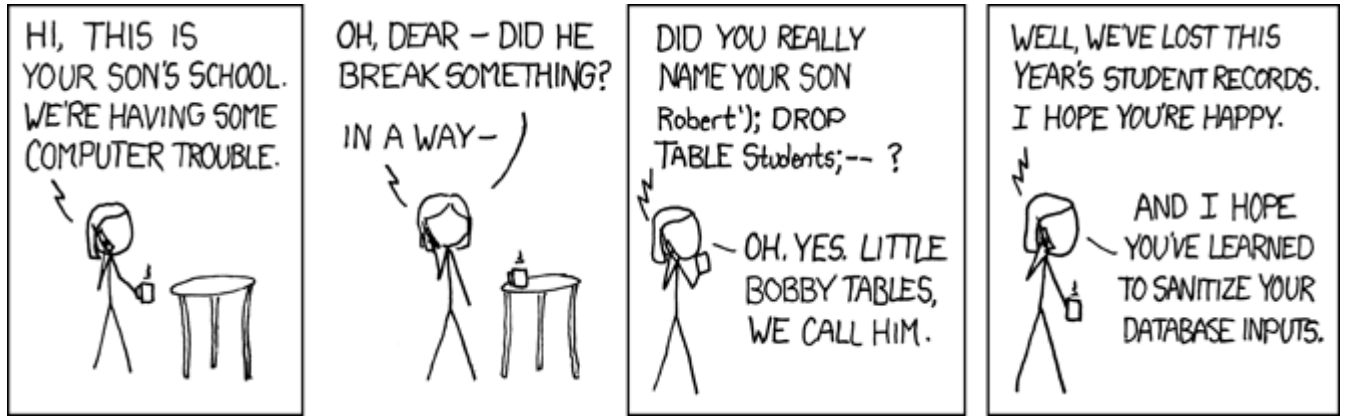
CSc 110, Autumn 2017

Lecture 28: Testing

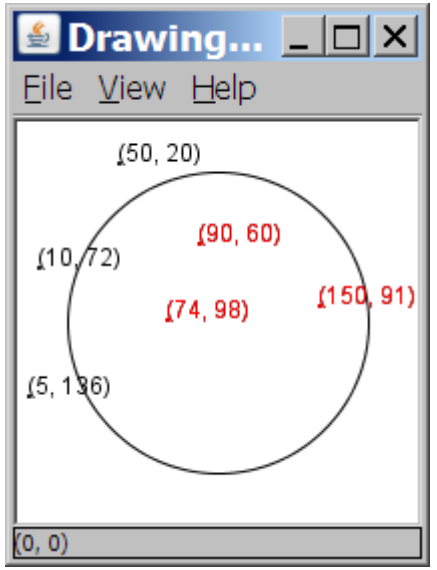
Thanks to Atif Memon from UMD for disaster examples



A programming problem

- Given a file of cities' names and (x, y) coordinates:

```
Winslow 50 20
Tucson 90 60
Phoenix 10 72
Bisbee 74 98
Yuma 5 136
Page 150 91
```

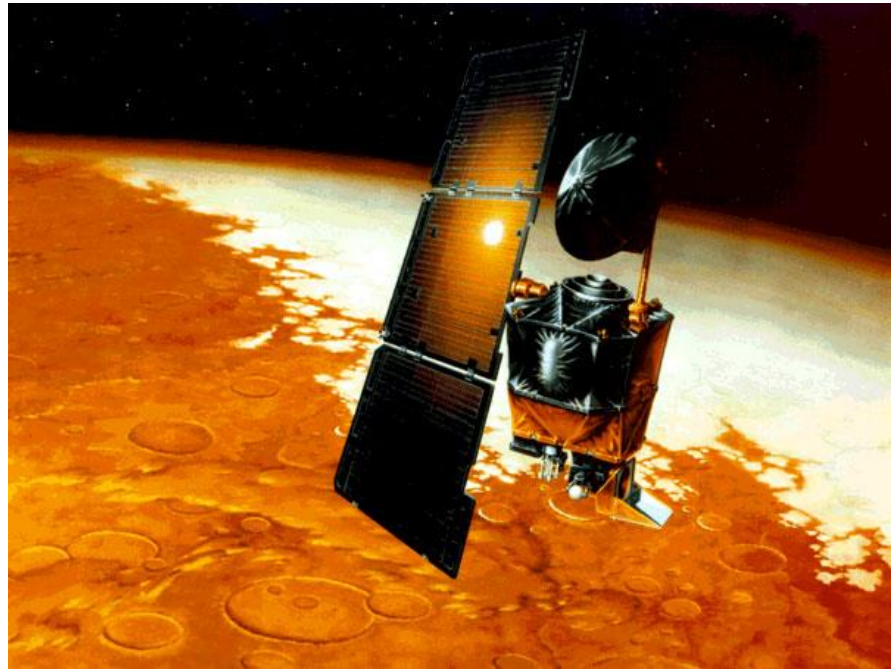


- Write a program to draw the cities on a `DrawingPanel`, then simulates an earthquake that turns all cities red that are within a given radius:

```
Epicenter x? 100
Epicenter y? 100
Affected radius? 75
```

Why talk about testing?

- Mars Climate Orbiter
  - Purpose: to relay signals from the Mars Polar Lander once it reached the surface of the planet
  - Disaster: smashed into the planet instead of reaching a safe orbit
  - Why: Software bug - failure to convert English measures to metric values



Why talk about testing?

- THERAC-25 Radiation Therapy
  - 1986: two cancer patients at the East Texas Cancer Center in Tyler received fatal radiation overdose
  - 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

Industry comparison

- “If the automobile industry had developed like the software industry, we would all be driving $25 cars that get 1,000 miles to the gallon.”
- “Yeah, and if cars were like software, they would crash twice a day for no reason, and when you called for service, they’d tell you to reinstall the engine.”

- Now days you can get a job as a software engineer developing cars!

Why talk about testing?

- You will spend a lot of time testing and debugging
- You can get a job as a Software Engineer in Test
- Terrible things can happen if you don't test well enough!

Types of testing

- **Black box testing**: testing the program does what the specification requires
- **White box testing**: examining code for potential problems
- **Unit testing**: 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

Testing example

```
def main():
      data = [3, 5, 2, 7, 45, 43, 5, 3]
      remove_bad_pairs(data)
      print(data)


remove_bad_pairs(list):
    if (len(list) % 2 != 0):
        v.remove(list[-1])
      i = 0
    while(i < len(list)):
        if (i % 2 != 0 and list[i - 1] > list[i]):
            list.remove(i - 1)
            list.remove(i)
```