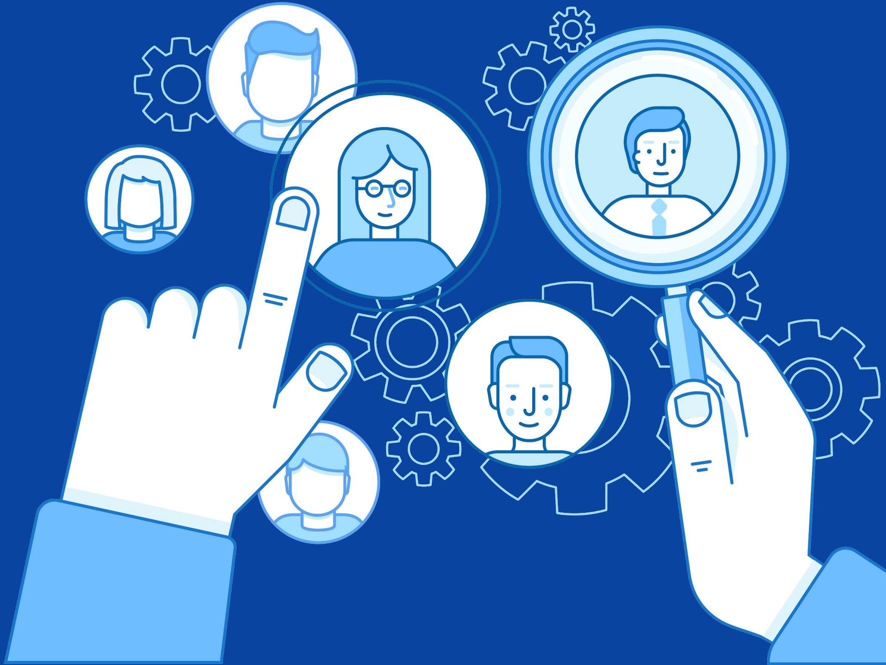
# Digital Career Institute

## Python Course - Coding Standards



Digital Career Institute

## Goal of the Submodule

The goal of this submodule is to help students learn why styling and coding standards are so important and how to follow and enforce them in Python. By the end of this submodule, the learners will be able to understand:

- Why styling is important.
- What is the common Python style guideline.
- How to ensure our code follows the standard.

Digital Career Institute

## Topics

- What is styling and how it differs from syntax.
- Motivation. Why should we conform to the standard styling?
- Examples of conformant and non-conformant code.
- What is the standard Python styling guide.
- How to use flake8 to enforce the standard Python styling guide into our code.

## What are Coding Standards?

# Coding Syntax, Style & Standards

Digital Career Institute

| Syntax | Style | Standards |
|---|---|---|
| **Specific** rules of a **particular language** | **Specific** rules for a **particular language** | **General** rules for a **particular codebase** |
| to produce | to produce | to produce |
| a **working** codebase | a **readable** codebase | a **readable** codebase |
| **PERFORMANCE** | **READABILITY** | |

# Coding Syntax, Style & Standards

Digital Career Institute

Syntax

If the proper syntax is not used, the code may **produce an error**.

Style & Standards

If the proper style is not used, the code may be **more difficult to read**, understand and maintain.

# Coding Syntax, Style & Standards

Syntax

We conform to the syntax so that the computer can understand our code.

The end user of our syntax is **the computer**.

Style & Standards

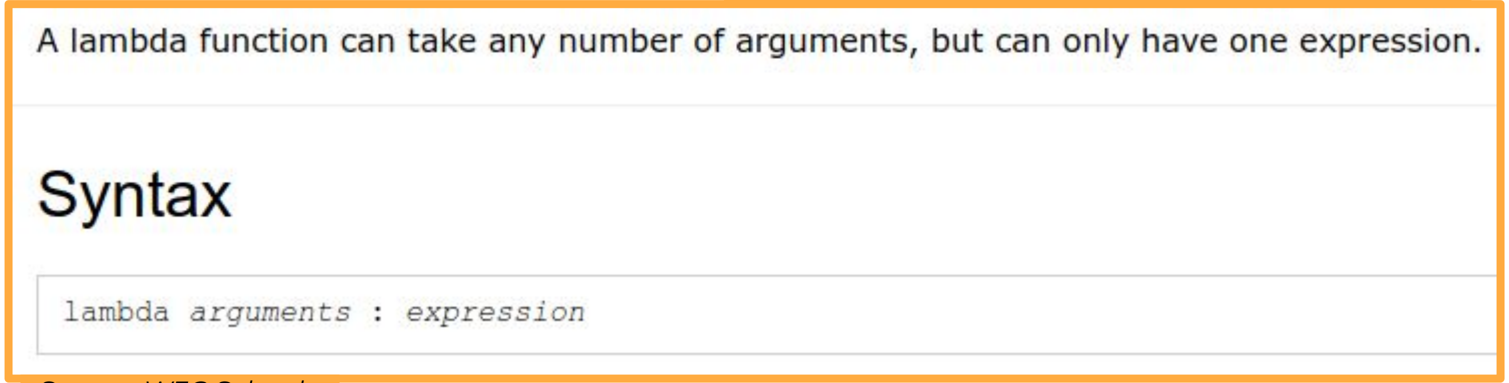
We conform to the style so that developers can understand our code.

The end users of our style are **the developers**.

## Syntax

Syntax is the way we define general programming concepts in a particular language.

**Python Lambda Syntax Example**


A lambda function can take any number of arguments, but can only have one expression.

Syntax

```
lambda arguments : expression
```

*Source: W3C Schools*

# Style

Digital Career Institute

Styling is how we write everything else that is not specified by the language syntax.

**Python Lambda Style Example**

```
○ lambda  x:  x  +  1        ○ lambda  x  :  x  +  1

○ lambda  x:  x+1            ○ lambda  x  :  x+1
```

Which one is the easiest to read?

# Style & Coding Standards

Digital Career Institute

A good codebase must:

**Work well**

*Syntax*

&

**Look good**

*Style*

## Why is Readability Important?

## Readability

Digital Career Institute

*“Code is read **much** more often than it is written.”*

Raymond Chen (Microsoft), Robert C. Martin (Agile) & Guido van Rossum (Founder of Python)

## Readability

$$\frac{\text{Read code}}{\text{Write code}} > 1$$

*The bigger the codebase is,*
*the higher the read/write ratio becomes.*

# The Coding Process

## What do we do when we write new code or refactor legacy code?

1. **We read the current code.**
2. **We process it.**
3. **We understand it.**
4. We decide how to do our changes.
5. We apply the changes.

**Readability** is the degree of difficulty with which we can get from step 1 to step 3.

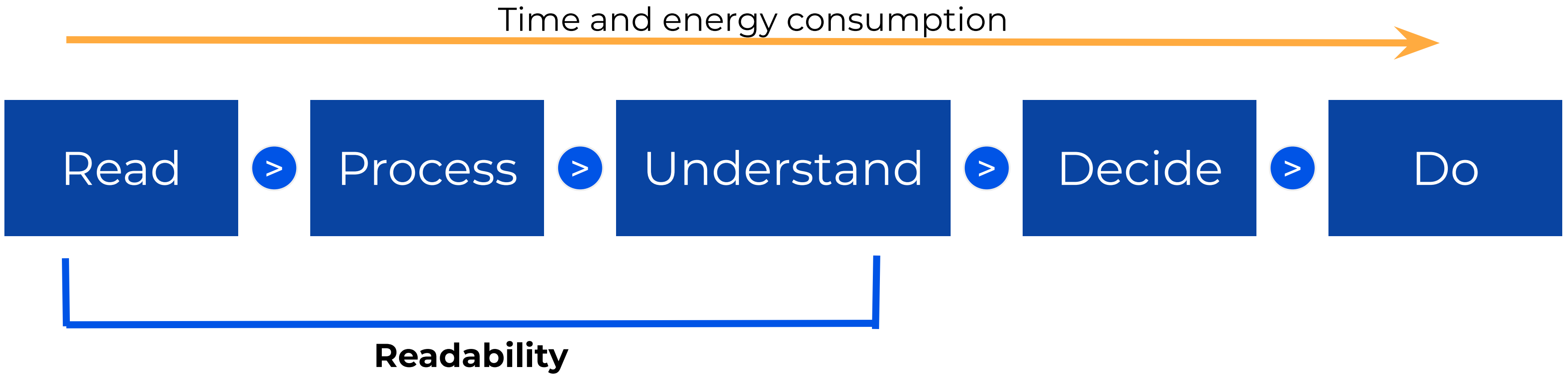
# The Coding Process

At every step, two things happen:

Time passes

Energy is consumed

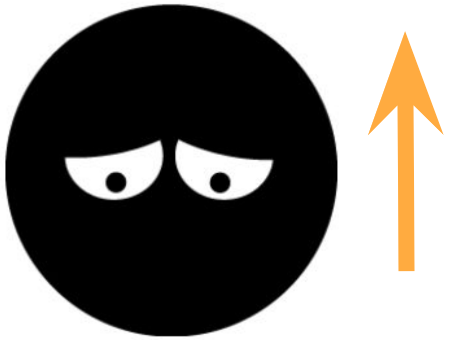
# The Coding Process



# The Coding Process

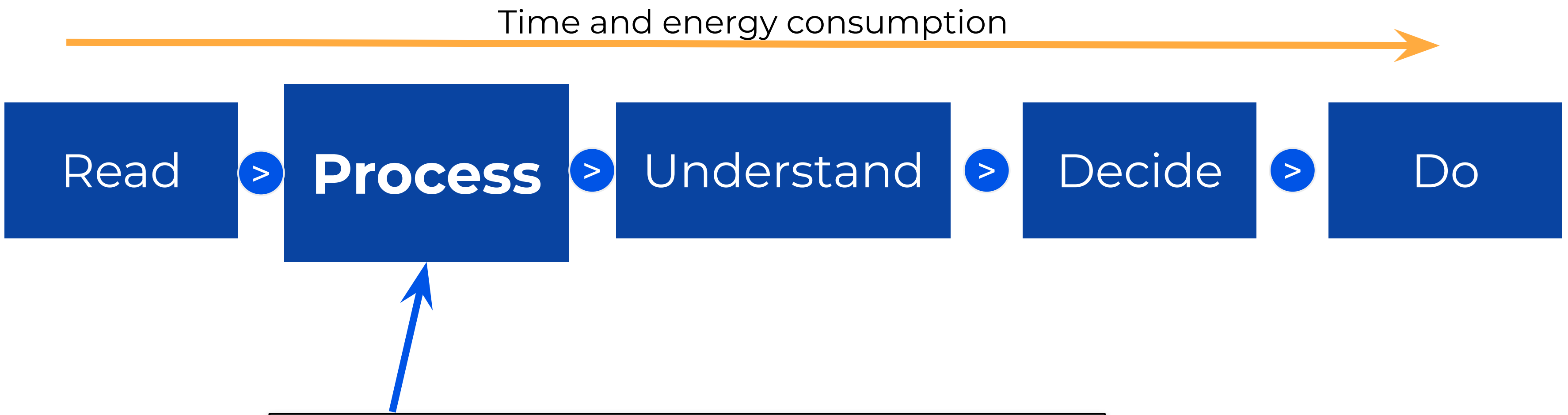
## This is equivalent to:

More money



More tired

# Readability



A **low readability** increases the time and effort spent on processing what is read, which will **increase the overall consumption**.

## Readability

A code that is difficult to read will be more **costly** to maintain and will reduce the developer's **productivity**. Both will reduce the business **scalability**.

The more time and energy is needed, the higher the chances of misunderstanding the code, taking wrong decisions and producing **bugs**.

## Readability

Digital Career Institute

**Poor Readability**

↓ Readability

=

↑ Time

↑ Energy

=

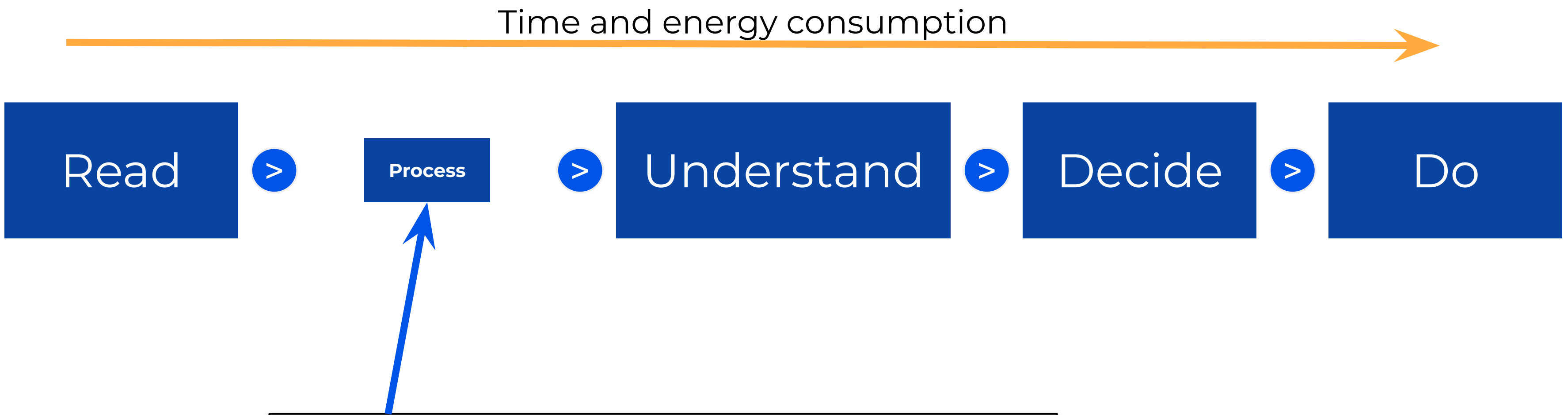
↑ Money

↑ Bugs

↓ Scalability

↓ Productivity

# Readability



A **high readability** reduces the time and effort spent on processing what is read, which will **reduce the overall consumption**.

# Readability

Digital Career Institute

**High Readability**

↑ Readability

=

↓ Time

↓ Energy

=

↓ Money

↓ Bugs

↑ Scalability

↑ Productivity

# Style Guides

Digital Career Institute

# What is a Style Guide?

A style guide is **a set of style rules** defined **for a particular codebase**.

Style rules example:

- Don’t write lines longer than N characters.
- Use an empty space to separate operators and operands.
- Do not import from different packages in one same instruction.
- Use lowercase and separate words with underscores in variable naming.
- ...

## Why a Style Guide?

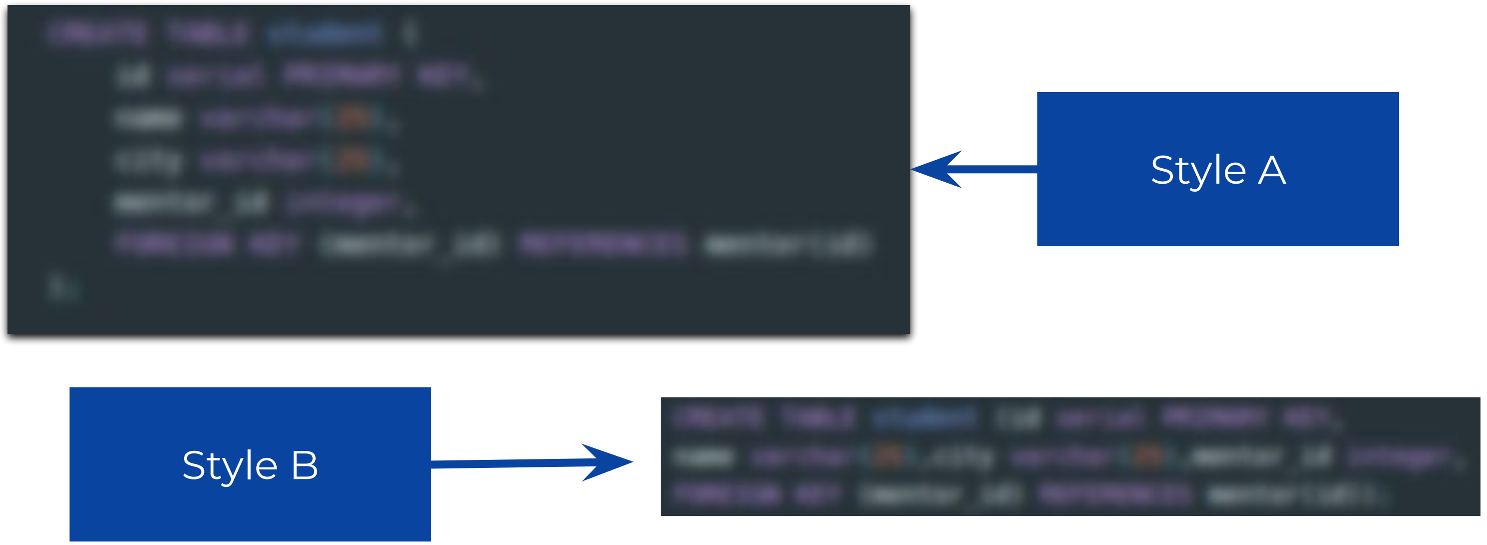
How does a <u>style guide</u> help increase readability?

How does a ***style guide*** help increase ***readability***?

# Why a Style Guide?

Digital Career Institute

The brain is very fast at identifying visual patterns.
A code style produces a visual pattern.



# Why a Style Guide?

It helps us identify the relevant parts of the code without reading.

Developer
Style Guide

Using a style consistently will reduce the time we need for reading, as we become accustomed to that particular pattern.

# Standard Style Guide

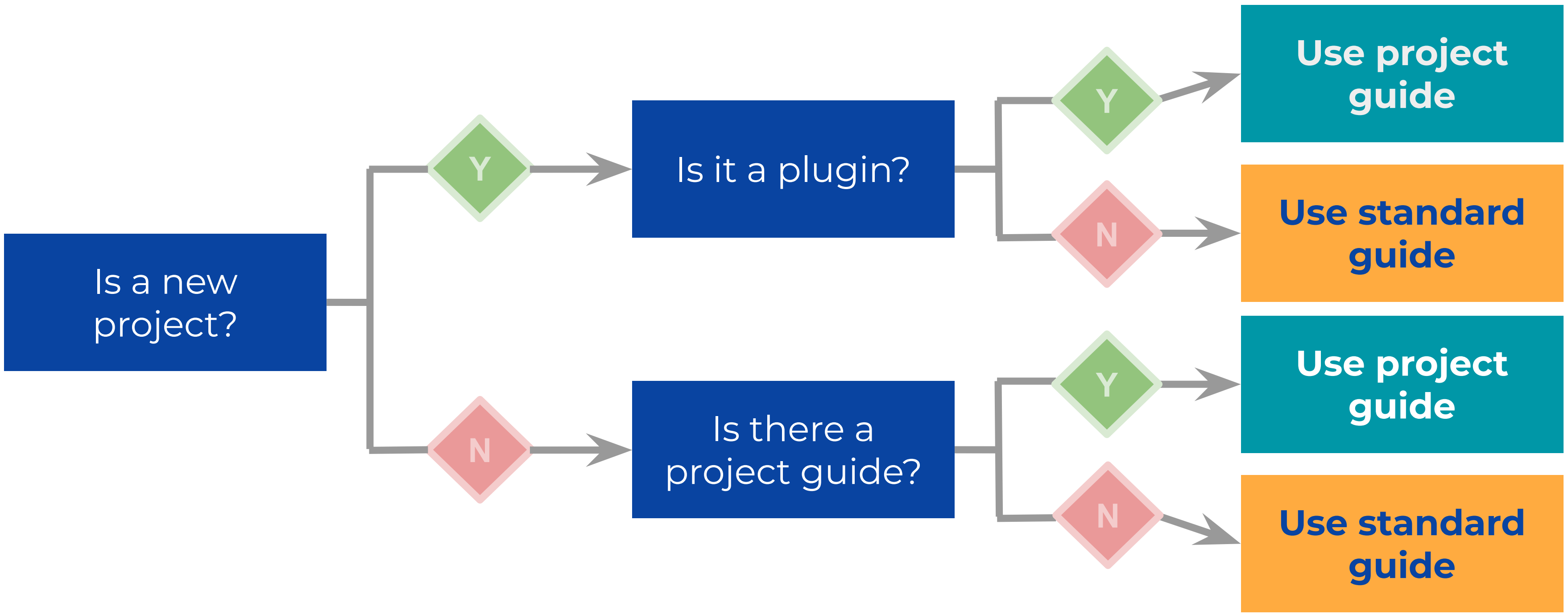
Digital Career Institute

**Why follow the standard?**

- To make it easy for other programmers to start typing code in the same project.
- To have a smoother transition if we work on multiple projects at the same time.
- To have more opportunities to work successfully with other people.
- Because there are tools available to automate and validate the standard styling.
- To spend more time (and money) typing code, and less time defining, automating and validating style rules.

# Standard vs. Custom Style Guide

**Rules of thumb**

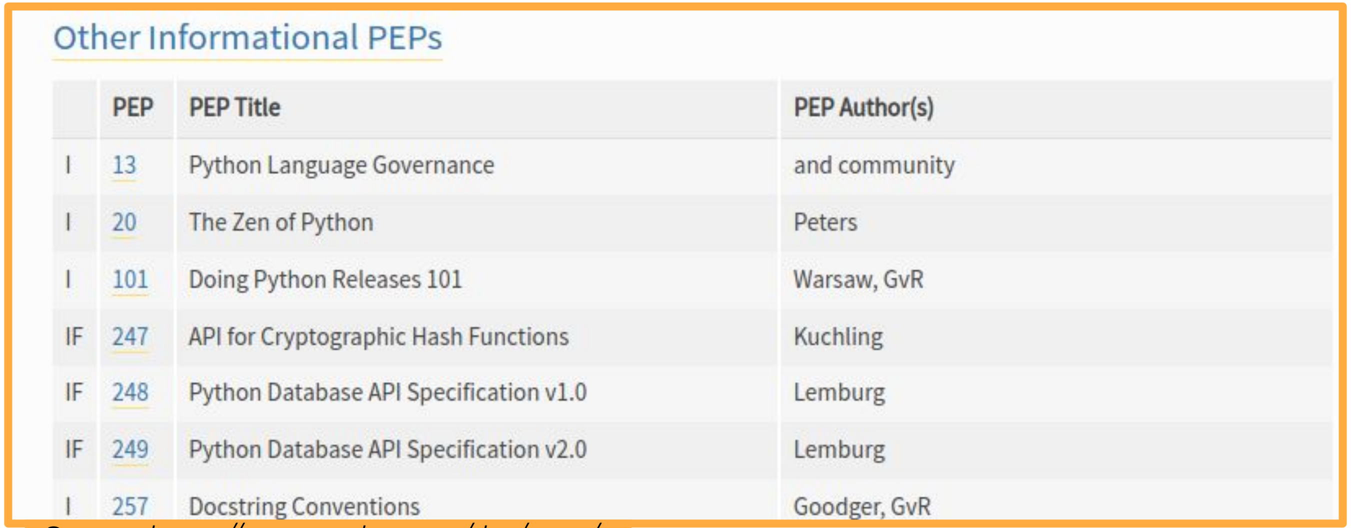


# Python Style Guide

Digital Career Institute

# Python Style Guide

## Python Enhancement Proposals



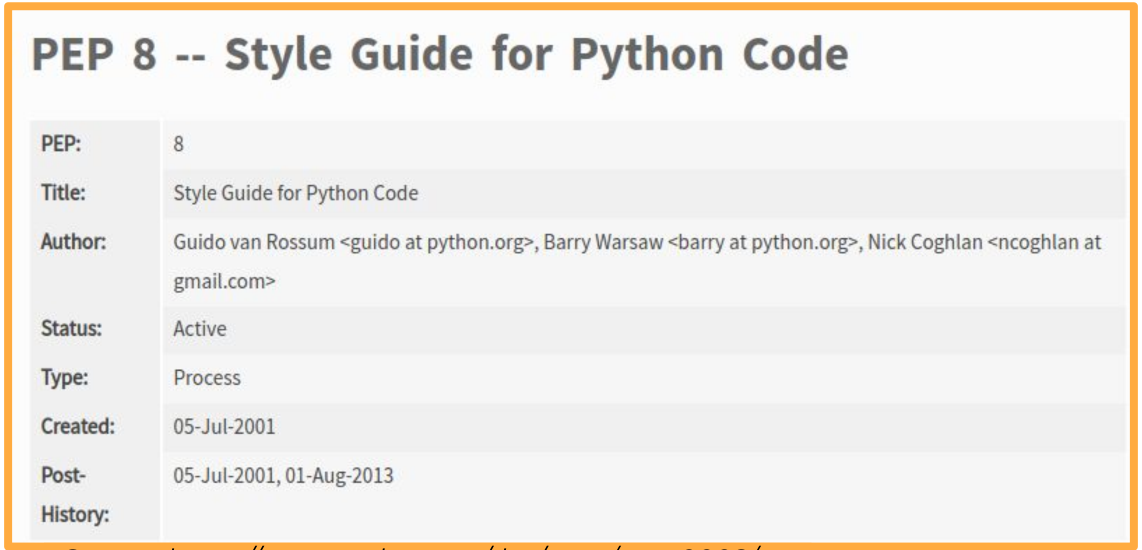
### Other Informational PEPs

| | PEP | PEP Title | PEP Author(s) |
|---|---|---|---|
| I | 13 | Python Language Governance | and community |
| I | 20 | The Zen of Python | Peters |
| I | 101 | Doing Python Releases 101 | Warsaw, GvR |
| IF | 247 | API for Cryptographic Hash Functions | Kuchling |
| IF | 248 | Python Database API Specification v1.0 | Lemburg |
| IF | 249 | Python Database API Specification v2.0 | Lemburg |
| I | 257 | Docstring Conventions | Goodger, GvR |

*Source: https://www.python.org/dev/peps/*

# Python Style Guide

## Python Enhancement Proposal 8



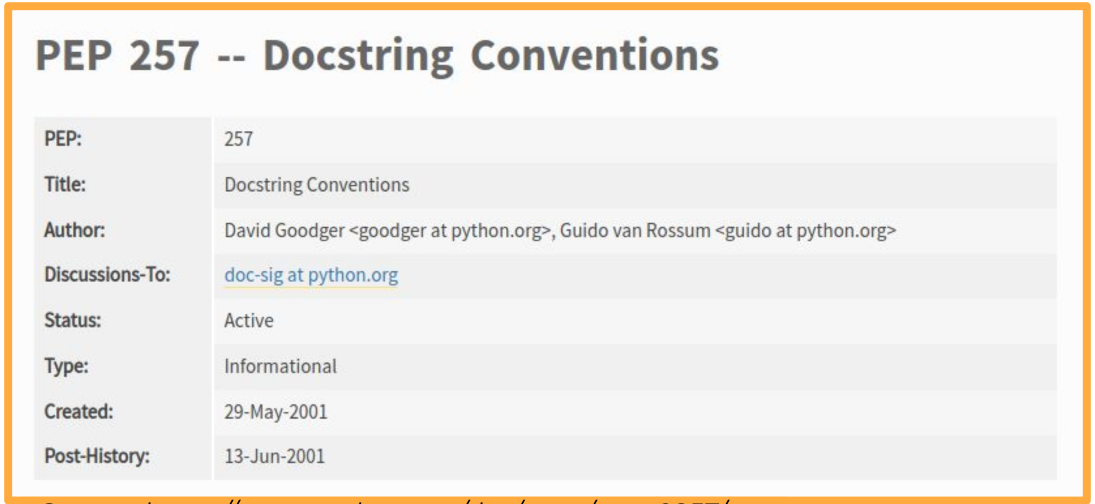
### PEP 8 -- Style Guide for Python Code

| | |
|---|---|
| PEP: | 8 |
| Title: | Style Guide for Python Code |
| Author: | Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com> |
| Status: | Active |
| Type: | Process |
| Created: | 05-Jul-2001 |
| Post-History: | 05-Jul-2001, 01-Aug-2013 |

*Source: https://www.python.org/dev/peps/pep-0008/*

# Python Style Guide

## Python Enhancement Proposal 257



### PEP 257 -- Docstring Conventions

| | |
|---|---|
| PEP: | 257 |
| Title: | Docstring Conventions |
| Author: | David Goodger <goodger at python.org>, Guido van Rossum <guido at python.org> |
| Discussions-To: | doc-sig at python.org |
| Status: | Active |
| Type: | Informational |
| Created: | 29-May-2001 |
| Post-History: | 13-Jun-2001 |

*Source: https://www.python.org/dev/peps/pep-0257/*

## PEP 8

### PEP8 is just a proposal

- *“Consistency with this style guide is important.”*
- *“Consistency within a project is more important.”*
- *“Consistency within one module or function is **the most important**.”*

*“A Foolish Consistency is the Hobgoblin of Little Minds”*

Quotes from Guido van Rossum, Barry Warsaw & Nick Coghlan
PEP 8

# PEP 8

## Indentation

- **Use 4 spaces per indentation level**

```
def hello_world():
    if True:
        print(“Hello World!”)
```

```
def hello_world():
  if True:
    print(“Hello World!”)
```

## PEP 8

### Indentation

- **Continuation lines vertically aligned**

```
bar = foo(arg1, arg2, arg3,
          kwarg1, kwarg2)
```

```
bar = foo(arg1, arg2, arg3,
    kwarg1, kwarg2)
```

# PEP 8

## Indentation

- **Add extra level on function headers**

```
def long_function_name(
        arg1, arg2, arg3,
        kwarg1, kwarg2):
    print(arg1)
```

```
def long_function_name(
    arg1, arg2, arg3,
    kwarg1, kwarg2):
    print(arg1)
```

# PEP 8

Digital Career Institute

## Blank lines

- **2 blank lines between top level functions and classes**

```
def bar():
    print('bar')


def foo():
    print('foo')
```

```
def bar():
    print('bar')

def foo():
    print('foo')
```

# PEP 8

## Blank lines

- **1 blank line between methods of a class**

```
class FooBar:
    def bar():
        print('bar')

    def foo():
        print('foo')
```

```
class FooBar:
    def bar():
        print('bar')


    def foo():
        print('foo')
```

# PEP 8

Digital Career Institute

## Imports

- **Should be placed at the top of the file, after the docstring**

```
import os

def bar():
    print(os)

bar()
```

```
def bar():
    print(os)

import os

bar()
```

# PEP 8

## Imports

- **Different lines for each package**
- **Same line for imports from the same packages**

```
import os
import sys
from foobar import foo, bar
```

```
import os, sys
from foobar import foo
from foobar import bar
```

# PEP 8

## Naming conventions

| | |
|---|---|
| **Classes** | PascalCase |
| **Methods, functions & variables** | snake_case |
| **Constants** | UPPER_CASE Upper snake case |

```
MY_CONSTANT = False
class FooBar:
    def foo_bar():
        is_true = True
```

```
myConstant = False
class Foo_Bar:
    def fooBar():
        istrue = True
```

## When in doubt

Digital Career Institute

Type:

```
>>> import this
```

# Tim Peter’s Zen of Python

- Explicit is better than implicit.
- Simple is better than complex.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- …

- Errors should never pass silently, unless explicitly silenced.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- …

## We learned ...

- What are coding styles and standards.
- What is the difference between them and the syntax.
- That there is a standard style guide for Python, known as PEP8.
- Why are style guides important and why should we conform to the standard.
- How and when to use the standard.
- Some of the standard styling rules of PEP8.

## Self Study

- Read *The Zen of Python*.
- Explore the *PEP8* recommendations.
- Explore the *PEP257* recommendations.
- Read about Python coding standards.

Digital Career Institute

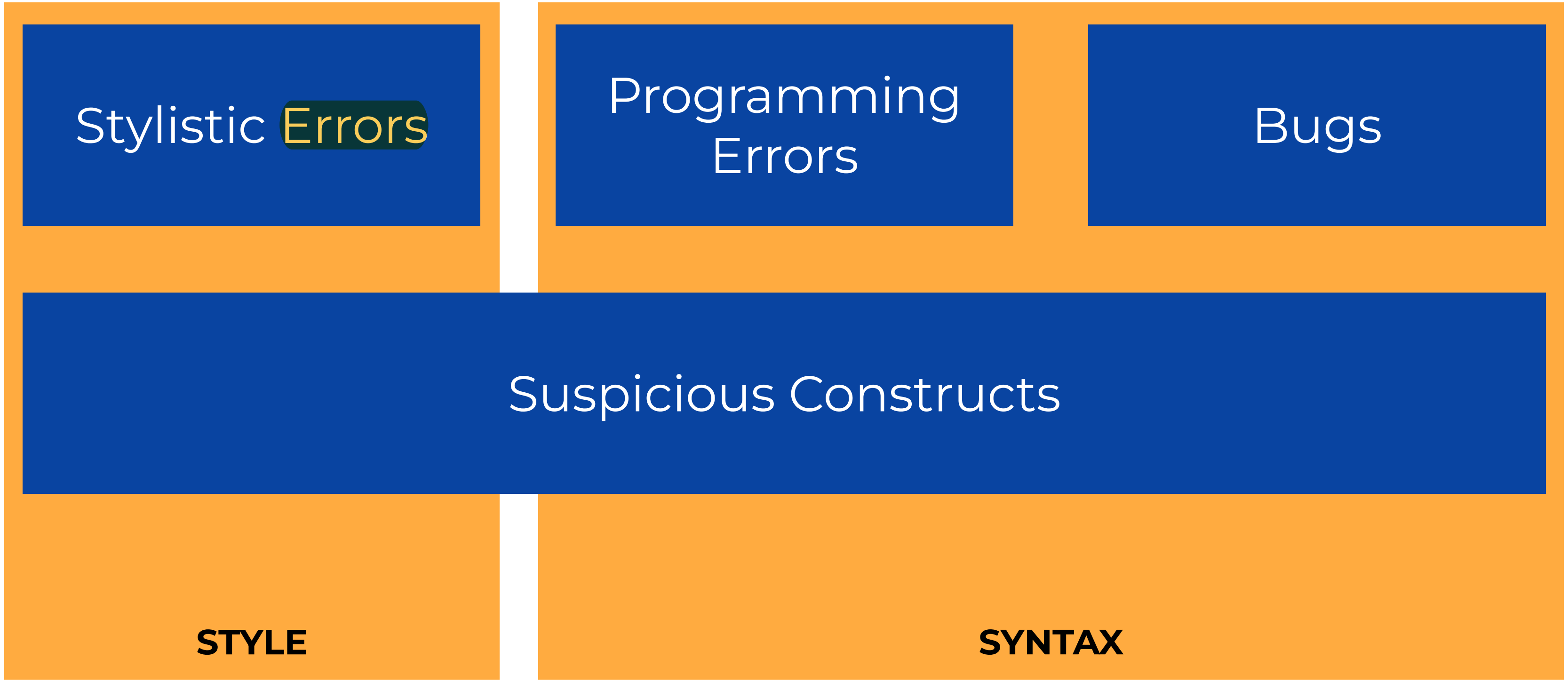
# Validation Tools

## Linter

Digital Career Institute

A **static code analysis tool** used to flag programming errors, bugs, **stylistic errors** or suspicious constructs.

# Linter

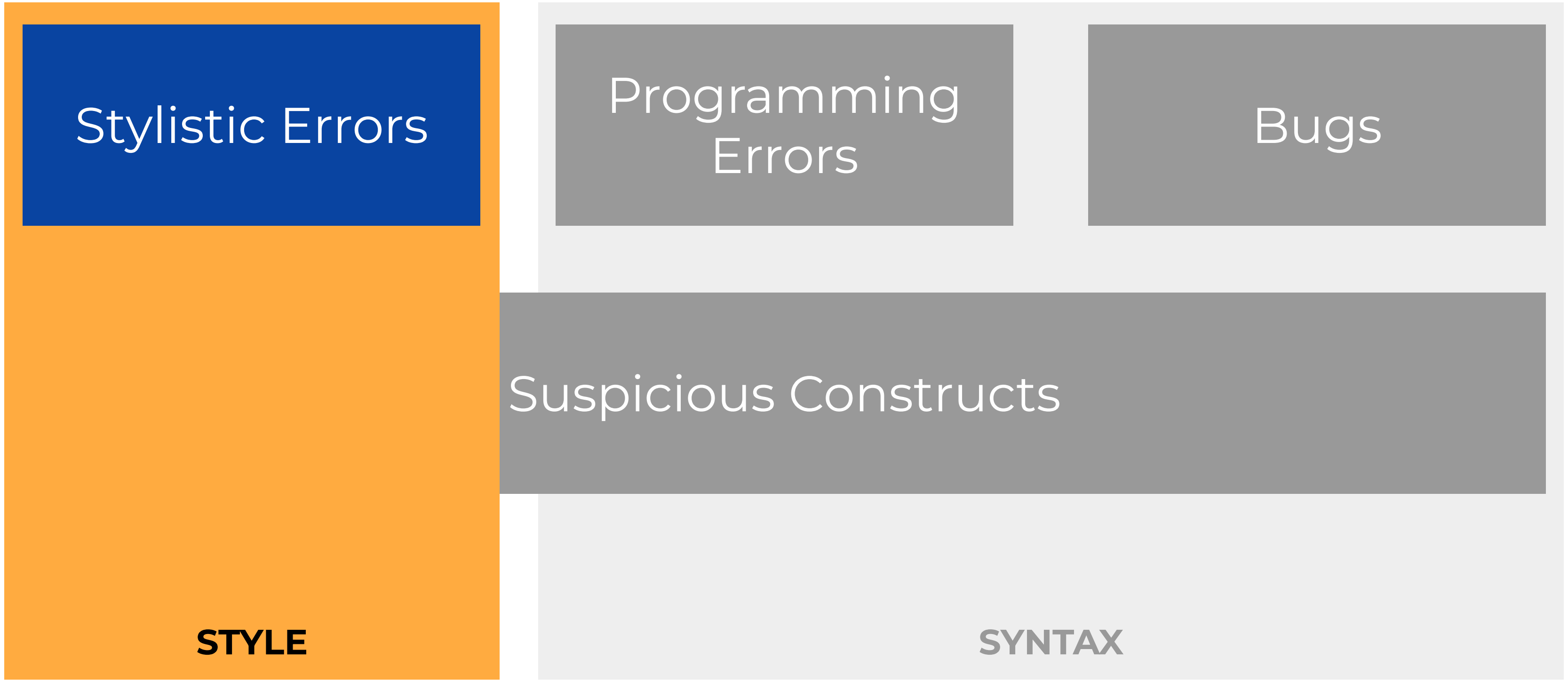


Stylistic Errors
Programming Errors
Bugs
Suspicious Constructs
STYLE
SYNTAX

## Tool: pycodestyle

**Pycodestyle** is a static code analysis tool used to check the code against **some of the PEP8** style conventions.

## Tool: pycodestyle



## Tool: pyflakes

**Pyflakes** is a static code analysis tool used to check the code for **non-stylistic errors**.

# Tool: pyflakes

Stylistic Errors

S

STYLE

Programming Errors

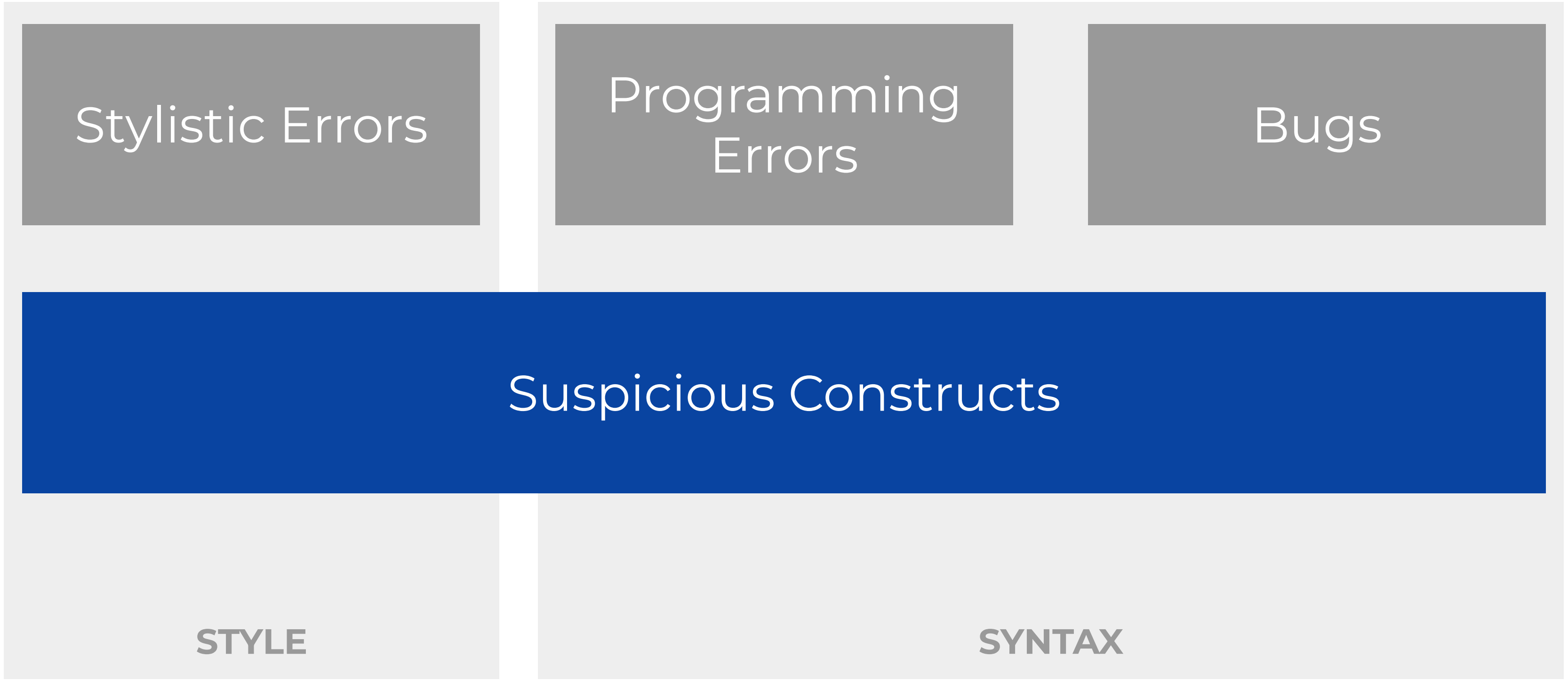
Bugs

SYNTAX

## Tool: McCabe

**McCabe** is a static code analysis tool used to check the code for methods and functions **exceedingly complex**.

# Tool: McCabe


Stylistic Errors
Programming Errors
Bugs
Suspicious Constructs
STYLE
SYNTAX

## Flake8

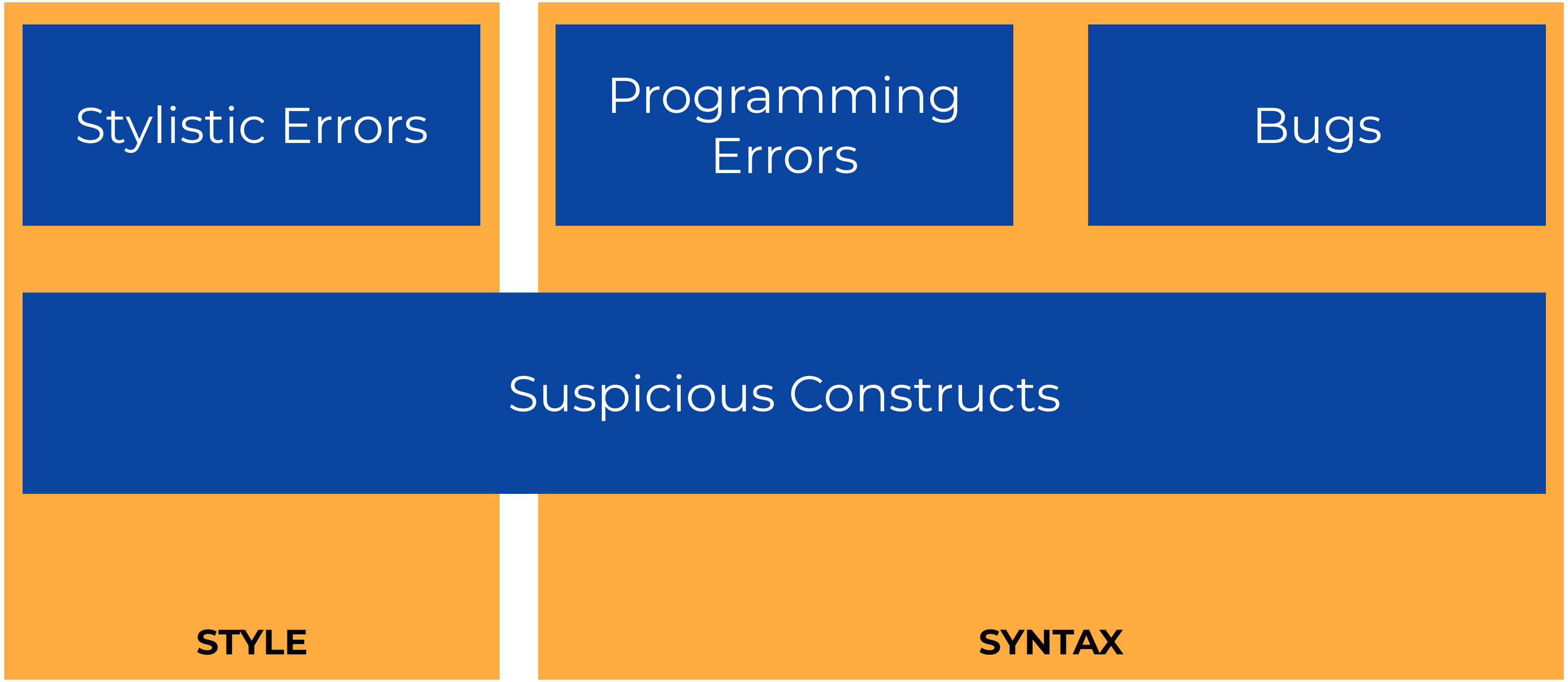
Digital Career Institute

# Flake8

**Flake8** is a linter that wraps together pycodestyle, pyflakes and McCabe complexity checker.

pycodestyle + pyflakes + McCabe

## Flake8


Stylistic Errors
Programming Errors
Bugs
Suspicious Constructs
STYLE
SYNTAX

## Flake8

Digital Career Institute

```
$ pip install flake8
```

## Flake8

Digital Career Institute

**playground.py**

```
a=2
b = 1

def sum(a,b):
     return a + b

print(sum(a,b))
```

```
$ flake8 playground.py
...
```

# Flake8

playground.py

```
a=2
b = 1

def sum(a,b):
        return a + b

print(sum(a,b))
```

```
playground.py:1:2: E225 missing
whitespace around operator

playground.py:4:1: E302 expected
2 blank lines, found 1

playground.py:4:10: E231 missing
whitespace after ','
```

# Flake8

playground.py

```
a=2
b = 1

def sum(a,b):
      return a + b

print(sum(a,b))
```

```
playground.py:5:6: E111 indentation
is not a multiple of 4

playground.py:5:6: E117
over-indented

playground.py:7:1: E305 expected 2
blank lines after class or function
definition, found 1

playground.py:7:12: E231 missing
whitespace after ','
```

# Flake8

**playground.py**

```python
a = 2
b = 1


def sum(a, b):
    return a + b


print(sum(a, b))
```

```
$ flake8 playground.py
$
```

## Flake8 # noqa

Digital Career Institute

**playground.py**

```python
a=2
b = 1

def sum(a,b): # noqa
     return a + b # noqa

print(sum(a,b)) # noqa
```

```
$ flake8 playground.py
playground.py:1:2: E225 missing
whitespace around operator
$
```

## Flake8 Options: Ignore Error Codes

Digital Career Institute

**playground.py**

```
a = 2
b = 1


def sum(a,b):
    return a + b


print(sum(a,b))
```

```
$ flake8 playground.py --ignore=E231
$
```

## Flake8 Options: Choose Error Codes

**playground.py**

```
a=2
b = 1

def sum(a, b):
      return a + b

print(sum(a, b))
```

```
$ flake8 playground.py --select=E231
$
```

## Flake8 Options: Exclude Directories

```
$ flake8 . --exclude=env
...
```

## Flake8 Error Codes

https://flake8.pycqa.org/en/latest/user/error-codes.html

| pycodestyle | pyflakes | McCabe |
|---|---|---|
| E??? W??? | F??? E999 | C901 |

## Other Tools & Linters

Digital Career Institute

## Other Tools

Flake8 provides a bare minimum with the most unanimously accepted recommendations.

Some of the features must be added with plugins:

| pep8-naming | flake8-docstrings | flake8-isort |
|---|---|---|
| **NAMING CONVENTIONS** | **DOCSTRINGS** | **IMPORTS ORDER** |

# Other Linters

Digital Career Institute

There are other linters that provide different flavours of linting.

PyLint

MyPy

Black

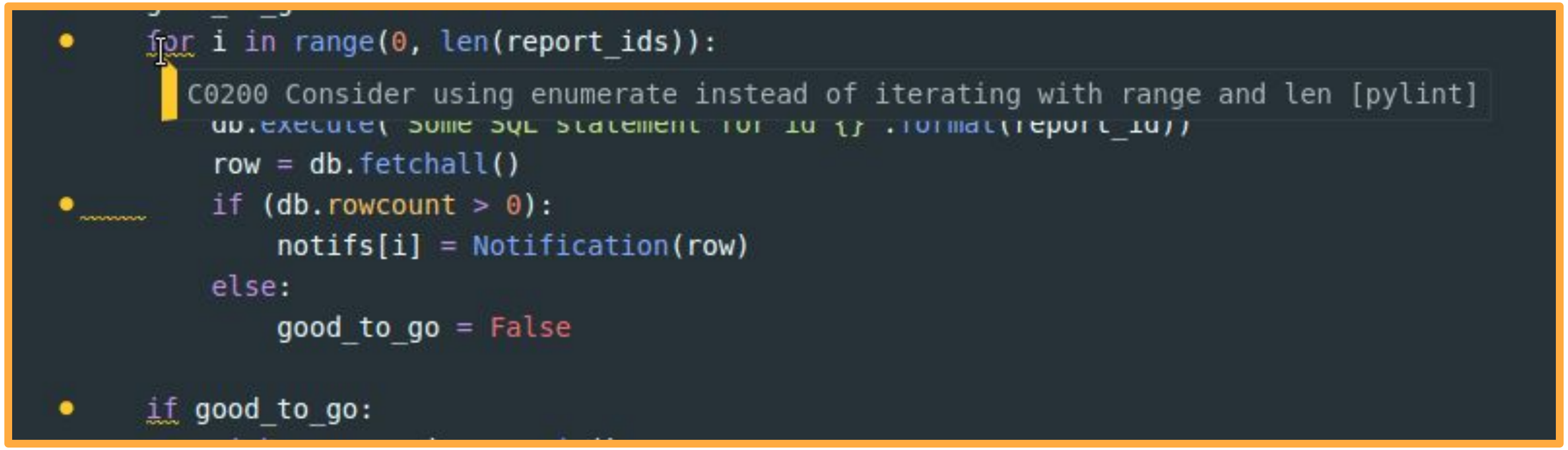
# Linter in the Editor

Digital Career Institute

- ✓ PyDev
- ✓ PyCharm
- ✓ Sublime
- ✓ Visual Studio Code
- ✓ Atom

Adding a linter to our editor will let us detect errors while we work.

## Linter in the Editor



```
for i in range(0, len(report_ids)):
 C0200 Consider using enumerate instead of iterating with range and len [pylint]
    db.execute('Some SQL statement for id {}'.format(report_id))
    row = db.fetchall()
    if (db.rowcount > 0):
        notifs[i] = Notification(row)
    else:
        good_to_go = False

if good_to_go:
```

## We learned ...

- That we can automate the detection of errors, both syntactic and stylistic, with a software called Lint.
- That there is a package called Flake8 that gathers together some tools for error detection.
- That we can customize which errors we want to detect.
- That we can integrate these tools into our editor to detect the errors earlier.

Digital Career Institute

## Documentation

## Documentation

Digital Career Institute
DCI

- Coding standards
  https://en.wikipedia.org/wiki/The_Elements_of_Programming_Style
- Python Style Recommendations
  https://www.python.org/dev/peps/pep-0008/
  https://www.python.org/dev/peps/pep-0257/
- Tools and linters
  https://flake8.pycqa.org/en/latest/
  https://pycodestyle.pycqa.org/en/latest/
  https://pypi.org/project/pyflakes/
  https://github.com/PyCQA/mccabe
- Editors
  https://code.visualstudio.com/docs/python/linting
  https://atom.io/packages/linter-flake8
  https://www.pydev.org/manual_adv_pylint.html

# THANK YOU

**Contact Details**
**DCI Digital Career Institute gGmbH**

Digital Career Institute

DCI