Testing Your Program

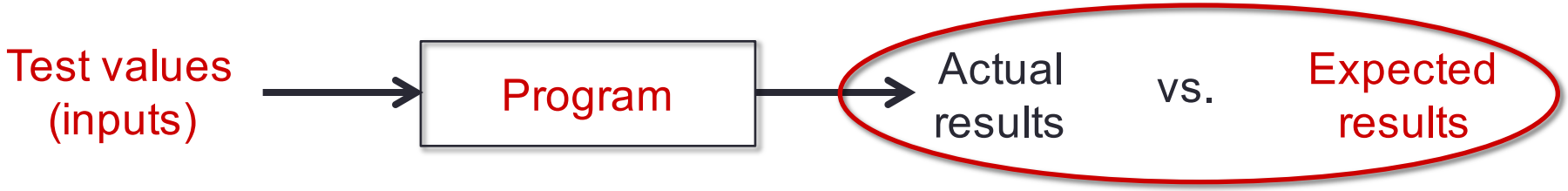
CS 1111
Introduction to Programming

Spring 2019

Testing and Debugging

- Testing = process of finding input values to check against a software

Test case consists of test input values and expected results



- Debugging = process of finding a defect given a failure

Why do We Test Software?

- Goal of testing
  - Not to prove correctness, but to increase our confidence in correctness
  - Improve quality
  - Reduce overall software development cost (budget, time, and effort)
  - Preserve customer satisfaction
  - **Get good grades** in CS 1110/1111 and any programming courses
- What fact does each test try to verify?
  - Know what to check and whether the program handles that properly
- Benefits
  - You are not biased that your code works
  - You will better understand what you need to build
  - You will get insights on how to built it

An Example in Python

```
# Return index of the first occurrence of a letter in string,
# Otherwise, return -1

def get_index_of(string, letter):
    index = -1
    for i in range(1, len(string)):
        if string[i] == letter:
            index = i
    return index


# Test1: inputs "python", "z"
print(get_index_of("python", "z")) # expected: -1, actual: -1

# Test2: inputs "python", "z"
print(get_index_of("python", "p")) # expected: 0, actual: -1
```

Defect: Should start at 0, not 1

Test1: not reveal a problem (defect)
actual output = expected output

Test2: reveal a problem (defect)
actual output ≠ expected output

For simplicity, this example assumes a function accept a letter of size 1

Testing: Choosing Test Inputs

```
# Return index of the first occurrence of a letter in string,
# Otherwise, return -1

def get_index_of(string, letter):
    index = -1
    for i in range(1, len(string)):
        if string[i] == letter:
            index = i
    return index
```

Focus on input values

1. Identify inputs
   - **string**, **letter**

2. What input values can be
   - **string** is empty or not
   - **letter** is empty or not
   - length of **string** (0, 1, 2, >2)

Focus on program functionality

1. Identify inputs
   - **string**, **letter**

2. What affect program's functionality
   - number of occurrence of **letter** in **string**
   - **letter** occurs first in **string**
   - **letter** occurs last in **string**

Testing: Comparing Results

- Given the test input values, compare the actual results with the expected results
- If the actual results == expected result, the program passes the test
- Otherwise, the program fails the test
  - The test input valules reflect the characteristics of the input parameters
  - The characteristics of the inputs signify the kinds of defects in program
  - The kinds of defects tell what to fix

Debugging

```
# Return index of the first occurrence of a letter in string,
# Otherwise, return -1

def get_index_of(string, letter):
    index = -1
    for i in range(1, len(string)):
        # check if we get the right character
        print(i, string[i])
        if string[i] == letter:
            index = I
            # check if the if-code-block is executed
            print(string[i], "=", letter, "i=", i)

    return index

# Test1: inputs "python", "z"
print(get_index_of("python", "z")) # expected: -1, actual: -1

# Test2: inputs "python", "z"
print(get_index_of("python", "p")) # expected: 0, actual: -1
```

Fix this

Run Test 2

```
1 y
2 t
3 h
4 o
5 n
-1
```

Summary

Testing

- Choose test values; check if the program fails (i.e., there is a problem or "defect" in the code, which causes the program to fail)
- Each test value serves a single purpose (or check for a certain aspect)
- Check
  - Normal cases
  - Corner, edge, boundary cases
  - Exceptional cases

Debugging

- The program failed; find where to fix
- Use "print" statement to ensure changes to the program variables are correct
- Verify forward: print from start, move forward until incorrect values are detected
- Verify backward: print from the end, move backward until incorrect values are detected