11102 Fall 2025 | Introduction to Computing

What is Software Engineering?

Ibrahim Albluwi

Discussion

What is the difference between building a **house for your cat** and a **skyscraper**?

vs.

Discussion

What is the difference between building a **house for your cat** and a **skyscraper**?

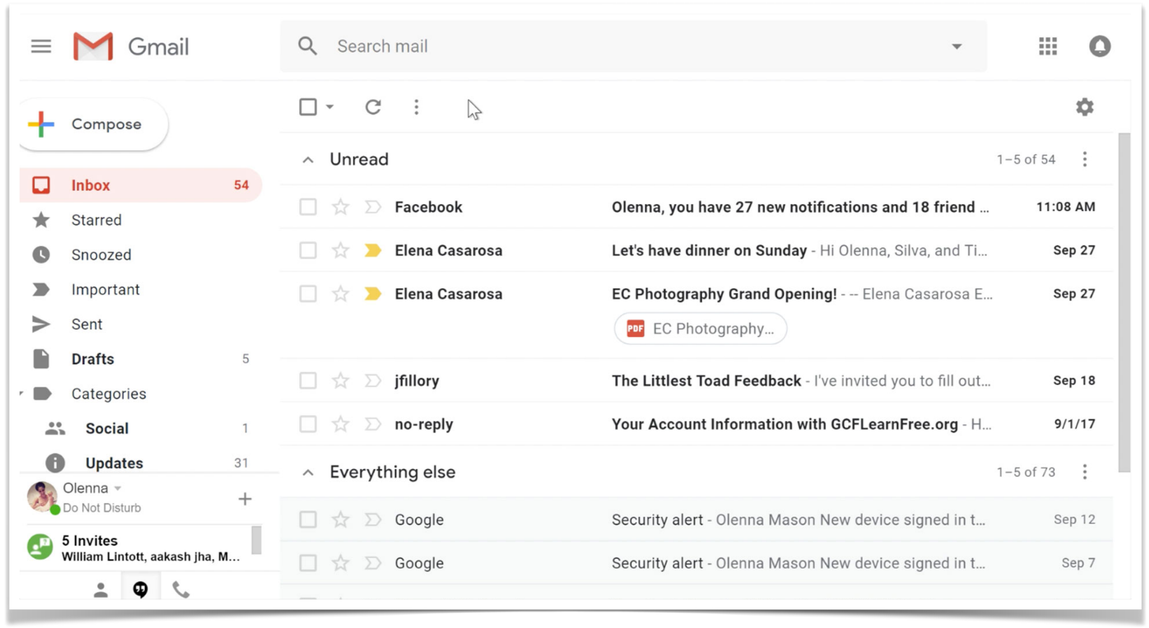
Skyscraper

Cat House

Skyscraper	Cat House
Requires a **large team** of builders, engineers, designers, etc.	Can be **built** by **you alone**.
Will be **used** by **thousands of people.**	Will be **used** by **your cat** only.
Will live for many **years**.	Might be thrown away next **week**.
High Stakes: If building fails, people might die or businesses might go bankrupt.	**Low Stakes:** If building fails, you can build or buy another one!
Must be completed within a **budget** and according to a **timeline.**	**Budget** and **time** are not a big issue!

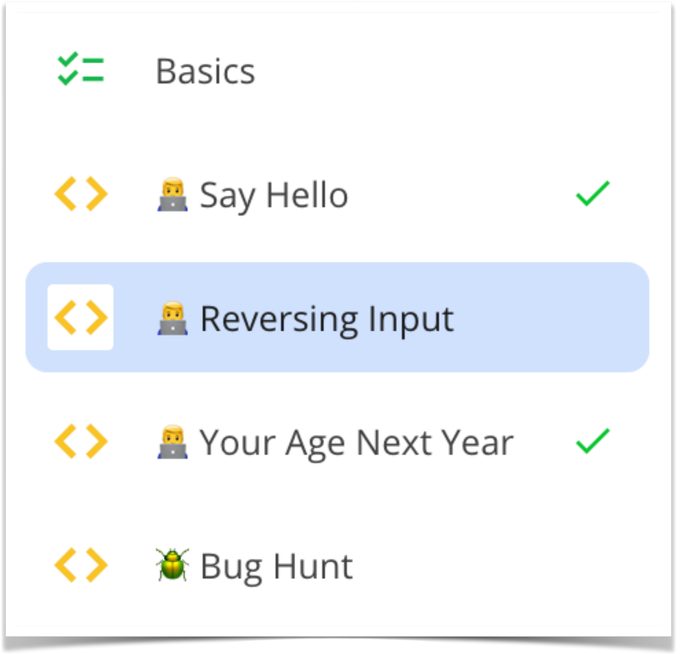
Discussion

What is the difference between solving a **homework exercise** and building **Gmail**?



Gmail

vs.



Homework Exercise

Discussion What is the difference between solving a **homework exercise** and building **Gmail**?

Gmail	Homework Exercise
Requires a **large team**.	Can be **solved** by **you alone**.
Will be **used** by **millions of people.**	Will be **used** by **you** only.
Will live for many **years**.	Thrown away after a few **months**.
High Stakes: People and businesses depend on it significantly.	**Low Stakes:** If you get it wrong, you lose 0.01 points.
Updates and bug fixes must be done under a **budget** and in a **timely** manner.	You build it for **free** and have a week to complete it in **< 15 hours.**
Requires **continuous updates**.	Why update it if there is a ✓

Summary

There is a big difference between:

• Building for **yourself**	vs.	Building **for others**.
• Building **alone**	vs.	Building in a **team**.
• Using **once**	vs.	Using **forever**.
• **Low** stakes	vs.	**High** stakes.
• No **budget**	vs.	Building with a **budget**.
• No **timeline**	vs.	Building with a **timeline**.
• **Static** requirements	vs.	**Changing** requirements.

Both skyscrapers and large **software** need a structured approach (a.k.a **engineering**!)

Software Engineering

What is it about?

Building software with others, for others, with high stakes, on a budget, and on a timeline.

Formally:

"The application of a *systematic*, *disciplined*, *quantifiable* approach to the *development*, *operation*, and *maintenance* of software; that is, the application of engineering to software." [1]

Practically:

"The job of *designing*, *developing*, *testing*, and *maintaining* software applications and systems." [2]

[1] IEEE Standard 610.12

[2] https://github.com/resources/articles/what-is-software-engineering

How Much Do SE's Get Paid?

GLASSDOOR

Accessed Nov. 2025

How much does a Software Engineer make in Amman, Jordan?

Experience: 0-1 year

Industries: Information Technology

Total pay range

JOD 456 - JOD 2K/mo

JOD 554/mo Median total pay

Experience: 10-14 years

Industries: Information Technology

Total pay range

JOD 885 - JOD 3K/mo

JOD 2K/mo Median total pay

Anecdotal Data

PSUT **fresh** graduates working as software engineers typically earn between **450** JOD and **800** JOD. Top **fresh** graduates earn **1400**+ JOD.

Examples of Software Engineering Tasks

Requirements Gathering

Key questions:
What does my client want?
What do my app users need?

Example. Conduct meetings and sign contracts with a client.

Gathering requirements is hard!

Examples of SE Tasks

Requirements Gathering

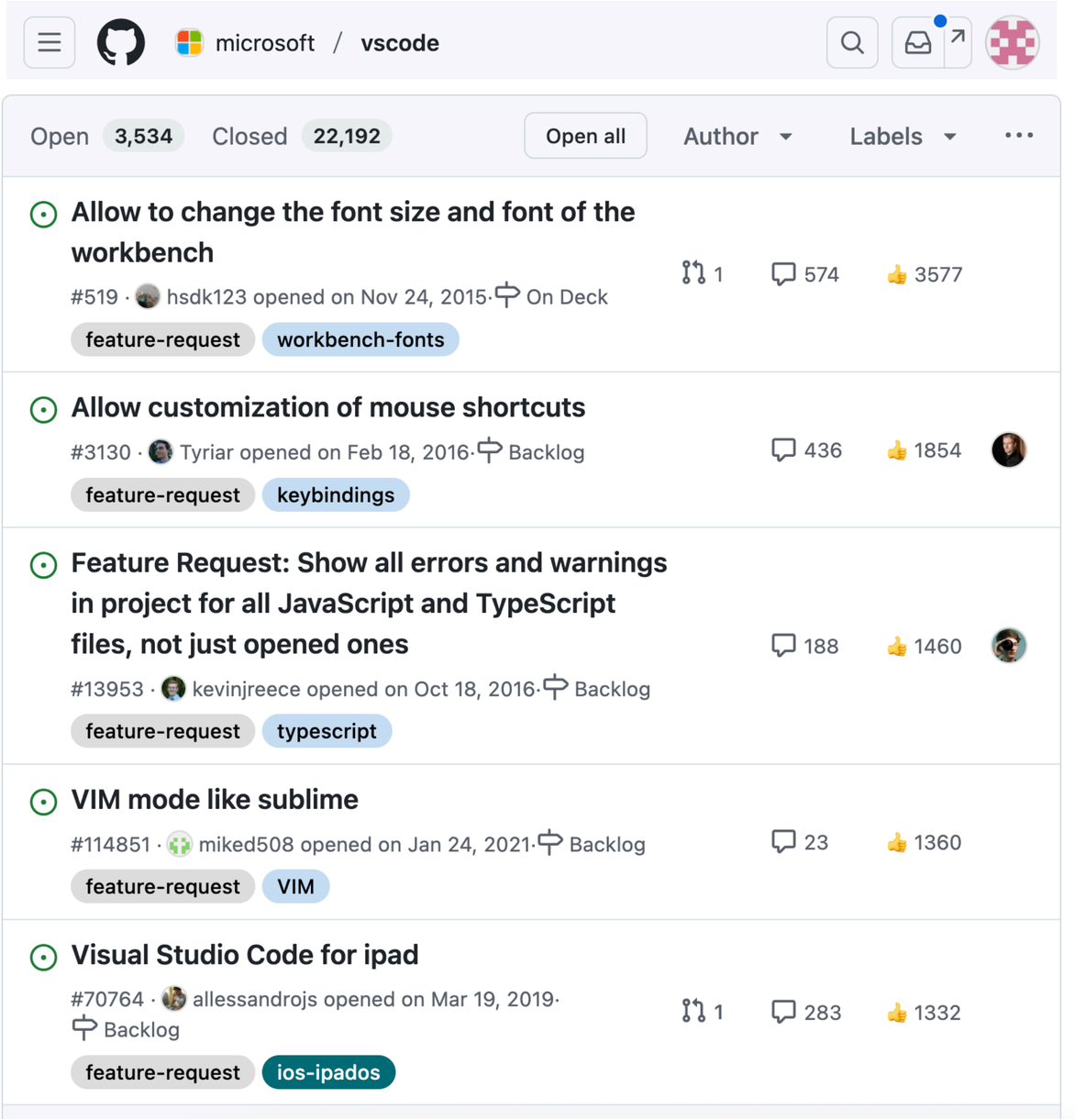
Key questions:

What does my client want?

What do my app users need?

Example. Conduct meetings and sign contracts with a client.

Example. Study market needs, survey actual or potential users, collect feature requests, etc.



VSCode Feature Requests

Accessed November, 21, 2025

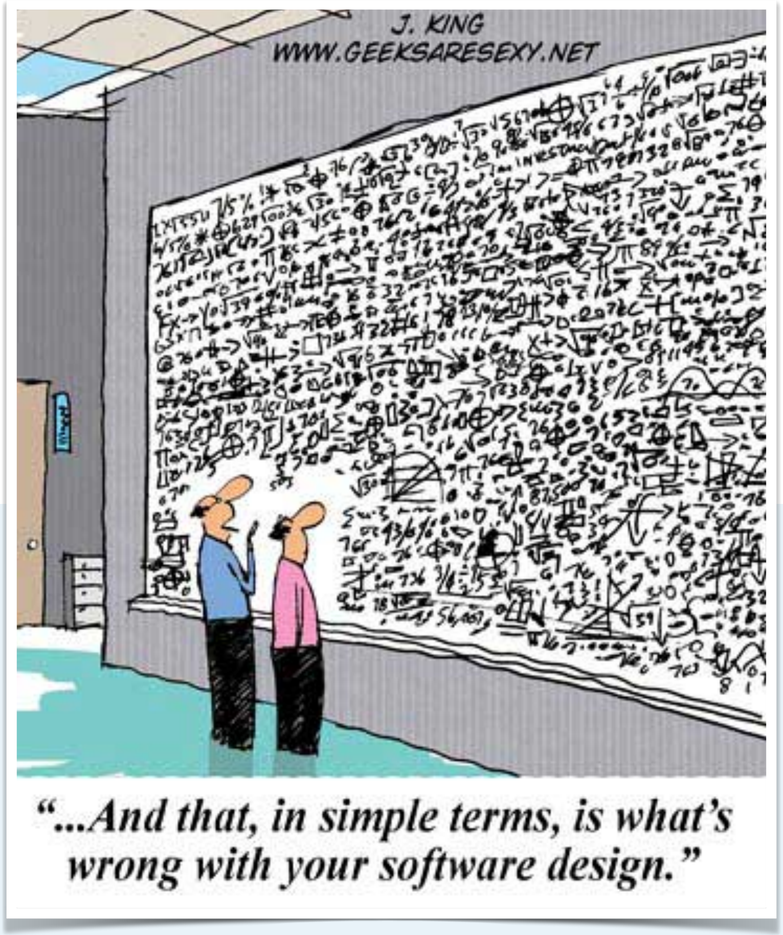
Examples of SE Tasks

Requirements Gathering
What does my client want?
What do my app users need?

Design

Key questions:
How should the software system be developed?
What should be done to:

- Ensure the system is *secure*, *reliable*, and *efficient*.
- Ensure the system can *scale* to a large number of users in the future.
- Ensure that *updates* and *new features* can be made at low cost.
- Allow *code reuse* during development and after development.
- etc.



Examples of SE Tasks

Requirements Gathering
What does my client want?
What do my app users need?

Design
How should the software system be developed?

Implementation
Actually writing the code!

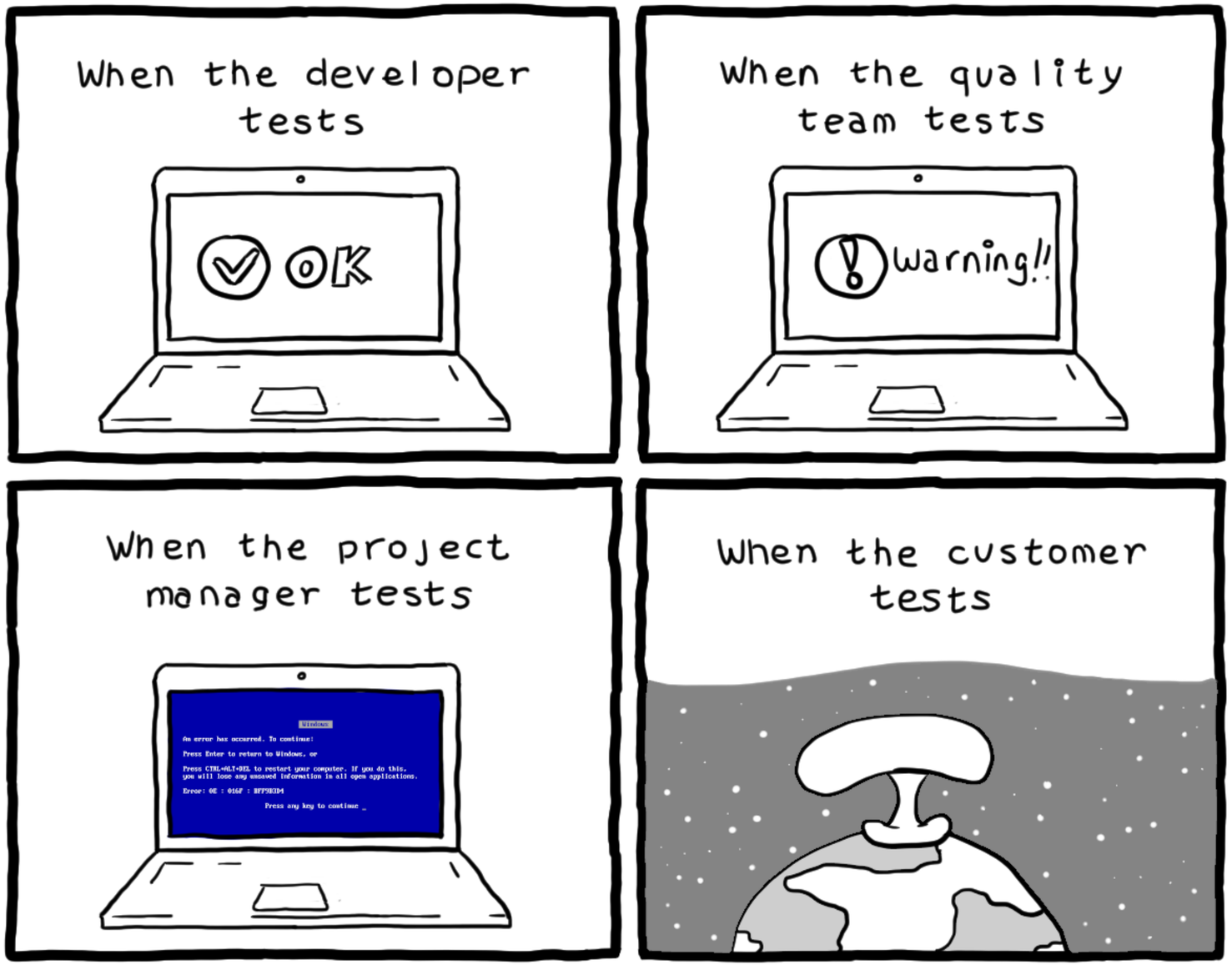
Examples of SE Tasks

Requirements Gathering
What does my client want?
What do my app users need?

Design
How should the software system be developed?

Implementation
Actually writing the code!

Testing
How do we ensure the software is bug-free?



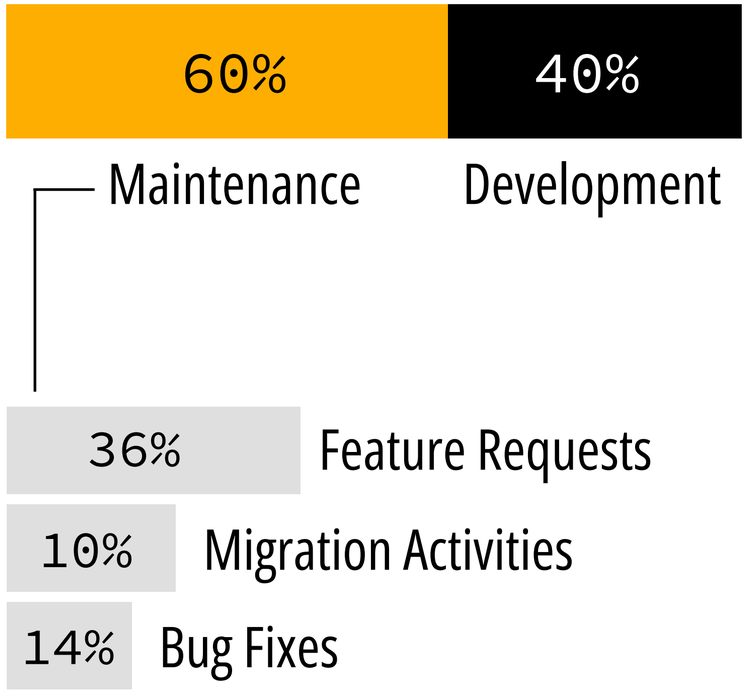
Examples of SE Tasks

Requirements Gathering
What does my client want?
What do my app users need?

Design
How should the software system be developed?

Implementation
Actually writing the code!

Testing
How do we ensure the software is bug-free?



Deployment, Maintenance, and Support
The larger portion of the work might be done in this phase!

A Sneak Peek at the **SE Study Plan** at PSUT

Common with Computer Science

- 3 11102 Introduction to Computer Science
- 3 11103 Structured Programming
- 3 11313 Algorithms Design and Analysis
- 3 11323 Database Systems
- 3 11206 Object Oriented Programming
- 3 11212 Data Structures and Introduction to Algorithms
- 3 12343 Visual Programming
- 3 12243 Webpage Design and Internet programming
- 3 22342 Computer Organization and Assembly Language
- 3 11335 Operating Systems
- 3 11435 Data Communications & Computer Networks
- 3 13211 Introduction to Software Engineering

Required for SE Majors

- 3 13325 Software Requirements Analysis
- 3 13324 System Analysis and Design
- 3 13212 Software Construction
- 3 13326 S.W. Eng. Approaches to Human Computer Interaction
- 3 13432 Software Project Management
- 3 13327 Sotware Design and Architecture
- 3 13428 Software Quality Assurance and Testing

A deep dive into each of the tasks performed by software engineers!

Shallow Dive # 1

Programming in a large team

Shallow Dive # 1

Programming in a large team

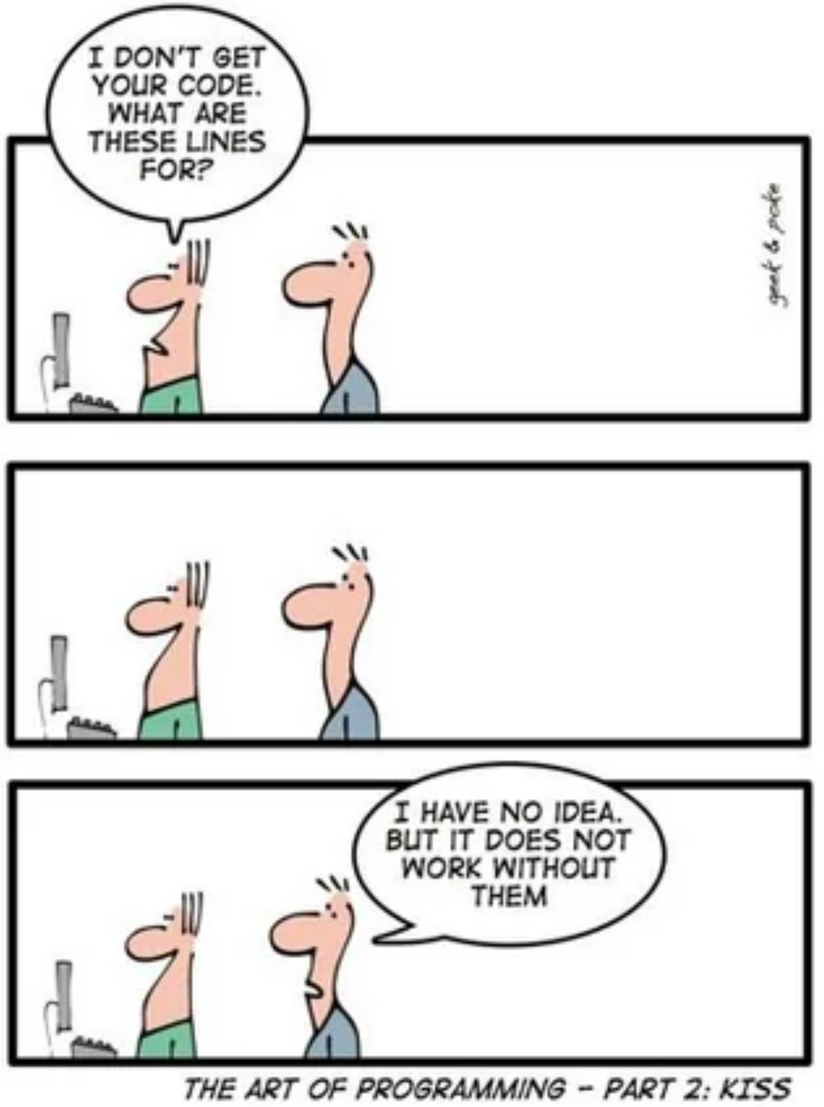
Coding Style

When working in a large team, your code will be read and used by others.

Most of the code you will be working with will be code written by others.

Teams typically agree on **coding style guidelines** for everyone to follow.

This makes reading code much easier!



Shallow Dive # 1

Programming in a large team

Coding Style

Example 1. PEP 8 Style Guide (a community standard)

https://peps.python.org/pep-0008/

Example 2. Google Coding Style Guidelines (used at Google and other places)

https://google.github.io/styleguide/pyguide.html

Shallow Dive # 1

Programming in a large team

Coding Style Example

PEP 8 Style Checker.

```
ialbluwi Documents $ pycodestyle test.py
test.py:2:17: E201 whitespace after '['
test.py:2:20: E203 whitespace before ','
test.py:2:25: E203 whitespace before ','
test.py:2:30: E202 whitespace before ']'
test.py:3:14: E225 missing whitespace around operator
test.py:4:21: E225 missing whitespace around operator
```

test.py

```
1   def compute(e1, e2, e3):
2       MyGrades = [ e1 , e2 , e3 ]
3       failgrade=50
4       if sum(MyGrades>50):
5           print('PASS')
```

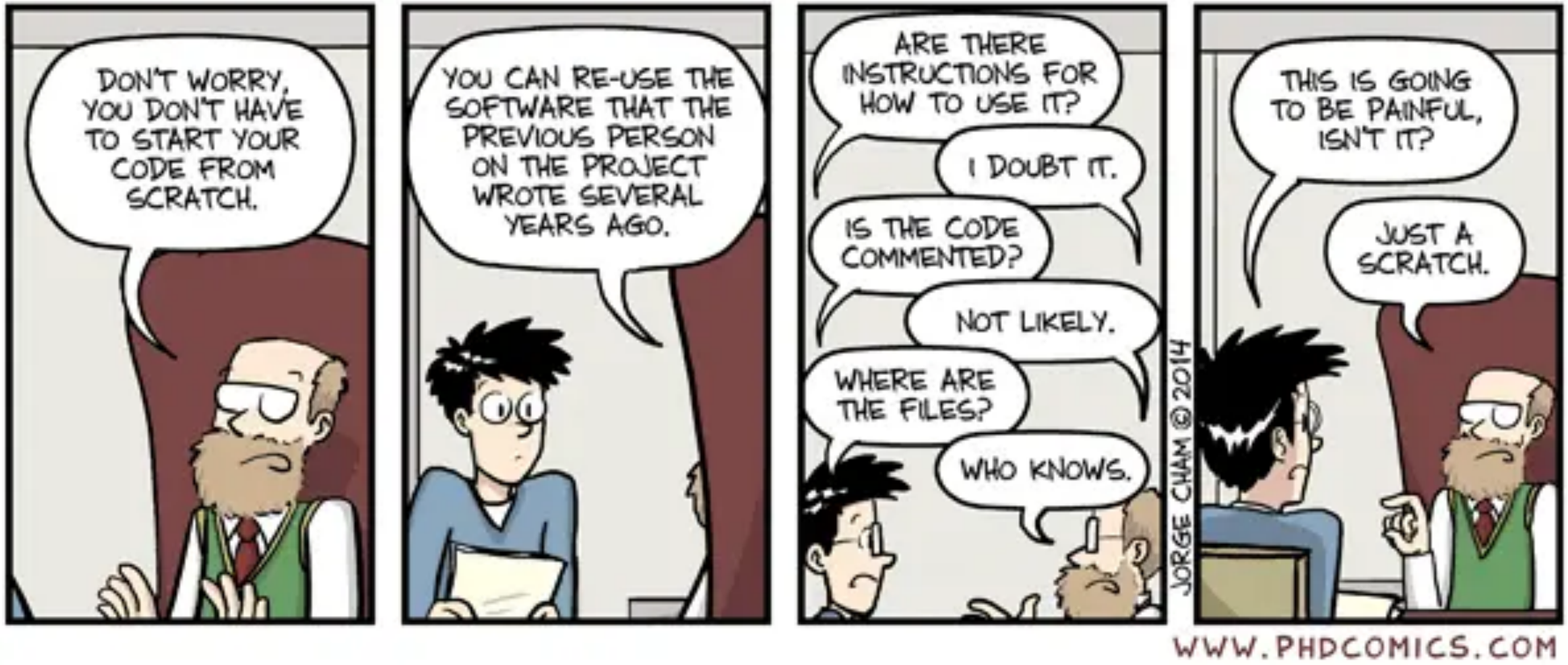
To install **pycodestyle** use the terminal:

- On Windows: `pip install pycodestyle`
- On Mac/Unix: `pip3 install pycodestyle`

Shallow Dive # 1

Programming in a large team

Documentation



Shallow Dive # 1

Programming in a large team

Documentation

Software engineers write lots of documentation! For example:

Code Documentation. Other team members need to understand the code. Even the person who wrote the code can forget as time passes!

Design Documentation. Team members need to communicate and justify designs and decisions to team leads (and the other way around, too!)

Onboarding Guides. New team members need guidance on how to navigate and deal with the code base!

User Documentation

Users of the system need to know how to use it!

Shallow Dive # 1

Programming in a large team

Code Documentation Example

Python provides a tool for automatically extracting documentation from code.

```
1  def is_palindrome(word):
2      """Check if a word can be read the same forwards and backwards."""
3      return word == word[::-1]
4
5  def is_sorted(lst):
6      """Check if a list is sorted in ascending order."""
7      for i in range(len(lst) - 1):
8          if lst[i] > lst[i + 1]:
9              return False
10     return True
```

A **Docstring**
(marked by triple quotes """ … """)

Shallow Dive # 1

Programming in a large team

Code Documentation Example

Python provides a tool for automatically extracting documentation from code!

```
ialbluwi Documents $ pydoc test
```

automatically reads the docstrings and generates the following:

```
Help on module test:

NAME
    test

FUNCTIONS
    is_palindrome(word)
        Check if a word can be read the same forwards and backwards.

    is_sorted(lst)
        Check if a list is sorted in ascending order.
```

Shallow Dive # 1

Programming in a large team

Code Documentation Example

Python provides a tool for automatically extracting documentation from code!

Try:

`ialbluwi Documents $ pydoc math`	Shows the documentation for `math`
`ialbluwi Documents $ pydoc random`	Shows the documentation for `random`
`ialbluwi Documents $ pydoc random.randint`	Shows the documentation for `randint`
`ialbluwi Documents $ pydoc -w random`	Creates an HTML file containing the documentation of `random`

A library that provides better HTML output is: `pdoc`
(requires installation: `pip install pdoc` then `pdoc filename.py`)

Shallow Dive # 1

Programming in a large team

Version Control

Teams need to:

- Keep track of *who changed what*.
- Manage *synchronous edits* to the same file.
- Allow *going back* to older versions.
- Allow *branching out* to try features without affecting others.

Example. Git and GitHub

Stay tuned for **Assignment 3!**