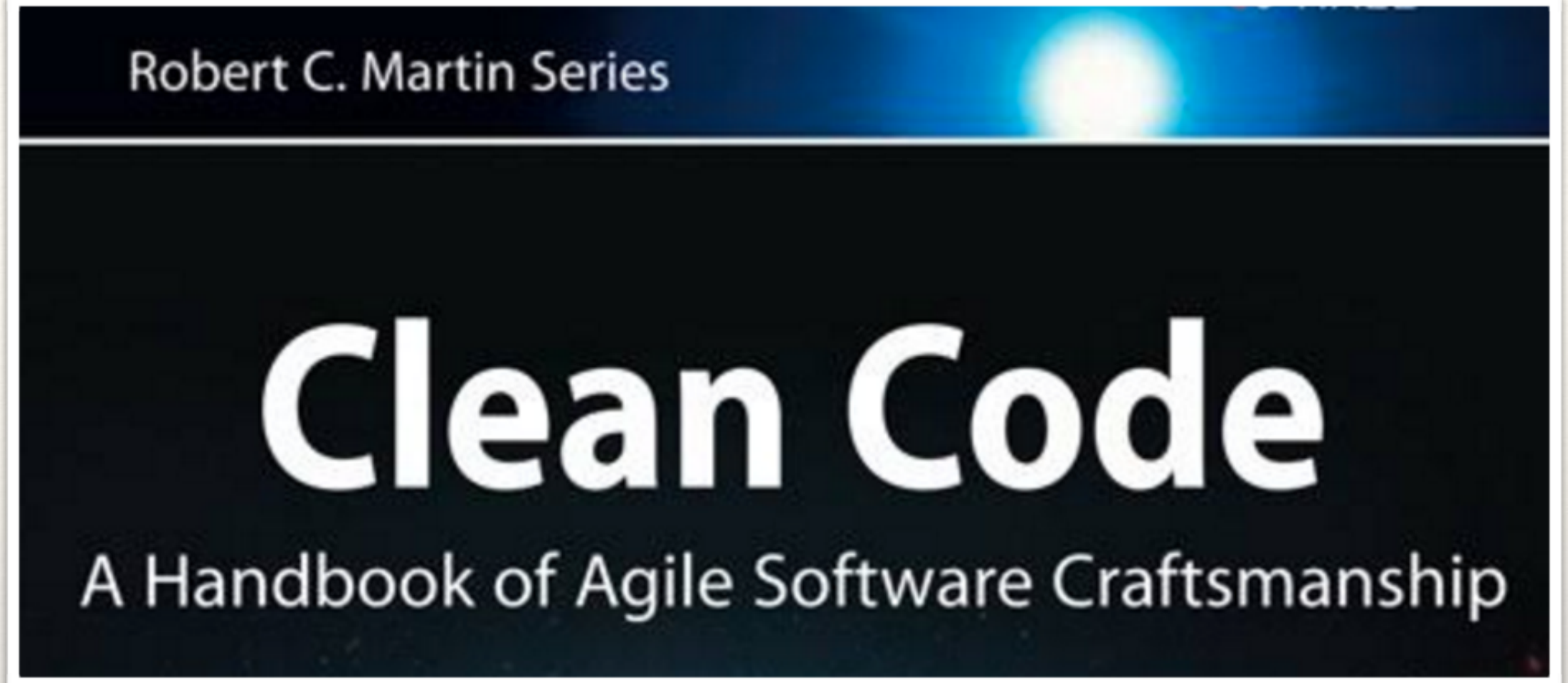
Oct 10 2016, ASU

Clean Code

Karen Pardos Olsen

(main source: 'Clean Code' by Robert C. Martin)



With examples written in java, but lessons can easily be transferred to e.g. python!

Get this book!

On Amazon/books.google…

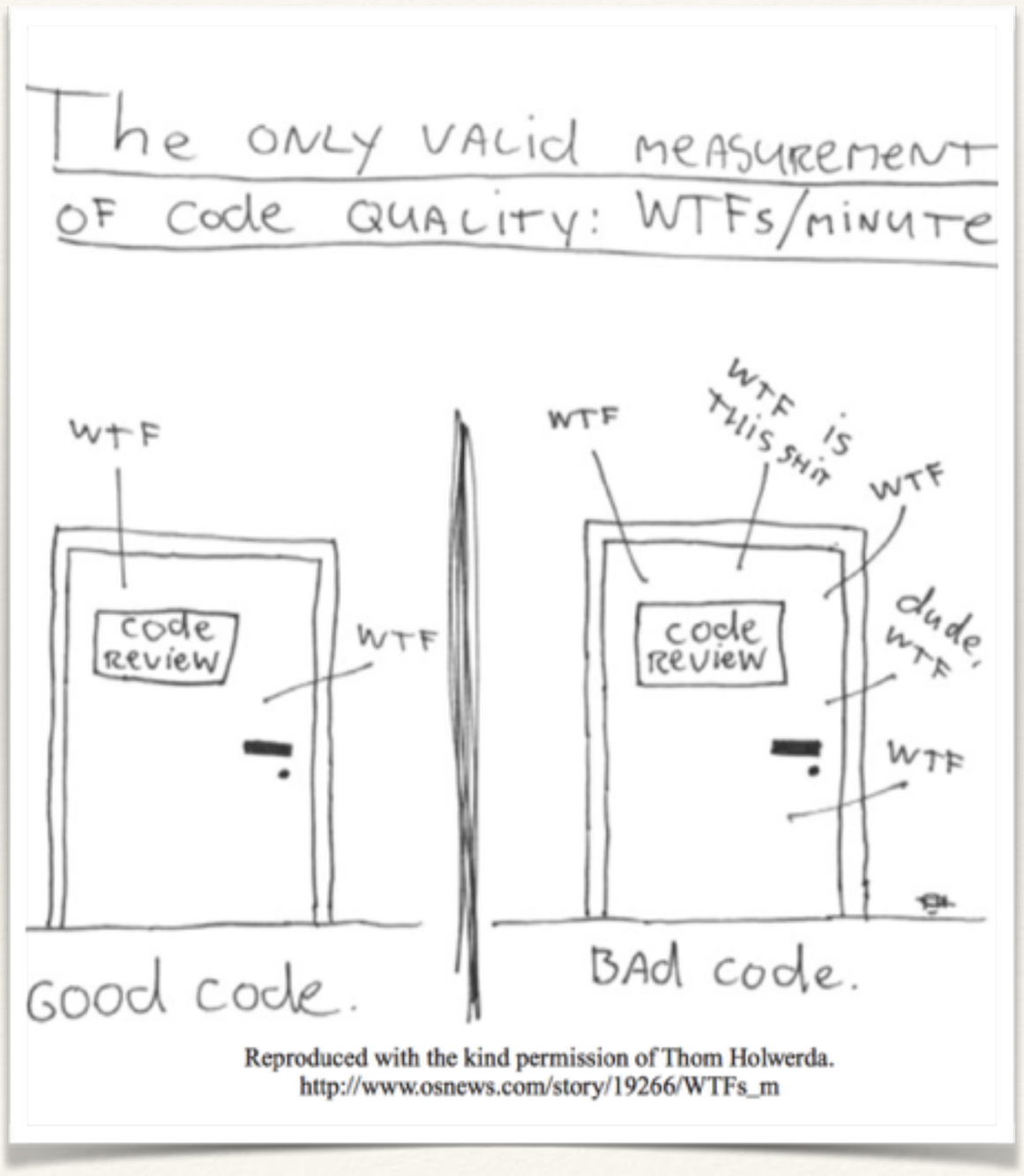
Today's plan:

- What is clean code? How do we recognize bad code?
- Meaningful names
- Functions

What is clean code?

Some signs:

- elegant, efficient, simple, direct…
- straightforward logic
- no duplication
- meaningful names, with comments where necessary
- can be read and enhanced by another developer
- complete error handling



Reproduced with the kind permission of Thom Holwerda.
http://www.osnews.com/story/19266/WTFs_m

What is clean code?

Becoming a better coder:

- Making the code easy to read
- Leaving the code a little cleaner after each check
- Caring for the code

Hard work that comes by practice!

How do we recognize bad code?

Being able to recognize clean code from dirty code does not mean that we know how to write clean code!

- When do we write bad code? When in a rush? Under pressure? Thinking "a working mess es better than nothing?"
- Something that could have been changed in one place, has been changed in many different places?
- Multiple names for the same object?
- Takes a long time to find bugs / enhance the code?

(Chapter 2)

Meaningful names

For variables, functions, modules, arguments, keywords…

Good names should:

1. Reveal intent (don't try to be smart)
2. Avoid disinformation
3. Be distinct from others
4. Pronounceable
5. Searchable

Meaningful names

1. Reveal intent (don't try to be smart)

```
d = 5   # elapsed time in days
```

Meaningful names

1. Reveal intent (don't try to be smart)

```
d = 5   # elapsed time in days
```

Any of the following would be more revealing:

```
elapsedTimeInDays = 5
daysSinceCreation = 5
daysSinceModification = 5
fileAgeInDays = 5
```

Meaningful names

2. Avoid disinformation

```
date_list = np.array([1,2,3])
```

Meaningful names

2. Avoid disinformation

```
date_list = np.array([1,2,3])
```

If it's an array, don't name it list…

```
date_array = np.array([1,2,3])
```

Meaningful names

3. Be distinct from others

```
def getActiveGalacticNuclei():
def getAGNs():
def getAGNsInfo():
```

How is a reader supposed to know the difference of these functions?

Meaningful names

4. Pronounceable

```
gensedZ = 5 # generate SED for a metallicity Z
```

“gen-es-ee-dee-zed”… or:

```
generateSEDforZ = 5
```

Meaningful names

5. Searchable

```
log = 5                          # log of grain size
```

Meaningful names

5. Searchable

```
log = 5                          # log of grain size
```

log probably exists many times in your code, so change to for example:

```
logGrainSize = 5          # log of grain size
```

(Chapter 2)

Meaningful names

Let's make functions that take redshift and spits out SFR and metallicity!

Meaningful names

```
import numpy as np
from scipy.interpolate import interp1d

print('\nMeaningful Names\n')

print('Two functions for a galaxy of mass 10^10 Msun [Speagle+14]\n')

# Observations:
z                       =    np.linspace(0,7,num=10)          # Range of redshifts probed
SFR                     =    np.linspace(0.7,1.7,num=10)      # Log of corresponding SFRs
interpol_matrix_SFR     =    interp1d(z, SFR)
Z                       =    np.linspace(0,-1,num=10)         # Log of corresponding Zs
interpol_matrix_Z       =    interp1d(z, Z)


def sfrz(z):
    # Function that takes a redshift and gives back a star formation rate (SFR).
    print('SFR for z = '+str(z)+' is:')
    SFR       =    10.**interpol_matrix_SFR(z)
    print(SFR)



def zz(z):
    # Function that takes a redshift and gives back a metallicity (Z).
    print('Metallicity for z = '+str(z)+' is:')
    z             =    10.**interpol_matrix_Z(z)
    print(z)
```

Meaningful names

```
import numpy as np
from scipy.interpolate import interp1d

print('\nMeaningful Names\n')

print('Two functions for a galaxy of mass 10^10 Msun [Speagle+14]\n')

# Observations:
redshifts          =    np.linspace(0,7,num=10)          # Range of redshifts probed
logSFR             =    np.linspace(0.7,1.7,num=10)      # Corresponding SFRs
interpol_SFR       =    interp1d(redshifts, logSFR)
logZ               =    np.linspace(0,-1,num=10)         # Corresponding Zs
interpol_Z         =    interp1d(redshifts, logZ)


def SFR_from_z(redshift):
    # Function that takes a redshift and gives back a star formation rate (SFR).
    print('SFR for z = '+str(redshift)+' is:')
    SFR      =    10.**interpol_SFR(redshift)
    print(SFR)



def Z_from_z(redshift):
    # Function that takes a redshift and gives back a metallicity (Z).
    print('Metallicity for z = '+str(redshift)+' is:')
    Z            =    10.**interpol_Z(redshift)
    print(Z)
```

(Chapter 3)

Functions

Modules -> submodules -> functions !

Some general rules:

1. Make it small! “Functions should hardly ever be 20 lines long” and “indent level of a function should not be greater than one or two”
2. Do only one thing! (and one level below that, to achieve the one thing) No hidden things.
3. From top to bottom!
4. As few arguments as possible!

Functions

```
import numpy as np
from scipy.interpolate import interp1d

print('\nMeaningful Names\n')

print('One function for a galaxy of mass 10^10 Msun [Speagle+14]\n')

# Observations:
redshifts         =    np.linspace(0,7,num=10)         # Range of redshifts probed
logSFR            =    np.linspace(0.7,1.7,num=10)     # Corresponding SFRs
interpol_SFR      =    interp1d(redshifts, logSFR)
logZ              =    np.linspace(0,-1,num=10)        # Corresponding Zs
interpol_Z        =    interp1d(redshifts, logZ)


def from_z(redshift,M,hubble_constant=0.7,output='SFR'):
    # Function that takes a redshift and gives back a star formation rate (SFR) OR metallicity.
    print('SFR for z = '+str(redshift)+' is:')

    print('Mass is: ',M)

    if output == 'SFR':
        print('SFR for z = '+str(redshift)+' is:')
        SFR      =    10.**interpol_SFR(redshift)
        print(SFR)
    if output == 'Z':
        print('Metallicity for z = '+str(redshift)+' is:')
        Z             =    10.**interpol_Z(redshift)
        print(Z)
```