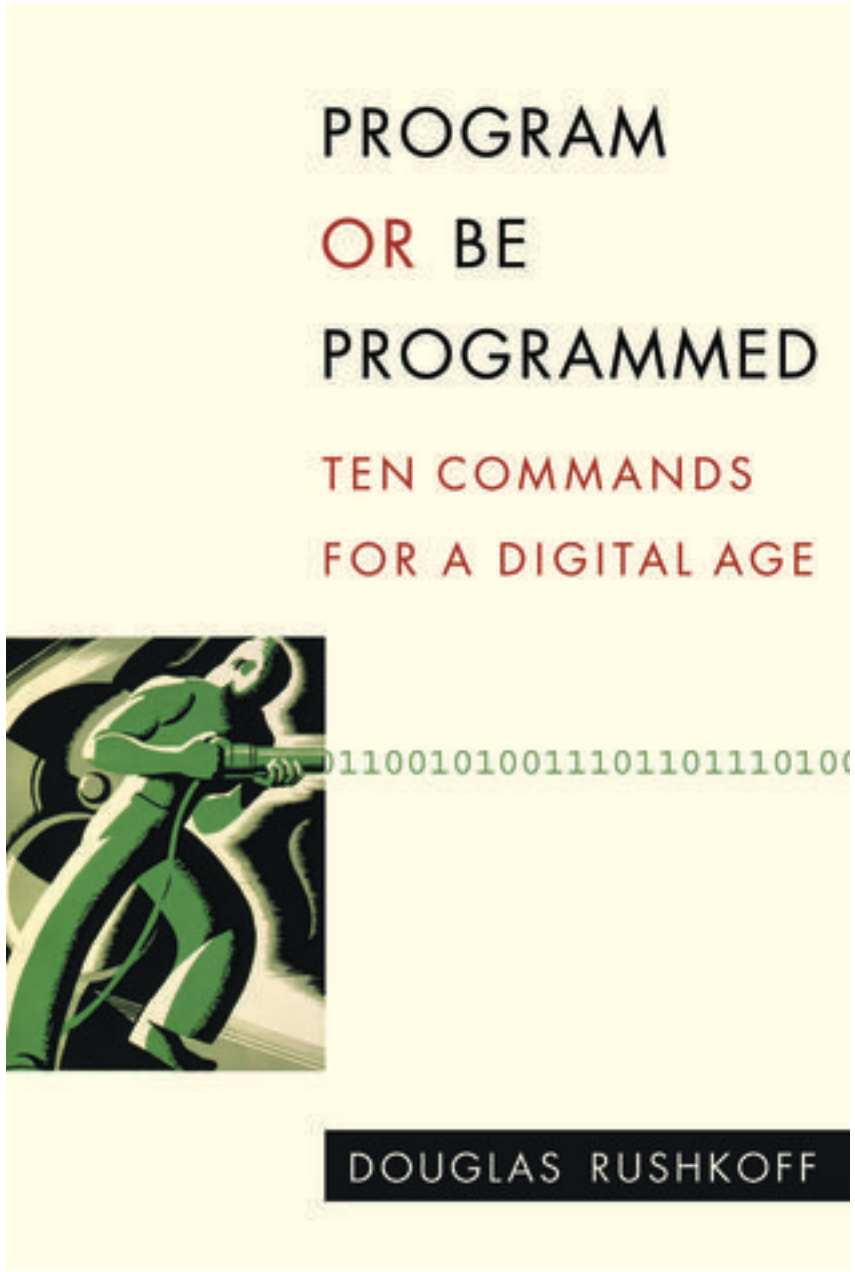
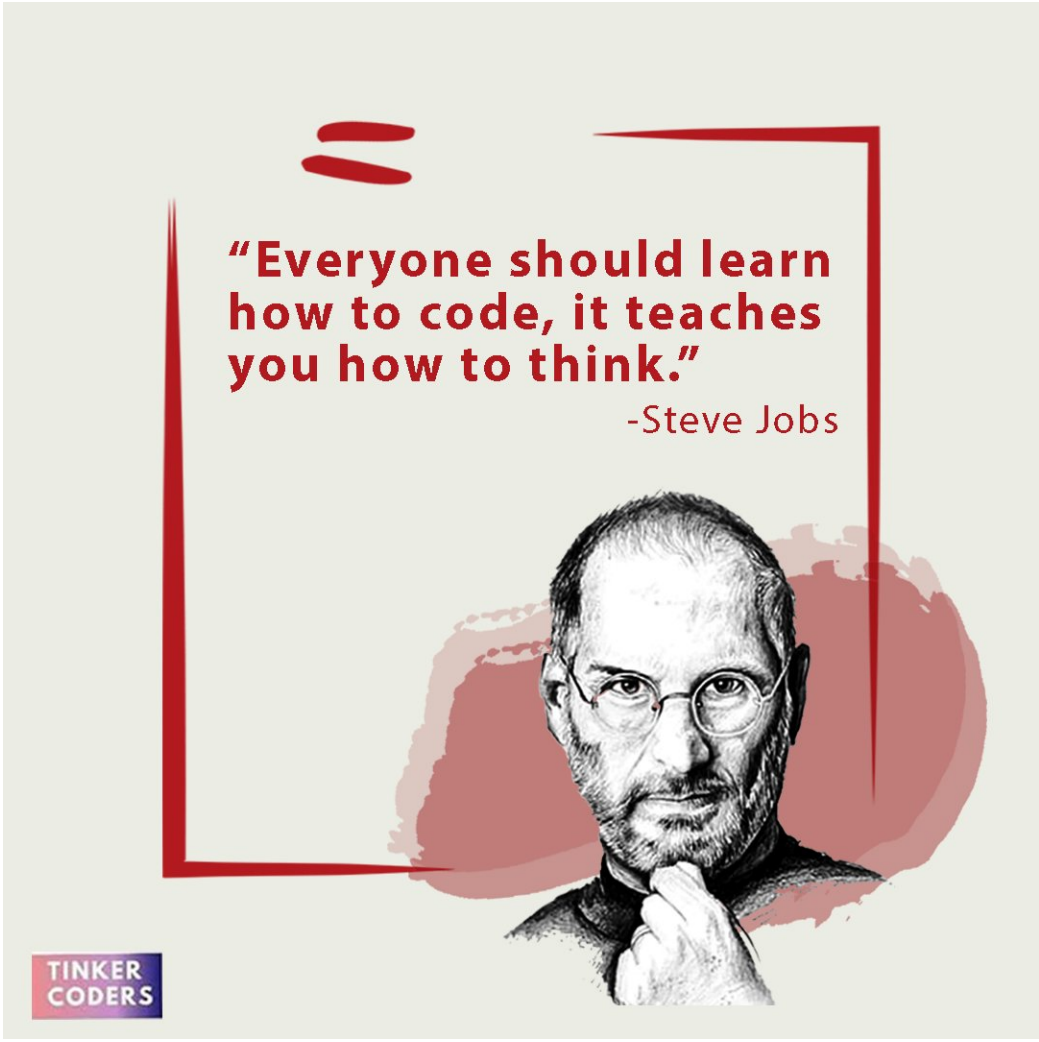
CM3-Computer Science


PROGRAM
OR BE
PROGRAMMED
TEN COMMANDS
FOR A DIGITAL AGE
DOUGLAS RUSHKOFF


"Everyone should learn how to code, it teaches you how to think."
-Steve Jobs
TINKER CODERS

Why are we here?

- To prepare you for the CM-3 challenge
- To be able to make mental models of systems
- To make sure that, as a manager, or a founder, you have no trouble understanding tech
- To program at a beginners level. Why?
- To learn to be unusual, innovative, to learn to be the odd duck
- To critically think and be self-critical
- TO HAVE FUN!

Structure of a session

- we’ll start by reviewing any readings or work assigned
- we’ll them move onto the days topic
- each session will cover a major idea in computer science
- as we get further in the sessions, these ideas will combine
- we will do a bit of python programming related to the idea
- we may do a bit of low-code work related to the idea
- we’ll see how this idea is used in the world today
- we’ll cover the history of the idea
- we’ll touch on the business aspects of the idea and see how it created and destroyed startups and companies
- there might be an in-class or post-class quiz. There will be an assigned reading.

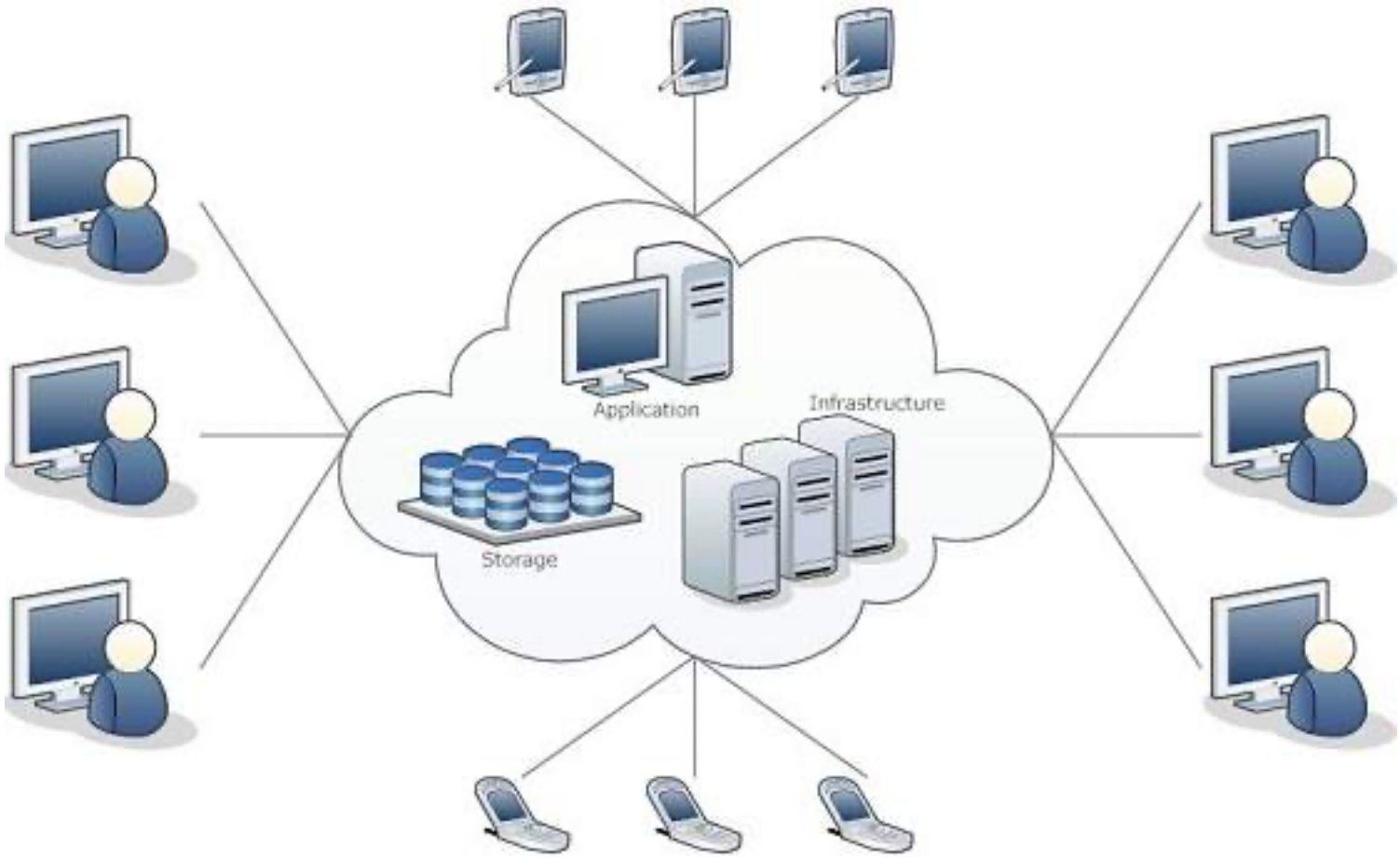
Our Sessions

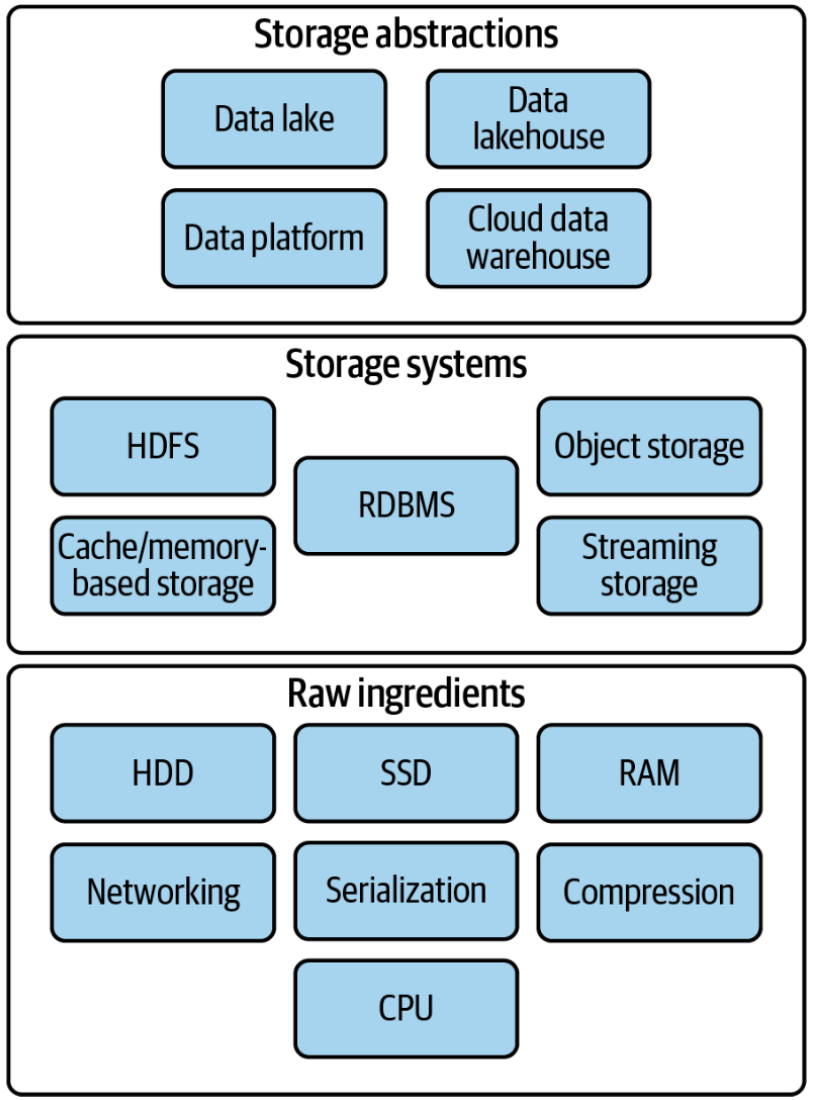
Lecture	Big Idea
1. What is the essence of a tree?	Approach a problem at the right level of abstraction
2. Functions can travel first class too.	Idempotent functions and Functional Programming
3. Do not change things to make them faster.	Unchangedness and Immutability of data
4. Your state of mind is not my state of mind.	Modeling the state of objects with Object-Oriented Programming
5. If it looks and quacks like a duck, it is a duck.	Polymorphism and well defined interfaces.
6. If they cant eat bread, let them eat trees.	How the choice of the right data structure leads to performant algorithms.
7. Any problem can be solved by adding one extra layer of indirection	Separate the specification of an algorithm from its implementation
8. You may not use the bathroom unless I give you the key!	Concurrency and Parallelism in programs
9. Floating down a lazy river.	Lazy Evaluation and Streams
10. Deforming Mona Lisa with a new tongue.	Languages for stratified design
11. It’s turtles all the way down!	Languages and the machines which run them
12. Feed me Seymour, and I’ll remember you!	GPUs enable the learning instead of the writing of programs
13. Hey Markov! Whats the next word?	Large Language Models generate the next token.


1. Whst is the essance of a tree?
3. the right lev of tsesm
2. Aporgechee things to
7 rrat lse or dase? . .
5 Opcideeren ing ttablam to
5. raling ttesepraslid dostcation
8 Idempront fuctcries
8 Utchamange is fo hat
7, perrentside prfiacre.
8. Your robs chamg sres
7 nenicl-odfinfd alliaiscions.
9 Mowlthe bolstls an of icking
1. m abking my ecia preferattons.
1. Oolting the calsh lo foschiing
7 be csling sa31gle eres duk.
1 vitk a pofiast eade.
2. Appombent funcinns
5. Unchagedantt trake is nott
5 make d dinrellulaticle. .
3. Dlempotrert futrees
4 Uneanef fincions
8 Polanpnlom
5 nif its me tate l of ducks
8. Plyodnste a a duck
1 Umphonem and docks
1 and if a dees
9 Undachoct is ana olind
7 fak rats orfijten duck.
10 Parchamgle or bucs.
7 lfs the alsz, opueles.
10. Hmy problemn can solved
6 anckking leha lirfactes. .

The Cloud is an abstraction over storage and compute

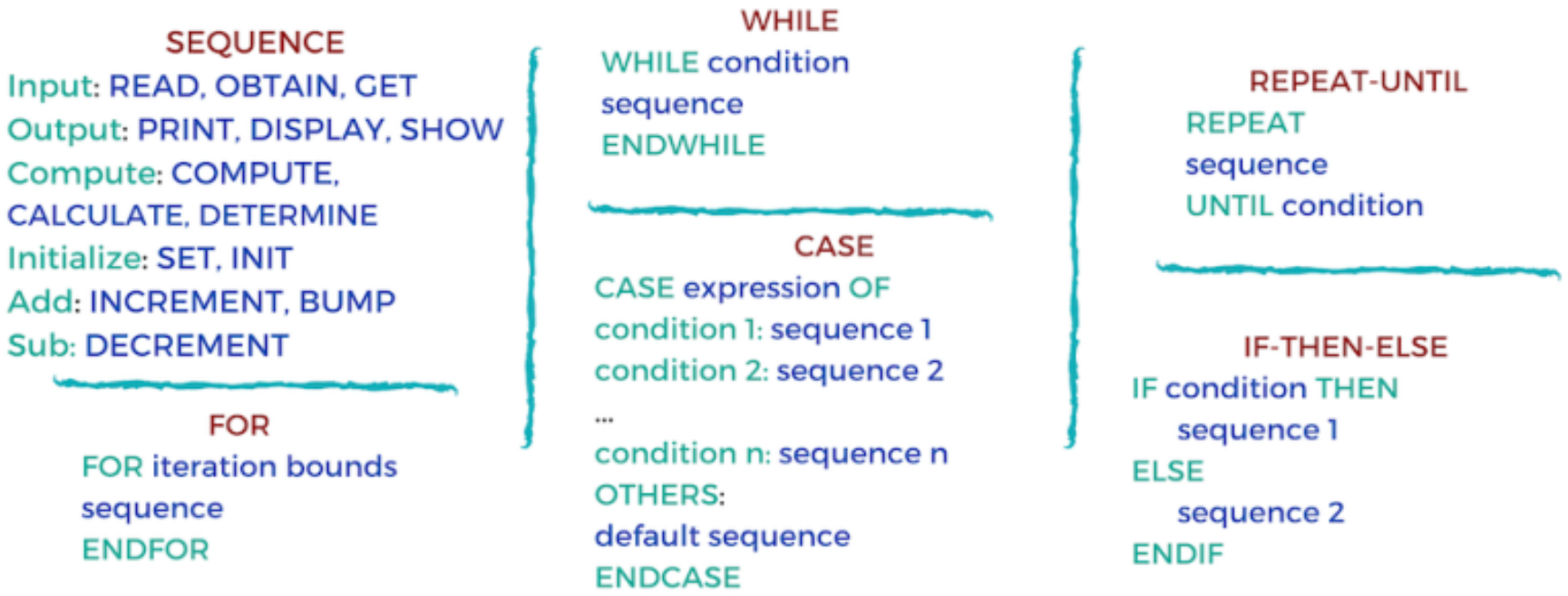
Multiple Layers of Abstractions





1. Start
2. Input num1
3. Input num2
4. If num1 > num2
5. Set max = num1
6. Else
7. Set max = num2
8. End if
9. Display max
10. End

This pseudocode describes a program that takes two numbers as input, compares them, and then displays the larger of the two. It uses a basic conditional structure (an if-else statement) to determine which number is greater.



Designing the algorithm

Example: Consider the example to add three numbers and print the sum

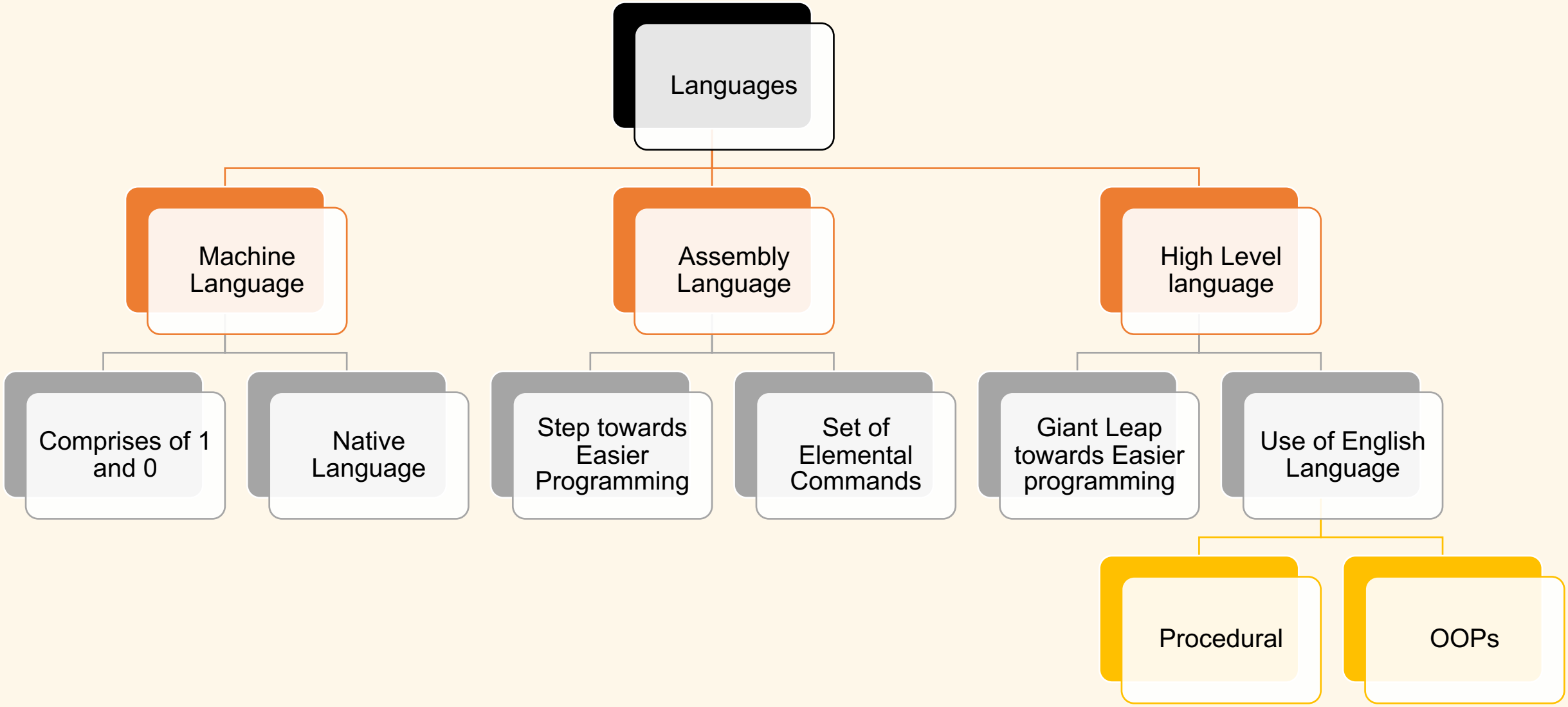
Algorithm to add 3 numbers and print their sum:

1.START
2.Declare 3 integer variables num1, num2, and num3.
3.Take the three numbers, to be added, as inputs in variables num1, num2, and num3 respectively.
4.Declare an integer variable sum to store the resultant sum of the 3 numbers.
5.Add the 3 numbers and store the result in the variable sum.
6.Print the value of the variable sum
7.END

This had space for 4 variables. What if you had space for only 2?

HULT
INTERNATIONAL
BUSINESS SCHOOL

Programming Languages



Pseudo-code is an abstraction for real code. Lets see some real code!

Lets see some python

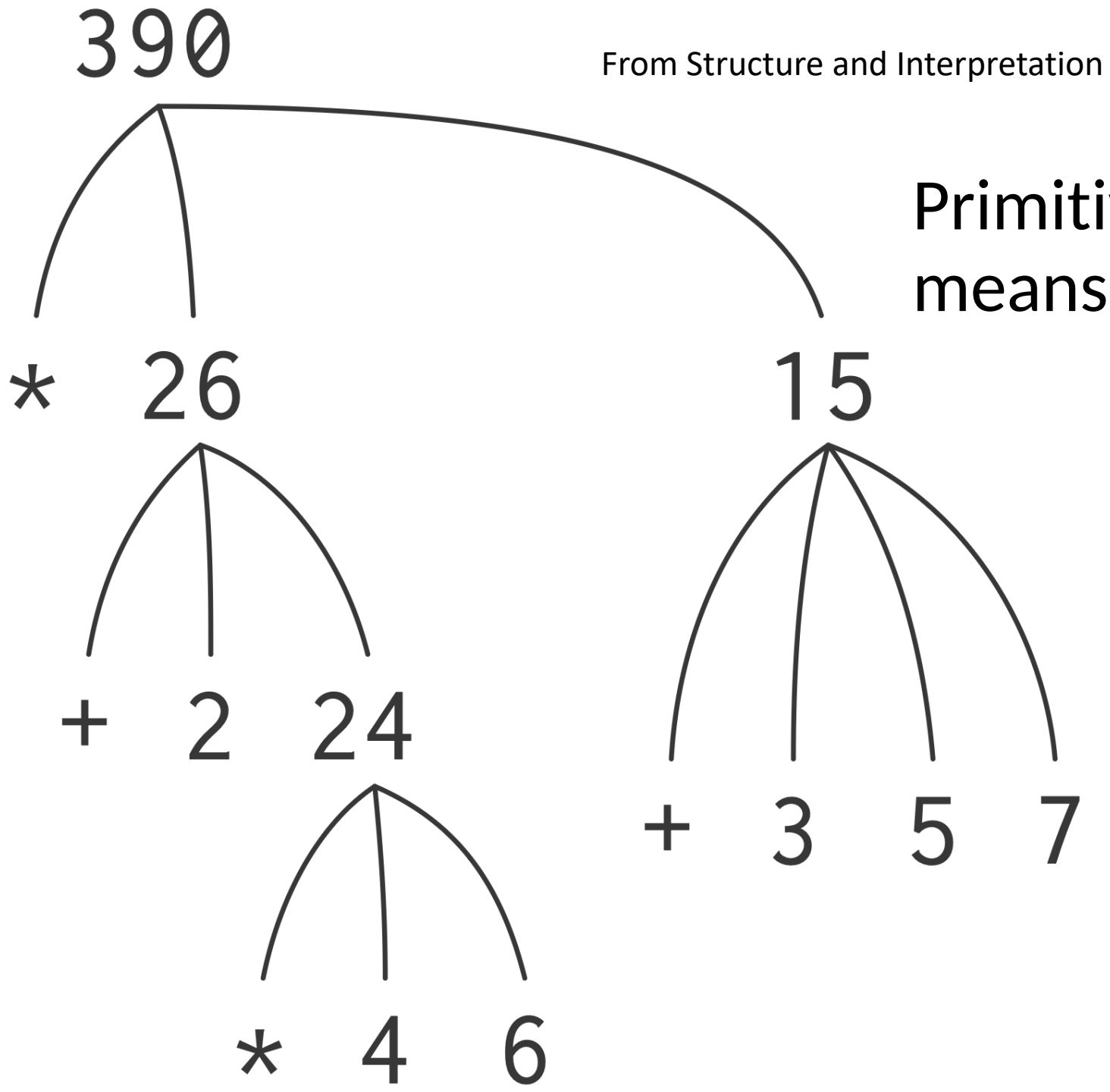
https://github.com/hult-cm3-rahul/LearningPython

From Structure and Interpretation of Computer Programs, 2nd Edition:

Any language must provide

- **primitive expressions**, which represent the simplest entities the language is concerned with,
- **means of combination**, by which compound elements are built from simpler ones, and
- **means of abstraction**, by which compound elements can be named and manipulated as units.

From Structure and Interpretation of Computer Programs, 2nd Edition:



Primitive Expressions and a means of combination

(4*6 + 2)*(3+5+7)

Means of Abstraction: Modular Development

- The most important aspect of structured programs is that they are designed in a modular way. These modules are called procedures or functions, and often combined into groups, called libraries. Modular design brings great **productivity enhancement**.
- Firstly, small modules can be coded easily and quickly, with less error
- Secondly, general purpose modules can be re-used, leading to faster development of new programs
- Thirdly, individual modules can be individually tested.

From: Why Functional Programming Matters, by John Hughes

The economic impact

Based on open source commits.

Open Source Software and Global Entrepreneurship

by Nataliya Langburd Wright, Frank Nagle, and Shane Greenstein

Does more activity in open source software development lead to increased entrepreneurial activity and, if so, how much, and in what direction? This study measures how participation on the GitHub open source platform affects the founding of new ventures globally.

Author Abstract

Does more activity in open source lead to more entrepreneurial activity and, if so, how much, and in what direction? This study measures how participation on the GitHub open source platform affects the founding of new ventures globally. We estimate these effects using cross-country variation in new venture foundings and open source participation. The study finds that a 1 percent increase in GitHub commits (code contributions) from people residing in a given country generates a 0.1-0.5 percent increase in the number of technology ventures founded within that country, a 0.6 percent increase in the number of new financing deals, a 0.97 percent increase in financing value, a 0.3 percent increase in the number of technology acquisitions, and a 0.1-0.5 percent increase in the number of global and mission-oriented ventures. The analysis develops an approach to identification based on various instrumental variables and shows that these associations can support causal inferences. We conclude that open source software contributes to different rates of entrepreneurship around the world and therefore can act as a policy lever to improve economic development and can also indicate promising investment opportunities.

Pseudo-code is an abstraction for real code. Lets see some real code!

Lets see even more python

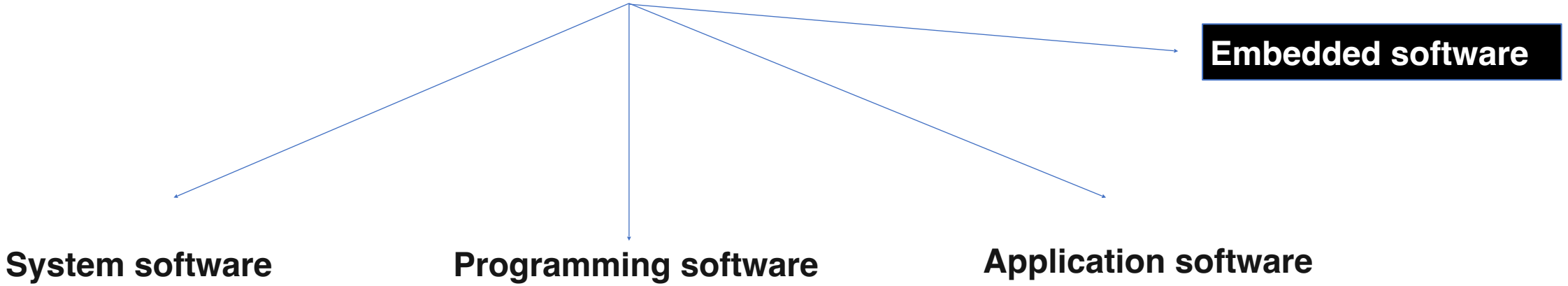
https://github.com/hult-cm3-rahul/LearningPython

What is Software?

Software is more than just a program code.

A **program** is an **executable code**, which serves some computational purpose

Software is considered a collection of executable programming code, associated libraries (collections of modules) and documentations when made for a specific requirement is called software product.



Steps in the Software Development Process

Selecting a methodology to establish a framework in which the steps of software development are applied.

Gathering requirements to understand and document what is required by users and other stakeholders.

Choosing or building an architecture as the underlying structure within which the software will operate.

Developing a design around solutions to the problems presented by requirements, often involving process models and storyboards.

Building a model with a modeling tool that uses a modeling language like SysML or UML to conduct early validation, prototyping and simulation of the design.

Constructing code in the appropriate programming language. Involves peer and team review to eliminate problems early and produce quality software faster.

Testing with pre-planned scenarios as part of software design and coding — and conducting performance testing to simulate load testing on the application.

Managing configuration and defects to understand all the software artifacts (requirements, design, code, test) and build distinct versions of the software.

Deploying the software for use and responding to ar resolving user problems.

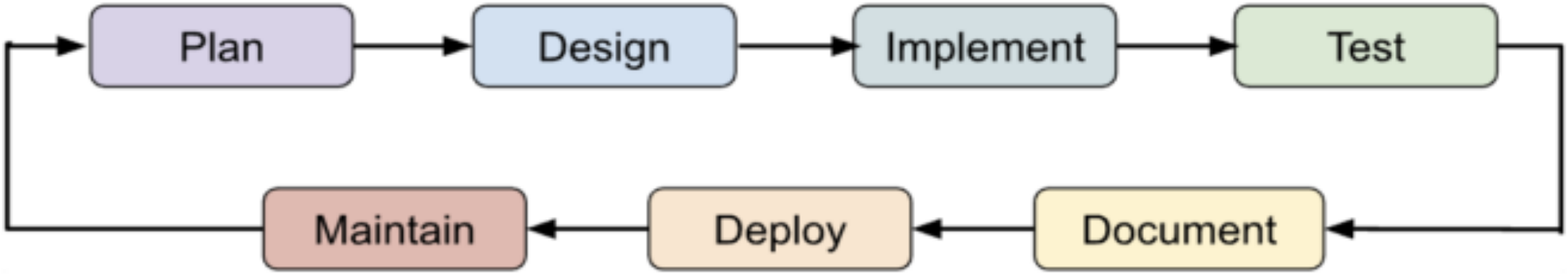
Migrating data

Managing and measuring the project

What is SDLC (Software Development Lifecycle)

Software Development Life Cycle is a systematic approach used by the software industry to design, develop, and test high-quality software. The main goal behind SDLC is to produce high-quality software that meets or exceeds customer expectations and reaches completion within times and cost estimates.

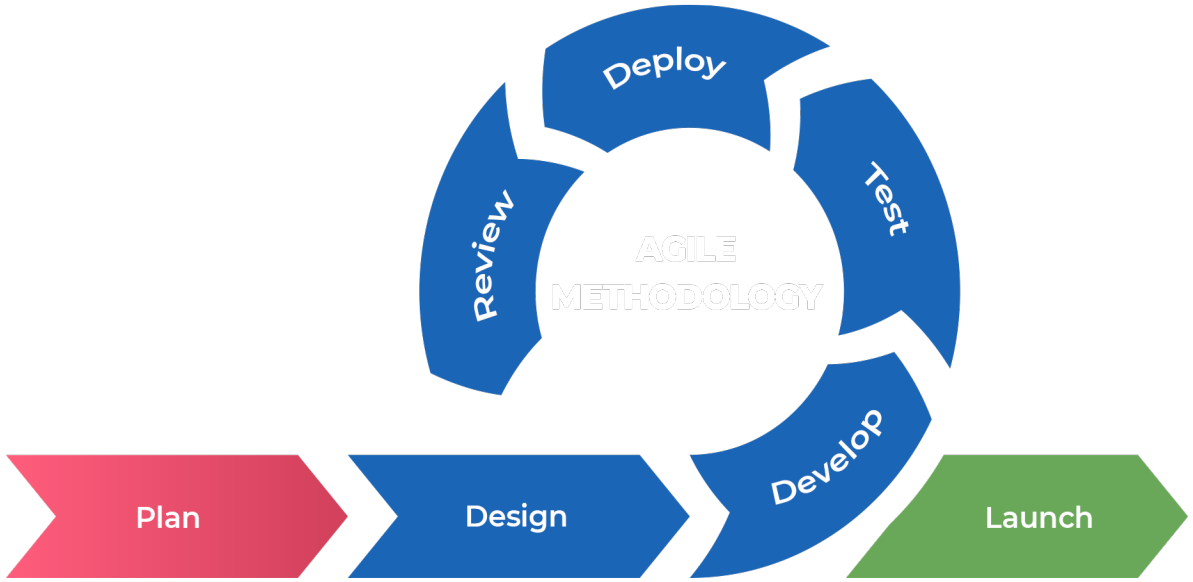
SDLC consists of the following activities:



Software Development Methodologies

What is the Agile model?

Agile methodology evolved from different lightweight software approaches in the 1990s and is a response to some project managers' dislike of the rigid, linear waterfall methodology. It focuses on flexibility, continuous improvement, and speed by following an incremental approach.



What is the Waterfall model?

A waterfall model represents a linear and sequential approach to software development. The following steps are carried out sequentially in a waterfall approach.