Final Lecture: Advanced Software Design and Development

Introduction

- Wrapping up key topics and introducing advanced tools for robust software development.

Learning Objectives

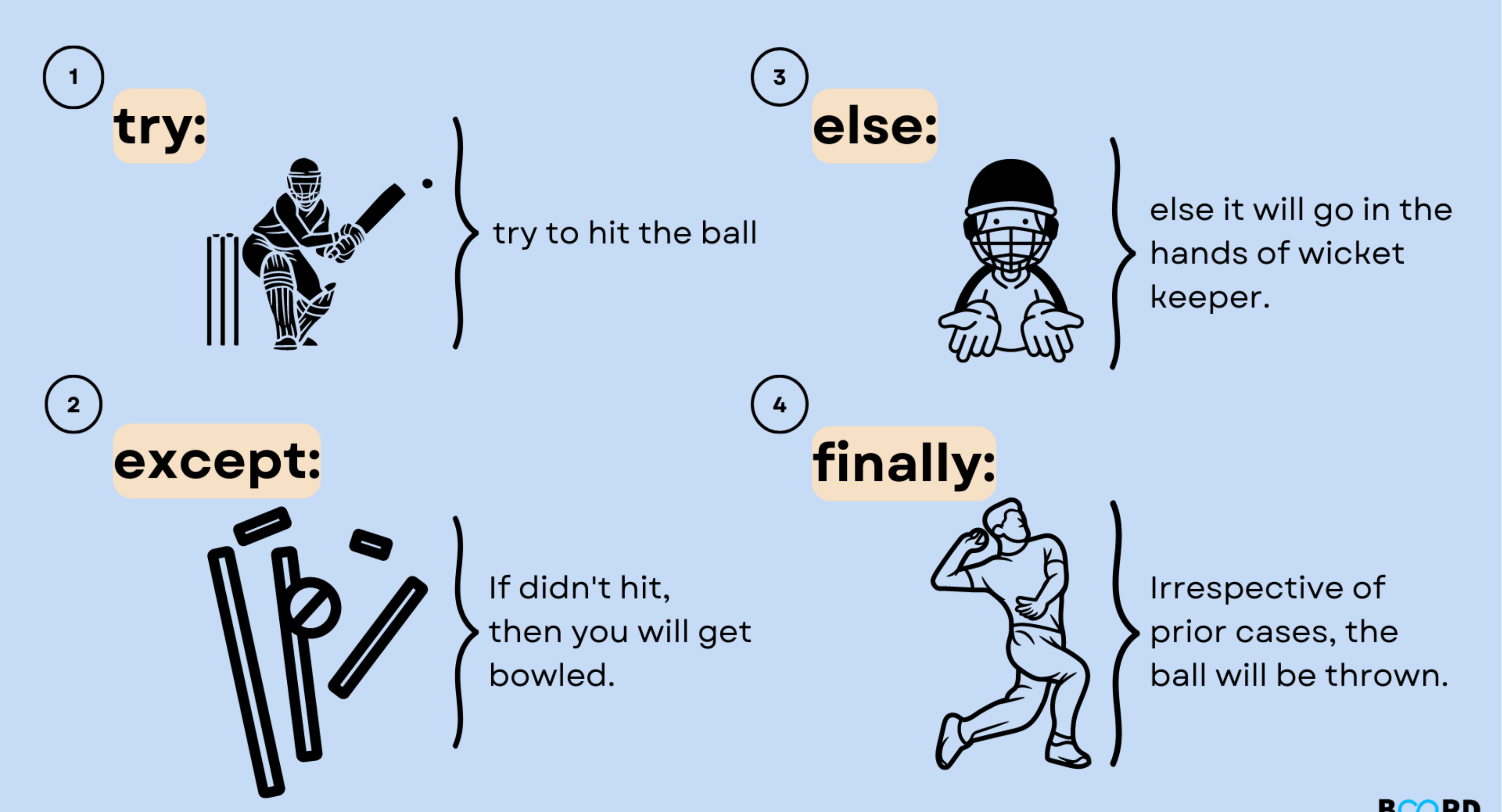
- 1. Strengthen programming skills.
- 2. Write robust, maintainable, and secure code.
- 3. Collaborate effectively using version control.
- 4. Test-driven development.

1. Defensive Coding

- Importance of writing code that anticipates and handles invalid inputs and errors.

Exception Handling

- Use try-except blocks to gracefully handle unforeseen runtime errors.



1 **try:**

try to hit the ball

2 **except:**

If didn't hit,
then you will get
bowled.

3 **else:**

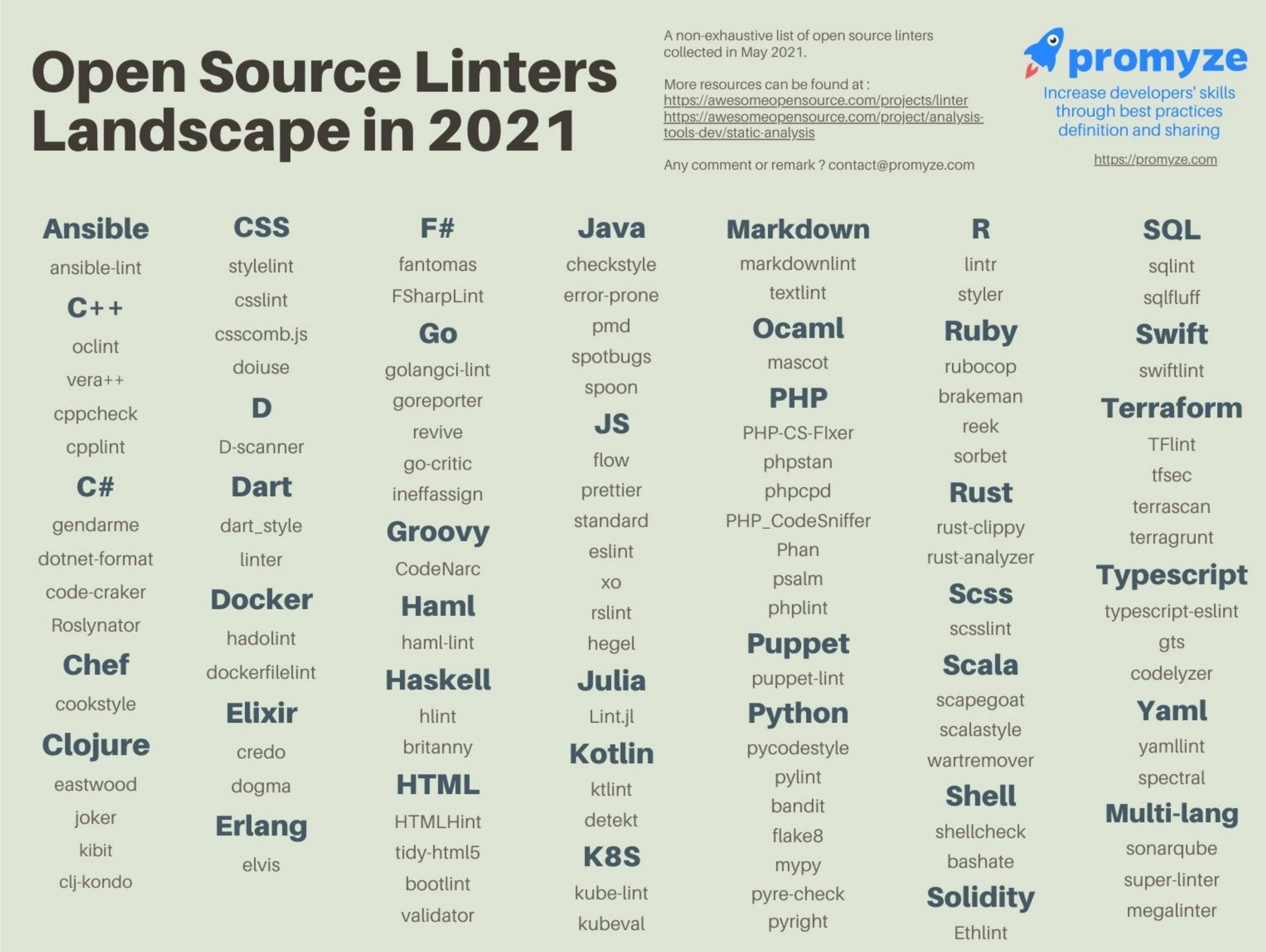
else it will go in the
hands of wicket
keeper.

4 **finally:**

Irrespective of
prior cases, the
ball will be thrown.

Static Code Analysis

- Tools like SonarQube and PyLint help detect bugs and code smells early.



Open Source Linters Landscape in 2021

A non-exhaustive list of open source linters collected in May 2021.

More resources can be found at :
https://awesomeopensource.com/projects/linter
https://awesomeopensource.com/project/analysis-tools-dev/static-analysis

Any comment or remark ? contact@promyze.com

promyze

Increase developers' skills through best practices definition and sharing

https://promyze.com

Ansible

ansible-lint

C++

oclint

vera++

cppcheck

cpplint

C#

gendarme

dotnet-format

code-craker

Roslynator

Chef

cookstyle

Clojure

eastwood

joker

kibit

clj-kondo

CSS

stylelint

csslint

csscomb.js

doiuse

D

D-scanner

Dart

dart_style

linter

Docker

hadolint

dockerfilelint

Elixir

credo

dogma

Erlang

elvis

F#

fantomas

FSharpLint

Go

golangci-lint

goreporter

revive

go-critic

ineffassign

Groovy

CodeNarc

Haml

haml-lint

Haskell

hlint

britanny

HTML

HTMLHint

tidy-html5

bootlint

validator

Java

checkstyle

error-prone

pmd

spotbugs

spoon

JS

flow

prettier

standard

eslint

xo

rslint

hegel

Julia

Lint.jl

Kotlin

ktlint

detekt

K8S

kube-lint

kubeval

Markdown

markdownlint

textlint

Ocaml

mascot

PHP

PHP-CS-FIxer

phpstan

phpcpd

PHP_CodeSniffer

Phan

psalm

phplint

Puppet

puppet-lint

Python

pycodestyle

pylint

bandit

flake8

mypy

pyre-check

pyright

R

lintr

styler

Ruby

rubocop

brakeman

reek

sorbet

Rust

rust-clippy

rust-analyzer

Scss

scsslint

Scala

scapegoat

scalastyle

wartremover

Shell

shellcheck

bashate

Solidity

Ethlint

SQL

sqlint

sqlfluff

Swift

swiftlint

Terraform

TFlint

tfsec

terrascan

terragrunt

Typescript

typescript-eslint

gts

codelyzer

Yaml

yamllint

spectral

Multi-lang

sonarqube

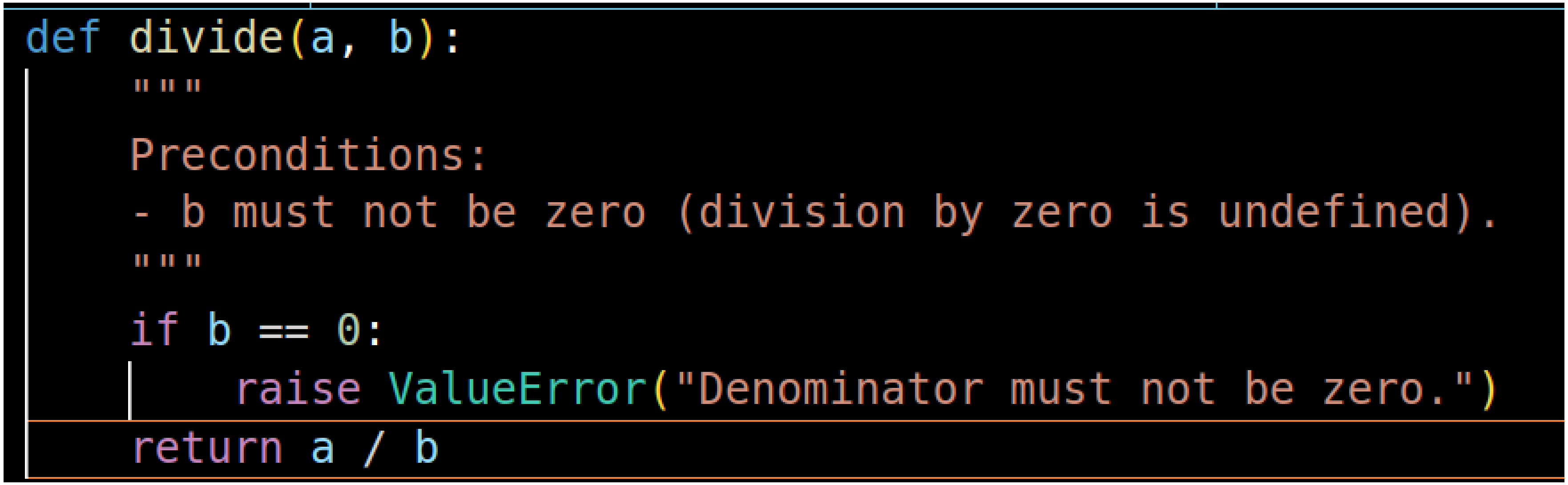
super-linter

megalinter

Design by Contract

- Define preconditions, postconditions, and invariants to ensure code correctness.

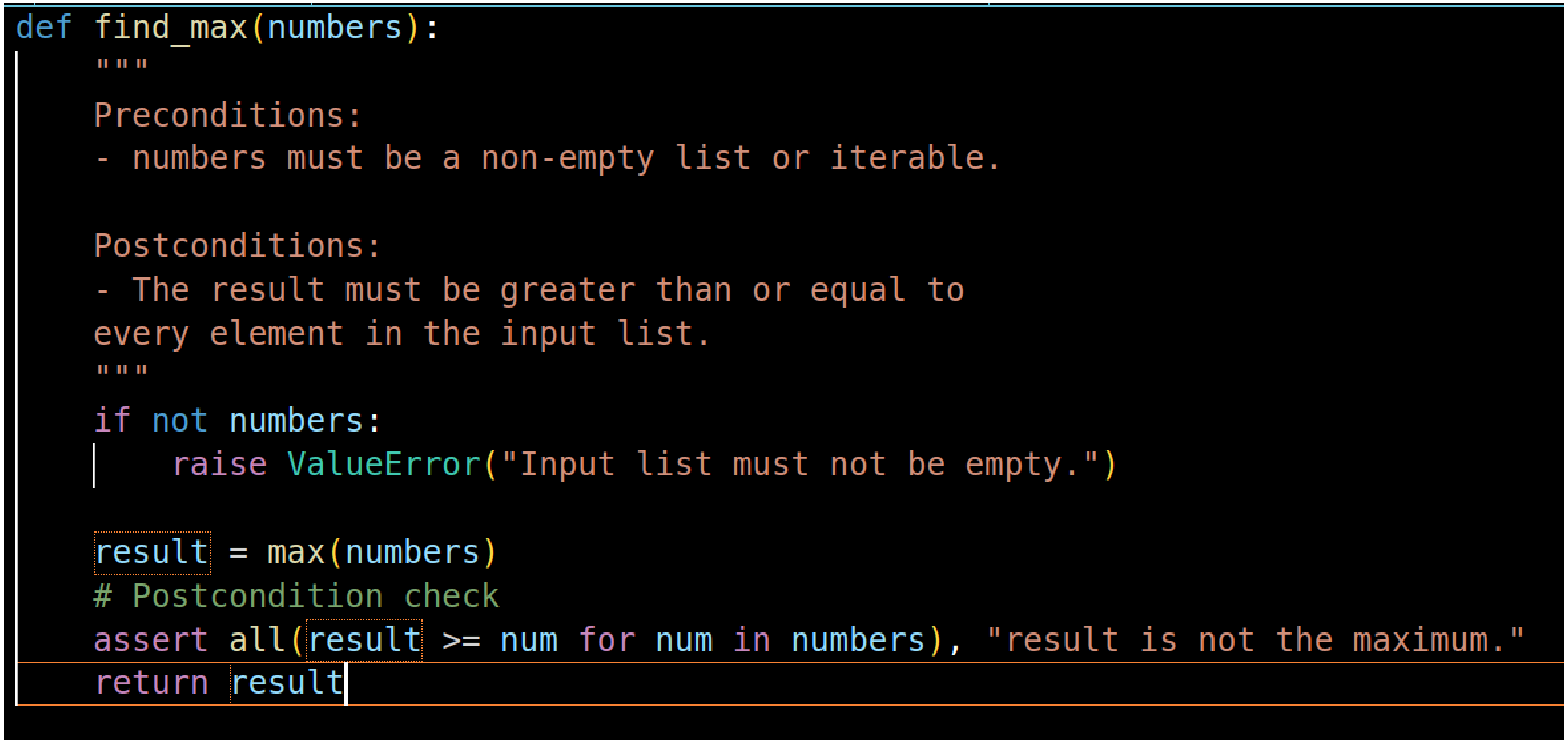
Preconditions



```python
def divide(a, b):
    """
    Preconditions:
    - b must not be zero (division by zero is undefined).
    """
    if b == 0:
        raise ValueError("Denominator must not be zero.")
    return a / b
```

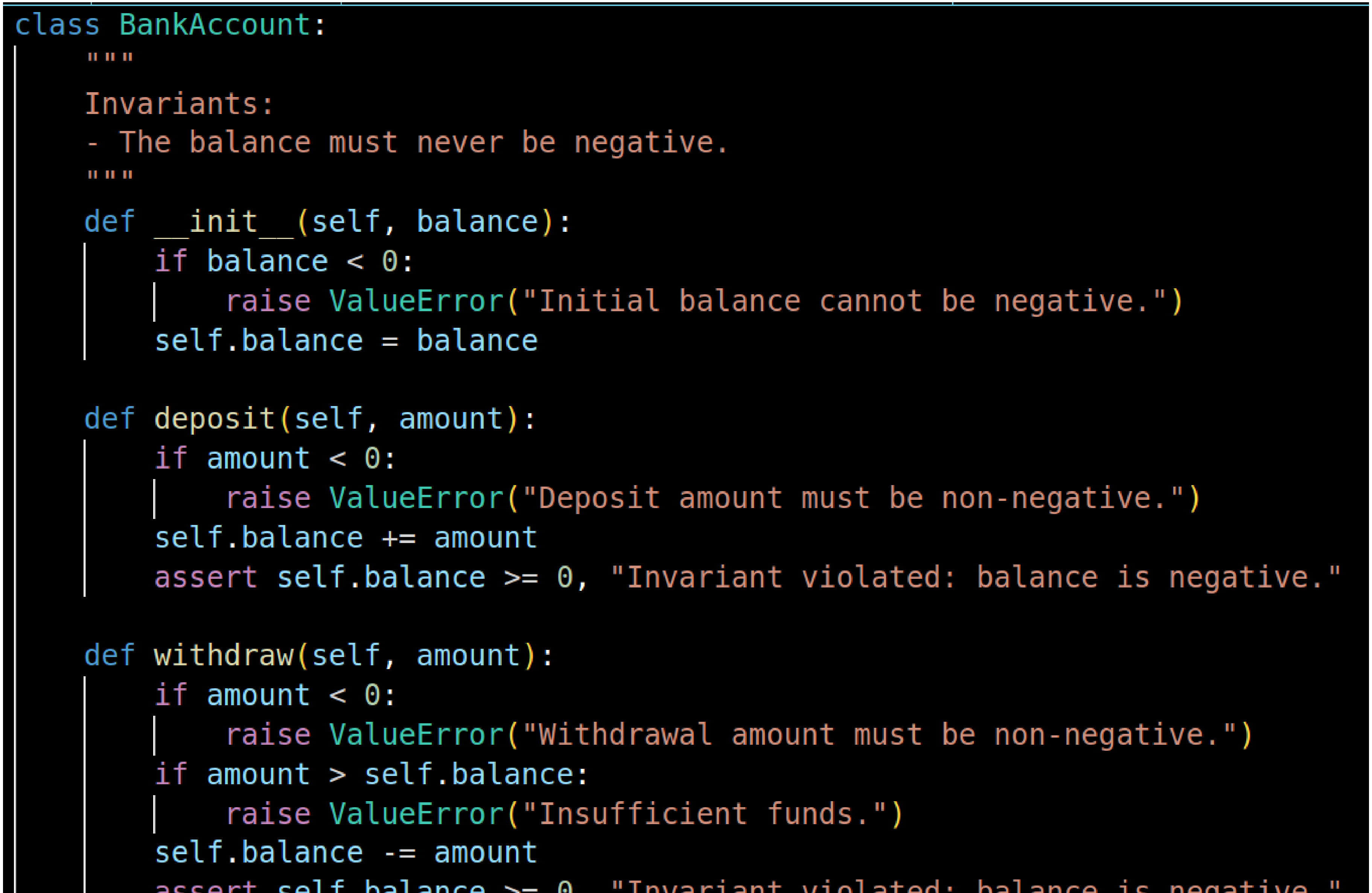
postconditions

- Define preconditions, postconditions, and invariants to ensure code correctness.



```python
def find_max(numbers):
    """
    Preconditions:
    - numbers must be a non-empty list or iterable.

    Postconditions:
    - The result must be greater than or equal to
    every element in the input list.
    """
    if not numbers:
        raise ValueError("Input list must not be empty.")

    result = max(numbers)
    # Postcondition check
    assert all(result >= num for num in numbers), "result is not the maximum."
    return result
```

invariants



```python
class BankAccount:
    """
    Invariants:
    - The balance must never be negative.
    """
    def __init__(self, balance):
        if balance < 0:
            raise ValueError("Initial balance cannot be negative.")
        self.balance = balance

    def deposit(self, amount):
        if amount < 0:
            raise ValueError("Deposit amount must be non-negative.")
        self.balance += amount
        assert self.balance >= 0, "Invariant violated: balance is negative."

    def withdraw(self, amount):
        if amount < 0:
            raise ValueError("Withdrawal amount must be non-negative.")
        if amount > self.balance:
            raise ValueError("Insufficient funds.")
        self.balance -= amount
        assert self.balance >= 0, "Invariant violated: balance is negative."
```

2. Modular Design

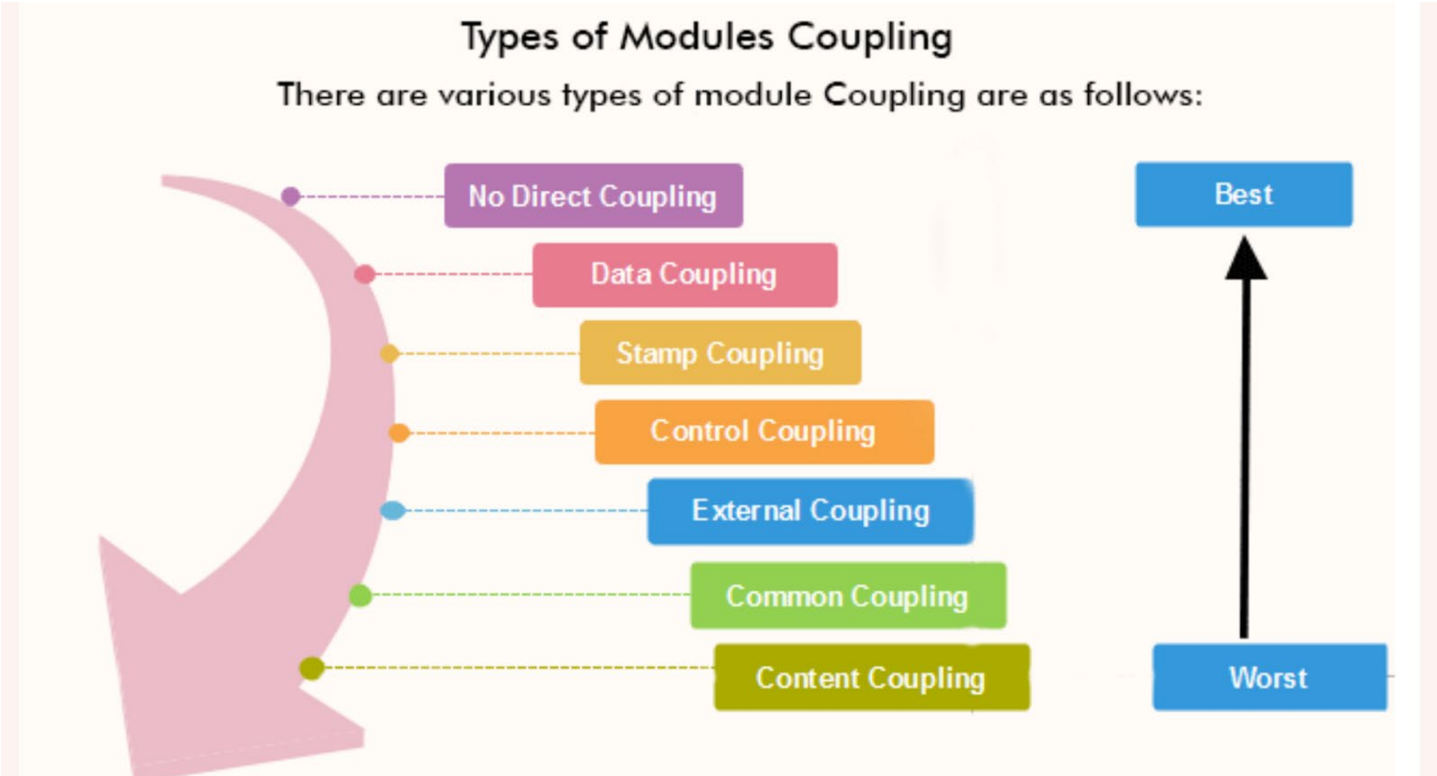
- Principles of high cohesion and low coupling for scalable systems.

Cohesion

- Modules should be organised around a single, well-defined purpose.

Coupling

- Minimise dependencies between modules to simplify testing and maintenance.

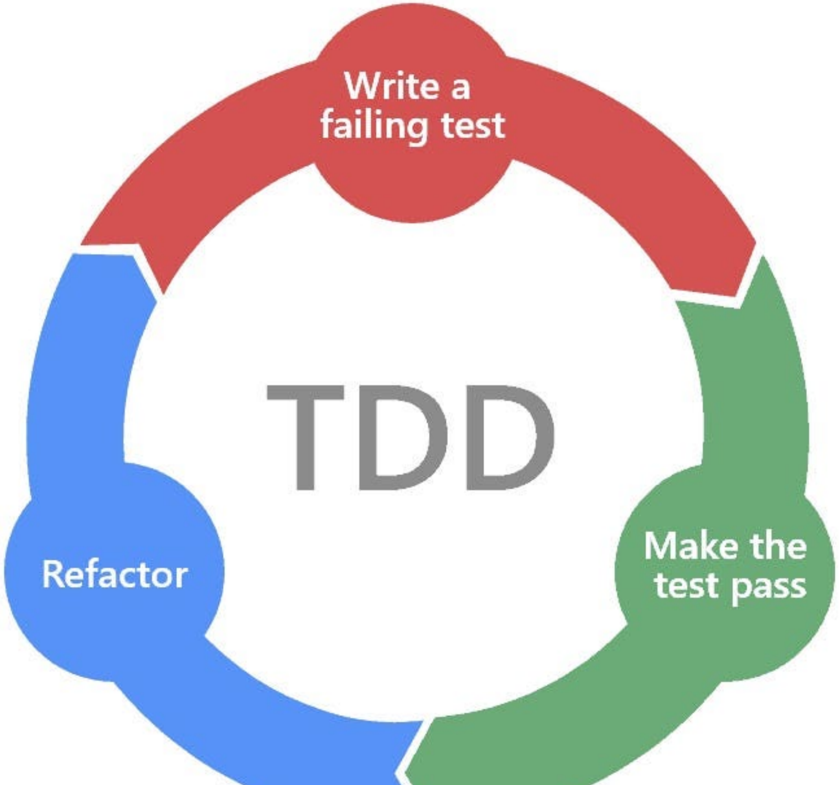


Types of Modules Coupling

There are various types of module Coupling are as follows:

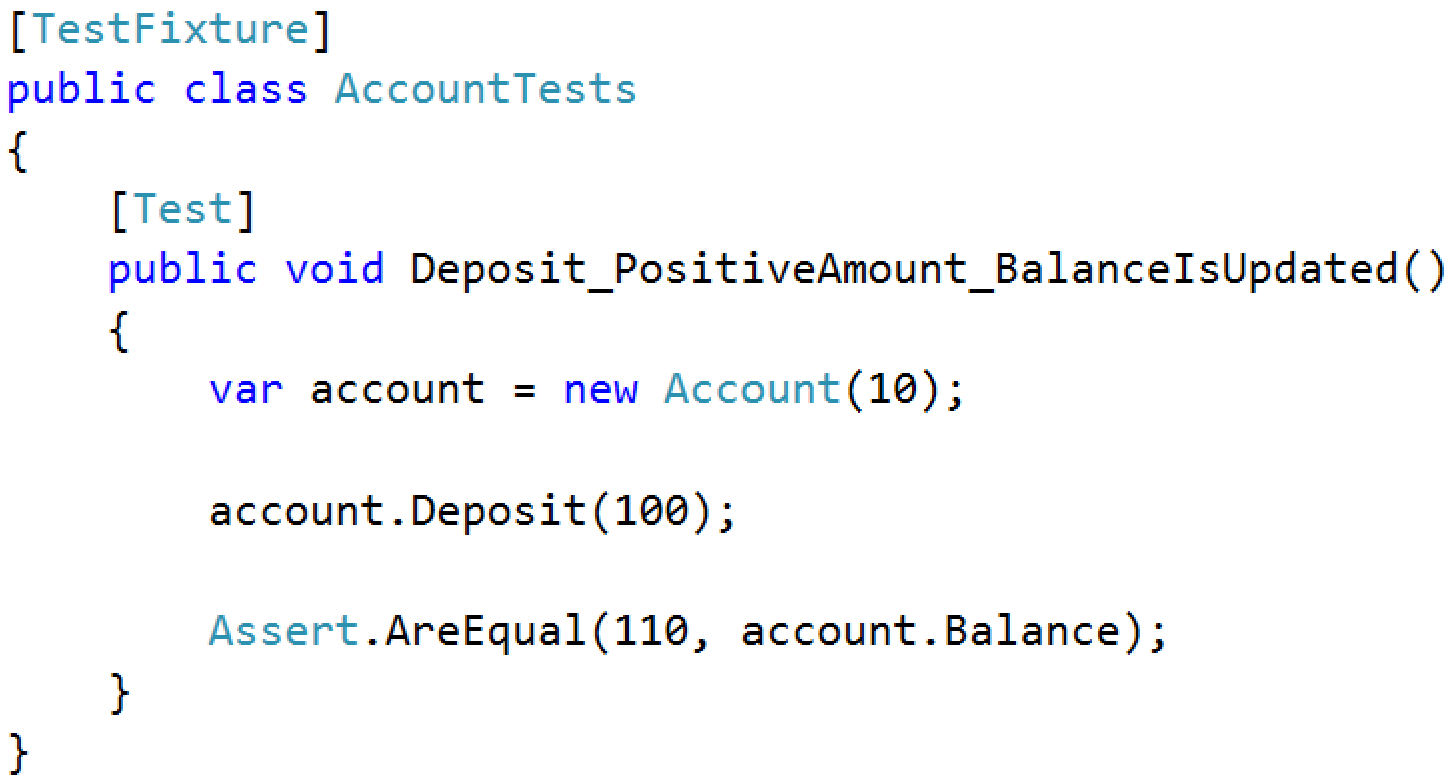
3. Test-Driven Development (TDD)

- Write tests before implementing functionality to ensure requirements are met.



Unit Tests

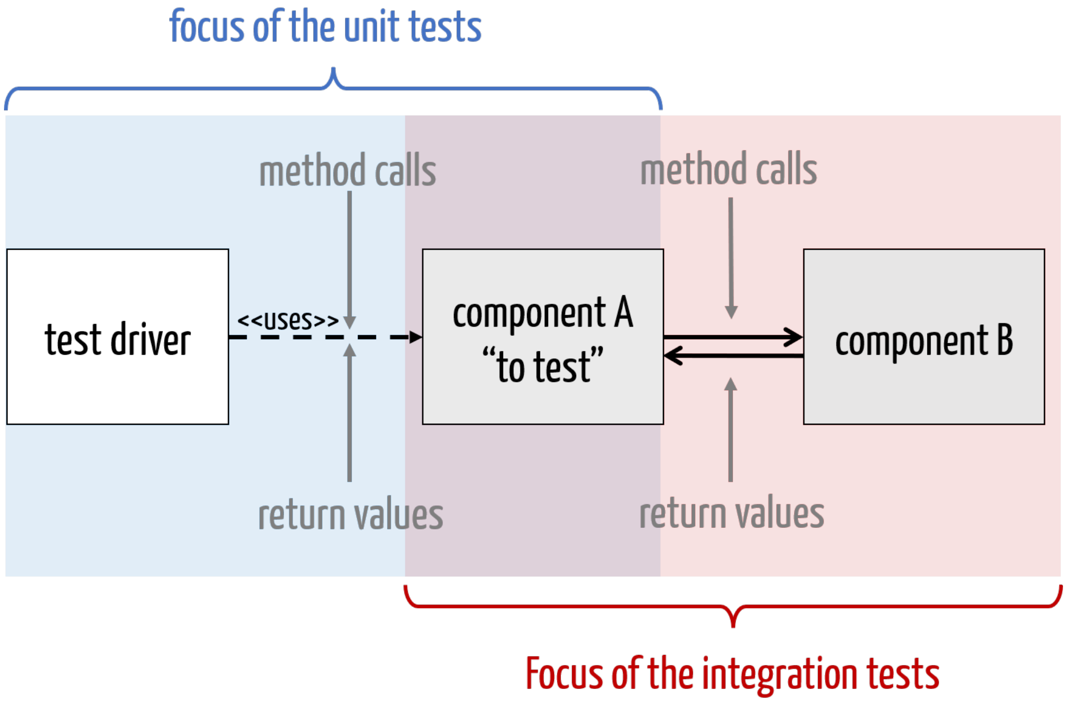
- Isolate and test small parts of the codebase using frameworks like PyTest.



```
[TestFixture]
public class AccountTests
{
    [Test]
    public void Deposit_PositiveAmount_BalanceIsUpdated()
    {
        var account = new Account(10);

        account.Deposit(100);

        Assert.AreEqual(110, account.Balance);
    }
}
```

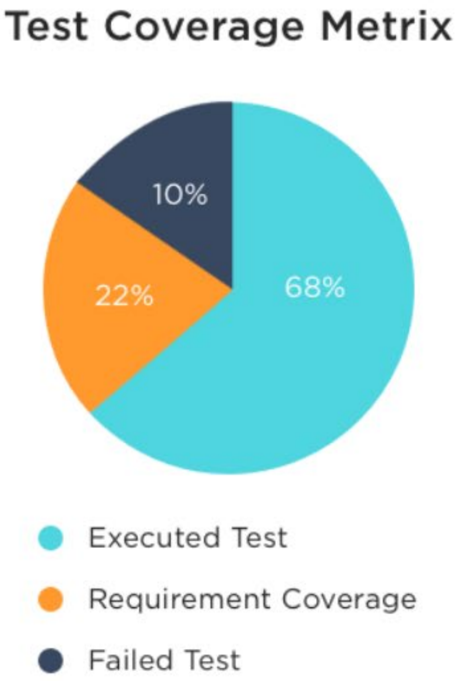
Integration Tests

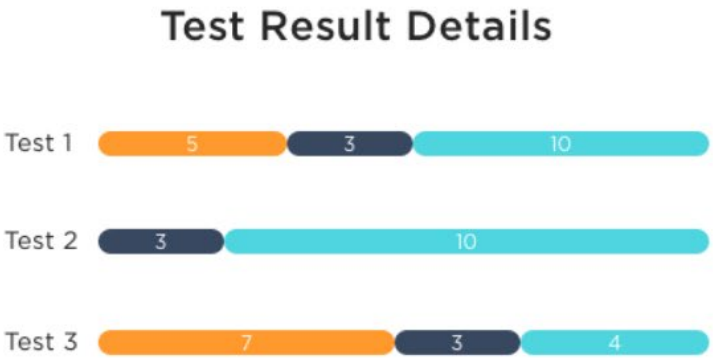
- Ensure that modules interact correctly using tools like Selenium.



Code Coverage

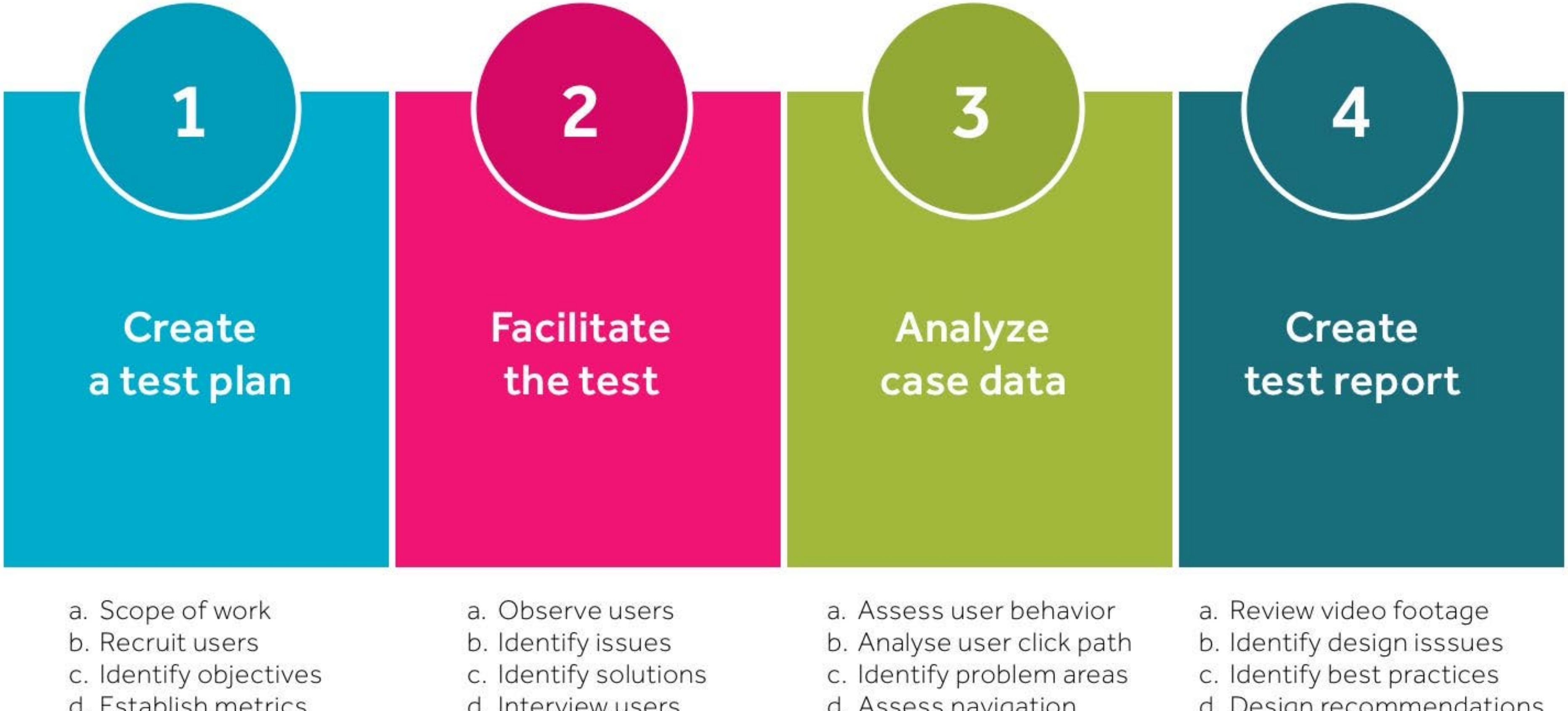
- Measure test coverage with tools like Coverage.py.





4. User Testing

- Involve users early and often to identify usability issues.



1

Create a test plan

a. Scope of work
b. Recruit users
c. Identify objectives
d. Establish metrics

2

Facilitate the test

a. Observe users
b. Identify issues
c. Identify solutions
d. Interview users

3

Analyze case data

a. Assess user behavior
b. Analyse user click path
c. Identify problem areas
d. Assess navigation

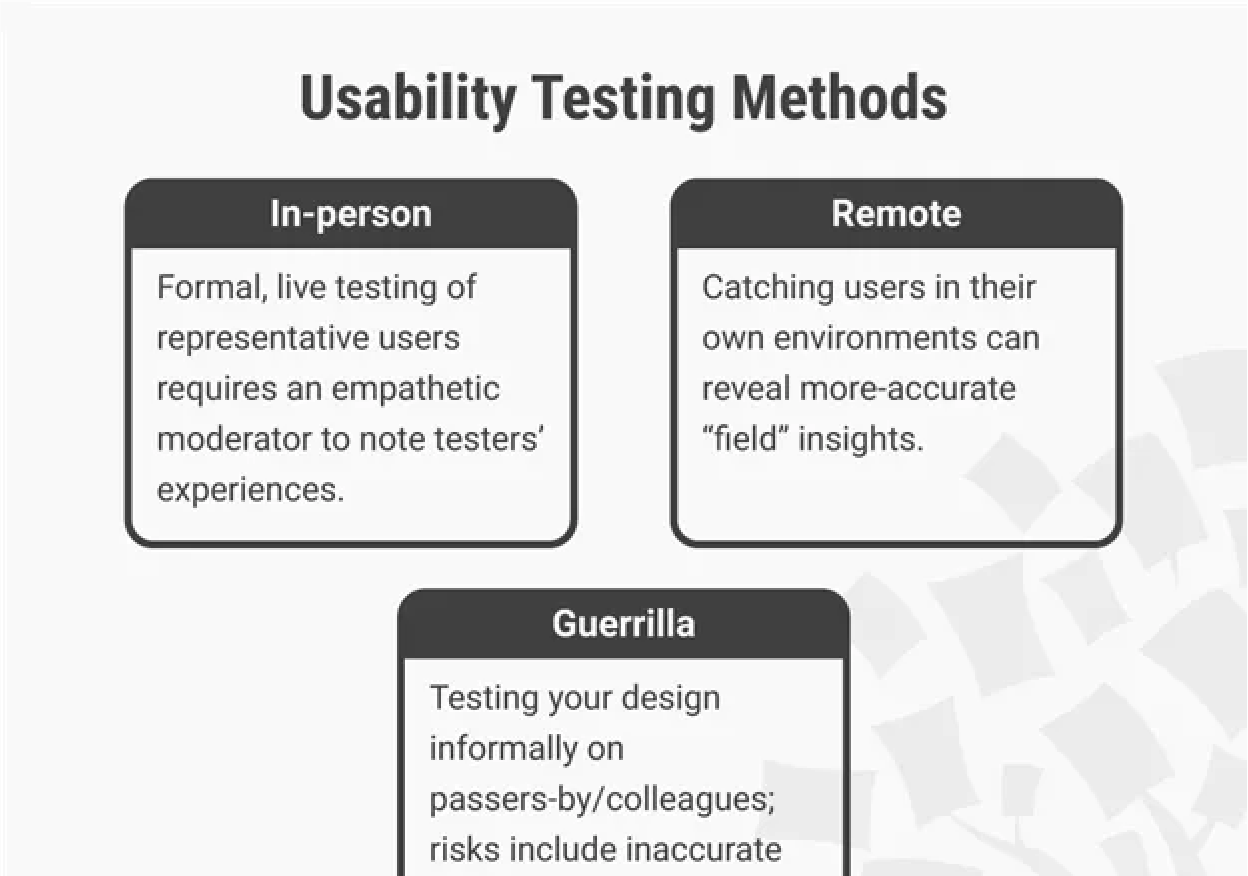
4

Create test report

a. Review video footage
b. Identify design isssues
c. Identify best practices
d. Design recommendations

Usability Testing

- Evaluate user interactions to improve interface design.



Usability Testing Methods

In-person

Formal, live testing of representative users requires an empathetic moderator to note testers' experiences.

Remote

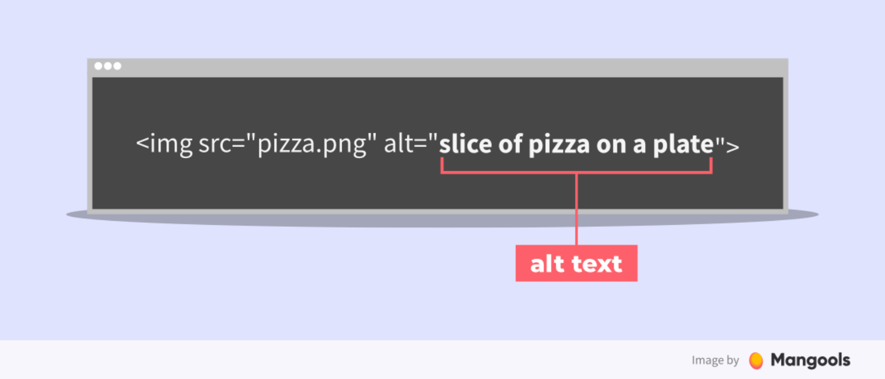
Catching users in their own environments can reveal more-accurate "field" insights.

Guerrilla

Testing your design informally on passers-by/colleagues; risks include inaccurate

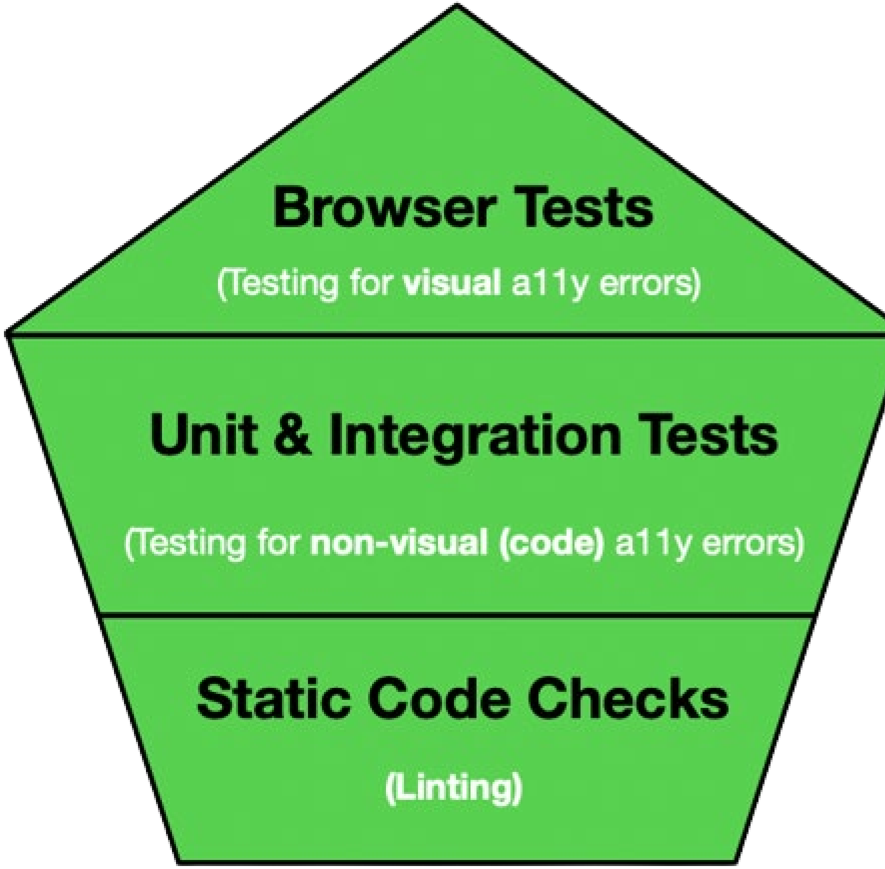
Accessibility Testing

- Ensure the software is usable by people with disabilities.



```
<img src="pizza.png" alt="slice of pizza on a plate">
```

Image by Mangools



Browser Tests

(Testing for **visual** a11y errors)

Unit & Integration Tests

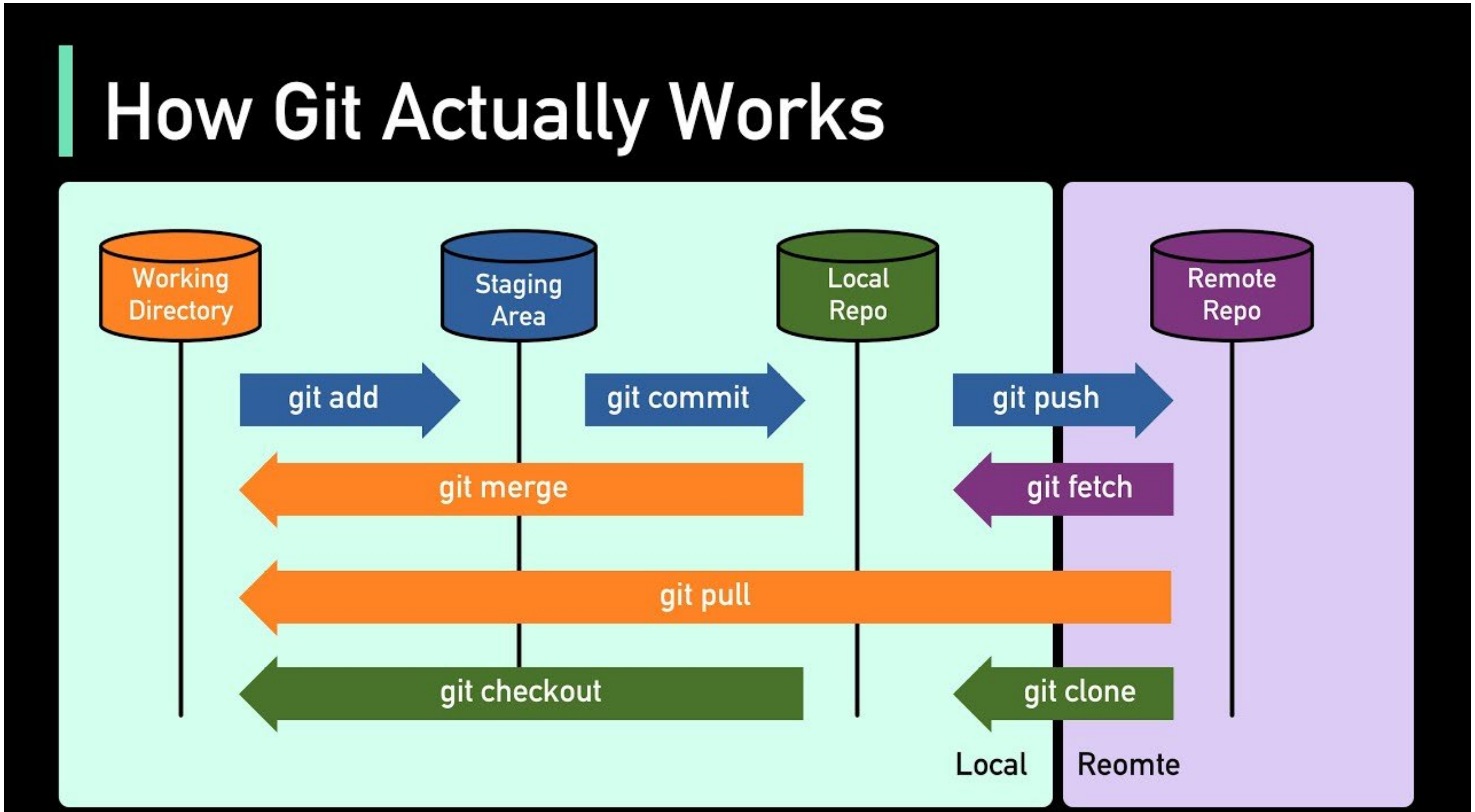
(Testing for **non-visual (code)** a11y errors)

Static Code Checks

(Linting)

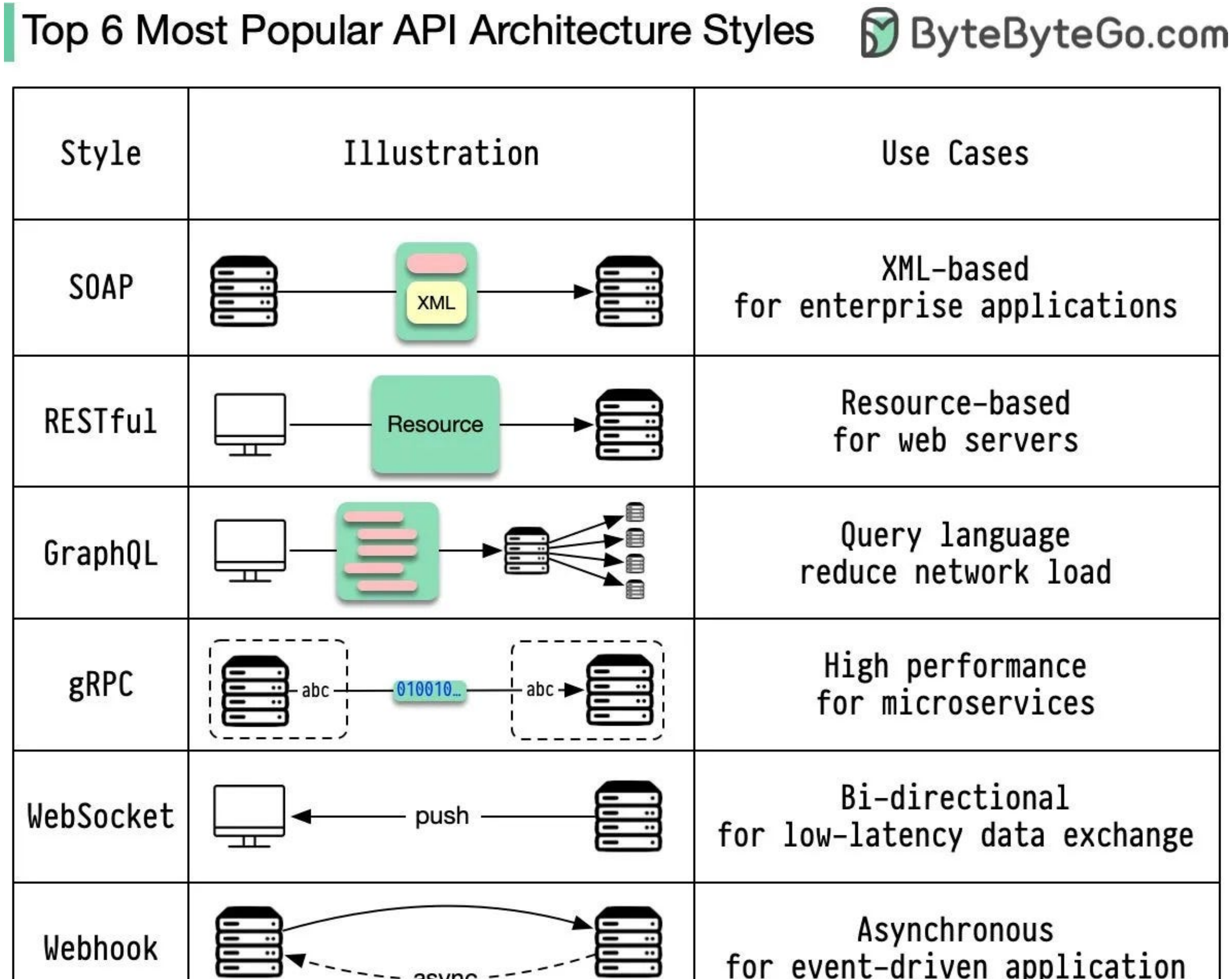
5. Version Control Best Practices

- Use Git effectively for individual and collaborative projects.



6. API Design and Interaction

- Use APIs to communicate between systems.



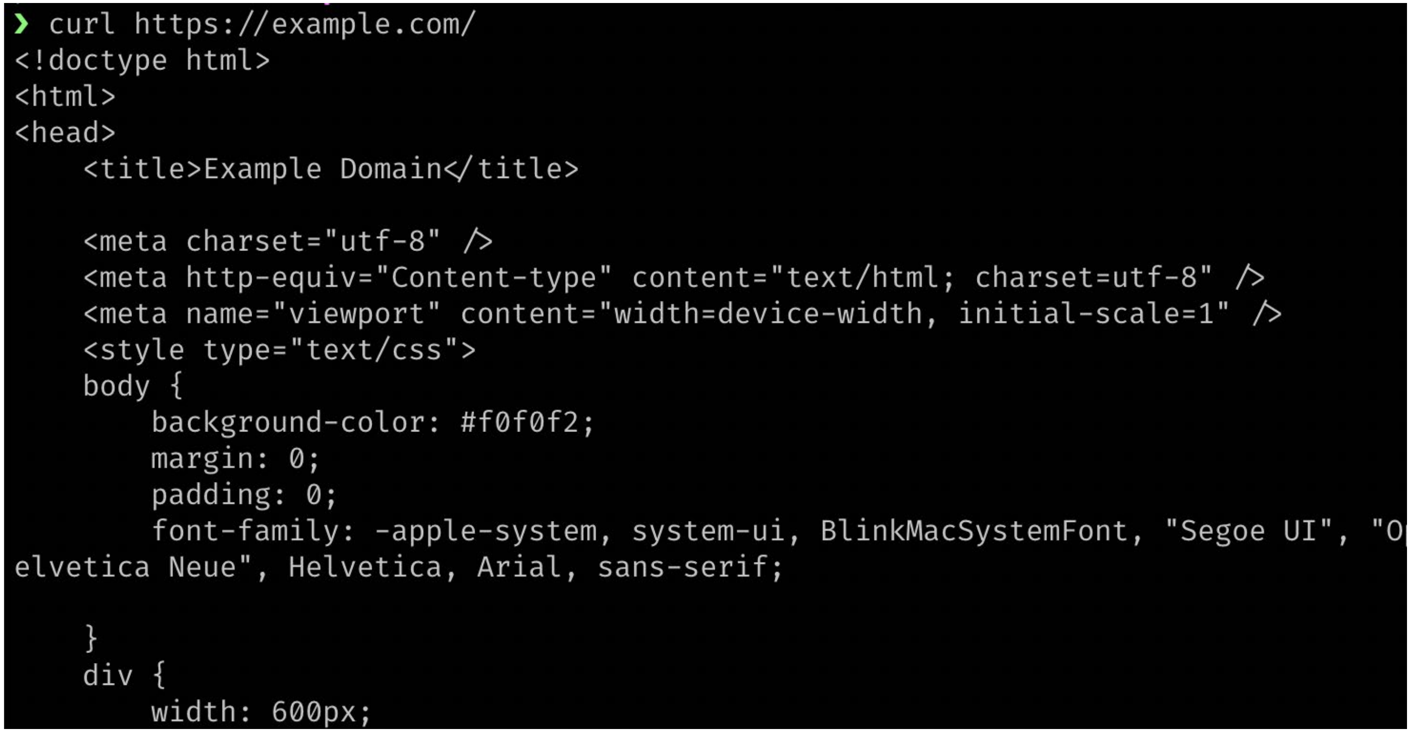
Top 6 Most Popular API Architecture Styles

ByteByteGo.com

Style	Illustration	Use Cases
SOAP	XML	XML-based for enterprise applications
RESTful	Resource	Resource-based for web servers
GraphQL		Query language reduce network load
gRPC	abc 010010... abc	High performance for microservices
WebSocket	push	Bi-directional for low-latency data exchange
Webhook	async	Asynchronous for event-driven application

cURL Basics

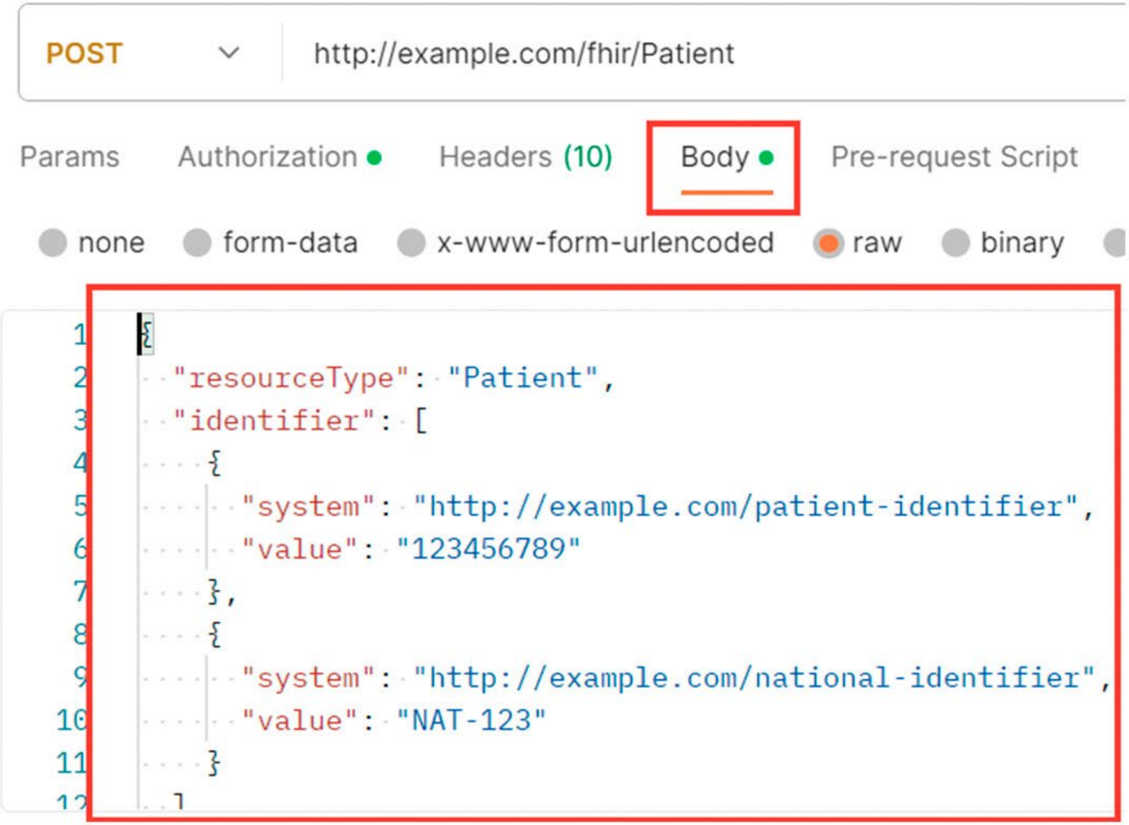
- Send HTTP requests and handle responses from APIs.



```
❯ curl https://example.com/
<!doctype html>
<html>
<head>
    <title>Example Domain</title>

    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
    body {
        background-color: #f0f0f2;
        margin: 0;
        padding: 0;
        font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "O
elvetica Neue", Helvetica, Arial, sans-serif;

    }
    div {
        width: 600px;
```

Postman and Insomnia

- Test and debug APIs using GUI tools.



POST http://example.com/fhir/Patient

○ none ○ form-data ○ x-www-form-urlencoded ● raw ○ binary

```
{
  "resourceType": "Patient",
  "identifier": [
    {
      "system": "http://example.com/patient-identifier",
      "value": "123456789"
    },
    {
      "system": "http://example.com/national-identifier",
      "value": "NAT-123"
    }
  ]
```

Continuous Integration

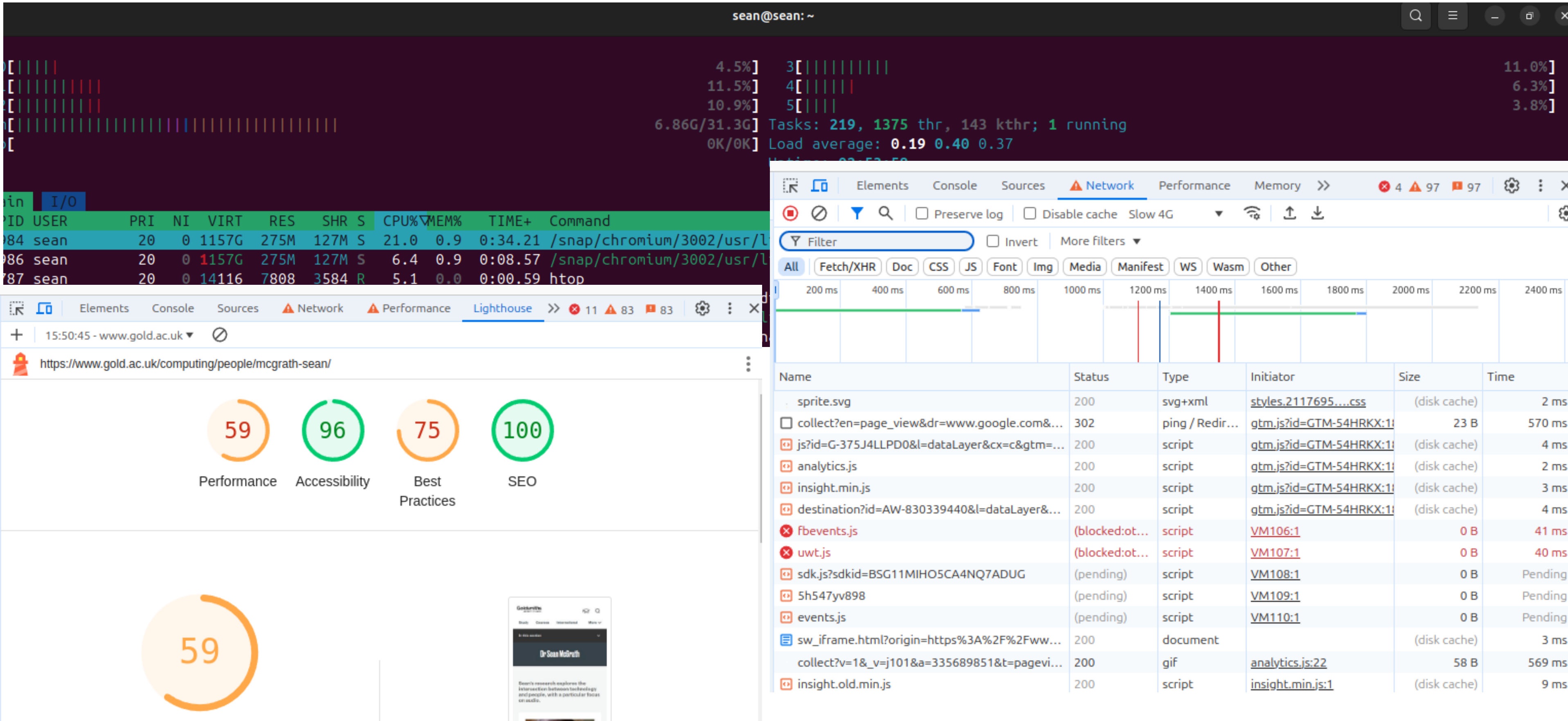
- Merge code frequently and test changes automatically.

Continuous Deployment

- Deploy tested code to production automatically.

8. Profiling and Performance

- Identify bottlenecks and optimise code performance.



```
[|||||                                   4.5%]   3[||||||||||                11.0%]
[|||||||||                              11.5%]   4[||||||                     6.3%]
[|||||||||                              10.9%]   5[||||                       3.8%]
[|||||||||||||||||||||||||||||||||||    6.86G/31.3G] Tasks: 219, 1375 thr, 143 kthr; 1 running
[                                        0K/0K] Load average: 0.19 0.40 0.37

in  I/O
ID USER       PRI  NI  VIRT   RES   SHR S  CPU%▽MEM%   TIME+  Command
84 sean        20   0 1157G  275M  127M S  21.0  0.9  0:34.21 /snap/chromium/3002/usr/l
86 sean        20   0 1157G  275M  127M S   6.4  0.9  0:08.57 /snap/chromium/3002/usr/l
87 sean        20   0 14116  7808  3584 R   5.1  0.0  0:00.59 htop
```

https://www.gold.ac.uk/computing/people/mcgrath-sean/

59 Performance

96 Accessibility

75 Best Practices

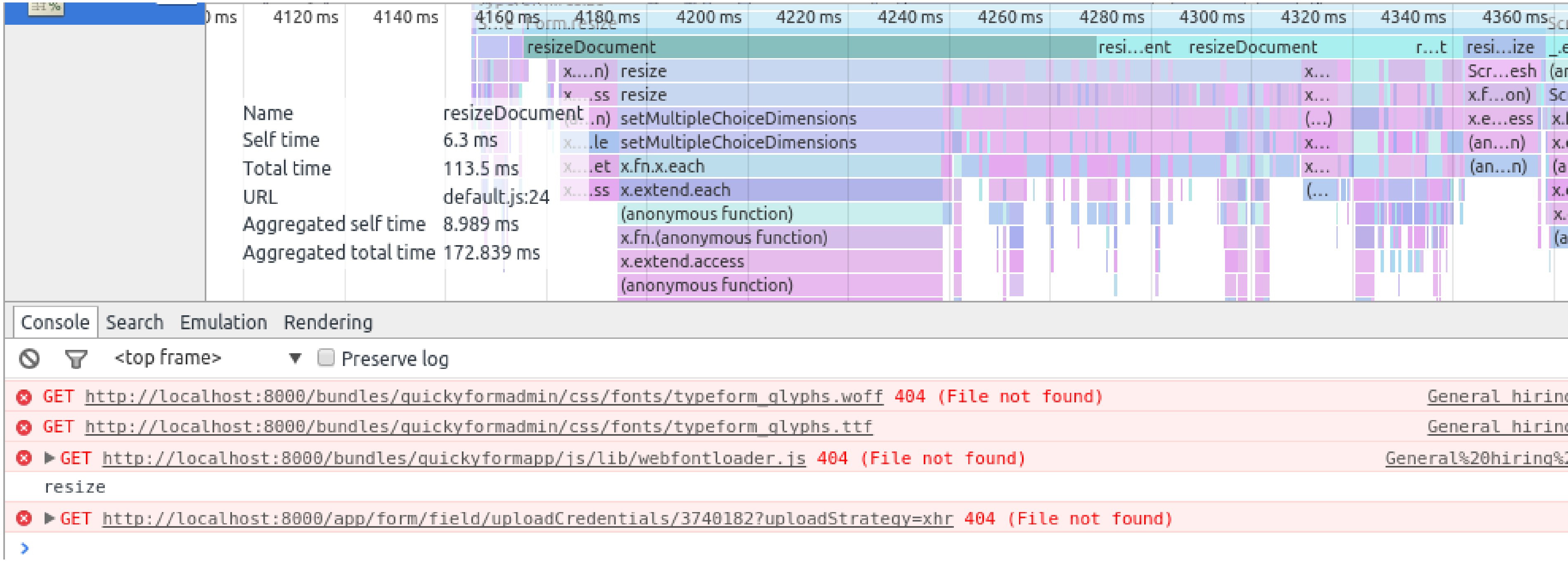
100 SEO

59

Name	Status	Type	Initiator	Size	Time
sprite.svg	200	svg+xml	styles.2117695….css	(disk cache)	2 ms
collect?en=page_view&dr=www.google.com&…	302	ping / Redir…	gtm.js?id=GTM-54HRKX:1	23 B	570 ms
js?id=G-375J4LLPD0&l=dataLayer&cx=c&gtm=…	200	script	gtm.js?id=GTM-54HRKX:1	(disk cache)	4 ms
analytics.js	200	script	gtm.js?id=GTM-54HRKX:1	(disk cache)	2 ms
insight.min.js	200	script	gtm.js?id=GTM-54HRKX:1	(disk cache)	3 ms
destination?id=AW-830339440&l=dataLayer&…	200	script	gtm.js?id=GTM-54HRKX:1	(disk cache)	4 ms
fbevents.js	(blocked:ot…	script	VM106:1	0 B	41 ms
uwt.js	(blocked:ot…	script	VM107:1	0 B	40 ms
sdk.js?sdkid=BSG11MIHO5CA4NQ7ADUG	(pending)	script	VM108:1	0 B	Pending
5h547yv898	(pending)	script	VM109:1	0 B	Pending
events.js	(pending)	script	VM110:1	0 B	Pending
sw_iframe.html?origin=https%3A%2F%2Fww…	200	document		(disk cache)	3 ms
collect?v=1&_v=j101&a=335689851&t=pagevi…	200	gif	analytics.js:22	58 B	569 ms
insight.old.min.js	200	script	insight.min.js:1	(disk cache)	9 ms

Command-line Tools

- Use `htop`, `perf`, and `ab` to monitor and benchmark systems. Chrome extensions also support profiling.



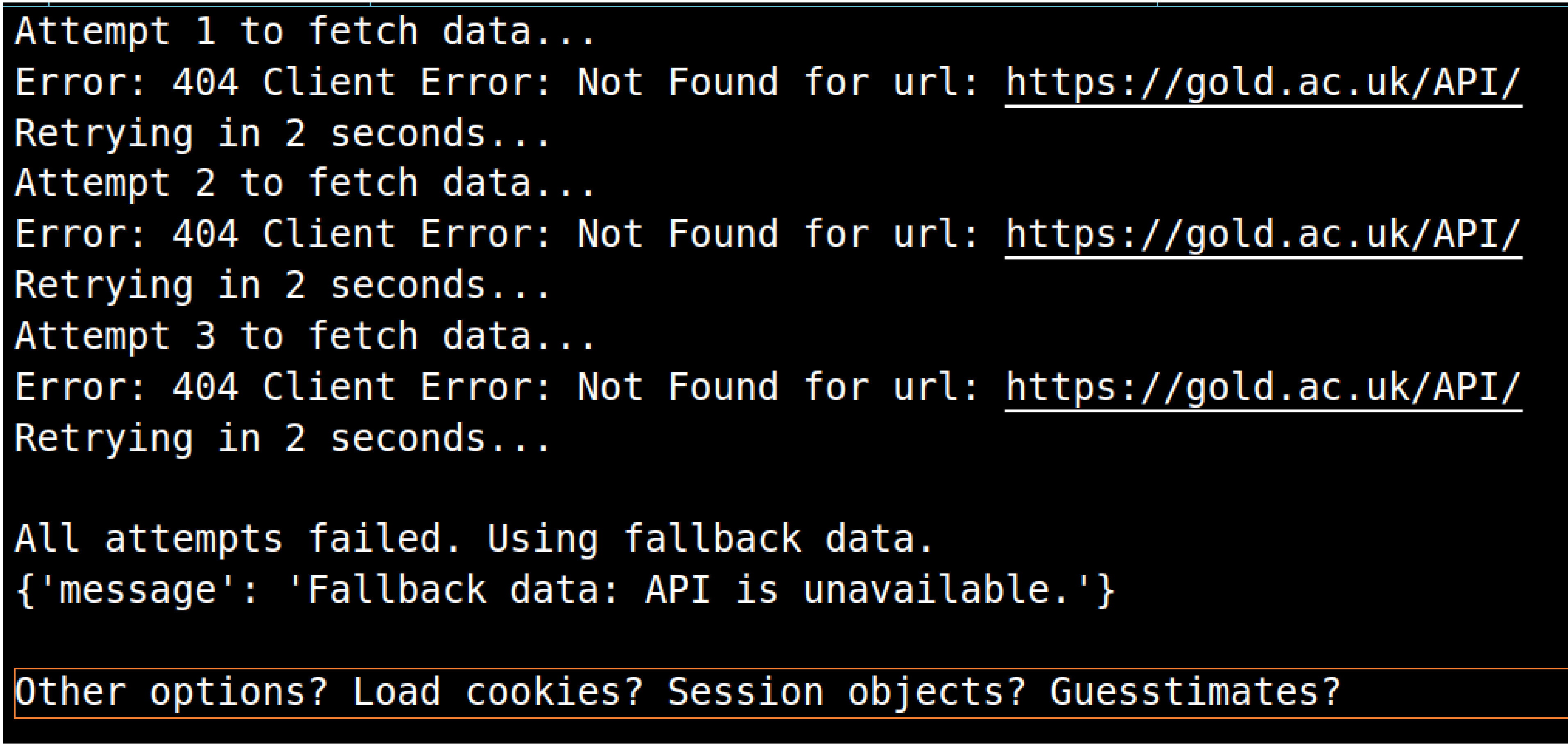
Name	resizeDocument
Self time	6.3 ms
Total time	113.5 ms
URL	default.js:24
Aggregated self time	8.989 ms
Aggregated total time	172.839 ms

```
GET http://localhost:8000/bundles/quickyformadmin/css/fonts/typeform_glyphs.woff 404 (File not found)     General_hirin
GET http://localhost:8000/bundles/quickyformadmin/css/fonts/typeform_glyphs.ttf                            General_hirin
▶ GET http://localhost:8000/bundles/quickyformapp/js/lib/webfontloader.js 404 (File not found)          General%20hiring%
resize
▶ GET http://localhost:8000/app/form/field/uploadCredentials/3740182?uploadStrategy=xhr 404 (File not found)
>
```

Error Handling Strategies

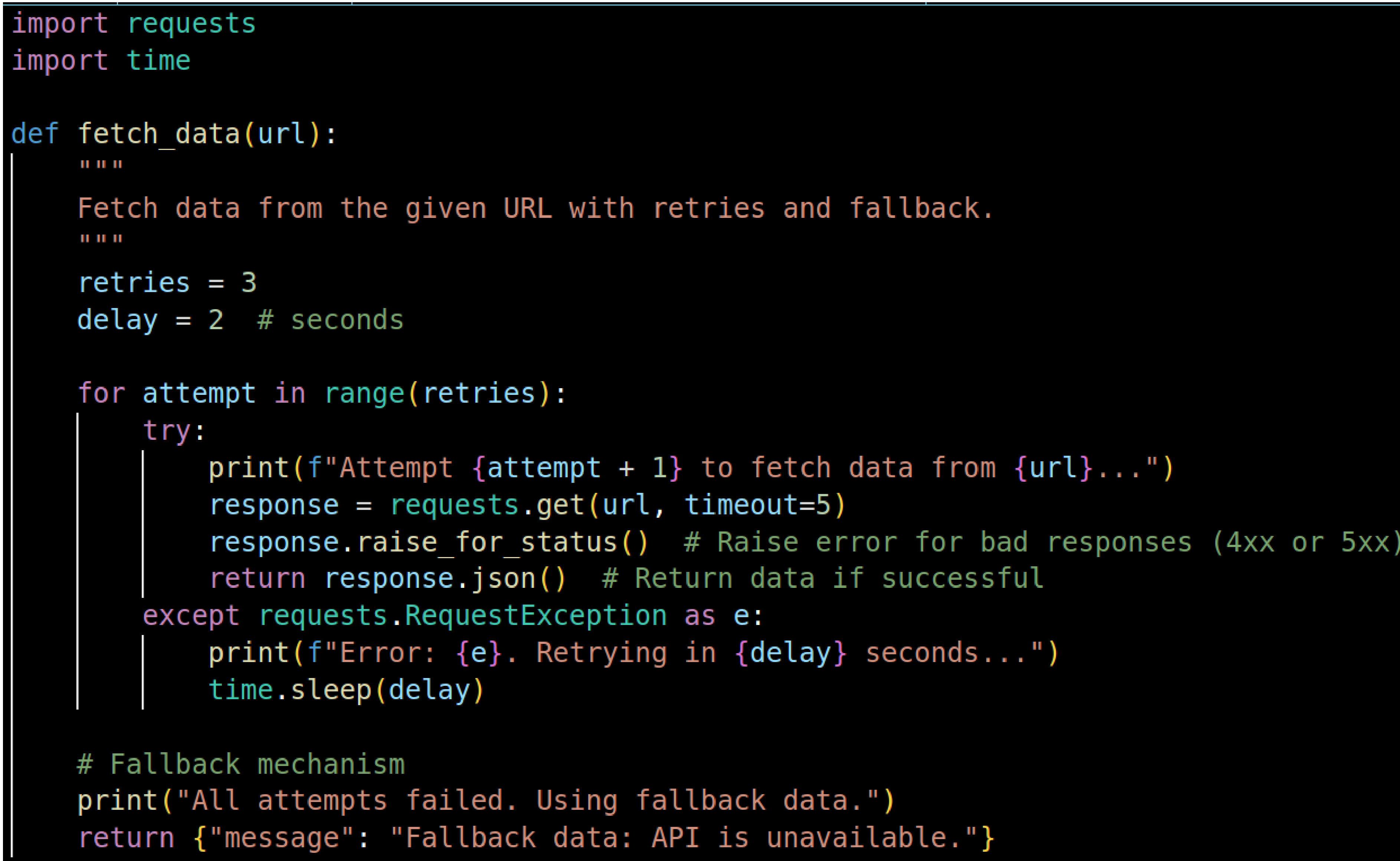
- Graceful degradation, retries, and fallback mechanisms.

Error Handling Strategies



```
Attempt 1 to fetch data...
Error: 404 Client Error: Not Found for url: https://gold.ac.uk/API/
Retrying in 2 seconds...
Attempt 2 to fetch data...
Error: 404 Client Error: Not Found for url: https://gold.ac.uk/API/
Retrying in 2 seconds...
Attempt 3 to fetch data...
Error: 404 Client Error: Not Found for url: https://gold.ac.uk/API/
Retrying in 2 seconds...

All attempts failed. Using fallback data.
{'message': 'Fallback data: API is unavailable.'}

Other options? Load cookies? Session objects? Guesstimates?
```

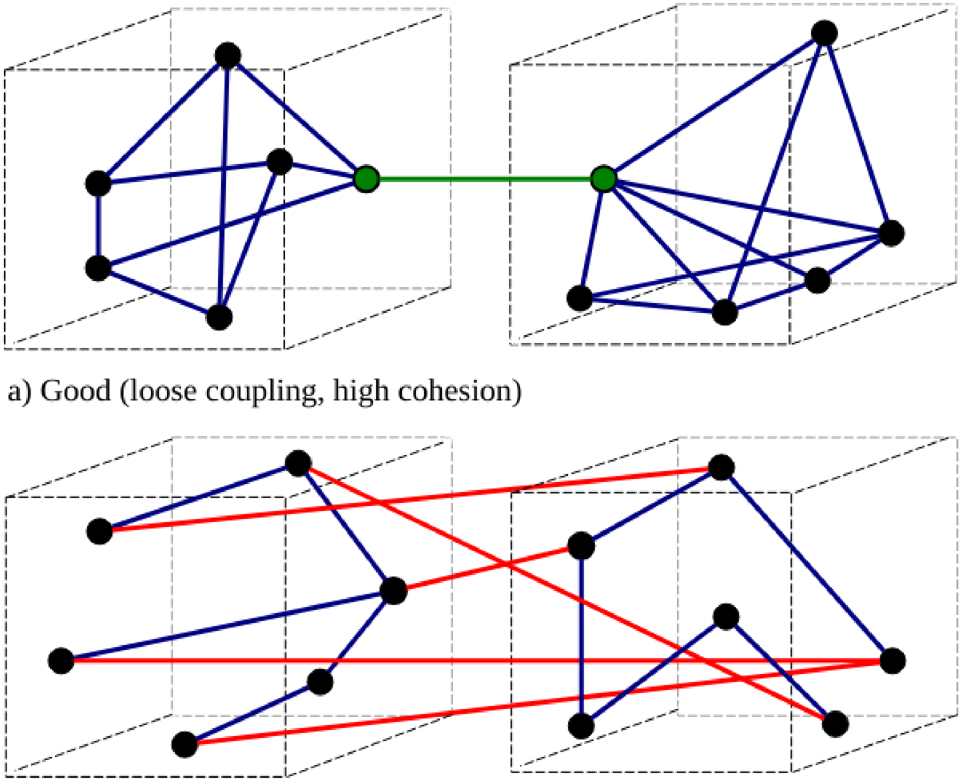
Error Handling Strategies



```python
import requests
import time

def fetch_data(url):
    """
    Fetch data from the given URL with retries and fallback.
    """
    retries = 3
    delay = 2  # seconds

    for attempt in range(retries):
        try:
            print(f"Attempt {attempt + 1} to fetch data from {url}...")
            response = requests.get(url, timeout=5)
            response.raise_for_status()  # Raise error for bad responses (4xx or 5xx)
            return response.json()  # Return data if successful
        except requests.RequestException as e:
            print(f"Error: {e}. Retrying in {delay} seconds...")
            time.sleep(delay)

    # Fallback mechanism
    print("All attempts failed. Using fallback data.")
    return {"message": "Fallback data: API is unavailable."}
```

Scalability Awareness

- Consider the impact of module design on scalability.

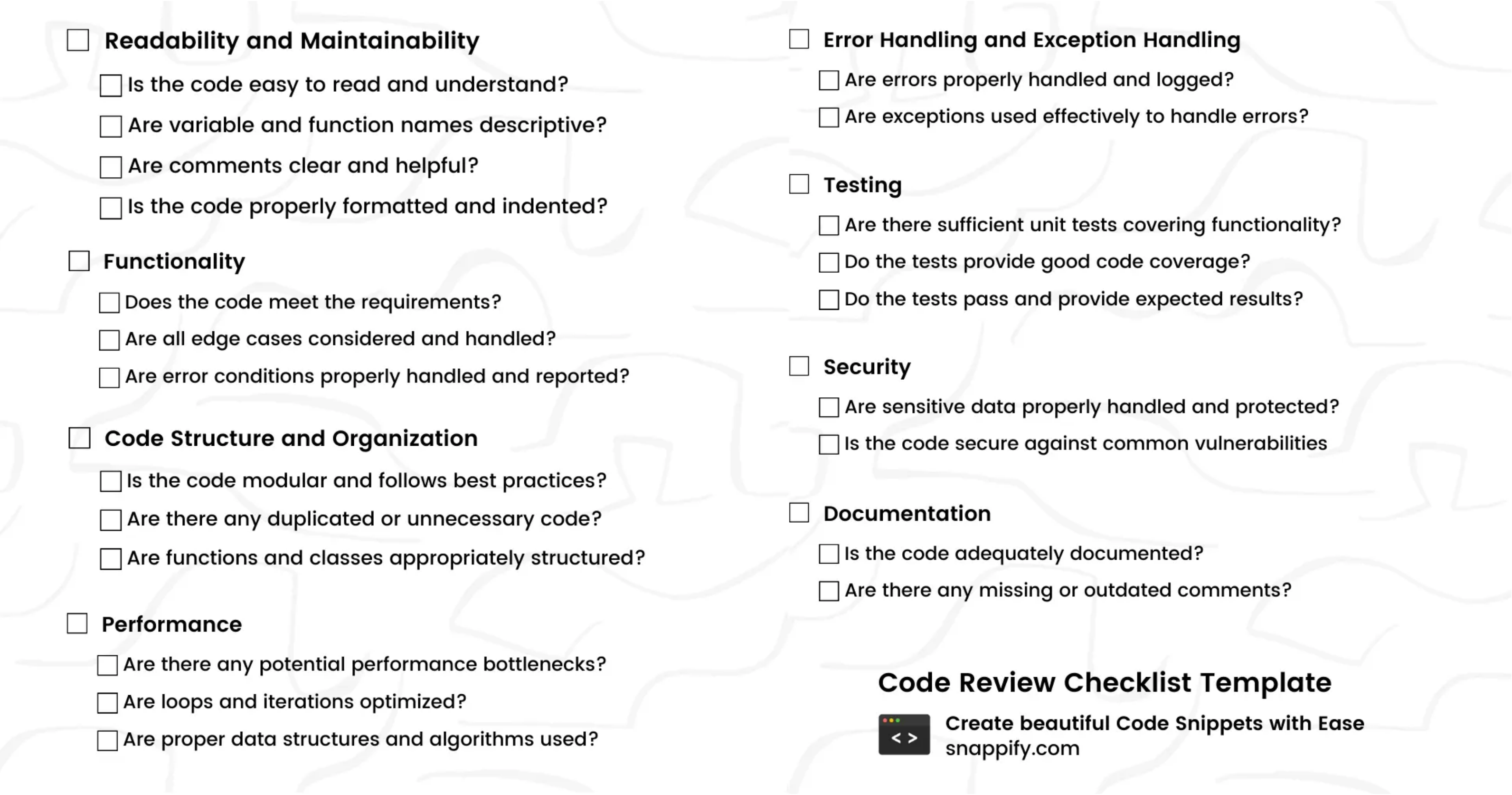


a) Good (loose coupling, high cohesion)

b) Bad (high coupling, low cohesion)

Real-World Practices

- Peer code reviews improve code quality and team collaboration.



□ **Readability and Maintainability**

- □ Is the code easy to read and understand?
- □ Are variable and function names descriptive?
- □ Are comments clear and helpful?
- □ Is the code properly formatted and indented?

□ **Functionality**

- □ Does the code meet the requirements?
- □ Are all edge cases considered and handled?
- □ Are error conditions properly handled and reported?

□ **Code Structure and Organization**

- □ Is the code modular and follows best practices?
- □ Are there any duplicated or unnecessary code?
- □ Are functions and classes appropriately structured?

□ **Performance**

- □ Are there any potential performance bottlenecks?
- □ Are loops and iterations optimized?
- □ Are proper data structures and algorithms used?

□ **Error Handling and Exception Handling**

- □ Are errors properly handled and logged?
- □ Are exceptions used effectively to handle errors?

□ **Testing**

- □ Are there sufficient unit tests covering functionality?
- □ Do the tests provide good code coverage?
- □ Do the tests pass and provide expected results?

□ **Security**

- □ Are sensitive data properly handled and protected?
- □ Is the code secure against common vulnerabilities

□ **Documentation**

- □ Is the code adequately documented?
- □ Are there any missing or outdated comments?

Code Review Checklist Template

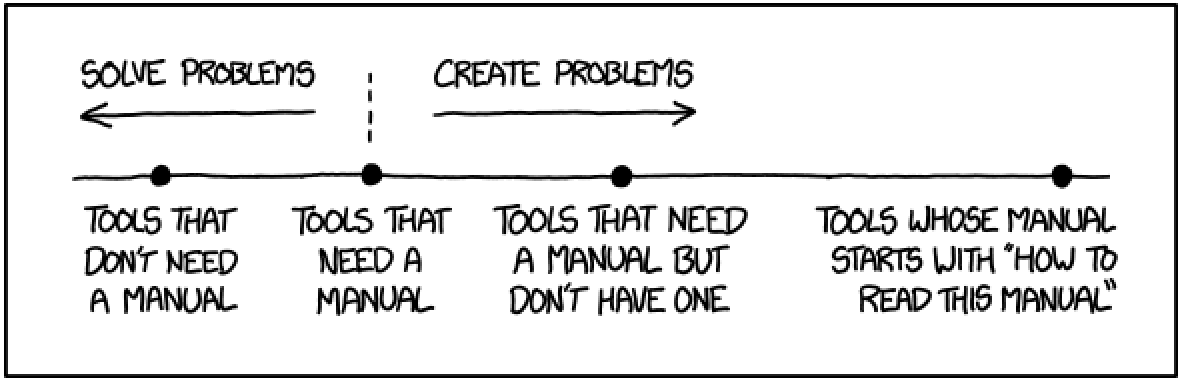
Create beautiful Code Snippets with Ease
snappify.com

Code Reviews

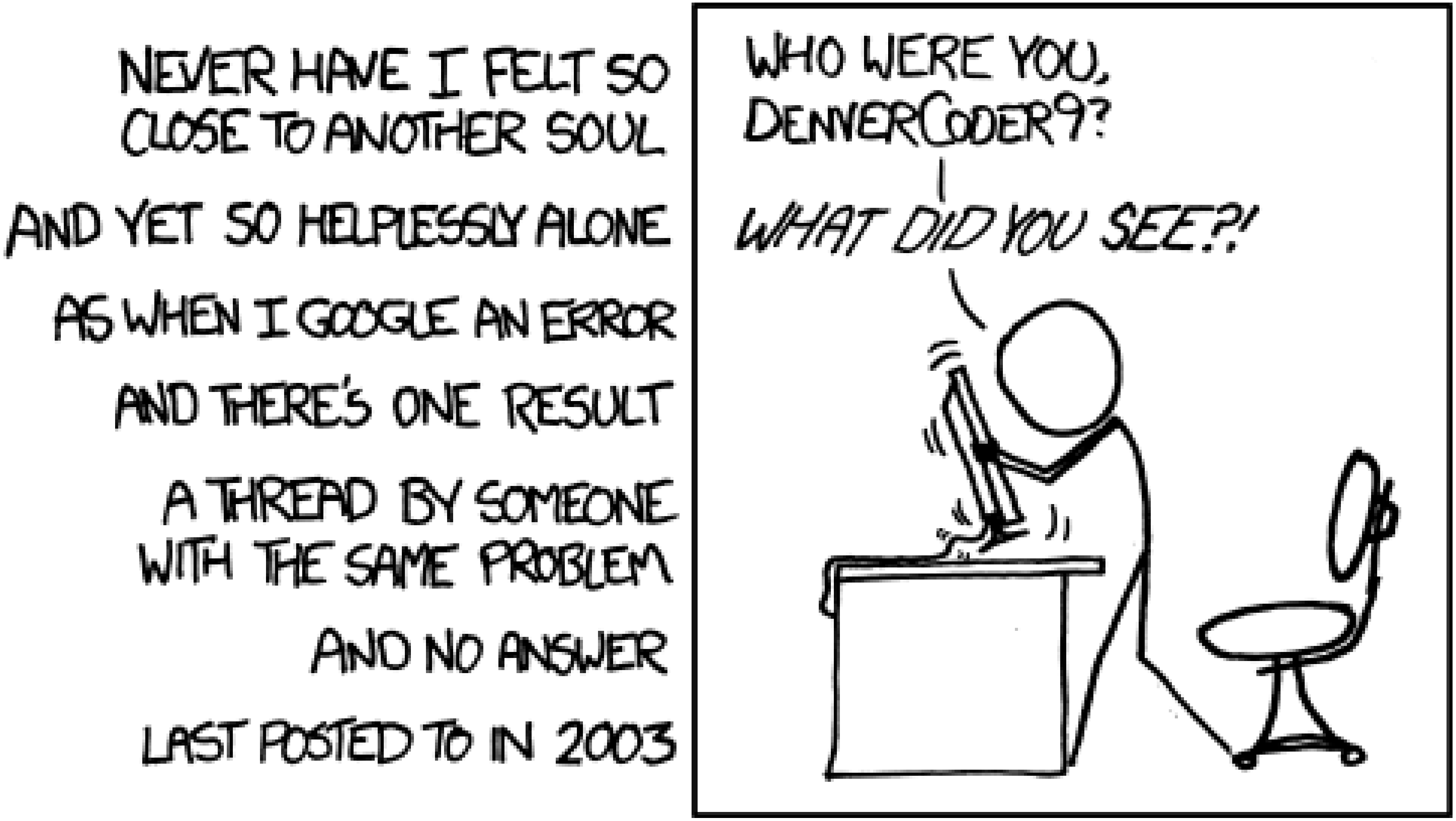
- Use GitHub or GitLab to conduct effective peer reviews.
- When you write a new bit of code you have to explain it to the team.
- One person ‘chairs’ the code review, takes notes, actions.
- One or more person assigned to critique, but others can also ask questions for clarity.

Documentation

- Document code clearly.
- Unit tests document functionality (or at least perceived expectations.)
- Comments are your friend e.g. docstrings.

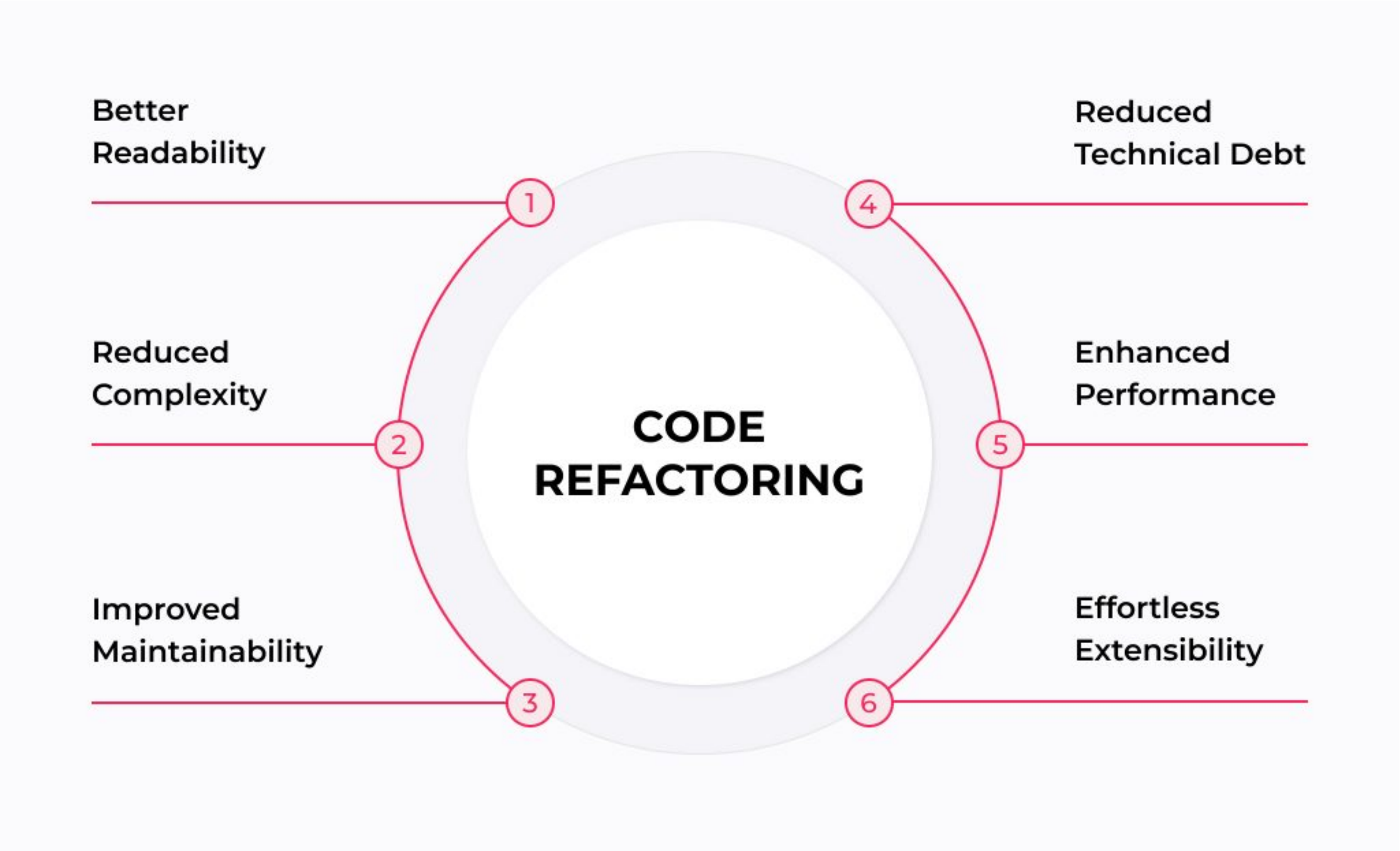

SOLVE PROBLEMS
CREATE PROBLEMS
TOOLS THAT DON'T NEED A MANUAL
TOOLS THAT NEED A MANUAL
TOOLS THAT NEED A MANUAL BUT DON'T HAVE ONE
TOOLS WHOSE MANUAL STARTS WITH "HOW TO READ THIS MANUAL"

Documentation


NEVER HAVE I FELT SO CLOSE TO ANOTHER SOUL
AND YET SO HELPLESSLY ALONE
AS WHEN I GOOGLE AN ERROR
AND THERE'S ONE RESULT
A THREAD BY SOMEONE WITH THE SAME PROBLEM
AND NO ANSWER
LAST POSTED TO IN 2003
WHO WERE YOU, DENVERCODER9?
WHAT DID YOU SEE?!

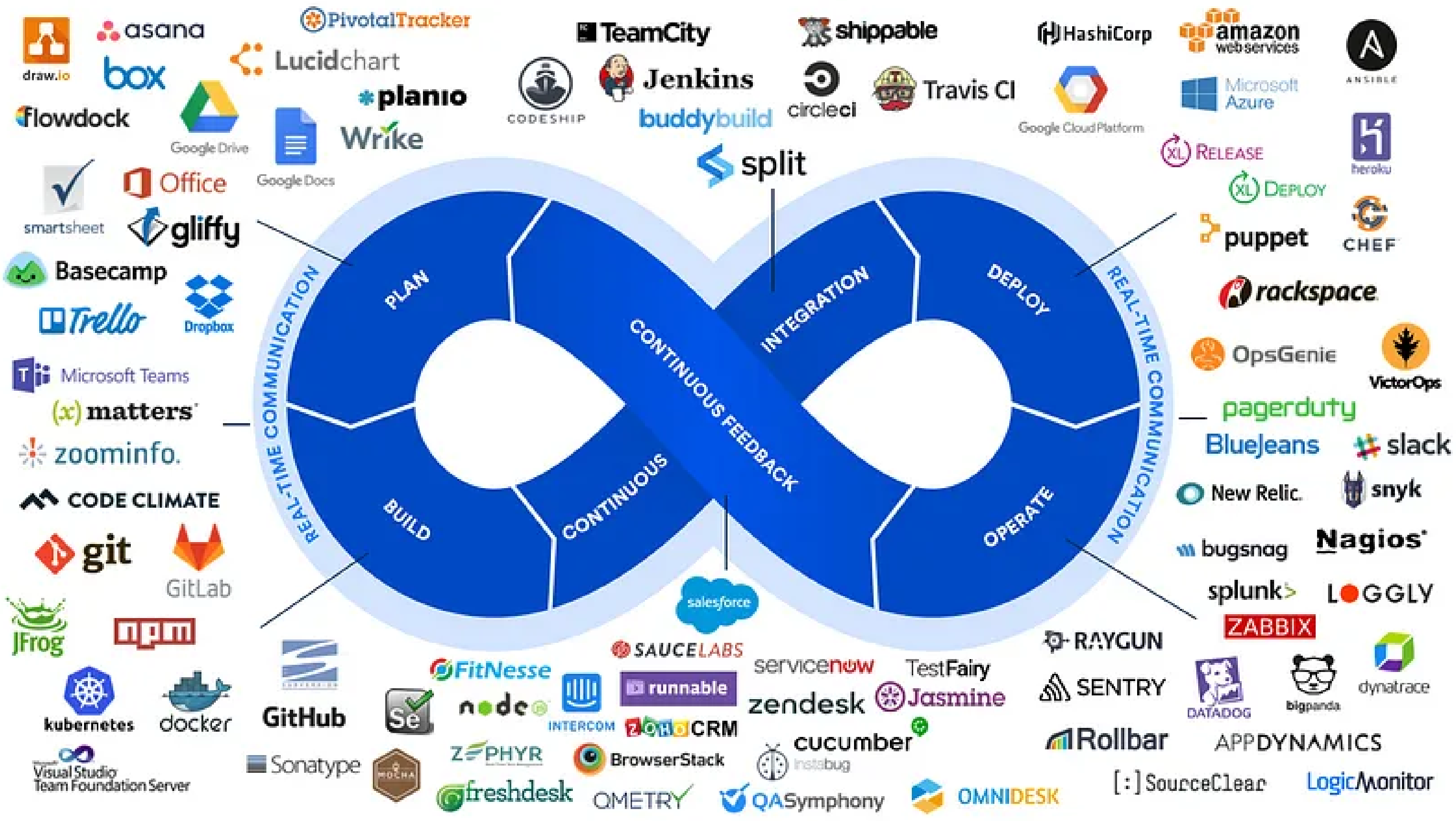
Future-Proofing Code

- Refactor to manage technical debt and maintain code quality.



Introduction to DevOps

- Learn about DevOps culture and tools for collaboration.



Final Thoughts

- Most people don’t know what they’re doing.
- Criticality of self is vital.
- Empirical testing will get you 80% of the way there.
- Some of this stuff is still subjective (e.g. too much code coverage for your tests is generally not desirable.)

Final Thoughts

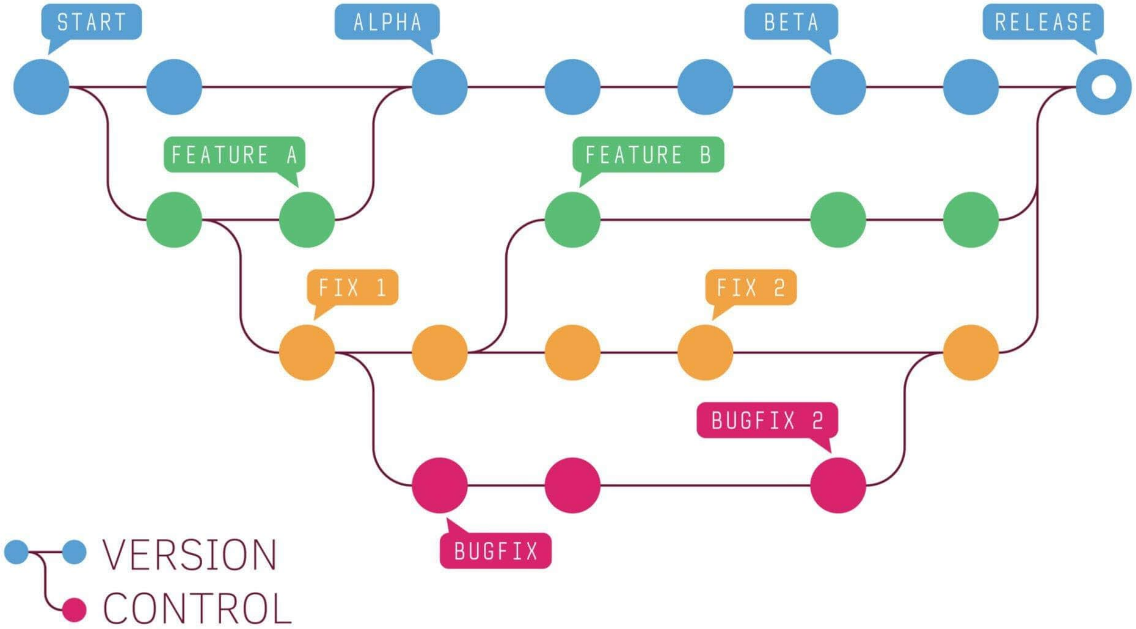
Antoine de Saint-Exupéry (Philosopher and Writer):

“Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away.”

Final Thoughts

Aristotle (Greek Philosopher):

“The whole is greater than the sum of its parts.”



Final Thoughts

Confucius (Chinese Philosopher):

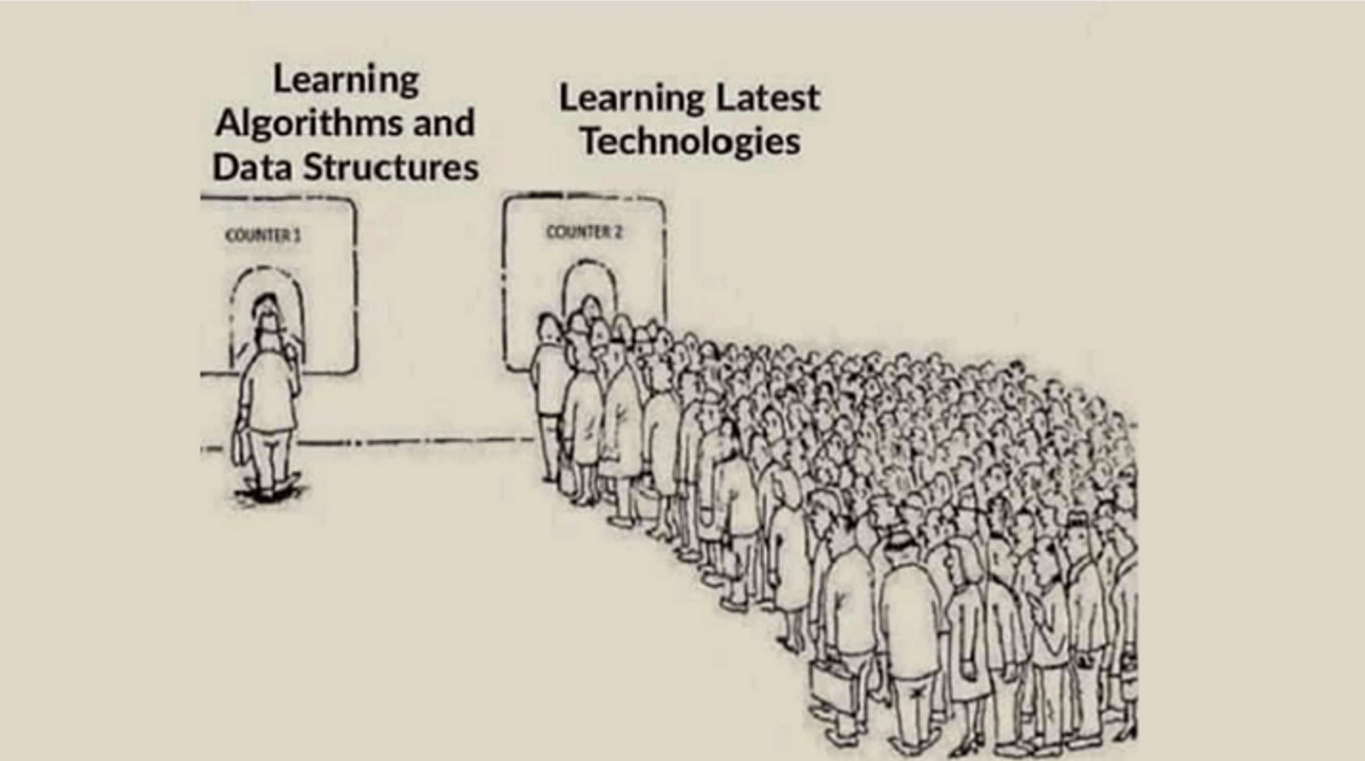
“Success depends upon previous preparation, and without such preparation, there is sure to be failure.”

Sprint Planning
We heard in Sprint Review...
How does this tie to our Product Goal??
Our Sprint Goal for this Sprint is...
Our Definition of Done is...
To Do
Doing
Done
Sprint Goal:

Final Thoughts

Voltaire (French Philosopher):

“The best is the enemy of the good.”


Learning Algorithms and Data Structures
Learning Latest Technologies
COUNTER 1
COUNTER 2

Final Thoughts

You are just getting started on what *could* be a **very** exciting journey.

Department of Computing

Text only

Degree Programmes

The Department of Computing offers single and joint honours undergraduate degree programmes, as well as MPhil and PhD research degrees. Computing students at Goldsmiths come from many different countries, age groups and backgrounds. There are currently around 400 undergraduate and 12 postgraduate students in the department.

Undergraduate Degrees

- BA Computing and Design for the World Wide Web
- BSc Computer Science
- BSc Information Systems
- BSc Internet Computing

Postgraduate Degrees

- Computer Science and Information Systems MPhil and PhD

Thank You!

- Coursework deadline is coming up.
- Exams in January.
- I’ll see you all again (soon) for Computing Project II...