Write Python Code Properly!

Video

https://youtu.be/D4_s3q038I0?si=wSzZoEI-9PPSwM8w

Notes

- Any line should not exceed 80 characters (79 characters + the newline characters)
  - Any line that does, should be split up
  - A formatter can be used to fix this when you save the file
    - The ‘black’ formatter is a popular choice
- Indentation should be done with four spaces
- Whatever style you go for should be consistent
  - If you use single quotations marks, always use single quotation marks

Naming Conventions

- Variables should be lower case and written in snake case
  - `this_is_a_variable`
- Constants should be upper case and written in snake case
  - `THIS_IS_A_CONSTANT`
- Modules (python files) should be lower case and ideally one word (if more words, use underscores)
  - `test.py`
  - `menu_test.py`

- Functions should be lower case and written in snake case
  - `def hello_world():`
- Classes should be in Pascal case
  - `class HelloWorld:`
- Exceptions should be in Pascal case
  - This is because in Python, exceptions are classes
  - `class AnException:`
- The first parameter of an instance method of class should be `self`

```
class Test:
  def test(self, name):
    pass
```

- The first parameter of a class method should be `cls`

```
class Test:
  def test(self, name):
    pass

  @classmethod
    def cls_method(cls, name):
      pass
```

Spacing

- Any top-level functions or classes should be separated by two blank lines

```
class One:
# blank line
# blank line
class Two:
```

- Methods inside a class are separated by one blank line

```
class One:
# blank line
# blank line
class Two:
  def test(self):
    pass
  # blank line
  def test_two(self):
    pass
```

Imports

- Should always be at the top of a file
    - Unless you have a module doc string (which is used to explain what the module contains)

```
"""
Contains the Person() class
"""

import sys
```

- Each import should be on its own line (one distinct module to each line)
- You canimport multiple things from a module on the same line
    - `from module import x, y`
- You should not use the wild card import
    - `from sys import *`

Single vs Double Quotes

- They are completely equivalent
- The only rule is that it should be consistent (pick one and stick with it, do not use a mixture)
    - Unless you need to embed the one you have chosen within a string, in which case you can use the other to do this

```
string = "a short string"
string_with_double_marks = 'this string should have a: "'
```

- However, when you are using a triple quoted string, use double quotes

```
"""
Use this
"""

'''
Not this
'''
```

Whitespaces

- No whitespaces immediately inside parentheses, brackets or braces

```
# Correct:
spam(ham[1], {eggs: 2})
# Wrong:
spam( ham[ 1 ], { eggs: 2 } )
```

- Not whitespaces between a trailing comma and a following close parenthesis

```
# Correct:
foo = (0,)
# Wrong:
bar = (0, )
```

- No whitespaces immediately before a comma, semicolon, or colon

```
# Correct:
if x == 4: print(x, y); x, y = y, x
# Wrong:
if x == 4 : print(x , y) ; x , y = y , x
```

- No whitespaces before the open parenthesis that starts the argument list of a function call

```
# Correct:
spam(1)
# Wrong:
spam (1)
```

- If operators with different priorities are used, consider adding whitespace around the operators with the lowest priority(ies). Use your own judgment; however, never use more than one space, and always have the same amount of whitespace on both sides of a binary operator

```
# Correct:
i = i + 1
submitted += 1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
# Wrong:
i=i+1
submitted +=1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a - b)
```

- Don’t use spaces around the = sign when used to indicate a keyword argument, or when used to indicate a default value for an unannotated function parameter

```
# Correct:
def complex(real, imag=0.0):
    return magic(r=real, i=imag)
# Wrong:
def complex(real, imag = 0.0):
    return magic(r = real, i = imag)
```

Inline Comments

- Should be used sparingly
- You should have two spaces after the code and one space after the hash sign

```
# Correct
COLOR = (0, 255, 0)  # This is correct

# Wrong
COLOR = (0, 255, 0) #This is incorrect
```

“is None” or “== None”

- To check if something is `None` , you should use `if foo is None:`
    - Doubles equals should not be used. For example, `if foo == None:`

Try and Except

- When catching exceptions, mention specific exceptions whenever possible instead of using a bare `except:` clause

```
try:
    import platform_specific_module
except ImportError:
    platform_specific_module = None
```

Resources

- PEP8 Style Guide - https://peps.python.org/pep-0008/