# TDD with Python unittest
for
embedded C

Ben6

2012-08-28

www.juluos.org

Agenda

- **TDD是什麼?**
- **為什麼使用TDD?**
- **python unittest**
  - C & Embedded C
- **Real Practice**

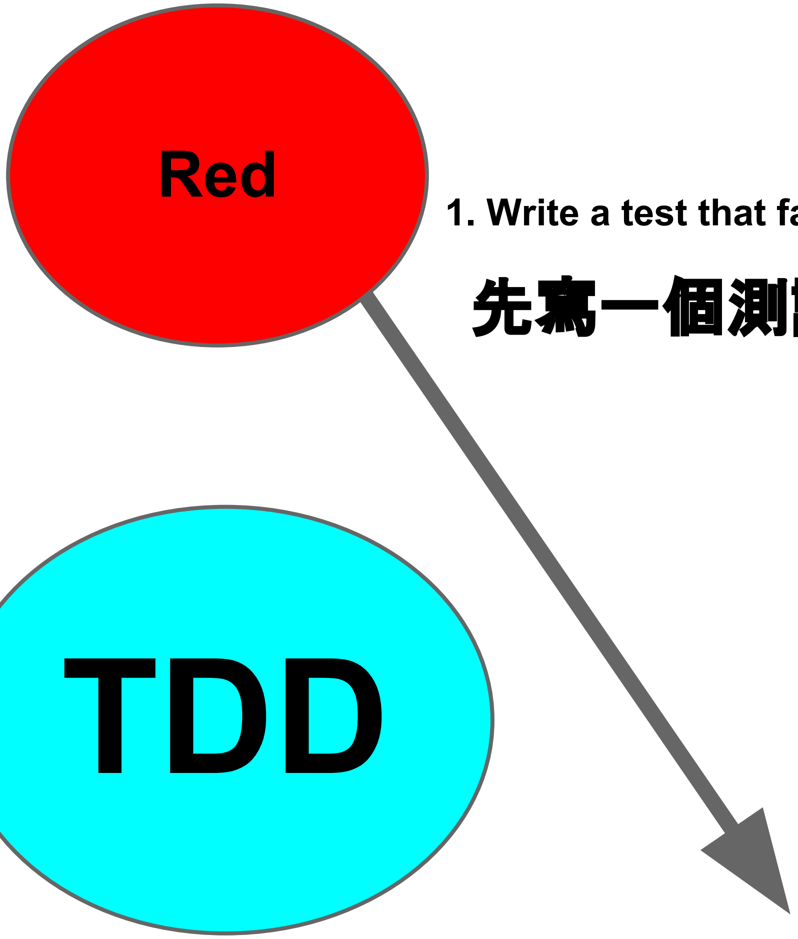
1. BOS 實際導入
2. Dummy test and LED Driver

TDD是什麼?

What's TDD?

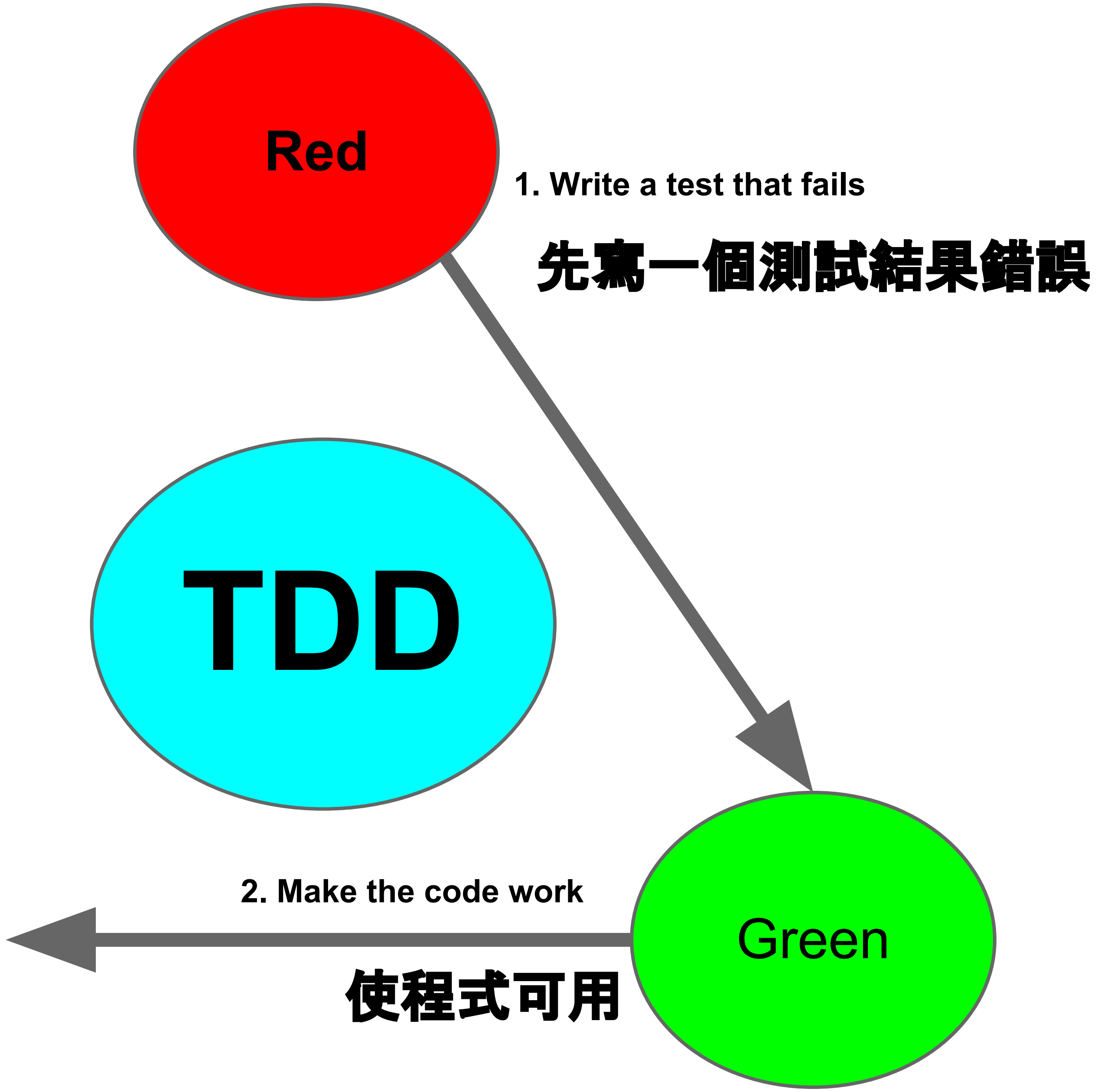
測試先行

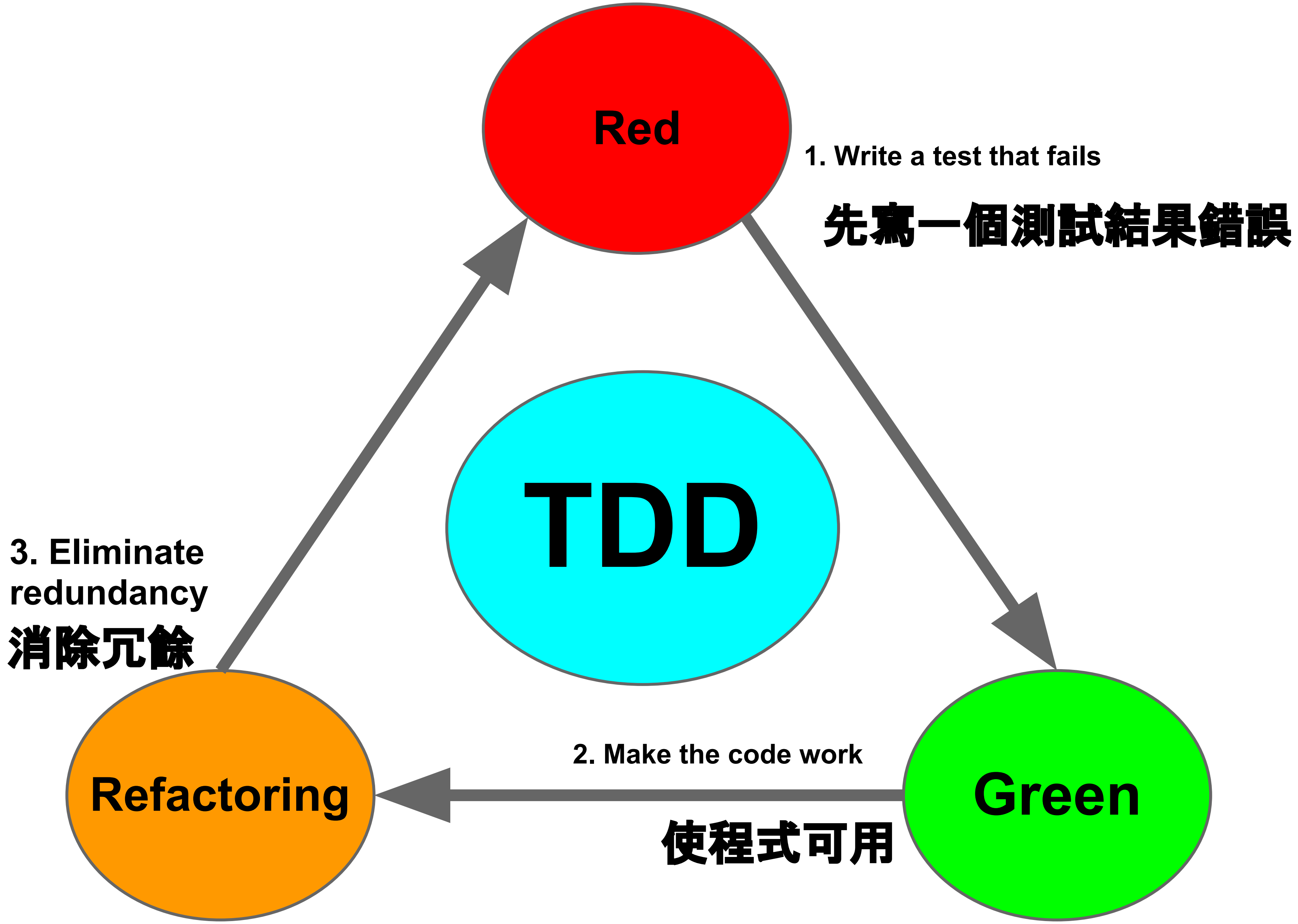
意思就是, 在開發程式時,
要以測試的角度來設計。

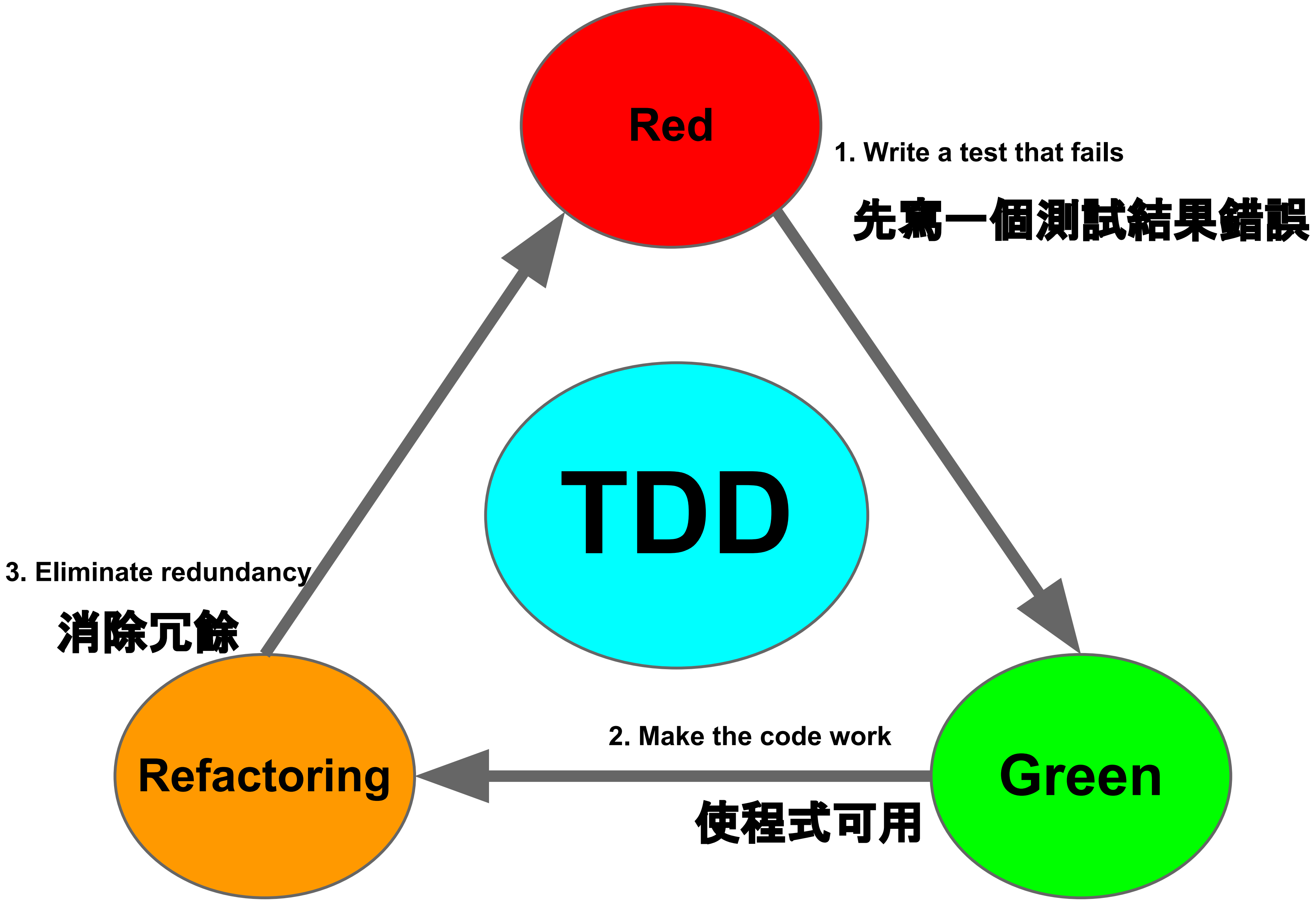


1. Write a test that fails

先寫一個測試結果錯誤


Red
1. Write a test that fails
先寫一個測試結果錯誤
TDD
2. Make the code work
使程式可用
Green


Red
1. Write a test that fails
先寫一個測試結果錯誤
TDD
Green
2. Make the code work
使程式可用
Refactoring
3. Eliminate redundancy
消除冗餘



TDD 的開發節奏 "Red, Green, Refactoring"

TDD的觀點

All code is guilty until proven innocent.

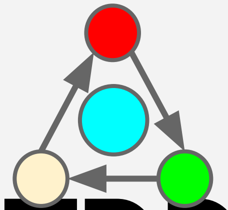
任何代碼都是有問題的，直到證明他無誤。

重構 (Refactoring)

Refactoring 是重構一詞，英文以現在進行式，意味，重構應該不間斷地持續進行。

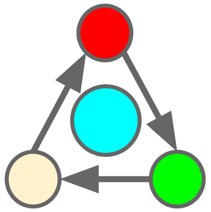
三項關鍵技能

- 對壞代碼的嗅覺
- 對更好代碼的遠見
- 轉化代碼



請問你有使用TDD嗎？

若有，為什麼使用？



我為什麼使用TDD?

為了不浪費生命在重複手動測試的事務上

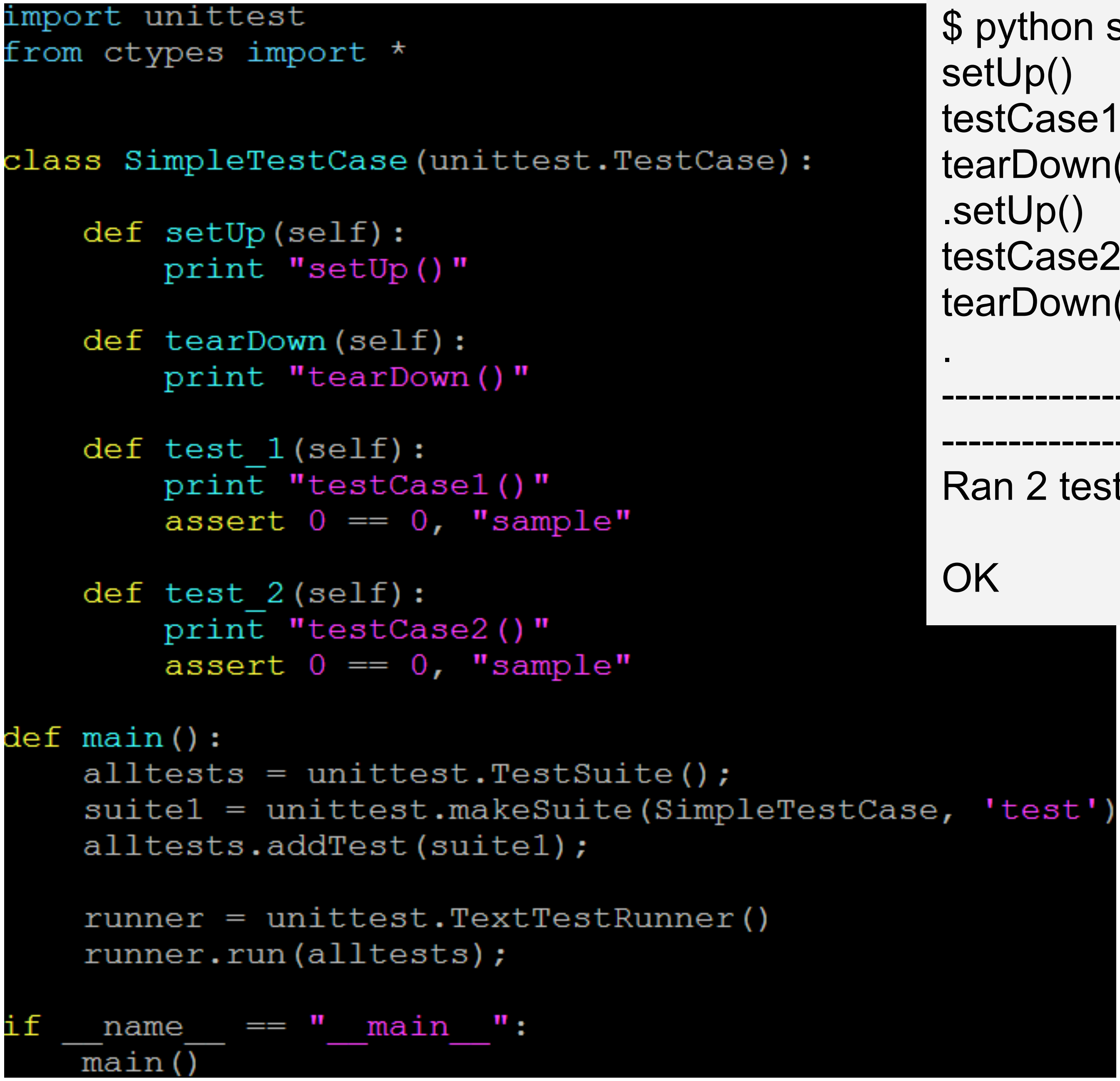
為了有自信重構程式使程式碼更易維護及理解

為了利用測試腳本將產品規格更清除的規範

更多...

Python unittest

Python unittest



```python
import unittest
from ctypes import *


class SimpleTestCase(unittest.TestCase):

    def setUp(self):
        print "setUp()"

    def tearDown(self):
        print "tearDown()"

    def test_1(self):
        print "testCase1()"
        assert 0 == 0, "sample"

    def test_2(self):
        print "testCase2()"
        assert 0 == 0, "sample"

def main():
    alltests = unittest.TestSuite();
    suite1 = unittest.makeSuite(SimpleTestCase, 'test')
    alltests.addTest(suite1);

    runner = unittest.TextTestRunner()
    runner.run(alltests);

if __name__ == "__main__":
    main()
```

```
$ python sample.py
setUp()
testCase1()
tearDown()
.setUp()
testCase2()
tearDown()
.
----------------------------------------
----------------------------
Ran 2 tests in 0.000s

OK
```

```python
import random
import unittest

class TestSequenceFunctions(unittest.TestCase):

    def setUp(self):
        self.seq = range(10)

    def test_shuffle(self):
        random.shuffle(self.seq)
        self.seq.sort()
        self.assertEqual(self.seq, range(10))

        self.assertRaises(TypeError, random.shuffle, (1,2,3))

    def test_choice(self):
        element = random.choice(self.seq)
        self.assertTrue(element in self.seq)

    def test_sample(self):
        with self.assertRaises(ValueError):
            random.sample(self.seq, 20)
        for element in random.sample(self.seq, 5):
            self.assertTrue(element in self.seq)

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

Python 與 C

c library: libfoo.so

```
int foo(unsigned int r);
```

python ctypes sample

```
from ctypes import *
foolib = CDLL.LoadLibrary('libfoo.so')
r = foolib.foo(c_uint(5))
```

Embedded C

Definitions

- Related to hardware environment

- **Ex: A driver**
    - LED
    - Netowrk Interface Card
    - VGA Card
    - self test
    - ...

ctypes

ctypes type	C type	Python type
c_bool	**_Bool**	bool (1)
c_char	**char**	1-character string
c_wchar	**wchar_t**	1-character unicode string
c_byte	**char**	int/long
c_ubyte	**unsigned char**	int/long
c_short	**short**	int/long
c_ushort	**unsigned short**	int/long
c_int	**int**	int/long
c_uint	**unsigned int**	int/long
c_long	**long**	int/long
c_ulong	**unsigned long**	int/long
c_longlong	**__int64** or **long long**	int/long
c_ulonglong	**unsigned __int64** or **unsigned long long**	int/long
c_float	**float**	float
c_double	**double**	float
c_longdouble	**long double**	float
c_char_p	**char *** (NUL terminated)	string or **None**
c_wchar_p	**wchar_t *** (NUL terminated)	unicode or **None**
c_void_p	**void ***	int/long or **None**

sample ctypes

```
>>> from ctypes import *
>>> p = create_string_buffer(3)
>>> print sizeof(p), repr(p.raw)
3 '\x00\x00\x00'

>>> p = create_string_buffer("Hello")
>>> print sizeof(p), repr(p.raw)
6 'Hello\x00'

>>> print repr(p.value)
'Hello'
```

sample ctypes

```
>>> from ctypes import *
>>> p = create_string_buffer("Hello", 10)
>>> print sizeof(p), repr(p.raw)
10 'Hello\x00\x00\x00\x00\x00'

>>> p.value = "Hi"
>>> print sizeof(p), repr(p.raw)
10 'Hi\x00lo\x00\x00\x00\x00\x00'
```

python cdll in different platform

```
from ctypes import *
import sys

platform = sys.platform
if sys.platform == "cygwin":
    libc = cdll.LoadLibrary("/bin/cygwin1.dll")
else:
    libc = CDLL('libc.so.6')
```

python cdll in different platform (cont.)

```
import sys
from math import log

def is64bit():
    return log(sys.maxsize, 2) == 63

arch=32
 if is64bit():
    arch = 64
```

案例研討(一)

BOS 實際導入

blibc pytest @ github

Makefile for unittest blibc

```
CFILES= ../../blibc/itoa.c
OBJS=$(CFILES:.c=.o)
CFLAGS= -I ../../include
BLIBC_SHARED = libbosc.so
all: $(BLIBC_SHARED)
	python alltests.py

%.o :%.c
	$(CC) -c $< -o $@ $(CFLAGS)

$(BLIBC_SHARED): $(OBJS)
	$(CC) -shared -o $@ $^

clean:
			rm -f $(BLIBC_SHARED) $(OBJS)
```

alltests.py

```
import unittest,os
from ctypes import *
app_path = os.path.dirname(os.path.abspath(__file__))
libpathname = os.path.join(app_path, "./libbosc.so")
bc = CDLL(libpathname);
class BOSTest_atoi(unittest.TestCase):
    s = (c_byte*9)()
    c = 427
    def test_atoi(self):
        # ... next page

def suite_blibc():
    bosTestSuite = unittest.makeSuite(BOSTest_atoi, 'test')
    return bosTestSuite

def main():
    suite1 = suite_blibc()
    alltests = unittest.TestSuite((suite1))
    runner = unittest.TextTestRunner()
    runner.run(alltests);
```

test 1 in alltests.py

```
def test_atoi(self):
  #  const char *itohex(uint32_t c, char *s, int size, int upper)
  b = bc.itohex(self.c, self.s, c_int(9), c_int(0))
  hex_str = string_at(self.s)
  assert hex_str == "000001ab", "atoi padding hex string"
  assert string_at(b) == "1ab"," atoi incorrect no-zero hex 
padding"
```

test 2 in alltests.py

```
 def test_auto_with_upper(self):
   upper_hex = bc.itohex(self.c, self.s, c_int(9), c_int(1))
   assert string_at(upper_hex) == "1AB", \
      " atoi incorrect no-zero upper hex padding"
```

實戰演練（一）

Dummy LED(s) Driver

實戰演練(一)發生錯誤的單元測試

空的測試腳本包含:

- a dummy python test (alltests.py)
  - use python ctypes to load lib function
- a sample Makefile to create shared library for python unitest

```
git clone -b tdd_leds_sample git@github.com:benwei/JuluOS.git tdd_leds
$ cd tdd_leds/tests
$ git checkout -b your_tdd1 a325e85366da5d0d3735863aa983a27700829a28
```

```
$ make
python alltests.py
E
======================================================================
ERROR: test_turn_onoff (__main__.BOSTest_led)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "alltests.py", line 21, in test_turn_onoff
    led.turn_on(c_char(0))
TypeError: one character string expected

----------------------------------------------------------------------
Ran 1 test in 0.000s

FAILED (errors=1)
```

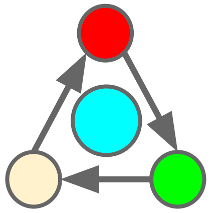
實戰演練(一)使程式通過測試

1. 建立 ../drivers/led.c
2. 並使之通過測試

結果如下:

```
$ make
cc -c ../drivers/led.c -o ../drivers/led.o -I ../inc
cc -shared -o libled.so ../drivers/led.o
python alltests.py
after turnon 1
after turnoff 0
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```



實戰演練(二)

Refactoring

及

TDD使用Mock LED(s) Driver

實際演練(二)LED Driver with TDD

Make a test that fails

- Create dummy test function
- Create a Mock LED Device for monitoring
- Add LED turn_on/off api to test function
- Check expect result of LED state

Make code work

- write the operation code for LED

Refactoring

- refined wording, duplicated code, naming ...

Mock LED(s) Device

- 可是一個記憶體位置
- 可為表示狀態的指示器
- 一個簡化後的虛擬裝置

Ex:

```
 uint mock_led = 0;
 led_plug(&mock_led, id_1);
```

Mock Object 在開發過程十分重要，因為當測試環境愈單純，將會更易於自動化測試的建構。http://en.wikipedia.org/wiki/Mock_object

實際演練(二)

- 練習題
  - 擴充為十個LED 燈，可以同時點亮，也可單獨開關其中之一。

完成後原始碼:

```
$ git checkout tdd_leds_sample
$ git checkout 453b80514da4793f6e608343742ba89bb870dcd6
```

實際演練(二)TDD完成輸入畫面

```
git clone -b tdd_leds_sample git@github.com:benwei/JuluOS.git tdd_leds
$ cd tdd_leds/tests
tests$ ls
alltests.py  Makefile
tests$ make
cc -c ../drivers/leds.c -o ../drivers/leds.o -I ../inc
cc -shared -o libleds.so ../drivers/leds.o
python alltests.py
after turnon 1
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
```

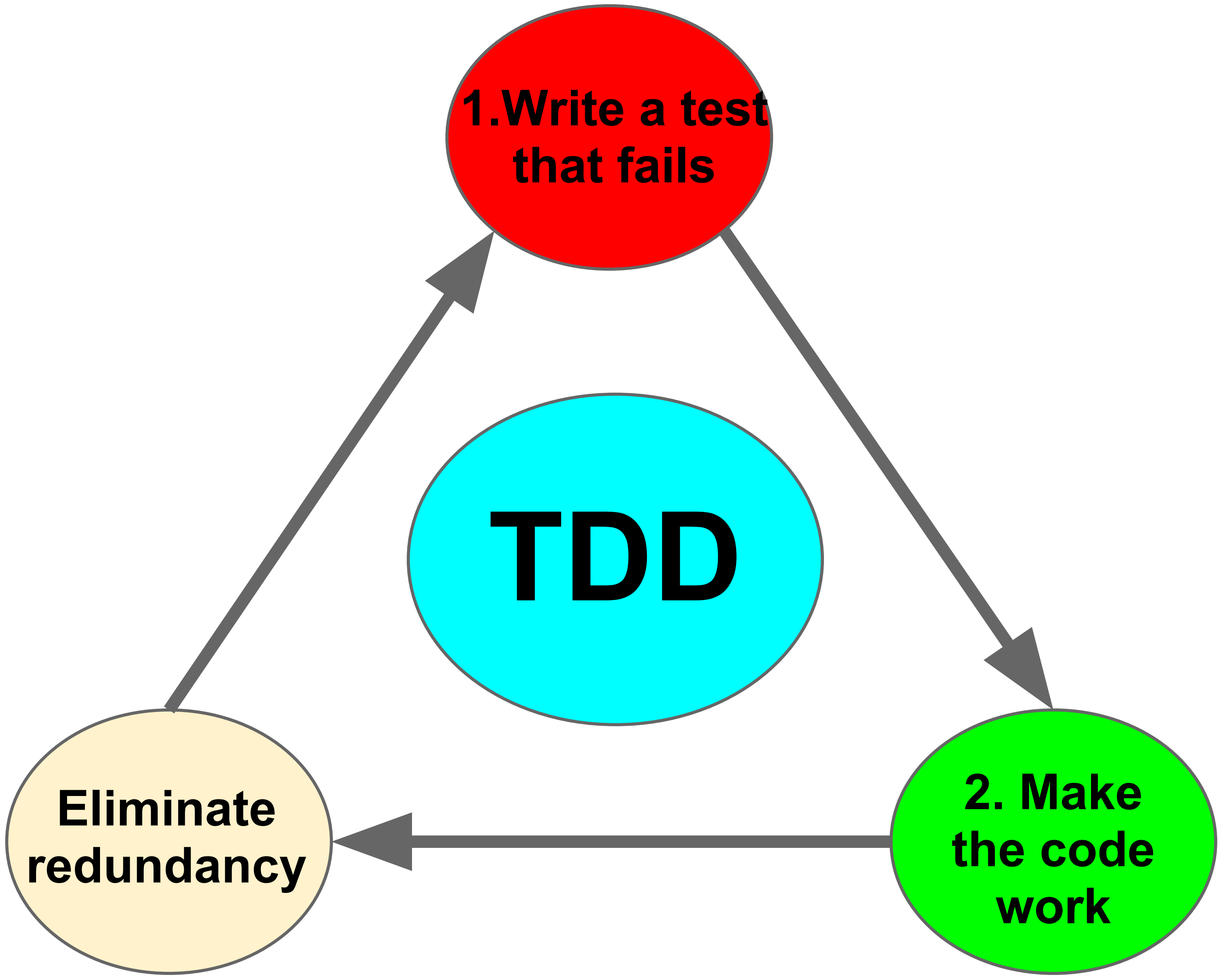
實際演練(二)延伸練習

- 使用者可由外部傳入LED Mock Objects
- 使用者可使用不同顏色LED
- 使用者可設任一LED燈亮1~10秒
    - 使用python + timer check 來確定其執行時間

Conclusion

TDD 好處

較少的bug
減少手動重複測試的時間
文件檔案
容易改善設計，因不將設計的耦合降低，將無法進行單元測試
根據測式程式後的記錄，能很清楚知道目前的進度。
很關卡遊戲，設置問題，填入解答(可不斷精鍊)，獲得完成時的成就感。



The mantra of TDD is "Red, green, refactoring"

Ending - Test-driven development

a programming technique that
requires you to write actual code
automated test code simultaneously

ensures you **test your code**
enables you to **retest** your code **quickly** and **easily**, since it's **automated**

Q&A?

Maintain

write a test fails

TDD?

Refactor

duplicate

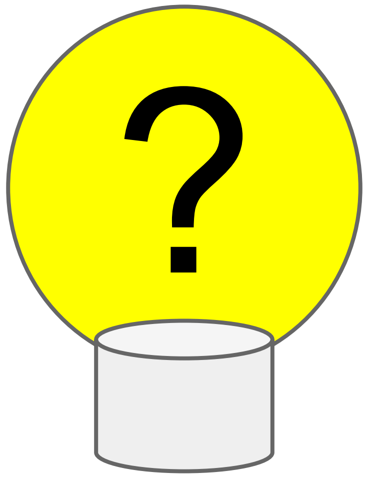
Backlogs

References

Mock Objects for TDD

- Why and When to Use Mock Objects
- library unittest mock
- wikipedia.org/wiki/Mock_object

How python with cli?



cli - interface

- input via arguments

- cli output
  - update the specific files
  - exit code
  - console output

- use unittest program testing cli output

ctypes use global variable in lib

```
led = CDLL("./libled.so")

v = c_uint.in_dll(led, "_led")
print v.value()
```

Reference:
http://docs.python.org/library/ctypes.html#accessing-values-exported-from-dlls

!

筆者的意見:儘量不要直接存取全域變數;經由封裝函數介面來使用,
如:uint get_led(void) { return _led; }

loremipsum

A Lorem Ipsum text generator

pypi.python.org/pypi/loremipsum/1.0.2

```
>>> from loremipsum import Generator
>>>
>>> sample = file('data/sample.txt').read()
>>> dictionary = file('data/dictionary.txt').read().split()
>>>
>>> g = Generator(sample, dictionary)
>>> g.generate_sentence() #doctest: +ELLIPSIS
(...)
>>>
```