Knight Capital (2012)

* Software engineers were making changes
* Once orders from broker dealer started coming in, the old code was activated
* Server started sending stocks with wrong conditions

Importance of software testing:

* Does the software work as expected?
* Can the correctness be verified independently?
* How confident are we in the accuracy of the results?

Automated testing

* Computers are great at repetitive tasks
* Define complex, repeatable processes with fewer errors
* Saves effort in the long run
* Unit tests: test specific units of functionality ensuring expected outputs from given inputs
  + Test if that unit of code is working correctly
  + Syntax is minimum
* At the higher level there are functional or integration tests: test functional paths through the code, especially useful for exposing faults in inter-unit interactions
* Regression tests: ensure unchanged program output despite code modifications
  + Write to ensure that the program behaves the same regards of the code modifications
  + Usually after the bug that got fixed
  + assert -> has Boolean statement that evaluates true or false
  + if true, no answer
  + if false, then it will give you the output

Digging deeper into errors with debugging

* Unit tests detect problems and provide hints towards their location, but they don’t give specific details
* Techniques for better understanding include:
  + Outputting program state at various points
  + Using logging capabilities for tracking the program’s progression
    - Python has classes that make it very easy
    - Then we can turn logging on and off
    - We can redirect to the file without changing the code
    - Program that has a lot of log statements is the same but all logging statements are print statements
    - You have to write them and replace them that’s why logging makes it much easier
    - Python package loggers?
  + Examining intermediate files
* When these aren’t sufficient a debugger can provide an in-depth exploration of running code
  + Debugger can be run in the command line or IDE such as VS code or RStudio etc -> they will have debugger built in
  + You can have a break point to see if it is what it should be
  + In python, it is useful to develop in debugger
  + Python has “evaluate code”, and we can do things in it, we can expect the members of the objects, run functions, run python code line by line

Other useful techniques:

* Defensive programming
  + Checks that input data satisfies preconditions before proceeding
    - Start of the function
    - Check if it is an integer
    - Then test if it is greater than zero
  + Raises an error if the preconditions are not met, preventing silent changes
    - A value error if the number is negative
    - If it is a string -> type error
  + Often checks for correct argument type, crucial for languages like Python
    - Because python is not picky for the types of variables
    - Every python function I write should start with something that raises a value error and type error
* Automated Code Style Checks
  + Linters analyse source code for errors, inconsistencies, and stylistic issues
    - Like blank spaces in python for example
  + They promote readability, maintainability, and adherence to best practises
* Code smell are hints of underlying design issues in code, despite it working
  + e.g. large classes or methods, methods with too many parameters, duplicated statements in conditional blocks, etc
    - things that work but doesn’t look right -> design issues in the code
    - method that has way too many parameters
    - data class in python remove the code smell
    - duplicated statements -> if something is done in the same way, you have to generalize it and package it into the function

Limits to testing:

* Testing investment should match the software’s complexity and usage
  + Automated testing saves time as software complexity increases
* Even thorough unit testing can’t catch all bugs
  + Manual testing and extensive data testing are vital
* No test suite can guarantee a bug-free application, but it significantly reduces risks