Code Quality

Code Quality

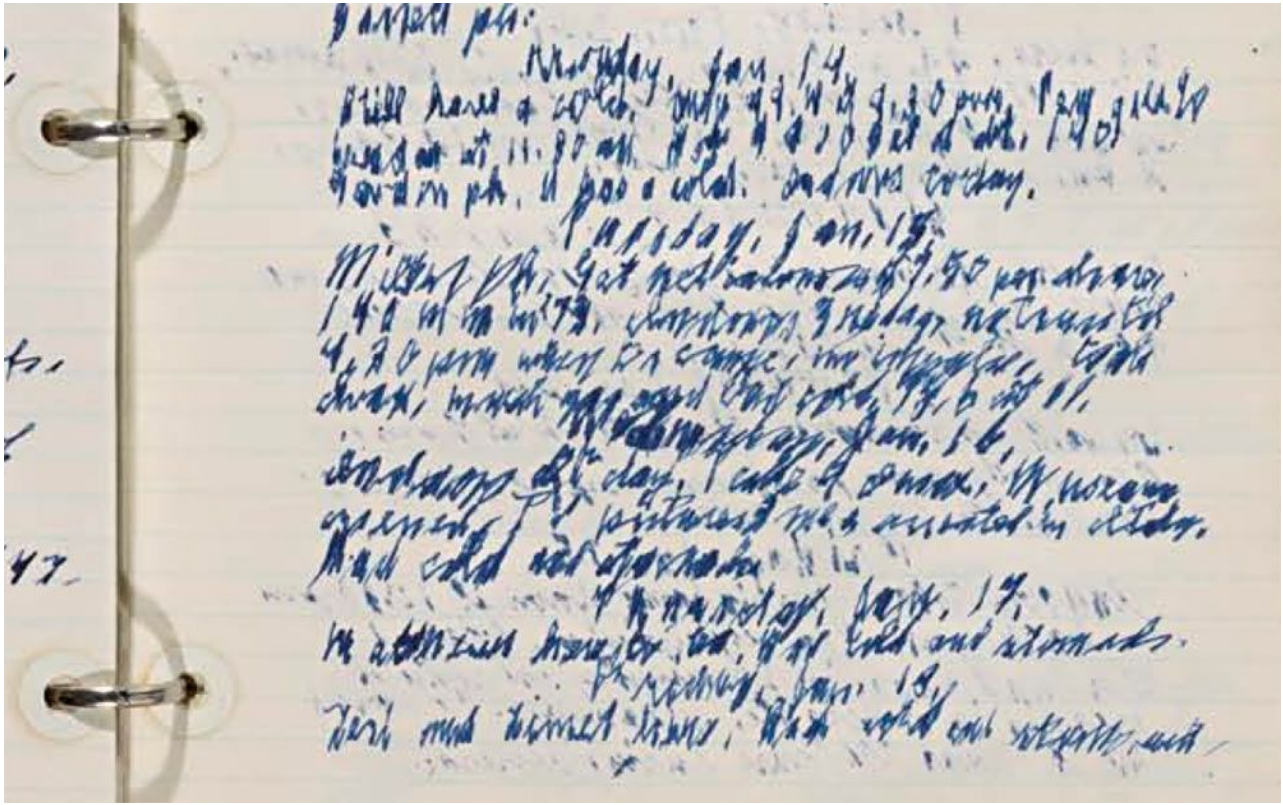
Readability

Coding Standards

Tools (pep8, pylint)

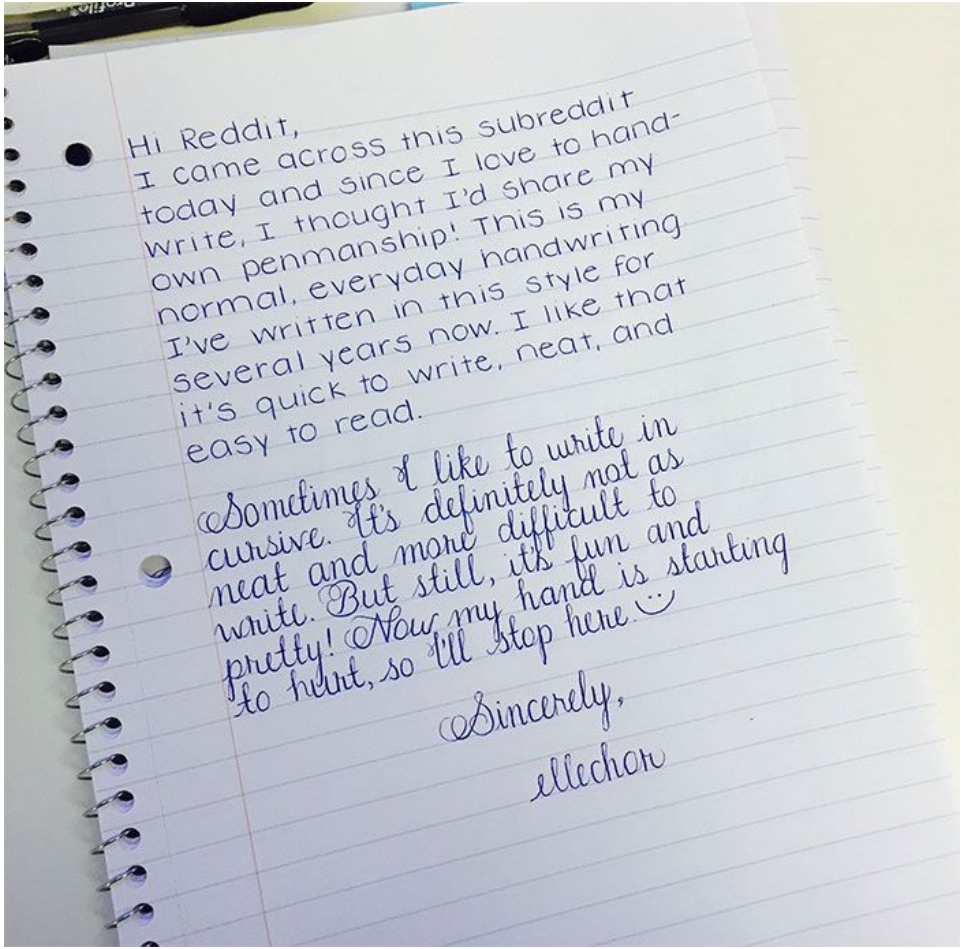
Cyclomatic Complexity (mccabe)

CAMBRIDGE SPARK



CAMBRIDGE SPARK

CAMBRIDGE SPARK



CAMBRIDGE SPARK

What about code?

CAMBRIDGE SPARK

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

CAMBRIDGE SPARK

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

CAMBRIDGE SPARK

Pylint: Python code smells (bad)

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

CAMBRIDGE SPARK

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
```

CAMBRIDGE SPARK

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
```

CAMBRIDGE SPARK

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*!**

CAMBRIDGE SPARK

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]
```

CAMBRIDGE SPARK

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

CAMBRIDGE SPARK

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!

CAMBRIDGE SPARK