Clean Code in Python

Akas Steven Tambunan

JCDS 0212 - Surabaya

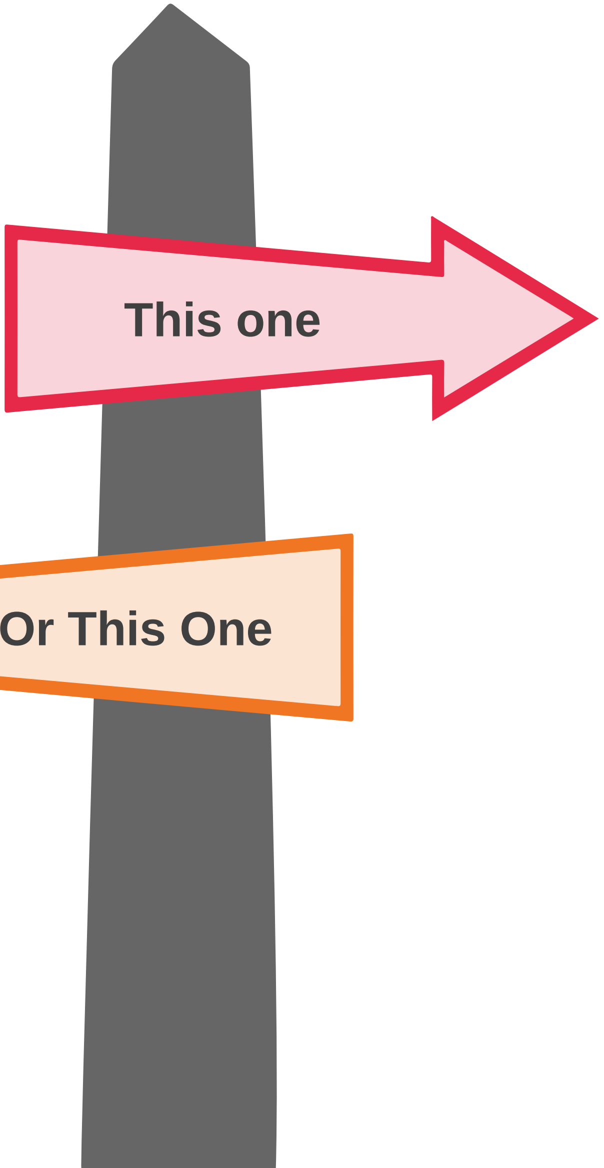
What is Clean Data ??

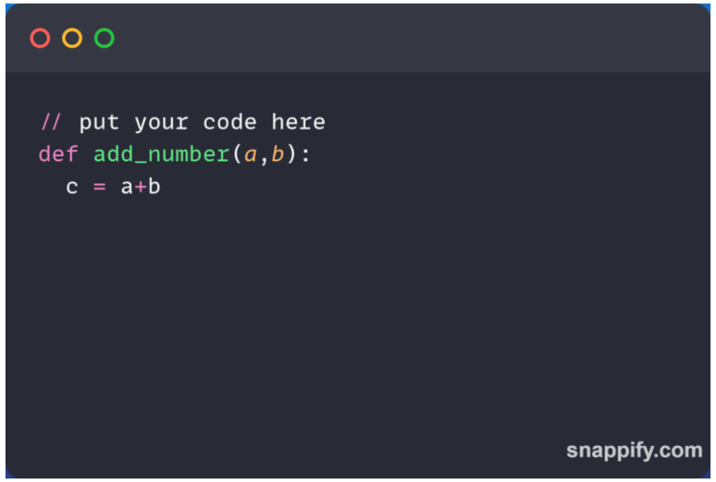
Makes it As Short as Possible
And more understandable

About
Efficiency

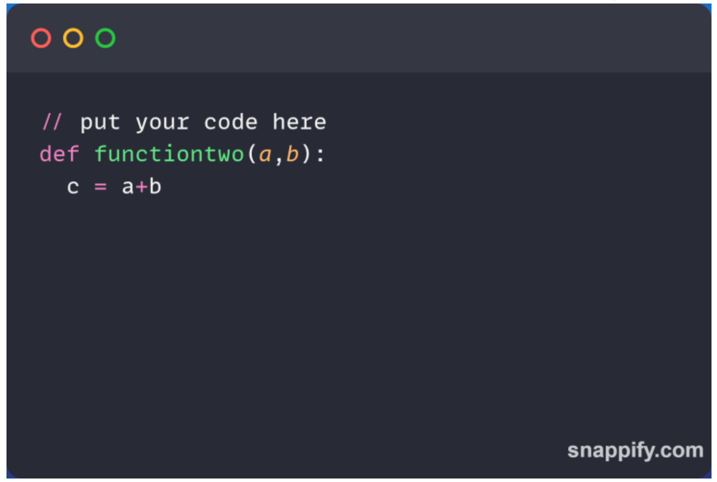
Which one is more understandable

Correct syntax is not enough for clean code but should have suggestive names (Avoid use ambiguous shorthand)





```
// put your code here
def add_number(a,b):
  c = a+b
```



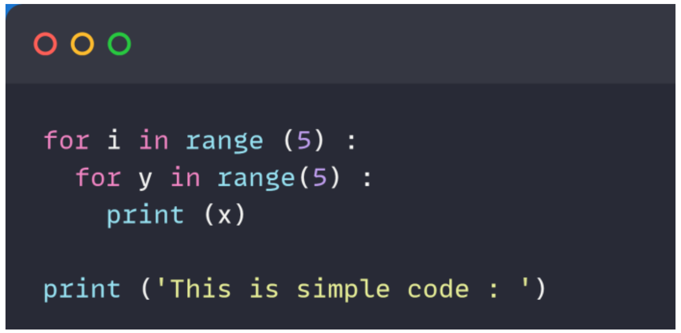
```
// put your code here
def functiontwo(a,b):
  c = a+b
```

Tips for writing clean code

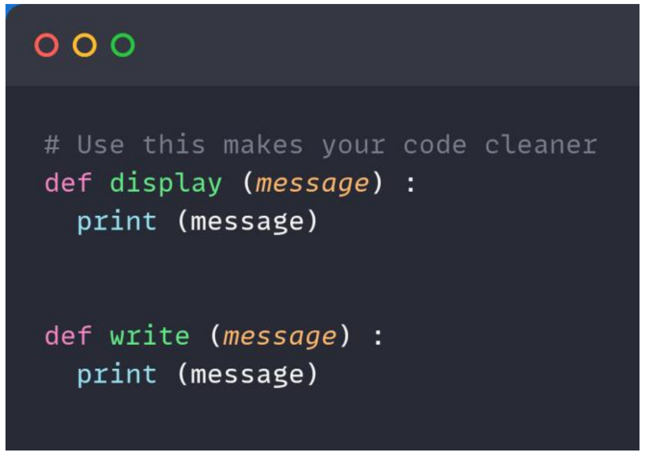
“

Here’s the tip

”



```
for i in range (5) :
  for y in range(5) :
    print (x)

print ('This is simple code : ')
```



```
# Use this makes your code cleaner
def display (message) :
  print (message)


def write (message) :
  print (message)
```

Use spaces than tab

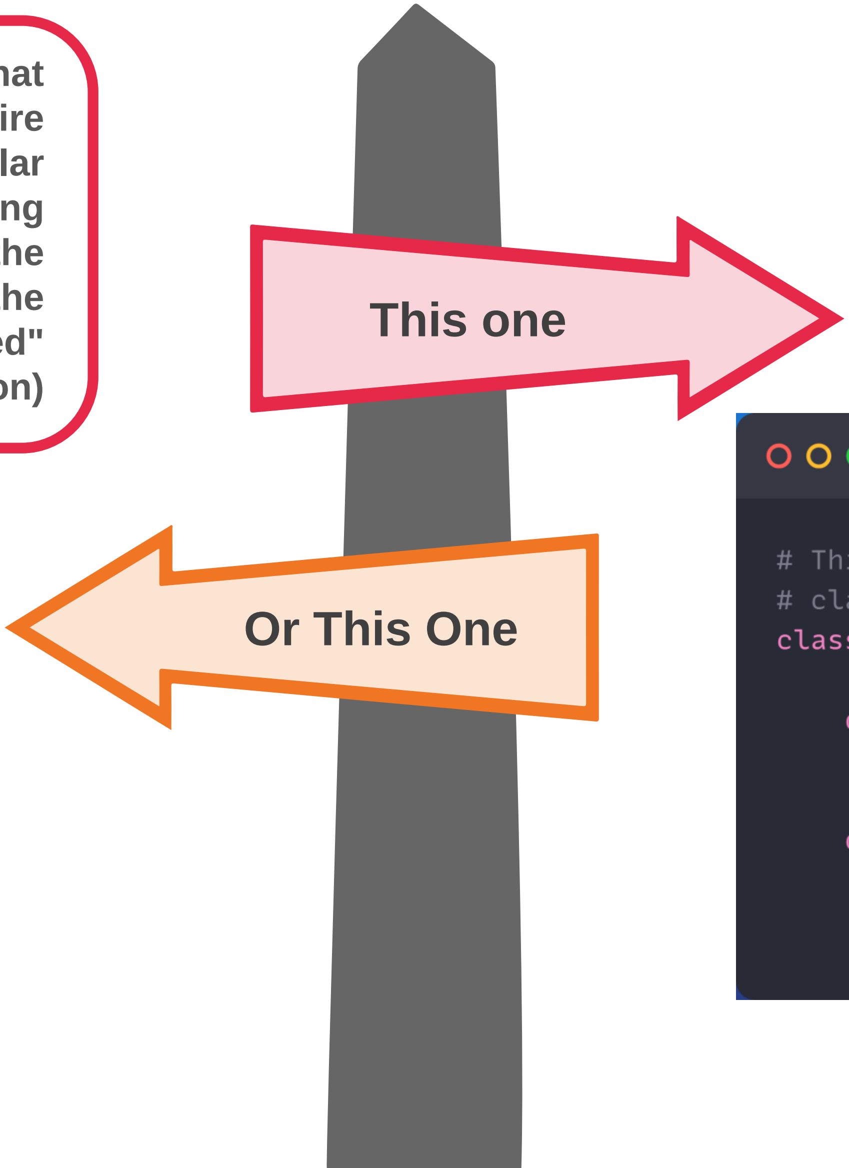
four spaces used for each level of indentation (4 spaces equal to one tab) that mean tab is used only when we will be use indentation

Don’t use long lines in python code

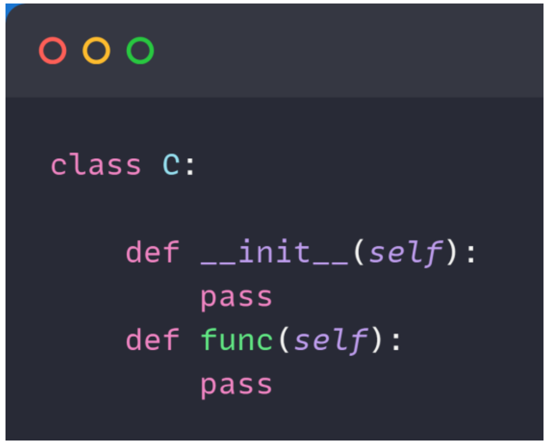
Instead long line it’s better to split it into multiple line

There should be two blank lines after each function/class

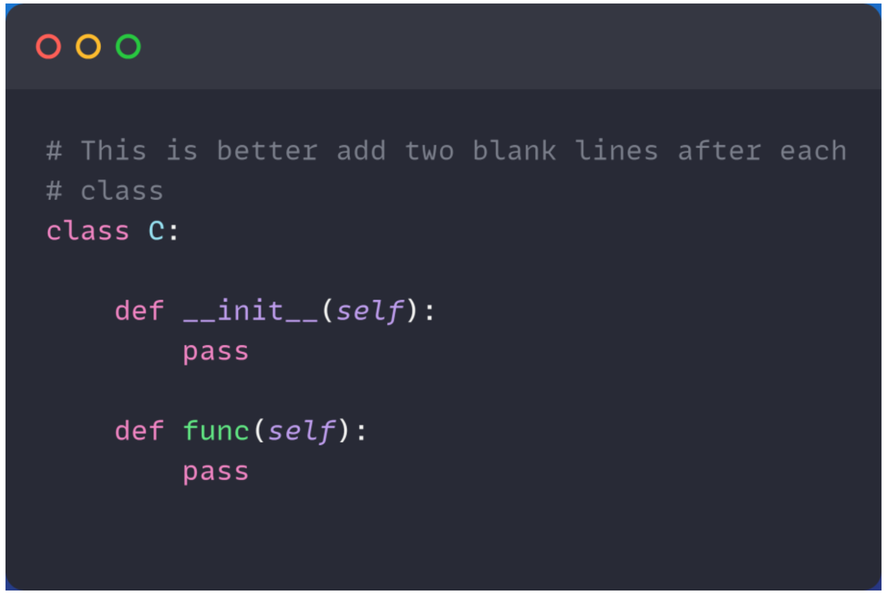
Which one is more understandable



It's extremely ugly to have that whitespace consistent in the entire codebase and just in this particular situation, where you don't have anything to add between class definition and the first inner method (which usually is the __init__ one), to end up with "glued" headers of class and method definition)



```python
class C:

    def __init__(self):
        pass
    def func(self):
        pass
```



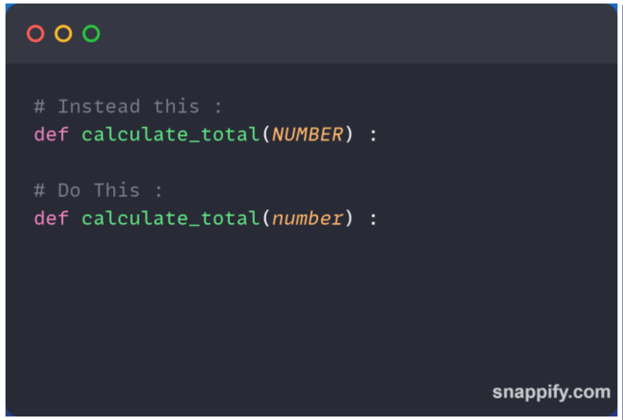
```python
# This is better add two blank lines after each
# class
class C:

    def __init__(self):
        pass

    def func(self):
        pass
```

Common name types in Phyton

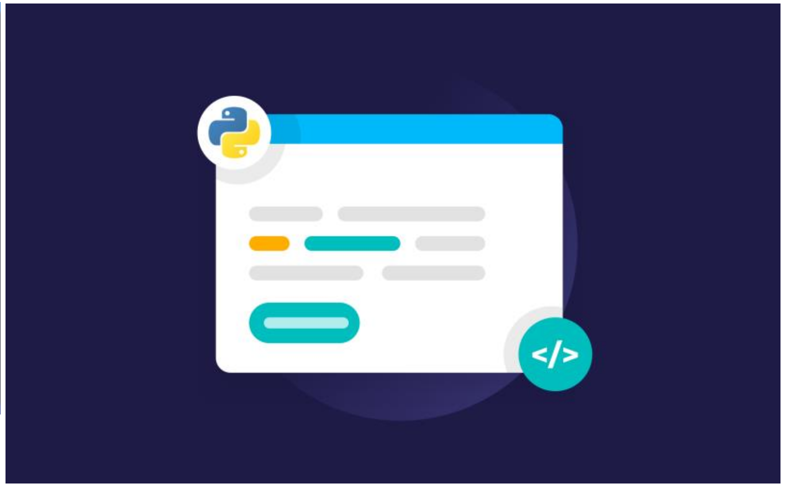
Camel Case

This should be uses in classes
Exp : newStudent, dataExplanatory, etc

Using CamelCase for classes makes it easier for others to read and understand the code and when Using lowercase for functions and variables helps programmers quickly identify their purpose



```
# Instead this :
def calculate_total(NUMBER) :

# Do This :
def calculate_total(number) :
```



Lower case

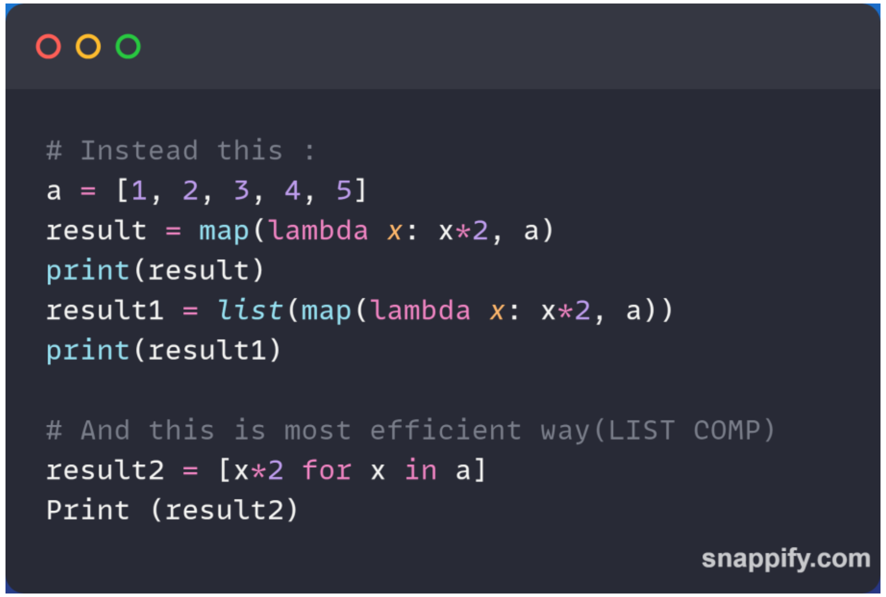
This use when defined function and variables

When defining a function, use lowercase letters and when passing variables, use lowercase letters

Compares using list comprehension with map and filter

List comprehension > lambda

When simple case it’s better to use lambda but not in complex case

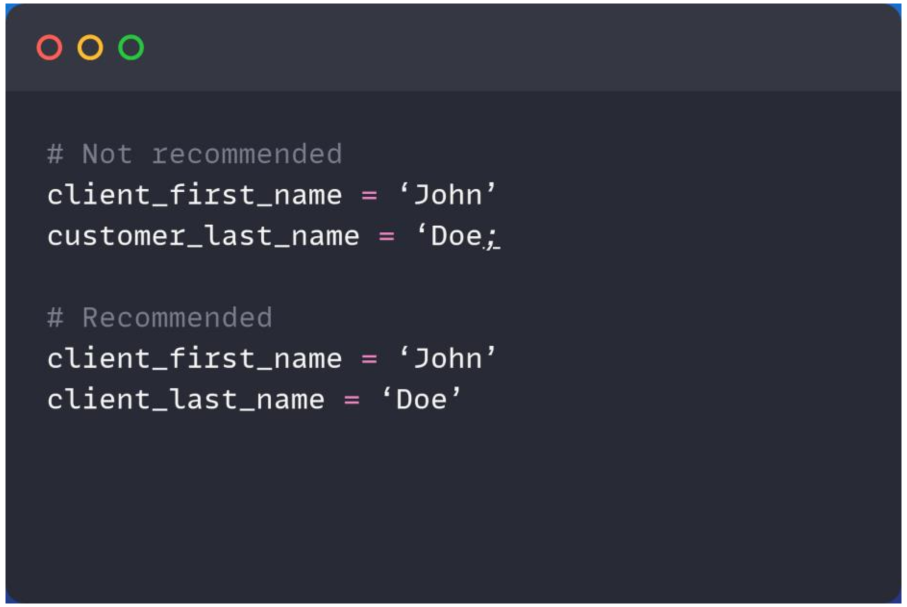


```
# Instead this :
a = [1, 2, 3, 4, 5]
result = map(lambda x: x*2, a)
print(result)
result1 = list(map(lambda x: x*2, a))
print(result1)

# And this is most efficient way(LIST COMP)
result2 = [x*2 for x in a]
Print (result2)
```

Actually

Each people have difference preferences

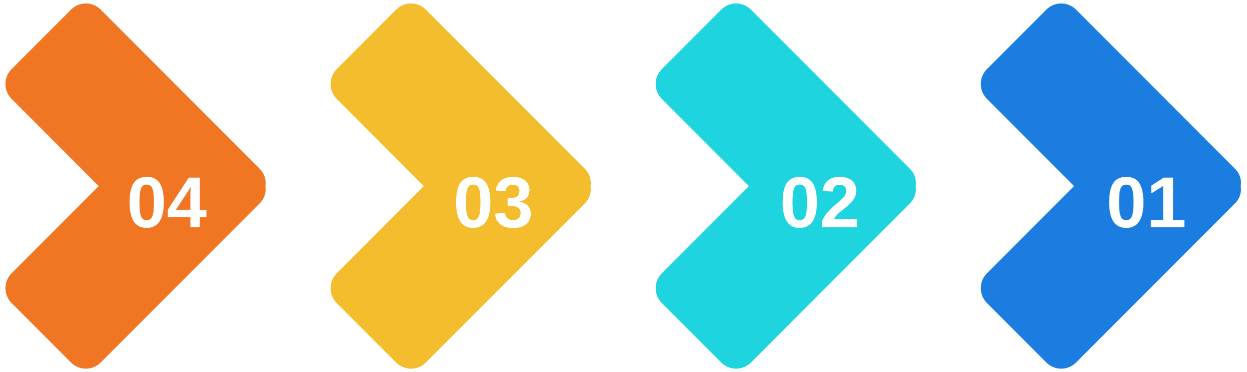
Consistent with your code



```
# Not recommended
client_first_name = ‘John’
customer_last_name = ‘Doe;

# Recommended
client_first_name = ‘John’
client_last_name = ‘Doe’
```

Be consistent with your function naming convention.

Use descriptive intention revealing names



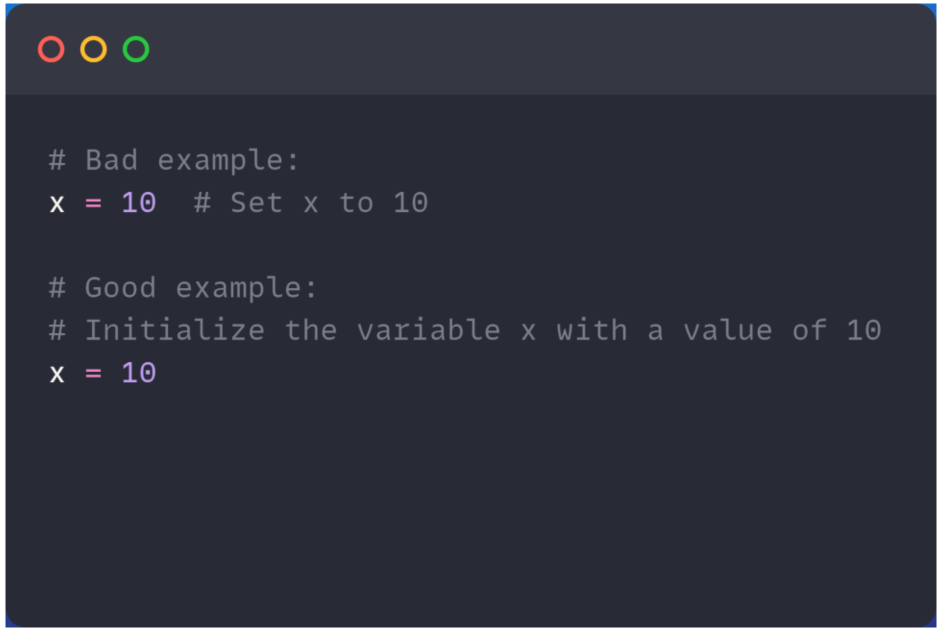
Use long descriptive names that are easy to read

Avoid using ambiguous shorthand

Use same vocabulary

Be consistent with your naming convention. Maintaining a consistent naming convention is important to eliminate confusion when other developers work on your code

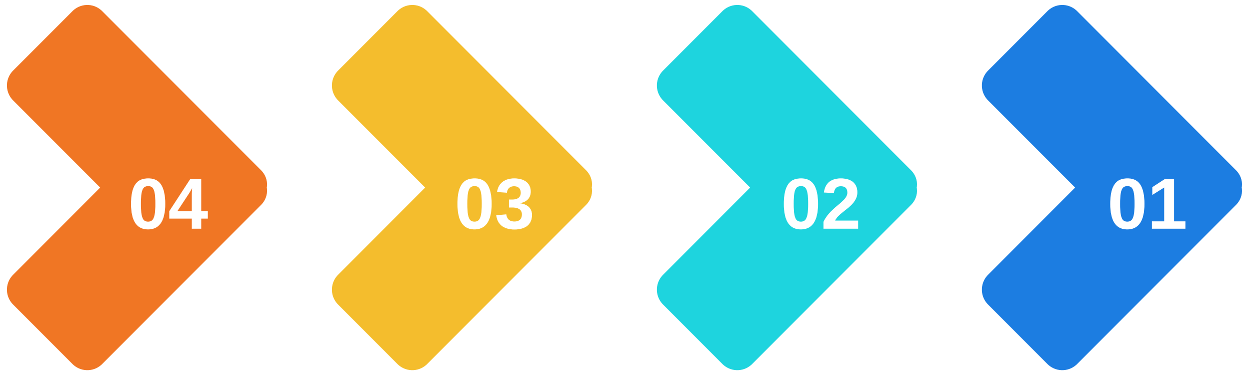
Commend your code



```
# Bad example:
x = 10  # Set x to 10

# Good example:
# Initialize the variable x with a value of 10
x = 10
```

Commend code sometimes needed

Commend code help to remember in step you doing when coding



Don't leave commented out code

Readable code doesn't need comments

Don't add noise comments

Don’t commend on bad Code rewrite it

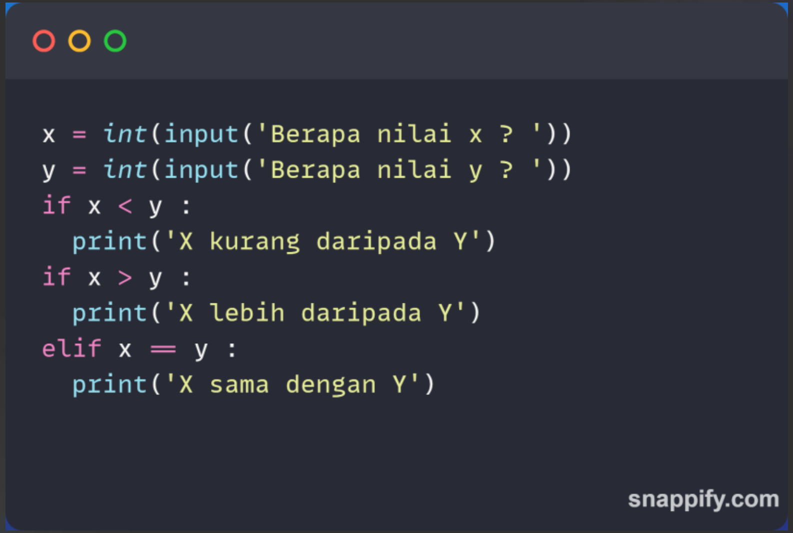
Section Break

Choose the better one

Choose one

Which one code is more clean

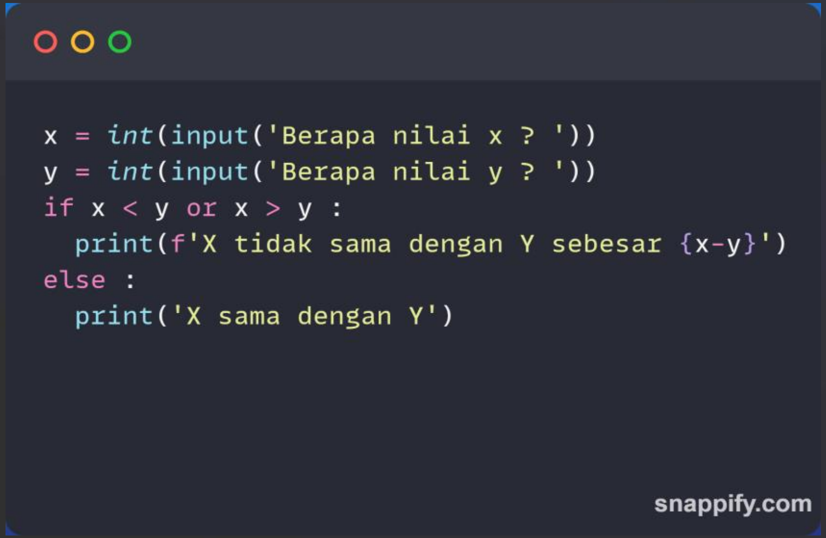
01 This one



```python
x = int(input('Berapa nilai x ? '))
y = int(input('Berapa nilai y ? '))
if x < y :
  print('X kurang daripada Y')
if x > y :
  print('X lebih daripada Y')
elif x == y :
  print('X sama dengan Y')
```

snappify.com

02 This one



```python
x = int(input('Berapa nilai x ? '))
y = int(input('Berapa nilai y ? '))
if x < y or x > y :
  print(f'X tidak sama dengan Y sebesar {x-y}')
else :
  print('X sama dengan Y')
```

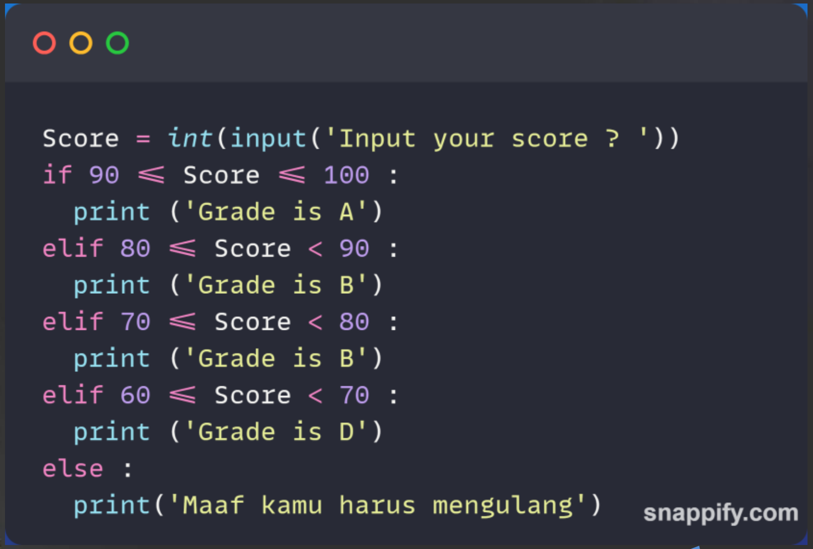
snappify.com

Code no 1 define variable x more or less than y by print word ‘ X kurang / lebih dari pada Y’ while in no 2 difference states by numeric although code no 2 is shorter but it’s not the same program with no 1

Choose one

Which one code is more clean

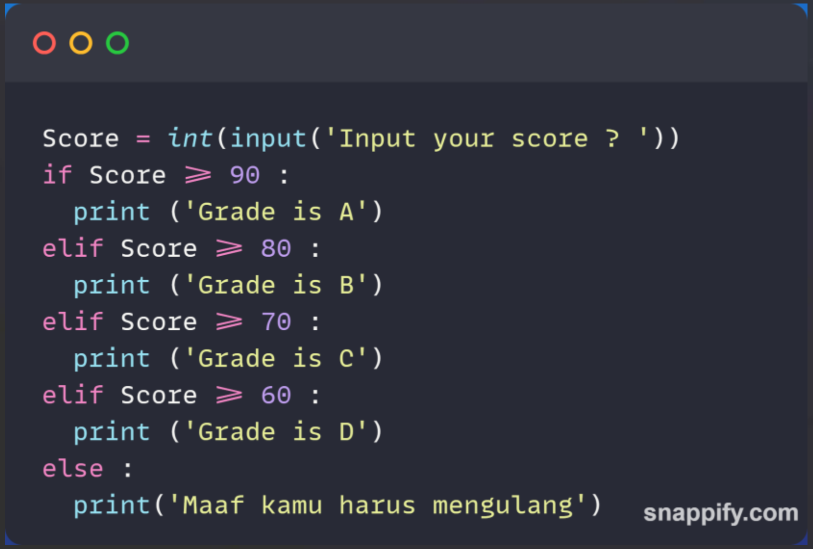
01 This one



```
Score = int(input('Input your score ? '))
if 90 ⩽ Score ⩽ 100 :
  print ('Grade is A')
elif 80 ⩽ Score < 90 :
  print ('Grade is B')
elif 70 ⩽ Score < 80 :
  print ('Grade is B')
elif 60 ⩽ Score < 70 :
  print ('Grade is D')
else :
  print('Maaf kamu harus mengulang')
```

snappify.com

02 This one



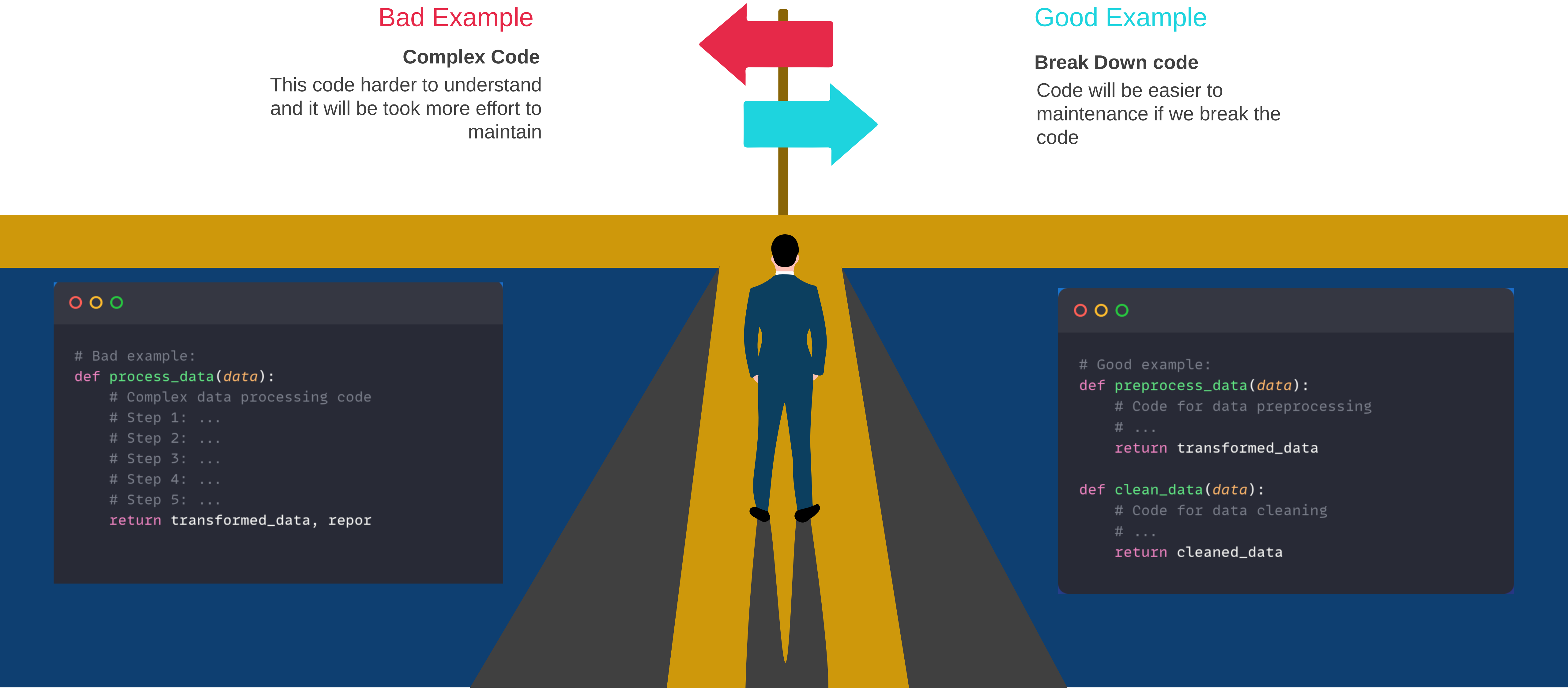
```
Score = int(input('Input your score ? '))
if Score ⩾ 90 :
  print ('Grade is A')
elif Score ⩾ 80 :
  print ('Grade is B')
elif Score ⩾ 70 :
  print ('Grade is C')
elif Score ⩾ 60 :
  print ('Grade is D')
else :
  print('Maaf kamu harus mengulang')
```

snappify.com

Code no 1 limit score if it have values more than 100 said ‘maaf kamu harus mengulang’ while in number 2 it will be printed score ‘A’ although two of the python have the same purpose it will be evaluated differently

Good and Bad Example

Good and Bad Example



Bad Example

Complex Code

This code harder to understand and it will be took more effort to maintain

Good Example

Break Down code

Code will be easier to maintenance if we break the code

```python
# Bad example:
def process_data(data):
    # Complex data processing code
    # Step 1: ...
    # Step 2: ...
    # Step 3: ...
    # Step 4: ...
    # Step 5: ...
    return transformed_data, repor
```

```python
# Good example:
def preprocess_data(data):
    # Code for data preprocessing
    # ...
    return transformed_data

def clean_data(data):
    # Code for data cleaning
    # ...
    return cleaned_data
```

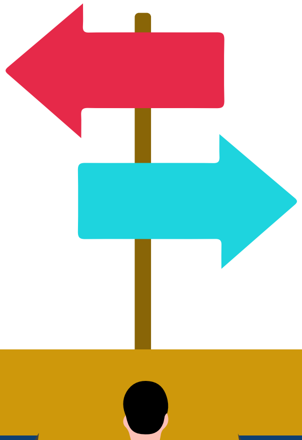
Good and Bad Example

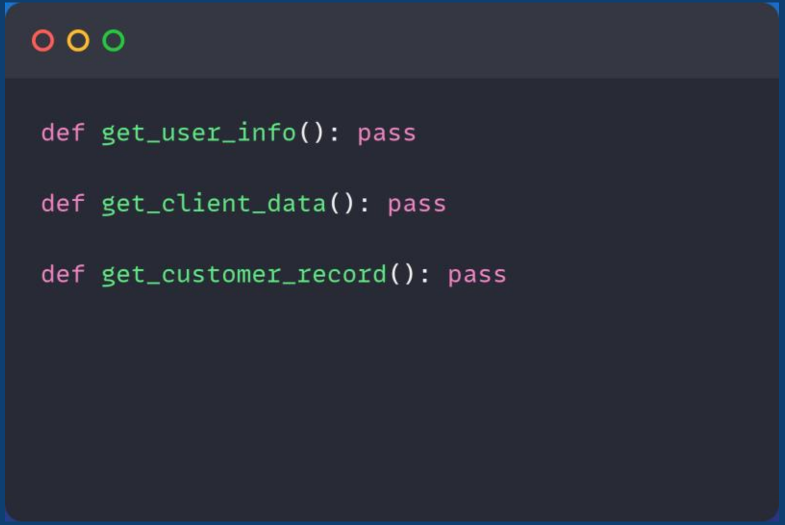
Bad Example

Here we use three different names for the same underlying entity

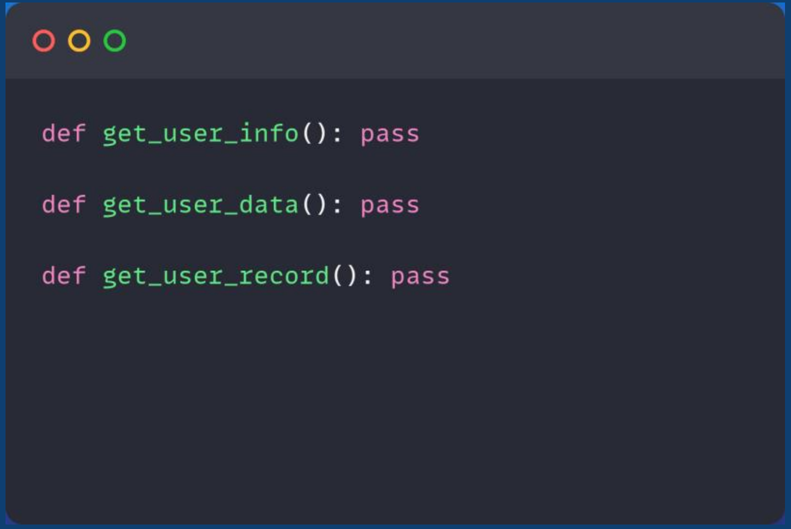
Good Example

If the entity is the same, you should be consistent in referring to it in your functions





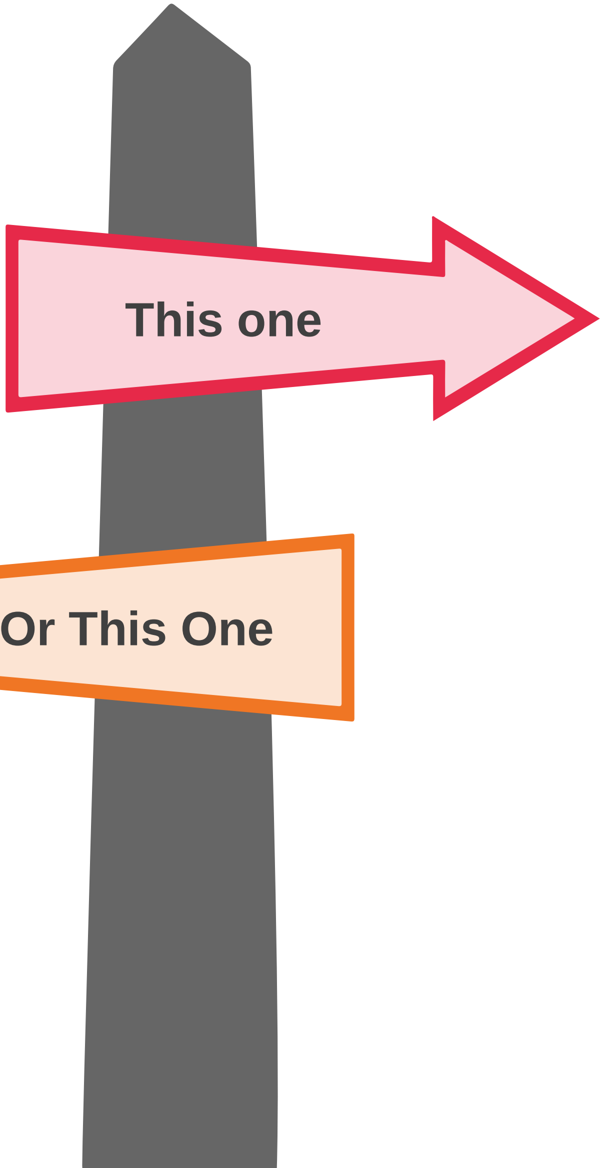
```
def get_user_info(): pass

def get_client_data(): pass

def get_customer_record(): pass
```

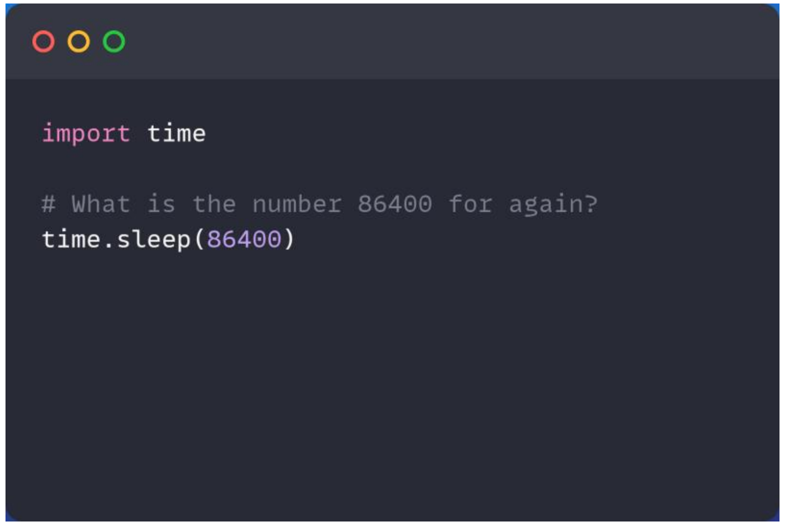


```
def get_user_info(): pass

def get_user_data(): pass

def get_user_record(): pass
```

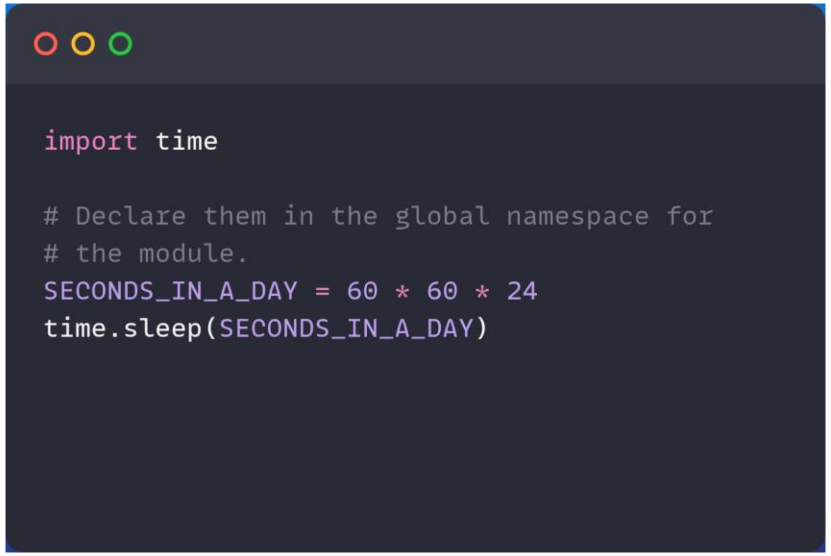
Which one is more better

It's important that the code we do write is readable and searchable. By not naming variables that end up being meaningful for understanding our program, we hurt our readers. Make your names searchable.





```python
import time

# What is the number 86400 for again?
time.sleep(86400)
```



```python
import time

# Declare them in the global namespace for
# the module.
SECONDS_IN_A_DAY = 60 * 60 * 24
time.sleep(SECONDS_IN_A_DAY)
```

Conclusion

01

Easy to Maintenance
Code with easier to read makes it easier to use and understand by another people in our teammates and also for yourself

02

Strength to face Bug
code which is easier to read and use makes it more strength with bug because this code process doesn't take more time

03

Easy to Develop
code that modular and efficient makes it easier to develop and combines with another features and application, doesn't take many PC RAM

04

The cleaner data, the more Advantage your level in coding

THANK YOU

Insert the Subtitle of Your Presentation