1.Use meaningful names for variables, functions, and classes:

```
# Bad example
x = 10
y = 20
def func(a, b):
    return a + b

# Good example
width = 10
height = 20
def calculate_area(length, width):
    return length * width
```

---

2. Write small and focused functions:

```
# Bad example
def process_data(data):
    # long and complicated function that does multiple things
    ...

# Good example
def clean_data(data):
    # function that cleans the data
    ...

def analyze_data(data):
    # function that analyzes the data
    ...

def plot_data(data):
    # function that plots the data
```

---

3. Use comments sparingly and only when necessary:

```
# Bad example
# Loop through the list and print each element
for i in range(len(my_list)):
    print(my_list[i])

# Good example
for element in my_list:
    print(element)
```

---

4. Follow the PEP 8 style guide:

```
# Bad example
def myFunction (argument1,argument2):
    result=argument1+argument2
    return result

# Good example
def my_function(argument1, argument2):
    result = argument1 + argument2
    return result
```

---
5. Use docstrings to document functions and classes:

```
def calculate_area(length, width):
    """Calculate the area of a rectangle.

    Args:
        length (float): The length of the rectangle.
        width (float): The width of the rectangle.

    Returns:
        float: The area of the rectangle.
    """
    return length * width
```

---
6. Write tests for your code:

```
import unittest

class TestCalculation(unittest.TestCase):
    def test_area_calculation(self):
        self.assertEqual(calculate_area(10, 20), 200)

if __name__ == '__main__':
    unittest.main()
```

---

7. Use exceptions to handle errors:

```
# Bad example
result = 0
if b != 0:
    result = a / b

# Good example
try:
    result = a / b
except ZeroDivisionError:
    result = None
```

---
8. Use constants or configuration files instead of hardcoding values:

```
# Bad example
def calculate_price(quantity):
    price_per_item = 5.0
    return quantity * price_per_item

# Good example
PRICE_PER_ITEM = 5.0

def calculate_price(quantity):
    return quantity * PRICE_PER_ITEM
```

---
9. Write simple and readable code:

```
# Bad example
result = a + (b * c) / (d ** e)

# Good example
term1 = b * c
term2 = d ** e
result = a + term1 / term2
```

---

10. Use built-in functions and libraries:

```
# Bad example
my_list = [1, 2, 3, 4, 5]
result = []
for item in my_list:
    result.append(item ** 2)

# Good example
my_list = [1, 2, 3, 4, 5]
result = list(map(lambda x: x ** 2, my_list))
```