce scope for bugs
- PEP8 checks for code conventions
- Pylint checks for code smells and potential bugs
- Cyclomatic Complexity (mccabe) is a software metric that tells you whether you should further modularise your code to ease comprehension
- Code checking tools can be quite verbose, it doesn't mean your code is necessarily bad!