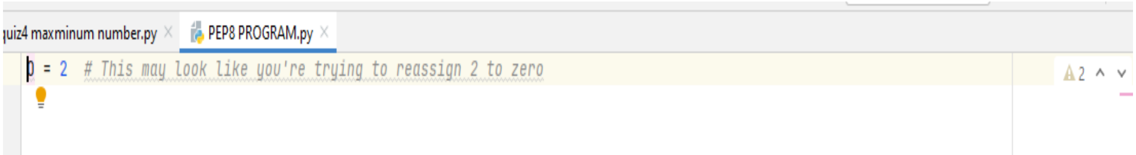
# PRACTICAL7: Implementing coding practices in Python using PEP8

PEP 8 exists to improve the readability of Python code.

## 1: Naming Conventions

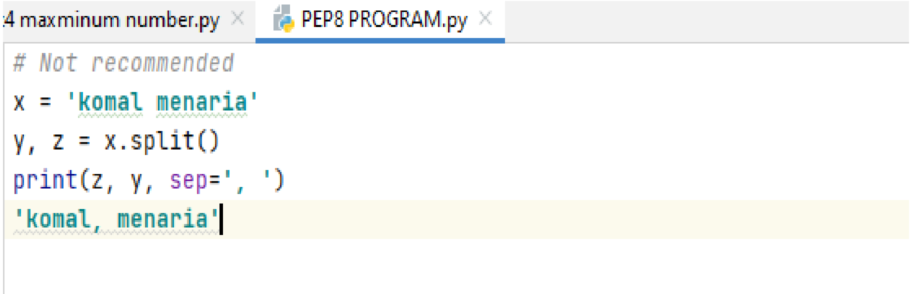
When you write Python code, you have to name a lot of things: variables, functions, classes, packages, and so on. Choosing sensible names will save you time and energy later. You'll be able to figure out, from the name, what a certain variable, function, or class represents. You'll also avoid using inappropriate names that might result in errors that are difficult to debug.



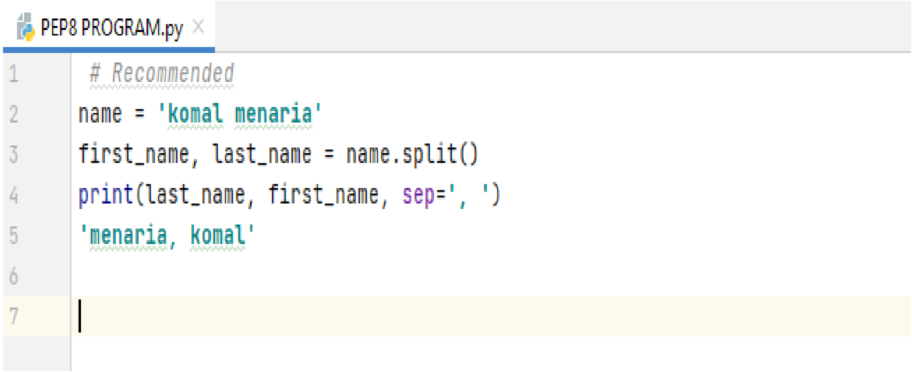
```
O = 2  # This may look like you're trying to reassign 2 to zero
```

## 2: How to Choose Names

When naming variables, you may be tempted to choose simple, single-letter lowercase names, like x. But, unless you're using x as the argument of a mathematical function, it's not clear what x represents.



```
# Not recommended
x = 'komal menaria'
y, z = x.split()
print(z, y, sep=', ')
'komal, menaria'
```



```
 # Recommended
name = 'komal menaria'
first_name, last_name = name.split()
print(last_name, first_name, sep=', ')
'menaria, komal'
```

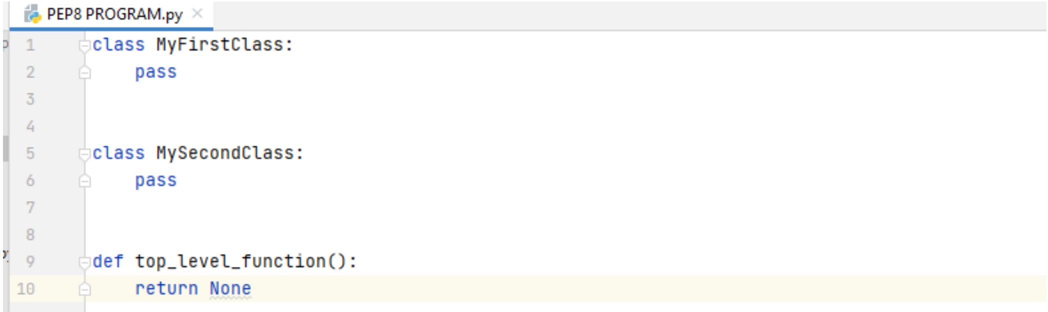
# 3: Code Layout

## Blank Lines

Vertical whitespace, or blank lines, can greatly improve the readability of your code. Code that's bunched up together can be overwhelming and hard to read. Similarly, too many blank lines in your code makes it look very sparse, and the reader might need to scroll more than necessary. Below are three key guidelines on how to use vertical whitespace.

### Surround top-level functions and classes with two blank lines.

Top-level functions and classes should be fairly self-contained and handle separate functionality. It makes sense to put extra vertical space around them, so that it's clear they are separate:



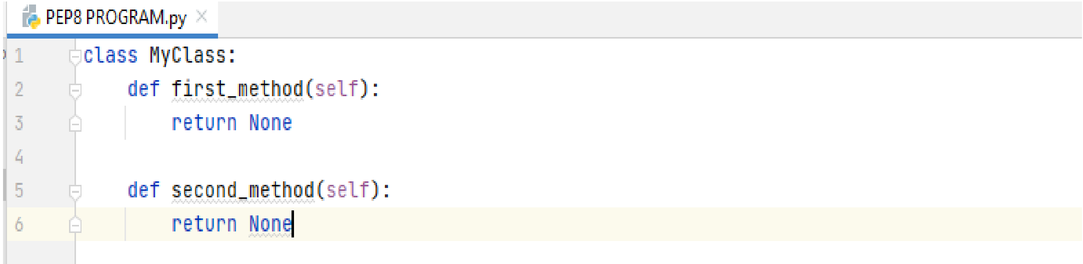
```python
class MyFirstClass:
    pass


class MySecondClass:
    pass


def top_level_function():
    return None
```

### Surround method definitions inside classes with a single blank line.

Inside a class, functions are all related to one another. It's good practice to leave only a single line between them:

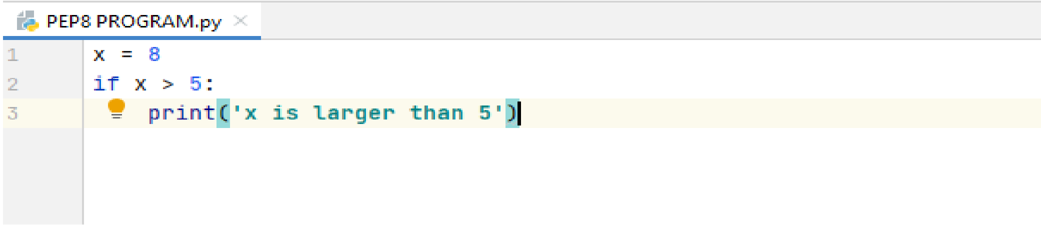


```python
class MyClass:
    def first_method(self):
        return None

    def second_method(self):
        return None
```

# 4: Indentation

Indentation, or leading whitespace, is extremely important in Python. The indentation level of lines of code in Python determines how statements are grouped together.



```python
x = 8
if x > 5:
    print('x is larger than 5')
```

# 5: Comments

You should use comments to document code as it’s written. It is important to document your code so that you, and any collaborators, can understand it. When you or someone else reads a comment, they should be able to easily understand the code the comment applies to and how it fits in with the rest of your code.

Here are some key points to remember when adding comments to your code:

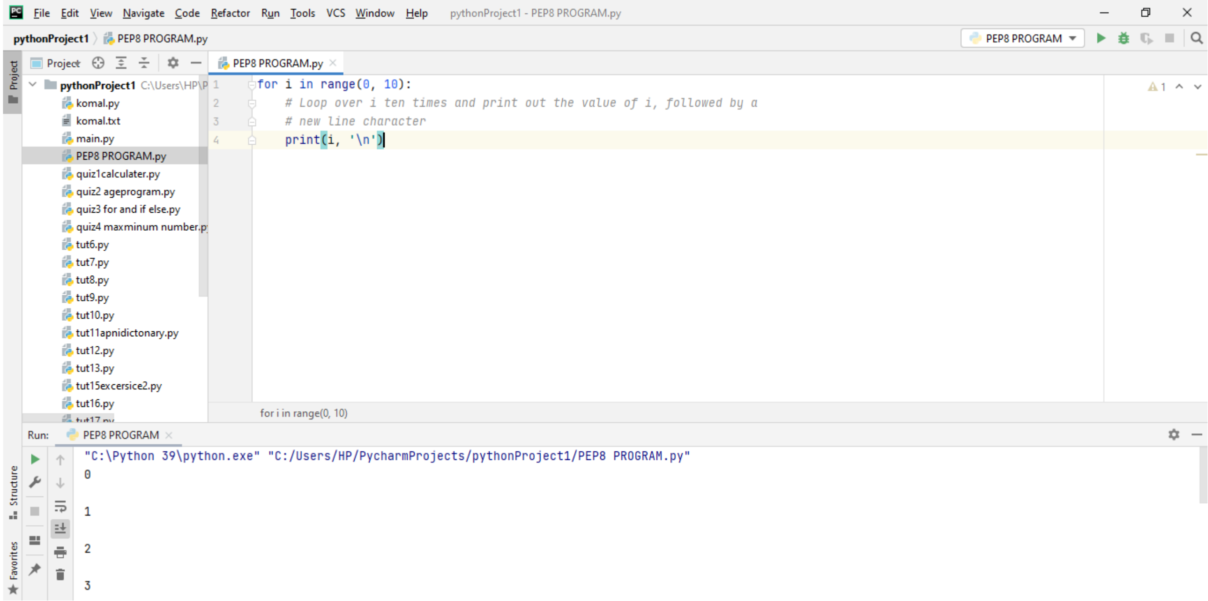
- Limit the line length of comments and docstrings to 72 characters.
- Use complete sentences, starting with a capital letter.
- Make sure to update comments if you change your code.

## Block Comments

Use block comments to document a small section of code. They are useful when you have to write several lines of code to perform a single action, such as importing data from a file or updating a database entry. They are important as they help others understand the purpose and functionality of a given code block.

PEP 8 provides the following rules for writing block comments:

- Indent block comments to the same level as the code they describe.
- Start each line with a # followed by a single space.
- Separate paragraphs by a line containing a single #.



```python
for i in range(0, 10):
    # Loop over i ten times and print out the value of i, followed by a
    # new line character
    print(i, '\n')
```

```
"C:\Python 39\python.exe" "C:/Users/HP/PycharmProjects/pythonProject1/PEP8 PROGRAM.py"
0

1

2

3
```