The good (correct and proper style)

```
def is_prime(number):
    """Primality test by trial division."""
    if number < 2:
        return False
    if number == 2:
        return True
    if number % 2 == 0:
        return False
    for candidate in range(3, int(sqrt(number)) + 1):
        if number % candidate == 0:
            return False
    return True
```

```
def reverse(string):
    """Revert the order of characters in <string>."""
    result = ""
    for character in string:
        result = character + result
    return result
```

```
class Dragon:
    """Represents a dragon in a fantasy computer game."""
    def __init__(self, param_name, param_level, param_life):
        self.name = param_name
        self.level = param_level
        self.life = param_life

object_smaug = Dragon("Smaug", 84, 112233)
object_norbert = Dragon("Norbert", 11, 2048)
```

The bad (syntactic, semantic, or logical error)

```
def sum_while(how_many):
    """Sum the first <how_many> natural numbers."""
    result = 0
    while how_many > 0:
        result = result + how_many
    how_many = how_many - 1
    return result
```

```
def recursive_list_sum(numbers):
    return numbers[0] + list_sum(numbers[1:])
```

```
def find_maximum(lst):
    maximum = lst[0]
    for number in lst:
        if number < maximum:
            maximum = number
    return maximum
```

```
class Player:
    """Represents the player's character."""
    def __init__(self, name, health_max):
        self.name = name
        self.health = health_max
        self.health_max = health_max
        self.level = 1


def level_up():
    """Level up the player."""
    self.level += 1
    self.health_max += 1
    self.health += health_max
    print("{} feels stronger!".format(self.name))
```

The ugly (correct but improper style)

```
def can_buy_beer(persons_age):
    if persons_age >= 18:
        return True
    else:
        return False
```

```
def find_maximum(lst):
    maximum = lst[0]
    for index in range (1, len(lst)):
        if lst[index] > maximum:
            maximum = lst[index]
    return maximum
```

```
def letter_distribution_analysis(string):
    dictionary = {}
    for letter in string:
        if letter not in dictionary: dictionary[ letter ] = 1
        else: dictionary[ letter ] += 1
    print_dictionary(dictionary)
```