Electronics
Good Coding Style

Terry Sturtevant

Wilfrid Laurier University

August 23, 2018
Good Coding Style

- Good coding style makes programs more readable.
- It minimizes the use of comments.
- It makes code more easily re-usable.
Good Coding Style

- Good coding style makes programs more readable
Good Coding Style

- Good coding style makes programs more readable
  It minimizes the use of comments
Good Coding Style

- Good coding style makes programs more readable
  - It minimizes the use of comments
  - It makes code more easily re-usable
Good coding style tips:

1. Use consistent case to distinguish variables, constants, etc.
2. Use consistent device prefixes in names
3. Create self-explanatory variable and function names
4. Don't use magic numbers
Good coding style tips:

1. Use consistent case to distinguish variables, constants, etc.
Good coding style tips:

1. Use consistent case to distinguish variables, constants, etc.
2. Use consistent device prefixes in names
Good coding style tips:

1. Use consistent case to distinguish variables, constants, etc.
2. Use consistent device prefixes in names
3. Create self-explanatory variable and function names
Good coding style tips:

1. Use consistent case to distinguish variables, constants, etc.
2. Use consistent device prefixes in names
3. Create self-explanatory variable and function names
4. Don’t use *magic numbers*
GPIO test sample code

```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BORDER)
GPIO.setup(12, GPIO.OUT)
while True:
    GPIO.output(12, False)
    time.sleep(1)
    GPIO.output(12, True)
    time.sleep(1)
```

Simple GPIO test

Terry Sturtevant
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(12, GPIO.OUT)

while True:
    GPIO.output(12, False)
    time.sleep(1)
    GPIO.output(12, True)
    time.sleep(1)
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BORDER)
GPIO.setup(12, GPIO.OUT)

while True:
    GPIO.output(12, False)
    time.sleep(1)
    GPIO.output(12, True)
    time.sleep(1)

Simple GPIO test
GPIO.setup(12, GPIO.OUT)
→ set up one pin to blink an LED
Good Coding Style

Example

```
GPIO.setup(12, GPIO.OUT)
→ set up one pin to blink an LED

The Raspberry Pi has two numbering schemes for GPIO pins; BOARD and BCM (Broadcom).
```
GPIO.setup(12, GPIO.OUT) → set up one pin to blink an LED

The Raspberry Pi has two numbering schemes for GPIO pins; BOARD and BCM (Broadcom).

GPIO.setmode(GPIO.BOARD)
Good Coding Style

**Example**

```python
GPIO.setup(12, GPIO.OUT)

→ set up one pin to blink an LED
```

The Raspberry Pi has two numbering schemes for GPIO pins; BOARD and BCM (Broadcom).

```python
GPIO.setmode(GPIO.BOARD)

This line was earlier.
```
The Raspberry Pi has two numbering schemes for GPIO pins; BOARD and BCM (Broadcom).

```
GPIO.setmode(GPIO.BOARD)
```

This line was earlier.

```
LED_PIN = 12
```

```
GPIO.setup(12, GPIO.OUT)
→ set up one pin to blink an LED
```

Terry Sturtevant
Good Coding Style

Example

```
GPIO.setup(12, GPIO.OUT)
→ set up one pin to blink an LED
```

The Raspberry Pi has two numbering schemes for GPIO pins; BOARD and BCM (Broadcom).

```
GPIO.setmode(GPIO.BOARD)
This line was earlier.
```

```
LED_PIN = 12
(The number depends on the numbering scheme.)
```
**Good Coding Style**

Example

```python
GPIO.setup(12, GPIO.OUT)
→ set up one pin to blink an LED
```

The Raspberry Pi has two numbering schemes for GPIO pins; BOARD and BCM (Broadcom).

```python
GPIO.setmode(GPIO.BOARD)
```

This line was earlier.

```python
LED_PIN = 12
```

*(The number depends on the numbering scheme.)*

Also, the delay is in seconds
Good Coding Style

Example

```python
GPIO.setup(12, GPIO.OUT)
→ set up one pin to blink an LED

The Raspberry Pi has two numbering schemes for GPIO pins; BOARD and BCM (Broadcom).

```python
GPIO.setmode(GPIO.BOARD)

This line was earlier.

```python
LED_PIN = 12

(The number depends on the numbering scheme.)

Also, the delay is in seconds

```python
TIME_SLEEP SECONDS = 1
```
**Example**

```python
GPIO.setup(12, GPIO.OUT)  
→ set up one pin to blink an LED
```

The Raspberry Pi has two numbering schemes for GPIO pins; BOARD and BCM (Broadcom).

```python
GPIO.setmode(GPIO.BOARD)
```

This line was earlier.

```python
LED_PIN=12
```

(The number depends on the numbering scheme.)

Also, the delay is in seconds

```python
TIME_SLEEP_SECONDS=1
```

This is much clearer.
A couple of functions can help.
A couple of functions can help.

```python
def turnOnLED():
    GPIO.output(LED_PIN,True)
return
```
A couple of functions can help.

```python
def turnOnLED():
    GPIO.output(LED_PIN,True)
    return

def turnOffLED():
    GPIO.output(LED_PIN,False)
    return
```
GPIO test sample code

```python
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BORD)
LED_PIN=12
TIME_SLEEP_SECONDS=1
def turnOnLED():
    GPIO.output(LED_PIN, True)
    return
def turnOffLED():
    GPIO.output(LED_PIN, False)
    return
```

GPIO test sample code

```python
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
LED_PIN=12
TIME_SLEEP SECONDS=1

def turnOnLED():
    GPIO.output(LED_PIN, True)
    return

def turnOffLED():
    GPIO.output(LED_PIN, False)
    return
```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BORDER)
LED_PIN=12
TIME_SLEEP_SECONDS=1

def turnOnLED():
    GPIO.output(LED_PIN, True)
    return

def turnOffLED():
    GPIO.output(LED_PIN, False)
    return

Definition section
From the definition section:
From the definition section:

- Using a different pin requires only one change in the code.
From the definition section:

- Using a different pin requires only one change in the code.
- Changing the time requires only one change in the code.
From the definition section:

- Using a different pin requires only one change in the code.
- Changing the time requires only one change in the code.
  (Also, it's clear what *units* are involved.)
From the definition section:

- Using a different pin requires only one change in the code.
- Changing the time requires only one change in the code.  
  (Also, it's clear what *units* are involved.)
- I can search through all my programs on **PIN**
From the definition section:

- Using a different pin requires only one change in the code.
- Changing the time requires only one change in the code. (Also, it’s clear what units are involved.)
- I can search through all my programs on PIN
- By making the LED definition I can search through all my programs for this specific type of device, i.e. “LED”
Good Coding Style

Example

GPIO test sample code

```python
GPIO.setup(LED_PIN, GPIO.OUT)
while True:
    turnOffLED()
time.sleep(TIME_SLEEP_SECONDS)
    turnOnLED()
time.sleep(TIME_SLEEP_SECONDS)
```

Operation section (Note: You could also have separate on and off times, etc.)
GPIO test sample code

```python
GPIO.setup(LED_PIN, GPIO.OUT)

while True:
    turnOffLED()
    time.sleep(TIME_SLEEP_SECONDS)
    turnOnLED()
    time.sleep(TIME_SLEEP_SECONDS)
```

(Note: You could also have separate on and off times, etc.)
GPIO test sample code

```python
GPIO.setup(LED_PIN, GPIO.OUT)

while True:
    turnOffLED()
    time.sleep(TIME_SLEEP_SECONDS)
    turnOnLED()
    time.sleep(TIME_SLEEP_SECONDS)
```

Operation section
**GPIO test sample code**

```python
GPIO.setup(LED_PIN, GPIO.OUT)

while True:
    turnOffLED()
    time.sleep(TIME_SLEEP SECONDS)
    turnOnLED()
    time.sleep(TIME_SLEEP SECONDS)
```

**Operation section** (Note: You could also have separate on and off times, etc.)