# pandas Coding Style Guide

## 1 Background

Writing good code is not just about what you write. It is also about how you write it. During Continuous Integration testing, several tools will be run to check your code for stylistic errors. Generating any warnings will cause the test to fail. Thus, good style is a requirement for submitting code to \*pandas\*. This document serves as a supplement to the script demonstrating the proper coding style required for contributing to \*pandas\*.

## 2 Patterns

### 2.1 Line Length

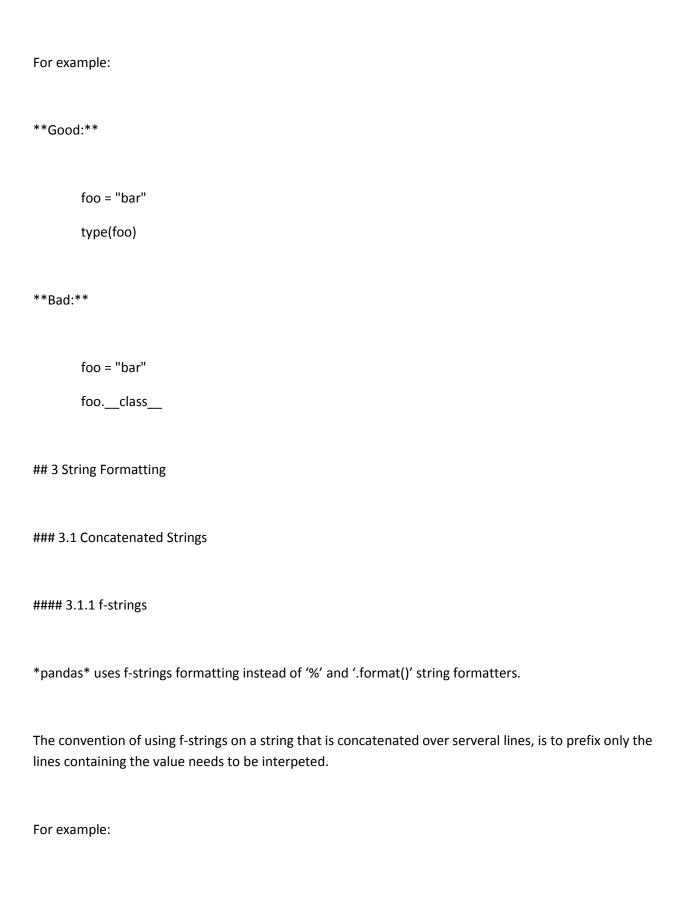
Line length is restricted to 80 characters to promote readability.

### 2.2 Header File

Every header file must include a header guard to avoid name collision if re-included.

### 2.3 foo.\\_class\\_

\*\*pandas\*\* uses 'type(foo)' instead 'foo.\_\_class\_\_' as it is making the code more readable.



```
foo = "old_function"
       bar = "new_function"
       my_warning_message = (
       f"Warning, {foo} is deprecated, "
       "please use the new and way better "
       f"{bar}"
       )
**Bad:**
       foo = "old_function"
       bar = "new_function"
       my_warning_message = (
       f"Warning, {foo} is deprecated, "
       f"please use the new and way better "
       f"{bar}"
       )
```

\*\*Good:\*\*

## #### 3.1.2 White Spaces

Putting the white space only at the end of the previous line, so there is no whitespace at the beggining of the concatenated string.

```
For example:
**Good:**
        example_string = (
        "Some long concatenated string,"
        "with good placement of the "
        "whitespaces"
       )
**Bad:**
       example_string = (
        "Some long concatenated string,"
        " with bad placement of the"
        " whitespaces"
       )
### 3.2 Representation function (aka 'repr()')
*pandas* uses 'repr()' instead of '%r' and '!r'.
```

The use of 'repr()' will only happend when the value is not an obvious string.

```
For example:
**Good:**
       value = str
       f"Unknown recived value, got: {repr(value)}"
**Bad:**
       value = str
       f"Unknown recived type, got: '{type(value).__name__}'"
## 4 Types
**pandas** strongly encourages the use of PEP 484 style type hints. New development should contain
type hints and pull requests to annotate existing code are accepted as well!
### 4.1 Imports
Types imports should follow the `from typing import ...` convention.
**Good:**
       from typing import List, Optional, Union
        primes: List[int] = []
```

```
**Bad:**

import typing

primes: typing.List[int] = []

Optional should be used where applicable

**Good:**

maybe_primes: List[Optional[int]] = []

**Bad:**
```