

GROUP 6





TABLE OF CONTENTS

01

Giới thiệu về testcase

02

Phân loại kiểm thử

03

Dynamic Verification

04

Bài tập



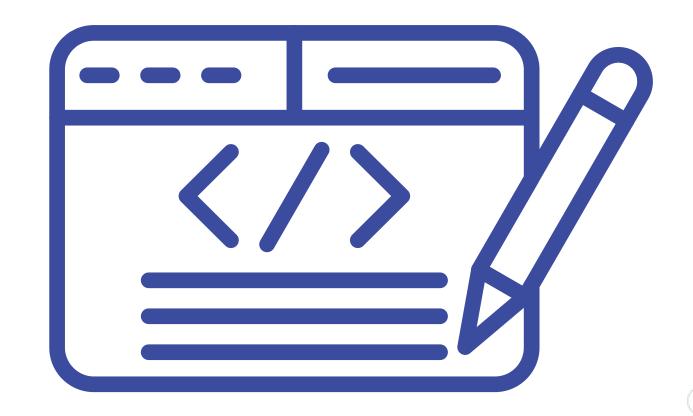
Testcase là gì?





Testcase là gì?

 Một bộ hoặc một tập hợp các điểm dữ liệu đầu vào, đầu ra dùng để đánh giá tính đúng đắn và hiệu suất của một chương trình

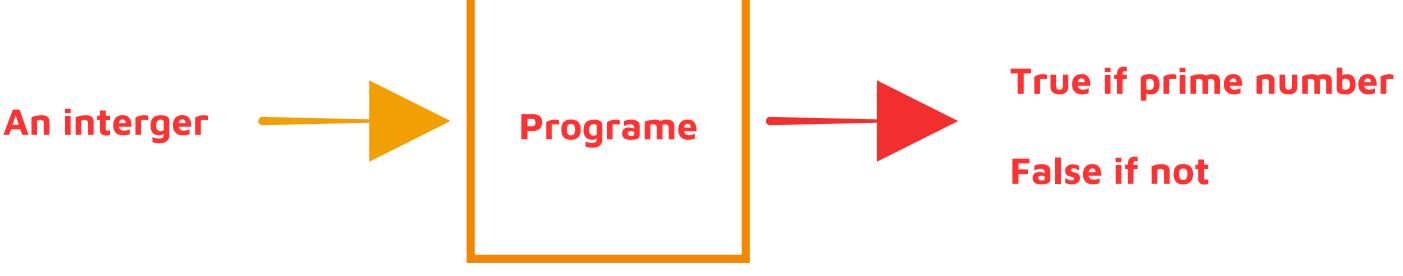


Mỗi testcase tương ứng với một tình huống cụ thể, trong đó các biến và điều kiện khác nhau được kiểm tra

→ Đảm bảo phần mềm hoạt động chính xác và như mong đợi



Testcase là gì?



Input	Output	Expected Time
2	True	< 1s
10	False	<1s
-69	Fasle	<1s



Tại sao cần Testcase?

Để chương trình, phần mềm thực hiện đúng các yêu cầu, thủ tục được đưa ra

Biết được hiệu năng chương trình

Giải thuật đúng đắn

Công cụ để kiểm tra đúng đắn và hiệu quả của chương trình



O1 Giới thiệu về testcase

O2
Phân loại kiểm thử

O3Dynamic Verification

O4 Bài tập



O2 Phân loại kiểm thử

> Kĩ thuật kiểm thử

Static Verification

- Thực thi kiểm chức mà không cần chạy chương trình
- Kiểm tra yêu cầu và tài liệu thiết kế
- Lỗi phát hiện là lỗi logic

Dynamic Verification

- Thực thi kiểm tra bằng cách chạy thử chương trình
- Kiểm tra kết quả phần mềm khi nó đang thực thi
- Lỗi logic và lỗi thực thi



Static Verification

def divide(x, y):
 return x / y

Không cần thực hiện ta vẫn biết:

Chương trình trên có lỗi nếu y = 0





O2
Phân loại kiểm thử

Static Verification

```
• • •
def read_file(file_path):
    with open(filename, "r") as f:
        return f.read()
def write_file(file_path, content):
    with open(file_path, "w") as f:
        f.write(content)
def main():
    file_path = input("Enter the file_path:
") content = input("Enter the content: ")
    print(read_file(file_path))
    write_file(file_path, content)
if __name__ == "__main__":
    main()
```



O2
Phân loại kiểm thử

```
def read_file(file_path):
    with open(filename, "r") as f:
        return f.read()
def write_file(file_path, content):
    with open(file_path, "w") as f:
        f.write(content)
def main():
    file_path = input("Enter the file_path:
   content = input("Enter the content: ")
    print(read_file(file_path))
   write_file(file_path, content)
if __name__ == "__main__":
   main()
```

Static Verification

Ở hàm read_file:

Chưa xử lí được trường hợp File không tồn tại



O1 Giới thiệu về testcase

02 Phân loại kiểm thử

O3Dynamic Verification

O4 Bài tập



- Thực thi kiểm tra bằng cách chạy thử chương trình
- Kiểm tra kết quả phần mềm khi nó đang thực thi
- Lỗi logic và lỗi thực thi

Black Box Testing

Boundary value testing

Equivalence class testing

White Box Testing

Coverage testing



Boundary value testing

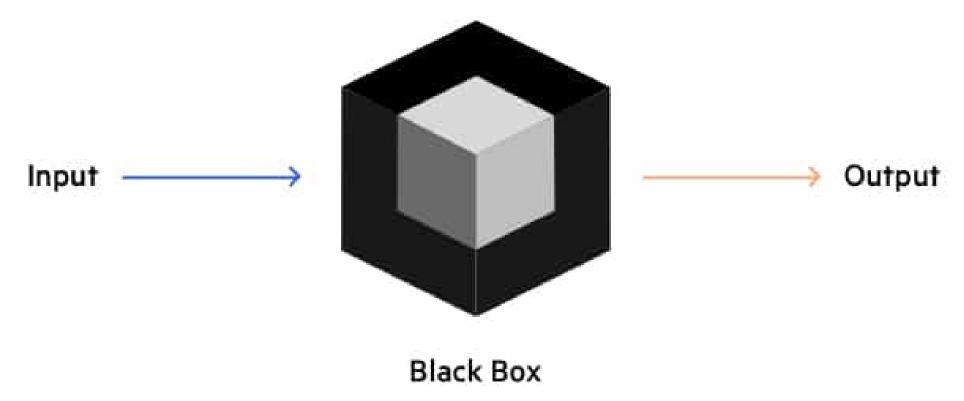
Equivalence class testing

White Box Testing

Coverage testing

Black Box Testing

- Phương pháp kiểm thử hộp đen: Coi hệ thống là một hộp đen, không thể thấy được cấu trúc logic bên trong. Người làm kiểm thử tập trung vào các yêu cầu chức năng của phần mềm dựa trên các dữ liệu lấy từ đặc tả yêu cầu phần mềm.
- Có thể áp dụng gần như cho tất cả chương trình





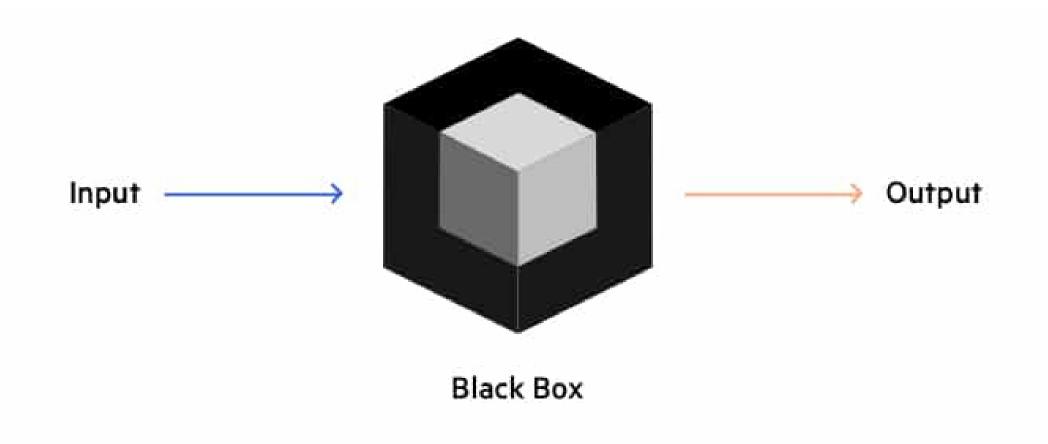
Boundary value testing
Equivalence class testing

White Box Testing

Coverage testing

Black Box Testing

- Kiểm thử hộp đen nhằm **tìm ra các lỗi**:
 - Chức năng thiếu hoặc không đúng đắn
 - Sai về giao diện của chương trình
 - Hành vi hoặc hiệu suất lỗi





Black Box Testing

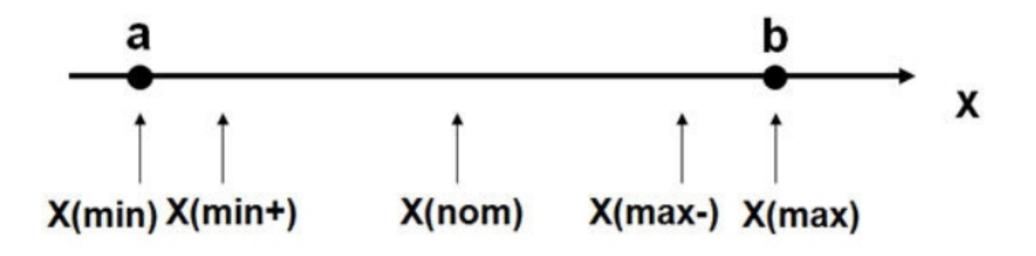
Equivalence class testing

White Box Testing

Coverage testing

Boundary value testing

- Tập trung vào việc **kiểm thử các giá trị biên của miền giá trị inputs** để thiết kế testcase do "lỗi thường tiềm ẩn lại các ngõ ngách và tập hợp tại biên" (Beizer)
- Kĩ thuật này hiệu quả nhất trong trường hợp: "Các đối số đầu vào (input variables) độc lập với nhau và mỗi đối số đều có một miền giá trị hữu hạn"





Black Box Testing

Equivalence class testing

White Box Testing

Coverage testing

Boundary value testing

Với x thuộc [a,b] thì ta sẽ test các trường hợp:

- 9
- a + 1
- **b**
- b 1
- (a+b)/2



Test cases for a variable x, where $a \le x \le b$



Black Box Testing

Equivalence class testing

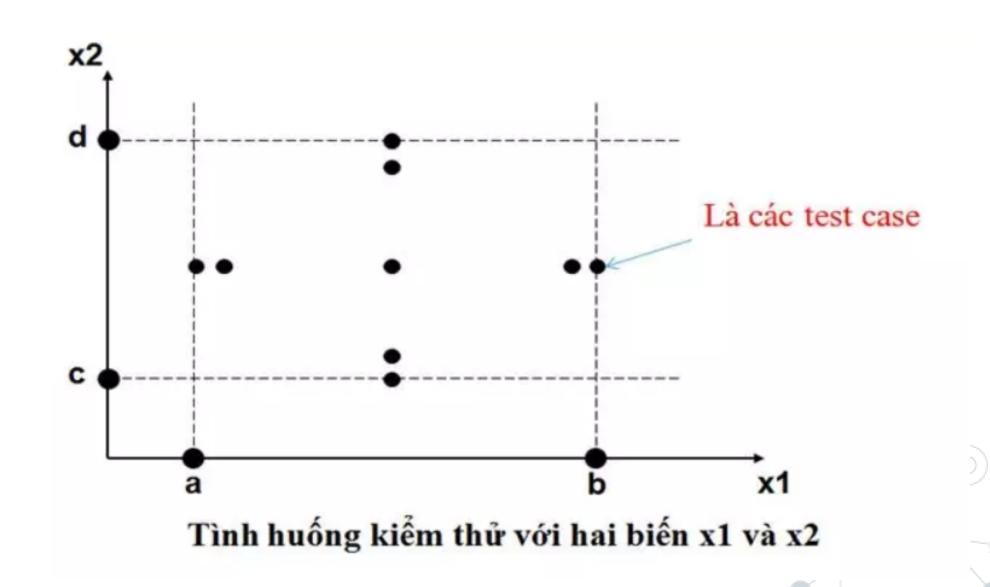
White Box Testing

Coverage testing

Boundary value testing

$$a \le x_1 \le b, c \le x_2 \le d$$

Thiết kế các testcase để test giá trị biên như đồ thị sau:





Black Box Testing

Equivalence class testing

White Box Testing

Coverage testing

Boundary value testing

 $1 \le day \le 31, 1 \le Month \le 12, 1812 \le Year \le 2012$

month	<u>day</u>
min = 1	min = 1
min+=2	min+=2
nom = 6	nom = 15
max - = 11	max - = 30
max = 12	max = 31

<u>year</u> min = 1812 min+ = 1813 nom = 1912 max- = 2011 max = 2012

Case	month	day	year	Expected Output
1	6	15		June 16, 1812
2	6	15	1813	June 16, 1813
3	6	15	1912	June 16, 1912
4	6	15	2011	June 16, 2011
5	6	15	2012	June 16, 2012
6	6	1	1912	June 2, 1912
7	6	2	1912	June 3, 1912
8	6	30	1912	July 1, 1912
9	6	31	1912	error
10	1	15	1912	January 16, 1912
11	2	15	1912	February 16, 1912
12	11	15	1912	November 16, 1912
13	12	15	1912	December 16, 1912

Boundary Value Analysis Test Cases



Black Box Testing

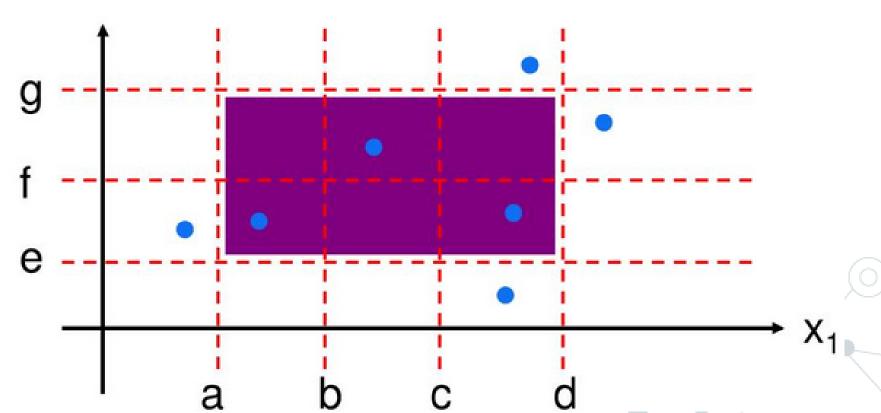
Boundary value testing

White Box Testing

Coverage testing

Equivalence class testing

- Ý tưởng: Chia miền vào chương trình thành các lớp dữ liệu. Xác định đầu vào hợp lệ và không hợp lệ để lập các ca kiểm thử theo các lớp đó
- Thay vì kiểm tra tất cả các giá trị đầu vào, có thể lựa chọn từ đầu vào đại diện riêng từng lớp, mỗi lớp gọi là một vùng tương đương
- Vùng tương đương đúng là vùng tương đương mà các các điểm dữ liệu ở đó cho kết quả thỏa mãn bài toán





Black Box Testing

Boundary value testing

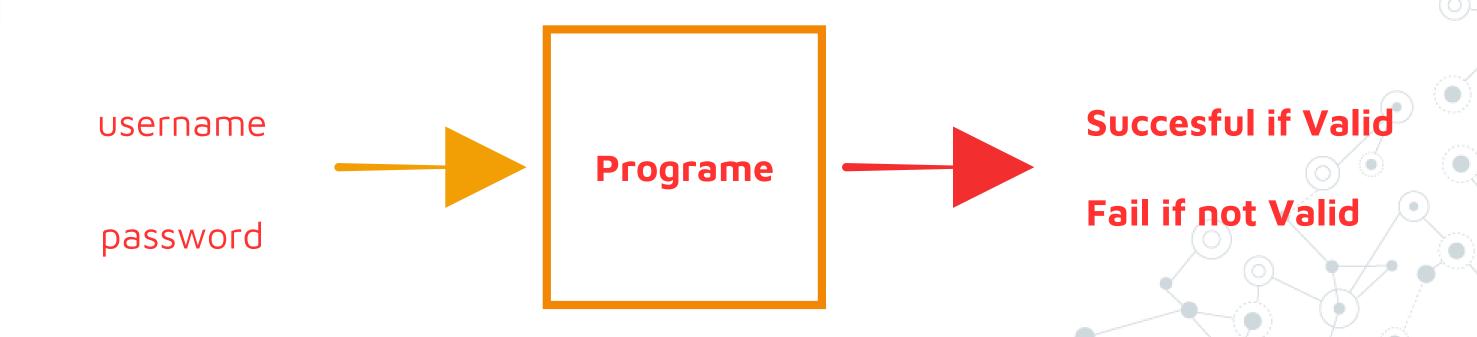
White Box Testing

Coverage testing

Equivalence class testing

Ví dụ: Viết chương trình đăng nhập với yêu cầu sau:

- Tên đăng nhập(username) và mật khẩu (password) không được rỗng
- Tên đăng nhập phải chứa ít nhất 6 ký tự và không nhiều hơn 20 ký tự
- Mật khẩu phải có ít nhất 8 ký tự và chứa ít nhất một chữ cái viết hoa và một chữ cái viết thường





Black Box Testing

Boundary value testing

White Box Testing

Coverage testing

Equivalence class testing

Ví dụ: Viết chương trình đăng nhập với yêu cầu sau:

- Tên đăng nhập(username) và mật khẩu (password) không được rỗng
- Tên đăng nhập phải chứa ít nhất 6 ký tự và không nhiều hơn 20 ký tự
- Mật khẩu phải có ít nhất 8 ký tự và chứa ít nhất một chữ cái viết hoa và một chữ cái viết thường.

Vùng tương đương đúng

Tên đăng nhập và mật khẩu phù hợp với tất cả yêu cầu

Vùng tương đương sai

- Một trong hai trường chuỗi rỗng
- Trường hợp tên đăng nhập có ít hơn 6 ký tự hoặc nhiều hơn 20 ký tự.
- Trường hợp mật khẩu có ít hơn 8 ký tự hoặc không chứa ít nhất một chữ cái viết hoa hay một chữ cái viết thường



Black Box Testing
Boundary value testing

White Box Testing

Coverage testing

Equivalence class testing

Input	Ouptut	Problem
Tên đăng nhập = " TranWoffy ", Mật khẩu = " P@ssw0rd "	Successful	None
Tên đăng nhập = "", Mật khẩu = " P@ssw0rd "	Fail	Usename Empty
Tên đăng nhập = " TranWoffy ", Mật khẩu = ""	Fail	Password Empty
Tên đăng nhập = " Woffy ", Mật khẩu = " P@ssw0rd "	Fail	Len(Usẻname) < 6
Tên đăng nhập = " TranWoffydeptraisieucap ", Mật khẩu = ""	Fail	Len(Username) > 20
Tên đăng nhập = " TranWoffy ", Mật khẩu = " P@ss "	Fail	Len(Pass) < 8
Tên đăng nhập = " TranWoffy ", Mật khẩu = " p@ssword "	Fail	Pass không chứa chữ viết hoa
Tên đăng nhập = " TranWoffy ", Mật khẩu = "P@SSWORD "	Fail	Pass không chứa chữ viết thường



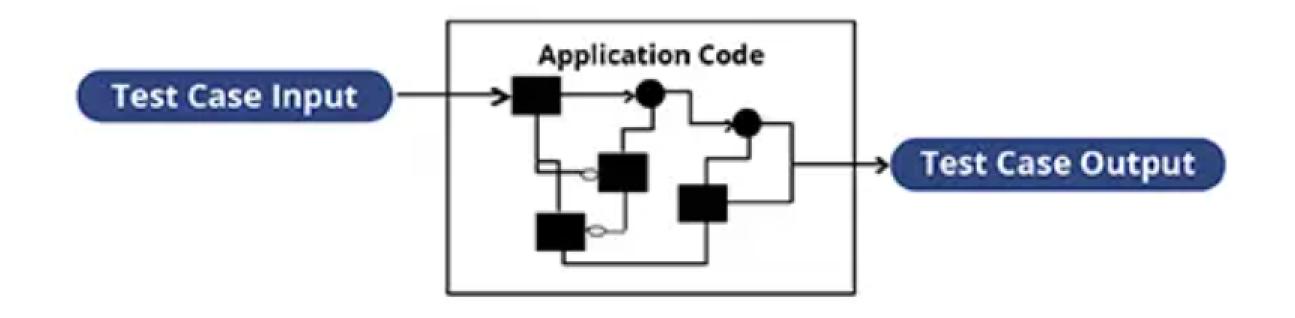
Black Box Testing

Boundary value testing
Equivalence class testing

Coverage testing

White Box Testing

- Thiết kế testcase dựa vào **cấu trúc** bên trong của đối tượng cần kiểm thử
- Đối tượng chính của kiểm thử hộp trắng là tập trung vào cấu trúc bên trong chương trình và tìm ra tất cả những lỗi bên trong chương trình.





Black Box Testing

Boundary value testing
Equivalence class testing

White Box Testing

Coverage testing

• Là kĩ thuật thiết kế testcase đảm bảo "cover" được tất cả các **câu lệnh,** biểu thức điều kiện trong code modulo cần test

Các tiêu chí đánh giá độ cover

- Method Coverage
- Statement Coverage
- Decision/Branch Coverage
- Condition Coverage





Black Box Testing

Boundary value testing

Equivalence class testing

White Box Testing

Coverage testing

Method Coverage

- Đo lường tỉ lệ mà các phương thức hoặc hàm trong chương trình đã được kiểm thử.
- Mục tiêu: Đảm bảo rằng tất cả các phương thức hoặc hàm trong mã nguồn đã
 được gọi ít nhất một lần trong quá trình kiểm thử
- Điều này bao gồm:
 - Các testcase cho các phương thức hoặc hàm đặc biệt
 - Các testcase kiểm tra các trường hợp biên
 - Các testcase kiểm tra các tình huống lỗi (nếu có) mà các phương thức hoặc hàm có thể gặp phải



Black Box Testing

Boundary value testing
Equivalence class testing

White Box Testing

Coverage testing

Method Coverage

```
def sum_two(a, b):
    return a + b

def calculate_total(a, b):
    if a == 0:
        return b

if b == 0:
    return a

return sum_two(a,b)
```

Với các bộ test mà **testcase có một trong 2 số bằng 0**

→ Method coverage chỉ bằng 50% (vì không thực hiện hàm sum_two(a,b))



Black Box Testing

Boundary value testing
Equivalence class testing

White Box Testing

Coverage testing

Method Coverage

```
def sum_two(a, b):
    return a + b

def calculate_total(a, b):
    if a == 0:
        return b

if b == 0:
    return a

return sum_two(a,b)
```

Với các bộ test mà **testcase có một trong 2 số bằng 0**→ Method coverage chỉ bằng **50%**(vì không thực hiện hàm sum_two(a,b))

Lưu ý: Method Coverage chỉ đo lường việc các phương thức hoặc hàm đã được gọi hay không - KHÔNG kiểm tra các hành vi cụ thể của các phương thức hoặc hàm đó.



Black Box Testing

Boundary value testing

Equivalence class testing

White Box Testing

Coverage testing

Statement Coverage

- Đo lường tỷ lệ các dòng mã đã được thực hiện so với tổng số dòng mã trong chương trình.
- Mục tiêu: Đảm bảo rằng mỗi dòng lệnh trong chương trình đã được thực hiện ít
 nhất một lần trong quá trình kiểm thử
- Phương pháp: Viết các testcase để điều hướng mã nguồn qua mọi dòng lệnh trong hàm hoặc phương thức.



Black Box Testing

Boundary value testing
Equivalence class testing

White Box Testing

Coverage testing

Statement Coverage

```
1 def test(a, b, c, d):
2   if a == 0:
3    return "Woffy"
4
5   x = 0
6   if a == b:
7    print("Woffy")
8
9   c -= d
10   return "Woffy"
```

Testcase (a,b,c,d) = (0,0,0,0)

→ Statement coverage sẽ đạt 25%

(vì chỉ chạy 3 dòng trên 12 dòng)

Testcase (a,b,c,d) = (1,1,2,1) → Statement coverage sẽ đạt 100% (vì các dòng lệnh sẽ chạy hết)



Black Box Testing

Boundary value testing

Equivalence class testing

White Box Testing

Coverage testing

- Decision/Branch Coverage
- **Tỷ lệ phần trăm** các **biểu thức điều kiện** trong chương trình được ước lượng giá trị trả về (true, false) khi thực thi các testcase
- Một biểu thức điều kiện (single hay complex) phải được kiểm tra trong cả hai trường hợp giá trị của biểu thức là true/false



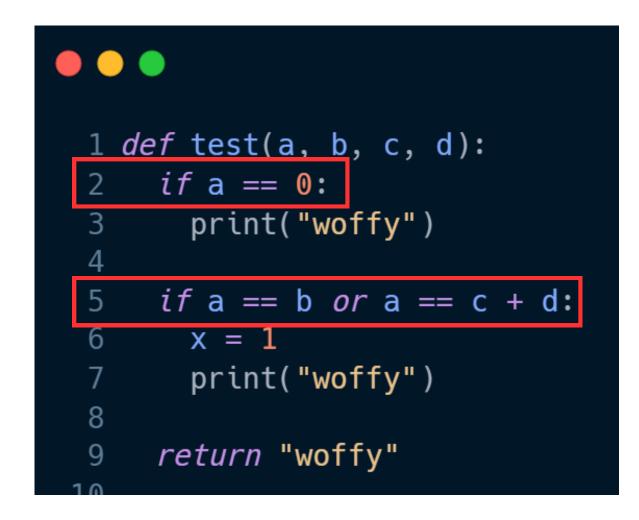
Black Box Testing

Boundary value testing
Equivalence class testing

White Box Testing

Coverage testing

Decision/Branch Coverage



	True	False
a == 0	test(0,0,0,0)	test(1,1,0,0)
a == b or a == c+d	test(0,0,0,0)	

- Chỉ đạt **75%** decision coverage
- Thêm một testcase là test(1,0,0,0) thì decision coverage sẽ đạt 100%



Black Box Testing

Boundary value testing

Equivalence class testing

White Box Testing

Coverage testing

- Condition Coverage
 - Tỷ lệ phần trăm các biểu thức điều kiện đơn trong biểu thức điều kiện phức của chương trình được ước lượng giá trị trả về (true, false) khi thực thi các testcase

```
1 def test(a, b, c, d):
2  if a == 0:
3   print("woffy")
4
5  if a == b or a == c + d:
6   x = 1
7   print("woffy")
8
9  return "woffy"
```

	True	False
a == 0	test(0,0,0,0)	test(1,0,0,0)
a == b	test(0,0,0,0)	test(1,0,0,0)
a == c + d	test(0,0,0,0)	test(1,0,0,0)

Đạt **100%** Condition Coverage với hai bộ testcase



O1 Giới thiệu về testcase

02 Phân loại kiểm thử

03Dynamic Verification

O4 Bài tập



04 Bài tập

Viết một bộ test cósố testcase nhỏ nhất sao cho Condition Coverage đạt 100%

```
1 def check_prime(number):
    if number <= 1:
      return False
    elif number == 2:
      return True
 6
    elif number % 2 == 0:
      return False
10
    else:
       for i in range(3, int(number**0.5) + 1, 2):
13
         if number % i == 0:
14
           return False
15
16
      return True
```



THANK YOU FOR LISTENING!