

---

# Python 單元測試 Unit Test

臺北科技大學資訊工程系

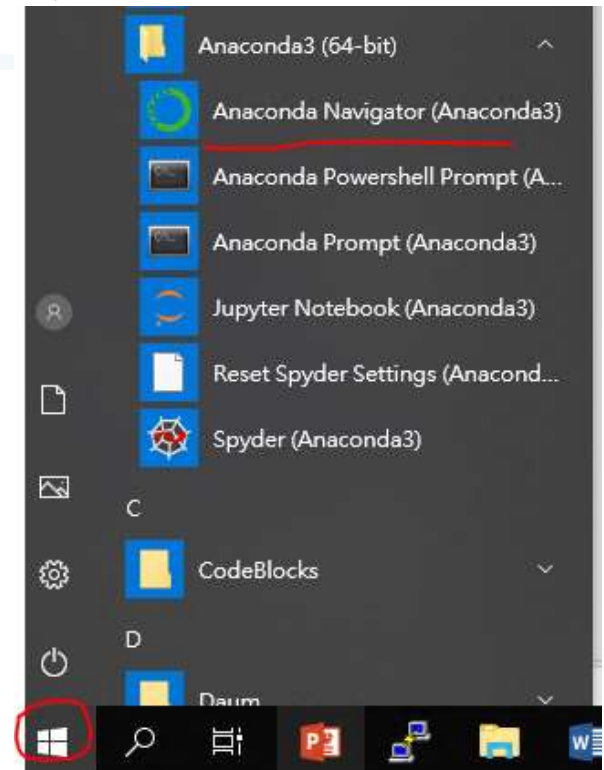
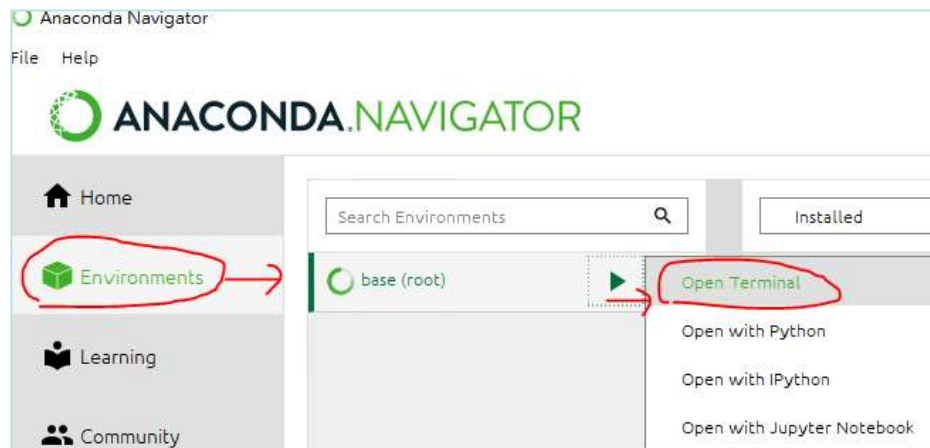
---

# 測試程式

- 測試程式/作業正確性，找出程式錯誤。
  - **設計** 測試案例(test case)，滿足問題/需求描述的各種條件/情境
  - **撰寫** 測試驅動程式(test driver)、測試方法(method)
  - **執行** 測試驅動程式，使得主要程式通過這些測試案例。
- 單元測試(Unit Test)概念
  - 程式設計師，測試自己所寫的程式模組是否有錯誤。
  - 撰寫測試方法(method)，驗證某功能在某測試案例，程式如預期運作。
- 單元測試(Unit Test)重點
  - 設計**足夠多**的測試案例，滿足各種問題條件
  - **正確**：測試主程式，讓結果正確
  - **完整**：測試主程式，使涵蓋度達到100%

# 安裝 coverage 套件

- ❑ 開始: Anaconda: Anaconda Navigator
- ❑ Environments: Open Terminal



- ❑ pip install coverage

○ pip install --index-url=https://pypi.org/simple/ --trusted-host pypi.python.org coverage

```
(base) C:\Users\jykuo>pip install --index-url=https://pypi.org/simple/ --trusted-host pypi.python.org coverage
Looking in indexes: https://pypi.org/simple/
Collecting coverage
  Obtaining dependency information for coverage from https://files.pythonhosted.org/packages/52/07/215f1373eeded0ba34fed
c7d61ff72ae771705692dfead54e74a97bbbfb/bb/coverage-7.3.2-cp311-cp311-win_amd64.whl.metadata
  Downloading coverage-7.3.2-cp311-cp311-win_amd64.whl.metadata (8.3 kB)
  Downloading coverage-7.3.2-cp311-cp311-win_amd64.whl (203 kB)
----- 203.3/203.3 kB 1.1 MB/s eta 0:00:00
Installing collected packages: coverage
Successfully installed coverage-7.3.2
```

# 題目

## □ 輸入：

- 體重(公斤), 身高(公尺)
- **合法範圍**：體重( $3 \leq \sim \leq 200$ )，身高( $0.50 \leq \sim \leq 2.50$ )。
- **公式**：BMI = 體重(公斤) / 身高<sup>2</sup>(公尺平方)

## □ 輸出：

- 身高、體重輸入範圍錯誤，輸出-1。
- **正常輸入範圍**：輸出**兩位小數**，第三位小數後面**去尾**(無條件捨去)
- **【輸出兩位小數，四捨五入】**

# 撰寫程式

## □ 撰寫測試BMI程式

- 造出 **c:\TEST** 目錄，將以下程式存入此目錄
- 程式名稱 **mybmi.py**

```
def computeBMI(kg, M):  
    if kg > 200 and M > 2.5:                #錯誤，範圍判斷錯誤  
        return -1  
    if kg <= 2 or M <= 0.05:  
        return -1  
    #BMI = round(kg/(M*M), 2)                #錯誤，四捨五入取兩位小數  
    BMI = ((100*kg/(M*M))//1)/100           #正確，去尾，乘100取整數，再除100取兩位小數  
    return BMI
```

# 設計測試案例

## □ 測試案例分類

### ○ Normal

- 輸入：52公斤，1.55公尺，
- BMI： $52(\text{公斤}) / (1.55 * 1.55)(\text{公尺平方}) = 21.64412$ (兩位小數 21.64)。

### ○ 錯誤身高

- 輸入：52公斤，0.05公尺
- 輸出：-1

### ○ 錯誤體重

- 輸入：2公斤，1.55公尺
- 輸出：-1

### ○ 錯誤身高、錯誤體重

- 輸入：500公斤，5公尺
- 輸出：-1

# 撰寫測試驅動程式

## □ 使用unittest (PyUnit)測試框架

○ 將以下程式 **mybmittest.py** 放入 **c:\TEST**

```
01 import unittest                # 匯入測試框架套件
02 #import coverage               # 匯入涵蓋度紀錄套件
03 import mybmi                   # 匯入 mybmi.py 內的程式
04 class TestBMI(unittest.TestCase): # 設計測試驅動程式類別
05     def test_computeBMIOK(self): # 設計測試主程式功能的方法，self 是物件本身
06         self.assertEqual(mybmi.computeBMI(52, 1.55), 21.64) # assertEquals(執行結果, 期望結果)
07
08
09
10
11
12
13
```

# 撰寫涵蓋度報表程式

## □ 使用unittest (PyUnit)測試框架

○ 將以下程式 **exec\_api.py** 放入 c:\TEST

```
01 # 使用 API 產生程式碼覆蓋率統計報告 exec_api.py
02 import coverage                      # 匯入涵蓋度紀錄套件
03 import unittest                     # 匯入測試框架套件
04 # 實體化一個涵蓋度紀錄物件
05 cov = coverage.coverage(branch=True, source=['mybmi'])
06 cov.start() # 啟動測試涵蓋度紀錄
07 suite = unittest.defaultTestLoader.discover("./", "mybmitest.py") # 載入測試套件
08 unittest.TextTestRunner().run(suite) # 執行測試套件組之測試
09 cov.stop()                          # 停止測試涵蓋度紀錄
10 cov.save()                          # 儲存測試涵蓋度資料
11 cov.report()                        # 命令列模式展示結果
12 cov.html_report(directory='cov')    # 製作測試涵蓋度結果報表，存放在 cov 子目錄
```



# 執行測試

- 測試結果，使用 spyder 打開 exec\_api.py，點擊執行

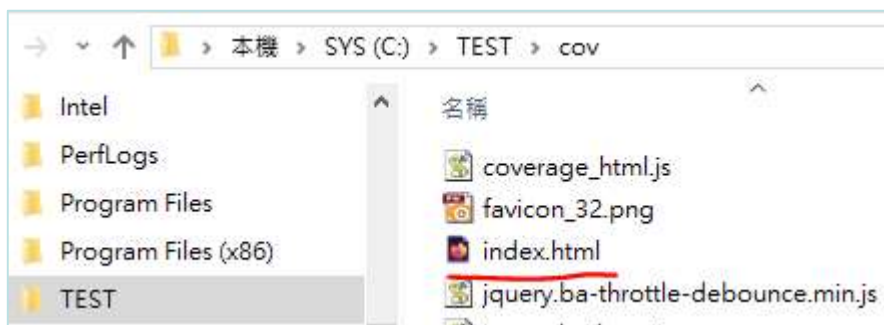
```
In [1]: runfile('C:/TEST/exec_api.py', wdir='C:/TEST')
.Name      Stmts  Miss Branch BrPart  Cover
-----
mybmi.py    7       2     4      2    64%
-----
TOTAL       7       2     4      2    64%
-----

Ran 1 test in 0.005s

OK
```

- 針對一個測試案例，程式測試正確，但涵蓋度64%

- 開啟檔案總管，執行(雙擊) index.html



# 執行測試

## ○ 測試涵蓋度(Coverage)/完整度 64%

Coverage report: 64%

Module ↑	statements	missing	excluded	branches	partial	coverage
mybmi.py	7	2	0	4	2	64%
<b>Total</b>	<b>7</b>	<b>2</b>	<b>0</b>	<b>4</b>	<b>2</b>	<b>64%</b>

coverage.py v6.0.2, created at 2021-10-25 14:49 +0800

## ○ 點選 mybmi.py，查看細節

- 紅色 return 沒有執行到
- if 只有執行部分條件判斷(沒有執行到False)

Coverage for mybmi.py : 64%

7 statements   5 run   2 missing   0 excluded   2 partial

```
1 def computeBMI(kg, M):
2     if kg>200 and M>2.5:                #錯誤，範圍判斷錯誤
3         return -1
4     if kg<=2 or M<=0.05:                #錯誤，範圍判斷錯誤
5         return -1
6     BMI = round(kg/(M*M),2)              #錯誤，四捨五入取兩位小數
7     #BMI = ((100*kg/(M*M))/1)/100        #正確，去尾，乘100取整數，再除100取兩位小數
8     return BMI
```

# Exercise測試問題

□ 1. 增修驅動程式，使主程式測試更完整(coverage 100%)

○ 測試案例 vs. 主程式

□ 2. 增加測試案例設計，使需求/問題測試更完整

○ 測試案例 vs. 需求/題目

# Exercise 測試問題(coverage 100%)

## □ 測試案例設計

### ○ 錯誤身高

- 輸入：52公斤，0.05公尺
- 輸出：-1

### ○ 錯誤體重

- 輸入：2公斤，1.55公尺
- 輸出：-1

## □ 1. 增修驅動程式，使 主程式測試 更完整(coverage 100%)

### ○ 【mybmittest.py】 驅動程式 07. 08. 加入

- `self.assertEqual(mybmi.computeBMI(52, 0.05), -1)`
- `self.assertEqual(mybmi.computeBMI(2, 1.55), -1)`

### ○ 測試案例 vs. 主程式

# Exercise測試問題(coverage 100%)

## ❑ 執行測試，未達100%

```
.Name      Stmts  Miss Branch BrPart  Cover
-----
mybmi.py    7      1      4      1    82%
-----
TOTAL       7      1      4      1    82%
```

Ran 1 test in 0.002s

OK

Coverage for **mybmi.py** : 82%

7 statements

6 run

1 missing

0 excluded

1 partial

```
1 def computeBMI(kg, M):
2     if kg>200 and M>2.5:                #錯誤，範圍判斷錯誤
3         return -1
4     if kg<=2 or M<=0.05:                #錯誤，範圍判斷錯誤
5         return -1
6     BMI = round(kg/(M*M),2)              #錯誤，四捨五入取兩位小數
7     #BMI = ((100*kg/(M*M))/1)/100        #正確，去尾，乘100取整數，再除100取兩位小數
8     return BMI
```

# Exercise測試問題(coverage 100%)

## □ 測試案例設計

### ○ 錯誤身高、錯誤體重

➤ 輸入：500公斤，5公尺

➤ 輸出：-1

## □ 1.增修驅動程式，使主程式測試更完整(coverage 100%)

### ○ 【mybmittest.py】 驅動程式09.加入

➤ self.assertEqual(mybmi.computeBMI(500, 5), -1)

.Name	Stmts	Miss	Branch	BrPart	Cover
-----	-----	-----	-----	-----	-----
mybmi.py	7	0	4	0	100%
-----	-----	-----	-----	-----	-----
TOTAL	7	0	4	0	100%

Ran 1 test in 0.001s

OK

Coverage for **mybmi.py** : 100%

7 statements

7 run

0 missing

0 excluded

0 partial

```
1 def computeBMI(kg, M):  
2     if kg>200 and M>2.5: #錯誤，範圍判斷錯誤  
3         return -1  
4     if kg<=2 or M<=0.05: #錯誤，範圍判斷錯誤  
5         return -1  
6     BMI = round(kg/(M*M),2) #錯誤，四捨五入取兩位小數  
7     #BMI = ((100*kg/(M*M))/1)/100 #正確，去尾，乘100取整數，再除100取兩位小數  
8     return BMI
```

# Exercise 問題

- 2. 增加測試案例設計，使需求/問題測試更完整
  - 測試案例 vs. 需求/題目
- 測試案例設計
  - 錯誤身高
    - 輸入：500公斤，1.55公尺
    - 輸出：-1
  - 錯誤體重
    - 輸入：200公斤，5公尺
    - 輸出：-1
- 2. 增加測試案例設計，使需求/問題測試更完整
  - 【mybmitest.py】 驅動程式11.12.加入
    - `self.assertEqual(mybmi.computeBMI(500, 1.55), -1)`
    - `self.assertEqual(mybmi.computeBMI(200, 5), -1)`

# Exercise 問題

- ❑ 執行測試，有錯誤，必須修改程式

```
FName      Stmts  Miss Branch BrPart  Cover
-----
mybmi.py    7      0    4      0  100%
-----
TOTAL       7      0    4      0  100%
```

```
=====
FAIL: test_computeBMIOK (mybmitest.TestBMI)
```

```
-----
Traceback (most recent call last):
```

```
  File "C:\TEST\mybmitest.py", line 11, in test_computeBMIOK
```

```
    self.assertEqual(mybmi.computeBMI(500, 1.55), -1)
```

```
AssertionError: 208.12 != -1
```

```
-----
Ran 1 test in 0.004s
```

```
FAILED (failures=1)
```



# Exercise 1 問題

- 請修改程式達到測試正確、涵蓋度100%



## Exercise 2 題目增加需求/條件

### □ 輸入：

- 體重(公斤), 身高(公尺)
- 體重( $3 \leq \sim \leq 200$ ), 身高( $0.50 \leq \sim \leq 2.5$ )。
- $BMI = \text{體重 (公斤)} / \text{身高}^2 \text{ (公尺平方)}$

### □ 輸出：

- 身高體重輸入範圍錯誤，輸出"ERROR INPUT"。

Level	BMI	OUTPUT
體重過輕	$BMI < 18.50$	Underweight
正常範圍	$18.50 \leq BMI < 24$	Normal
體重過重	$24.00 \leq BMI < 27$	Overweight
肥胖	$27.00 \leq BMI$	Obesity 肥胖

## Exercise 2 題目增加需求/條件

- 設計一個 function，呼叫 computeBMI，處理輸出
  - def **printResult**(weight, height):
- 針對**測試驅動程式**，設計**測試方法** testPrintResult(self):
- 設計足夠的**測試案例**
  - 使**主程式測試** coverage 100%
  - 修改使**主程式正確**

## Exercise 2 設計測試案例

### □ 測試案例分類

#### ○ 錯誤身高

➤ 輸入：52公斤，5公分，輸出：ERROR INPUT

#### ○ 錯誤體重

➤ 輸入：2公斤，155公分，輸出：ERROR INPUT

#### ○ 輸出：Underweight

#### ○ 輸出：Normal

➤ 輸入：52公斤，155公分，

➤ BMI：52(公斤)/(1.55\*1.55)(公尺平方)= 21.64412(兩位小數 21.64)。

#### ○ 輸出：Overweight

#### ○ 輸出：Obesity 肥胖

---

---

# 附錄

# Exercise 3

- ❑ 測試計算平均
- ❑ 測試計算年齡
- ❑ 測試計算BMI值的function
  - BMI值計算公式:  $BMI = \text{體重(公斤)} / \text{身高}^2(\text{公尺平方})$
  - 例如：一個52公斤的人，身高是155公分，則BMI為：
  - $52(\text{公斤}) / (1.55 * 1.55)(\text{公尺平方}) = 21.64412$
  - 正常範圍為 BMI=18.5~24
  - 請設計一個 function，傳入身高、體重，回傳BMI。
  - 請設計一個 function，從鍵盤輸入姓名name、身高、體重。
  - 當BMI太大，輸出 Hi name, Your BMI: xx too HIGHT。
  - 當BMI太小，輸出 Hi name, Your BMI: xxx too LOW。

# Exercise 3

```
def BMIOutput(name, BMI):
    #print('Hi ', name, end=', Your BMI: %d' %BMI)
    str01 = 'Hi {name}, Your BMI: {BMI}'.format(name=name, BMI=BMI)
    if BMI<=0:
        out = str01 + ' ERROR'
    elif BMI>24:
        out = str01 + ' too HIGH'
    elif BMI<18.5:
        out = str01 + ' too LOW'
    else:
        out = str01 + ' OK'
    return out
```

```
def testBMI():
    name, kg, M = inputData()
    BMI = computeBMI(kg, M)
    print(BMIOutput(name, BMI))
```

```
def computeBMI(kg, M):
    if kg<=0 or M<=0:
        return 0
    #BMI = round(kg/(M*M),2)
    BMI = ((100*kg/(M*M))//1)/100
    return BMI
```

```
def inputData():
    name = input('name: ')
    kg = float(input('KG: '))
    M = float(input('M: '))
    return name, kg, M
```

# Exercise 3

```
import unittest

class TestBMI(unittest.TestCase):
    def test_computeBMIOK(self):
        self.assertEqual(computeBMI(52, 1.55), 21.64)
        self.assertEqual(computeBMI(50, 1.5), 22.22)
        self.assertAlmostEqual(computeBMI(52, 1.55), 21.63, 1)

    def test_computeBMIOK_Almost(self):
        self.assertAlmostEqual(computeBMI(50, 1.5), 22.21, 1)

    def test_computeBMILow(self):
        self.assertEqual(computeBMI(60, 2), 15)

    def test_computeBMHigh(self):
        self.assertEqual(computeBMI(50, 1), 50)

    def test_computeBMINE(self):
        self.assertEqual(computeBMI(50, -1), 0)
        self.assertEqual(computeBMI(-50, 1), 0)

    def test_BMIOutput(self):
        out = 'Hi John, Your BMI: 50.0 too HIGH'
        self.assertEqual(BMIOutput('John', 50.0), out)

    def test_BMIOutputNe(self):
        out = 'Hi John, Your BMI: -50.0 ERROR'
        self.assertEqual(BMIOutput('John', -50.0), out)

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



# Exercise 4

- 給予一組學生名單，包括名字、學號以及其三科成績，
  - 計算每位學生的平均分數，
  - 並將最高分與最低分的學生姓名分數印出。
  
- 給予一組郵遞區號(北北基)的資料，試著寫出兩個函式：
  - area\_to\_zip(area):
    - 傳入值為區域名稱回傳此區域的郵遞區號，
    - ex. 呼叫 area\_to\_zip("信義區")，回傳 201
  - zip\_to\_area(zip):
    - 傳入值為郵遞區號回傳區域名稱，
    - ex. 呼叫 area\_to\_zip(106)，回傳 "大安區"。

# 單元測試 (Unit Test)

## ❑ unittest (PyUnit)測試框架

- 測試套件(Test Suite): 一組 **Test Case**, **Test Suite** 或兩者組合。
- Test runner: 負責**執行測試**並提供**測試結果**。
- **測試執行器**
  - 在每個測試執行前，執行 setUp 方法，
  - 在每個測試執行後，執行 tearDown 方法。
- 將某些**測試方法**組成 Test Suite
- 使用 TextTestRunner 執行一組 Test Suite

## ❑ **function 傳入多個不定參數 \***

- self.args = (52, 1.55)
- computeBMI(\*self.args)

## ❑ **assertAlmostEqual(執行結果, 期望結果, 比對小數位數)**

- **self.assertAlmostEqual(computeBMI(50, 1.5), 22.21,1)**

# 單元測試 (Unit Test)

## □ unittest (PyUnit)測試框架

```
import unittest
class TestBMI(unittest.TestCase):
    def setUp(self):
        self.args = (52, 1.55)
    def tearDown(self):
        self.args = None
    def test_computeBMIOK(self):
        self.assertEqual(computeBMI(*self.args), 21.64)
        #self.assertEqual(computeBMI(50, 1.5), 22.22)
        self.assertAlmostEqual(computeBMI(*self.args), 21.63, 1)

def suite():
    suite = unittest.TestSuite()
    suite.addTest(TestBMI('test_computeBMIOK'))
    return suite

if __name__ == '__main__':
    runner = unittest.TextTestRunner()
    runner.run(suite())
```

```
-----
Ran 1 test in 0.002s

OK
```

# 單元測試 (Unit Test)

```
import unittest
class TestStringMethods(unittest.TestCase):
    def test_upper(self):
        self.assertTrue('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        #self.assertTrue('Foo'.isupper())
        self.assertTrue('foo'.islower())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        with self.assertRaises(TypeError):
            s.split(2)

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

# 單元測試 (Unit Test)

## ▣ 程式碼涵蓋度 (Code Coverage) – 指令行執行

### ○ 執行指令涵蓋與分支涵蓋

- `coverage run -branch 10802.py`
- 每一條指令是否執行過?
- 每一個分支判斷 True/False 是否執行過?

### ○ 產生html報表，目錄 d 名稱為 cov

- `coverage html -d cov`

---

# END

---

