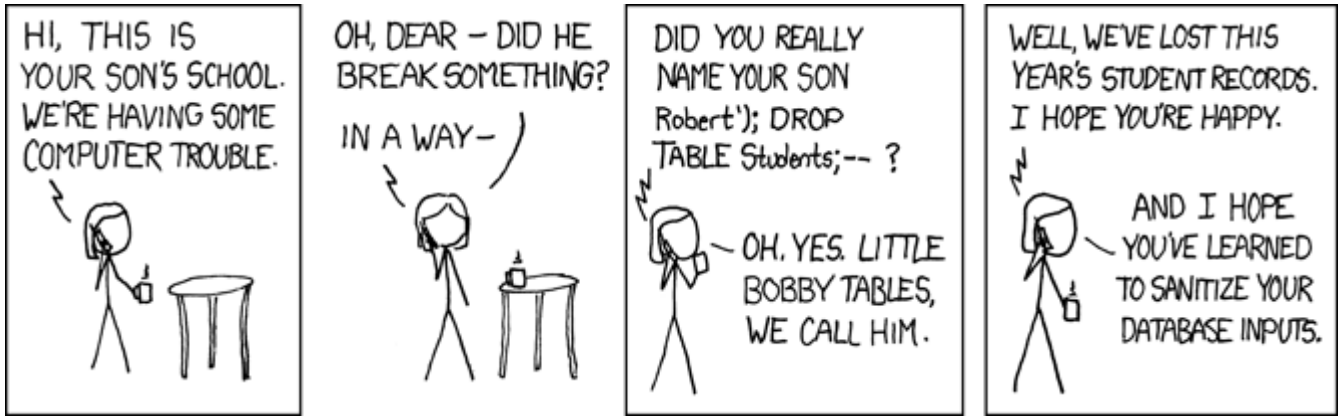
CSc 110, Spring 2017

Lecture 22: Testing

Thanks to Atif Memon from UMD for disaster examples



Count Vowels

- Write a function `vowel_count(s)` that accepts a string `s` as a parameter and returns a list of integers representing the counts of each vowel of string `s`.

- There are five vowels: a, e, i, o, u

- What data mapping would help to count the vowels?

- Write a helper function that returns a number representing the index of a vowel in the above mapping.

Vowel helper function

```
# Maps the characters a,e,i,o,u to the numbers 0,1,2,3,4,
# respectively. If the parameter c is not a vowel, returns -1
 def is_vowel(c):
    if (c == "a"):
        return 0
    elif (c == "e"):
        return 1
    elif (c == "i"):
        return 2
    elif (c == "o"):
        return 3
    elif (c == "u"):
        return 4
    return -1                  # parameter c is not a vowel
```

Count vowels

```
def main():
    vlist = vowel_count("i think, therefore i am")
    print("vlist = ", vlist)

# Return a list containing the counts of the number of vowels
# in string s
def vowel_count(s):
    # indices of list vowels map to a, e, i, o, u
    vowels = [0] * 5
    s = s.lower()
    for c in s:
        i = is_vowel(c)
        if (i >= 0):
            vowels[i] += 1
    return vowels
```

Count Vowels

- Consider an alternative for function `is_vowel(c)`
- Look at this string: `v = "aeiou"`
- What does `v.find("e")` return?
- Previous call to `is_vowel(c)`:

```
i = is_vowel(c)
```

- Now becomes:

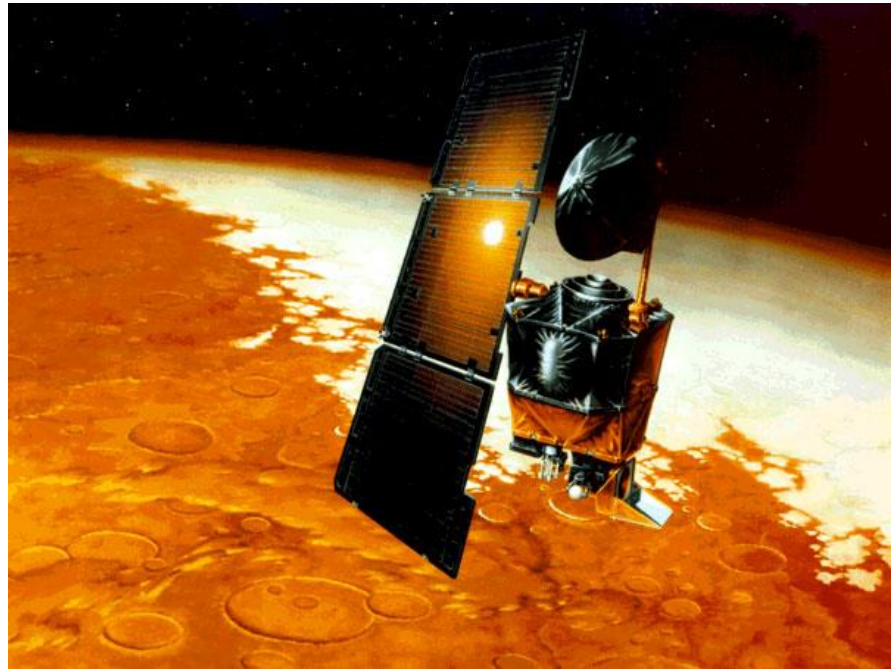
```
i = "aeiou".find(c)
```

Why talk about testing?

- Software engineers spend a lot of time testing and debugging.
- There are positions specifically titled "Software Engineer in Test."
- Game development companies have "play testers."
- Software testing is expensive but necessary.
- Terrible things can happen if software isn't tested thoroughly.

Why talk about testing?

- Mars Climate Orbiter
  - Purpose: to study the Martian climate and to serve as a relay for the Mars Polar Lander
  - Disaster: Bad trajectory caused it to disintegrate in the upper atmosphere of Mars
  - Why: Software bug - failure to convert English units to metric values (pound-seconds vs. newton-seconds)



Why talk about testing?

- THERAC-25 Radiation Therapy
  - 1985 to 1987: two cancer patients at the East Texas Cancer Center in Tyler received fatal radiation overdose (a total of 6 accidents)
  - Why: Software bug - mishandled race condition (i.e., miscoordination between concurrent tasks)

Why talk about testing?

- London Ambulance Service
  - Purpose: automate many of the human-intensive processes of manual dispatch systems associated with ambulance services in the UK – functions: Call taking
  - Failure of the London Ambulance Service on 26 and 27 November 1992
    - Load increased, emergencies accumulated, system made incorrect allocations

Types of testing: Black box testing

- Testing that verifies the code does what the specification requires
- The testing is not dependent on the internal structure of the code
- Recall testing your code for Gradanator
  - The specification gave values to enter and the responses to expect
  - This was a very repetitious task (as you know!)
  - Could we automate this task?

Black box testing example

Function: `remove_bad_pairs(plist)` - accepts a list of integers and removes any adjacent pair of integers in the list if the left element of the pair is larger that the right element of the pair.

Consider passing in the following list:

```
[3, 7, 9, 2, 5, 5, 8, 5, 6, 3, 4, 7, 3, 1]
```

Think of the list as being paired as follows.

```
[3, 7, 9, 2, 5, 5, 8, 5, 6, 3, 4, 7, 3, 1]
```

The pairs below are "bad" because the left of the pair is greater than the right:

```
9, 2  8, 5, 6, 3 3, 1
```

The function would modify the list as follows:

```
[3, 7, 5, 5, 4, 7]
```

Black box testing example – cont.

Function: `remove_bad_pairs(plist)`

If the list has an odd length, the last element is not part of a pair and is also considered "bad". It would be removed. The list

```
[10, 20, 2, 5, 4]
```

would be modified to become

```
[10, 20, 2, 5]
```

If an empty list is passed in, the list should still be empty after the call.

What should we test for?

- Removing one pair: `[9, 2]` → `[]`
- Removing multiple pairs in a row: `[5, 3, 8, 1, 7, 0]` → `[]`
- The empty list: `[]` → `[]`
- List of an odd size: `[10]` → `[]`
- List containing an equal pair: `[2, 4, 8, 8]` → `[2, 4, 8, 8]`
- List containing negative numbers: `[-2, -3, 7, 8]` → `[7, 8]`

Black box testing – 1 Manually in Idle

```
>>> pl = [9, 2]
>>> remove_bad_pairs(pl)
[]
>>> pl
[]
>>> pl = [5, 3, 8, 1, 7, 0]
>>> remove_bad_pairs(pl)
[]
>>> pl
[]
>>> pl = []
>>> remove_bad_pairs(pl)
[]
>>> pl
[]
>>> pl = [10]
>>> remove_bad_pairs(pl)
[]
```

Black box testing – 1 Manually in Idle

```
>>> pl = [10]
>>> remove_bad_pairs(pl)
[]
>>> pl
[]
>>> pl = [2, 4, 8, 8]
>>> remove_bad_pairs(pl)
[2, 4, 8, 8]
>>> pl
[2, 4, 8, 8]
>>> pl = [-2, -3, 7, 8]
>>> remove_bad_pairs(pl)
[7, 8]
>>> pl
[7, 8]
```

Black box testing – 2 Coded with test calls

- Manual testing is tedious, expensive, and error-prone.
- Let's automate this.
- Let's write a program to make these calls.

- Programmer changes the implementation of `remove_bad_pairs(plist)`

- Let's see if the new version works.

Black box testing – 3 Further automation

- The program which called tests was better, but let's automate further.
- Remove the repetition of testing the outcome
- A test harness:
  - Automate the testing process
  - Execute the test suite (set of test cases)
  - Report the outcome

Types of testing: White box testing

In contrast to Black box testing, consider White box testing which has these characteristics:

- Bases tests on the structure of the code
- Ensures that all paths through the code are executed
- Requires programming knowledge

What should you test?

- Focus on edge cases
- Common edge cases
  - 1, 0, -1
  - Empty list
  - Range out of bounds

Make sure every path is taken in the code.

Tester must understand all language functionality and concepts.

What is wrong with the version of `remove_bad_pairs(plist)` on the next slide?

White box testing example

```
#Version of the function with an error
def remove_bad_pairs(pairs_list):
    if (len(pairs_list) % 2 != 0):
        pairs_list.remove(pairs_list[-1])
    i = 0
    while (i < len(pairs_list)):
        if (i % 2 != 0 and pairs_list[i - 1] > pairs_list[i]):
            list.remove(i-1)
            list.remove(i)
        else:
            i +=2
```

White box test a more complex problem

- Consider the following Internet Movie Database (IMDb) data:

```
1 9.1 196376 The Shawshank Redemption (1994)
2 9.0 139085 The Godfather: Part II (1974)
3 8.8 81507 Casablanca (1942)
```

- Write a program that displays any movies containing a phrase:

```
Search word? part

Rank    Votes   Rating  Title
2       139085  9.0     The Godfather: Part II (1974)
40      129172  8.5     The Departed (2006)
95      20401   8.2     The Apartment (1960)
192     30587   8.0     Spartacus (1960)
4 matches.
```

Types of testing

- Unit test: verifies correctness of a small piece of testable code in isolation
- Integration test: verifies different small already tested components of the program work together correctly
- Regression testing: a complete retesting of a modified program
- Stress testing: tests the behavior under peak user volumes
- Performance, security, usability and many more

Test driven development

- A software engineering philosophy
- Tests are written before the code is written