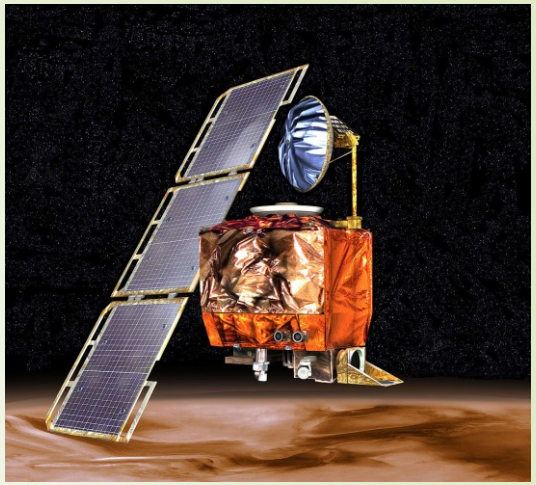
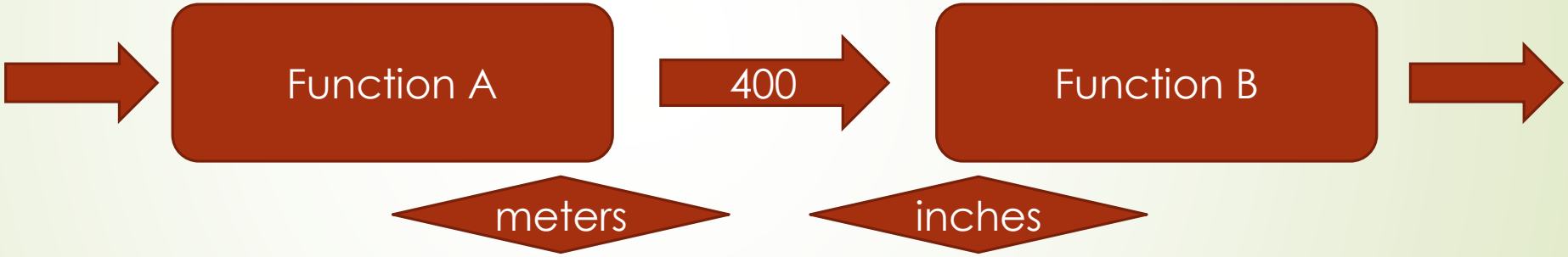
Principles of Programming I

- Errors and testing
- Unit testing in Python
- Git, GitHub

Testing



- Famous cases of big failures due to software errors
  - NASA Mars Climate Orbiter crashed because of errors in flight controller code



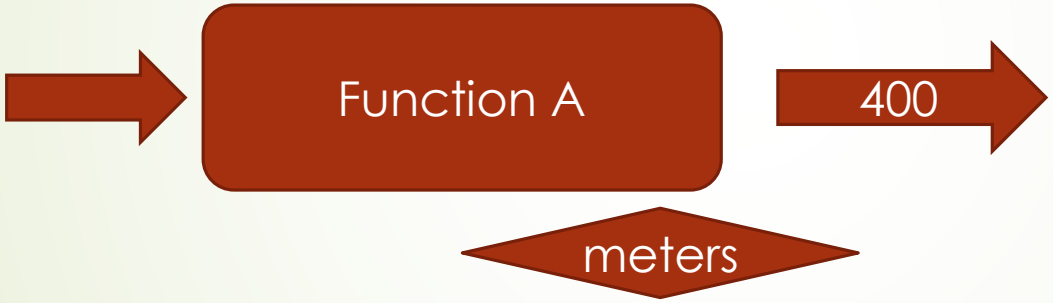
- Facebook has 30 000 000 lines of code: suppose one error occurs per 1000 lines of code - disaster

Kinds of tests

- Unit test: when it fails, it tells you what piece of your code needs to be fixed
- Integration test: when it fails, it tells you that the pieces of your application are not working together as expected
- Acceptance test: when it fails, it tells you that the application is not doing what the customer expects it to do
- Regression test: when it fails, it tells you that the application no longer behaves the way it used to

Unit testing

- In this course we are concerned with unit testing



- Unit testing is applied to fruitful functions

Unit testing in Python

Pytest

- Assuming your implemented functions are in a module my_module.py
- The test file:
  - has name of the shape test_something.py
  - imports my_module
  - has the following structure:

```
import pytest
from  my_module  import  *

 def test_somefunction():
        #  You  can  define  variables  and  functions  here
        #  Test  methods  go  here-  the  name  of  each  test  method
        #  begins  with  "test_"
        # and uses "assert" statement

def test_someotherfunction ():
 ....
```

Pytest (cont.)

- Pytest has extended functionality but the most common use is via assert statement:
  - assert Boolean_expression
  - the test is considered as "passed" if Boolean_expression evaluated to True, otherwise it "failed"

  Example:

```
def test_ addition():
    assert (5 == addition(3,2))
# here addition(x,y) is the function being tested
```

  - Caution: do not use A == B when A or B are floats
    - It is not always the case that 1.2 + 1.8 == 3.0 (it could be 3.00000001)
    - Use abs(A-B) < 0.0001 instead, use as many 0 as required by your application
  - To execute the tests, run pytest in the command line when in the directory where both test_something.py and my_module.py are located

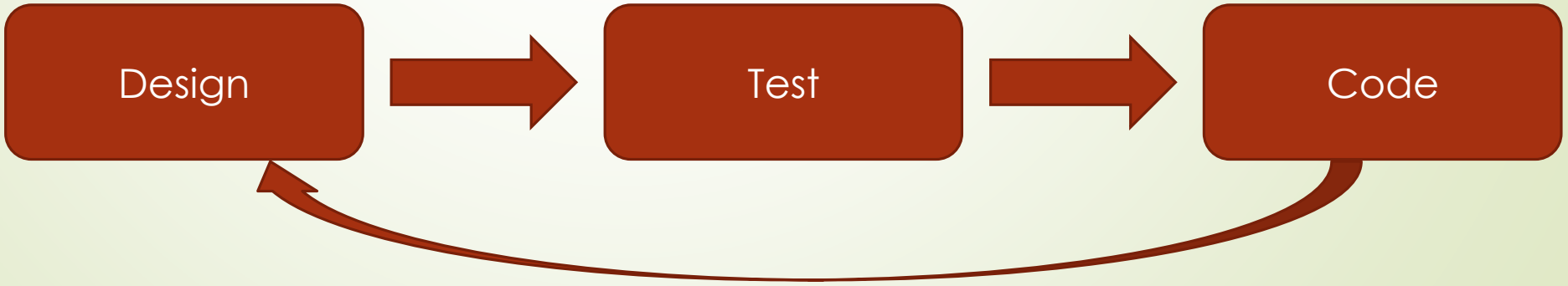
  Demo of Pytest will follow in several minutes.

Test Drive Development (TDD)

Classic vs. Modern World of Software Development

- Classic

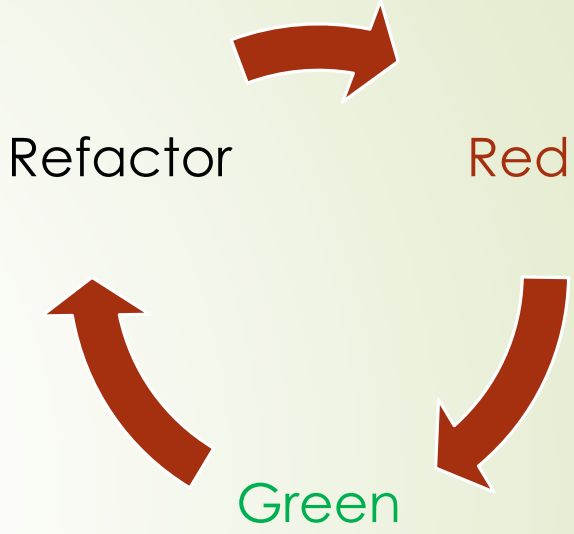
- Modern



TDD

- TDD is a technique for building software that guides the process of development by writing tests
- Developers write unit tests (not testers as in alternative approaches)
- First they write tests, and then code
- Common pipeline:
  - Add a little of tests
  - Run the current code against them and see where it fails
  - Make minimal necessary changes to make it pass the tests
  - Run the tests and see that they pass
  - Refactor our code to remove redundancy

Slang of TDD



- Green
  - All tests pass
- Red
  - Some tests fail
- Refactor
  - Eliminate all the duplication and smells created in just getting the tests to work
- Make it green, then make it clean! (cit.)
- Stub
  - A shortest possible implementation of a function that returns some meaningful value but (usually) not a correct value
- TDD commonly starts with stubs that fail even minimal tests

Version control with Git and GitHub

What is version control?

- A system that keeps record of all your changes
- Allows for collaborative development
- Allows you to know who made what changes and when
- Allows you to revert any changes and go back to the previous state
- Distributed version control:
  - Users keep entire code and history on their local machines
  - Users can make any changes without internet access
    - (Except getting or sending the updates to/from remote server)

Git

- The most widely used version control system in software development
- The most important concept behind git is snapshots:
  - By means of them, git keeps track of your code history
  - A snapshot is a record of what all your files look like at a given point of time
  - You decide when to take a snapshot and of what files
  - You can go back in history and visit any shapshot

Key concepts of Git

- Commit
  - The act of creating a snapshot
  - Each project is made of a bunch of commits
  - Each commit contains the following important information:
    - New content of files
    - How files changed from a previous version
- Staging
  - Before the file is committed, it needs to be "indexed" or "staged"
  - You can think of staging as of marking for the next commit

Key concepts of Git (cont.)

- Repository (repo)
  - A collection of all the files and the history of those files
  - Comprises all your commits
  - Place where all your work related to a project is stored
  - Can live on a local machine or remote server (GitHub)
- Cloning
  - The act of copying a repository from a remote server
  - Allows teams to work together
- Pulling
  - The process of downloading commits that don't exist on your machine from a remote repository
- Pushing
  - The process of adding your local changes (commits) to the remote repository

Git and GitHub: Basic Functionality