INTRODUCTION TO SOFTWARE DEVELOPMENT

Week 2 Day 2

Led by: Emily Crose

for

Oakland University

PREVIOUS SESSION RECAP

QUESTION OR CLARIFICATIONS?

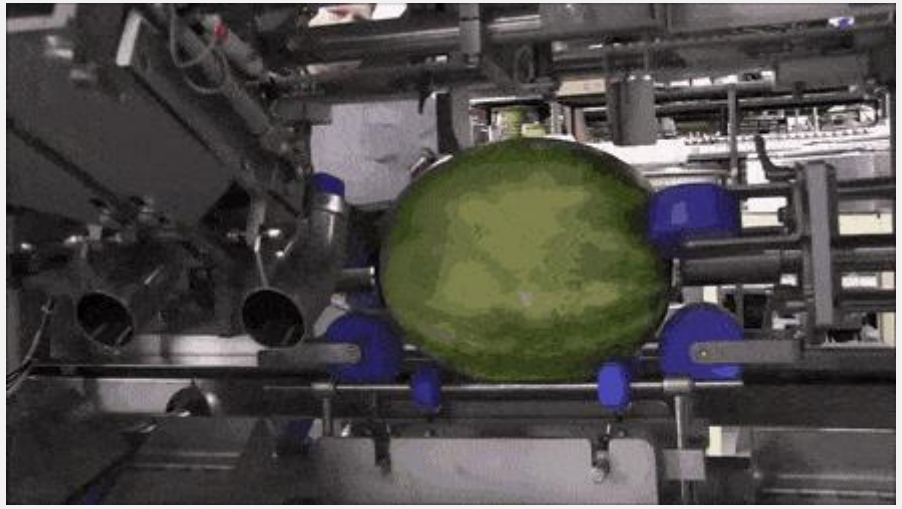
TERMS TO LISTEN FOR

Loops	Repeated routines
Nesting	A routine within another routine
Libraries	Pre-made, coded routines which we may utilize in our program

PROGRAMMING CORE CONCEPTS

- Source code
  - We develop the program's source code. Our program will then run using these instructions
- Program execution
  - The program's operation is transferred to memory and runs
- Routines
  - Processes that a program will execute
- Compiling
  - Building an executable binary

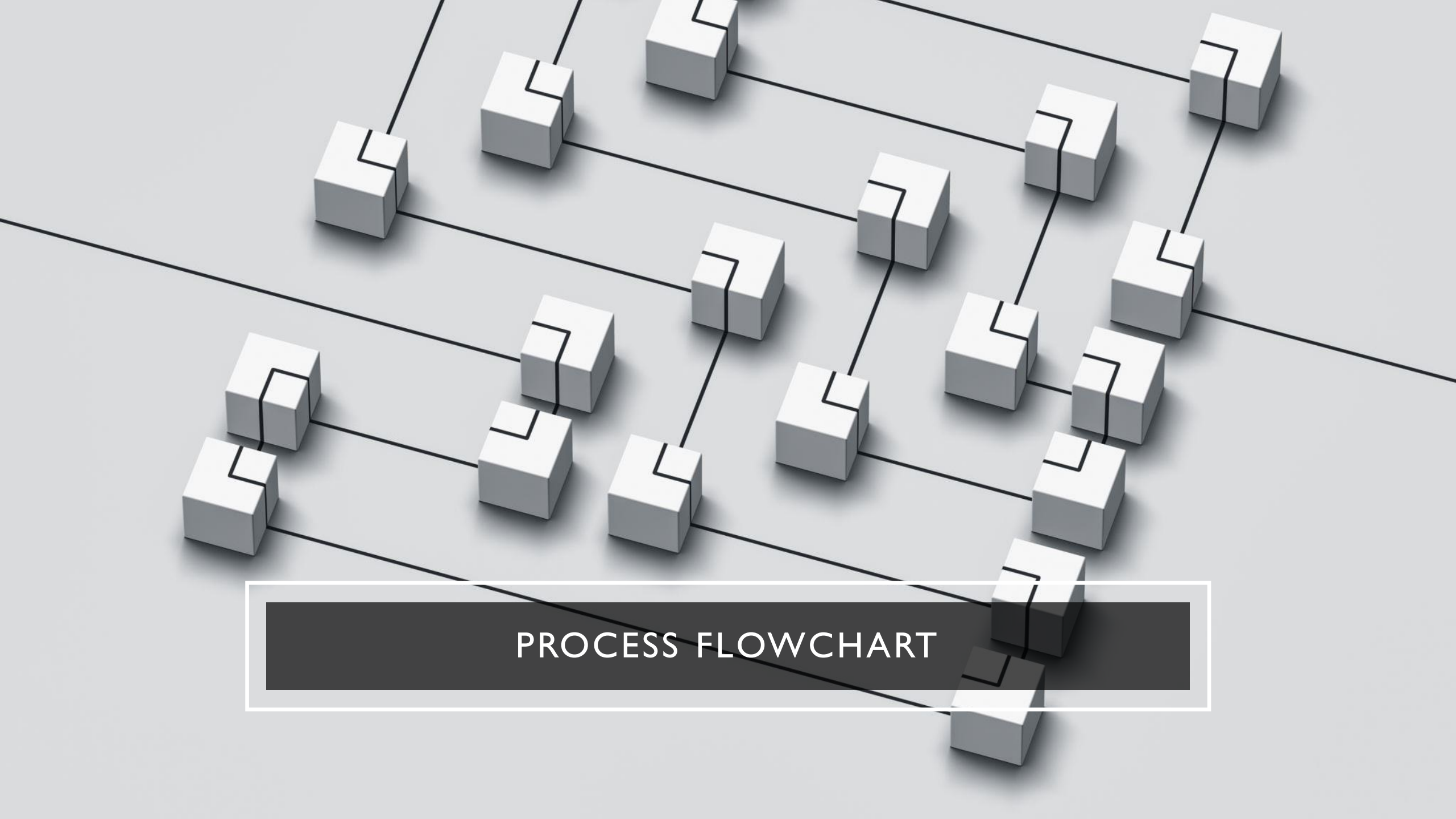
MANUAL VS. AUTOMATION



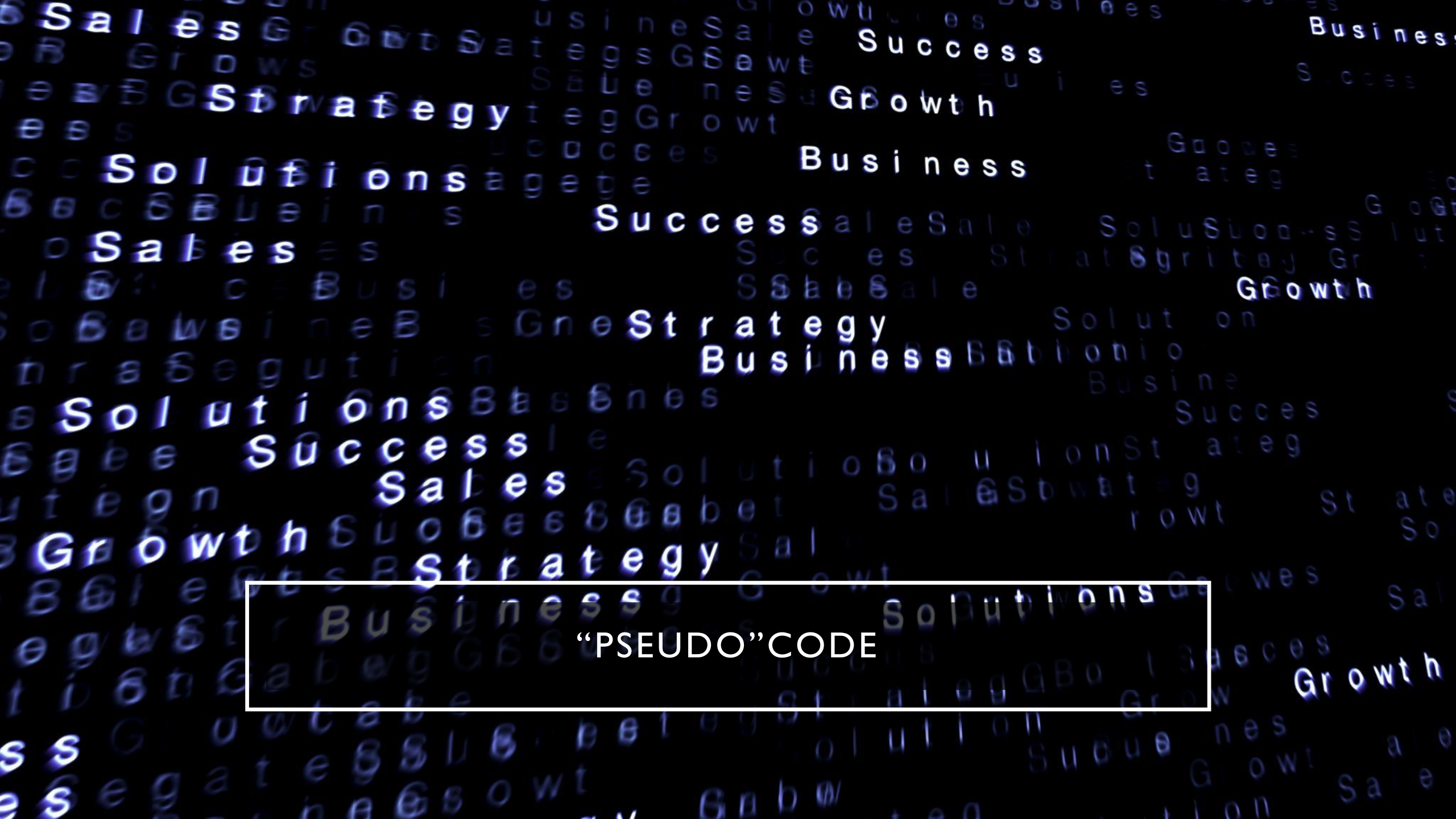

WHAT IS PROGRAMMING?

WHAT IS PROGRAMMING?

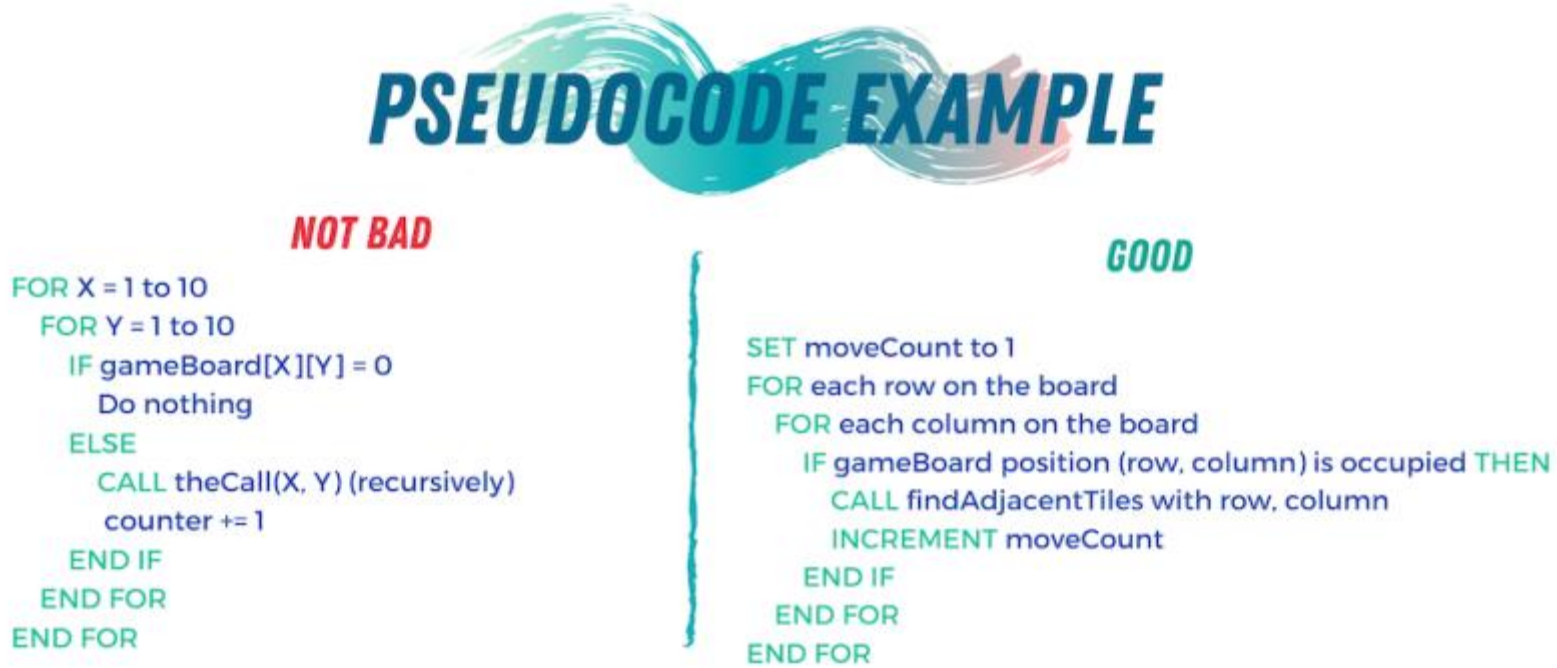
ORGANIZING PROGRAM OPERATIONS



PROCESS FLOWCHART



“PSEUDO”CODE



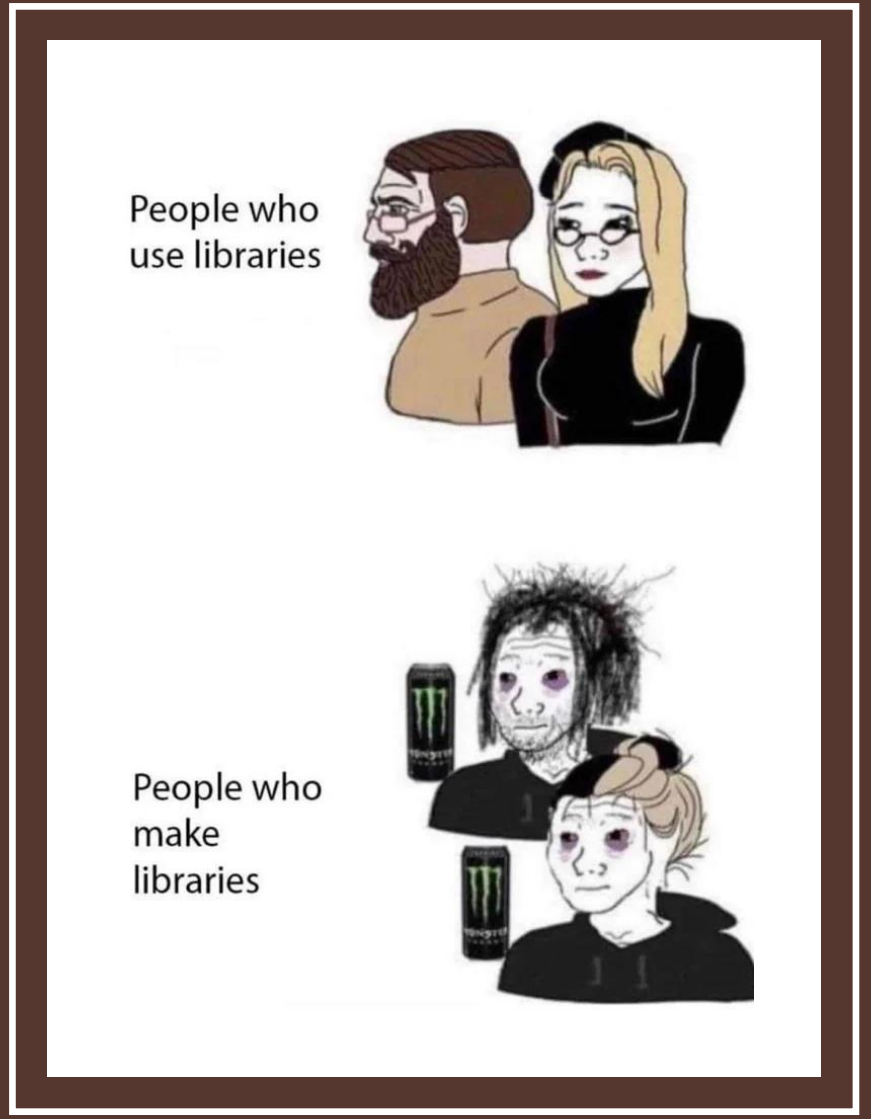
PSEUDOCODE EXAMPLE

DEPENDENCIES

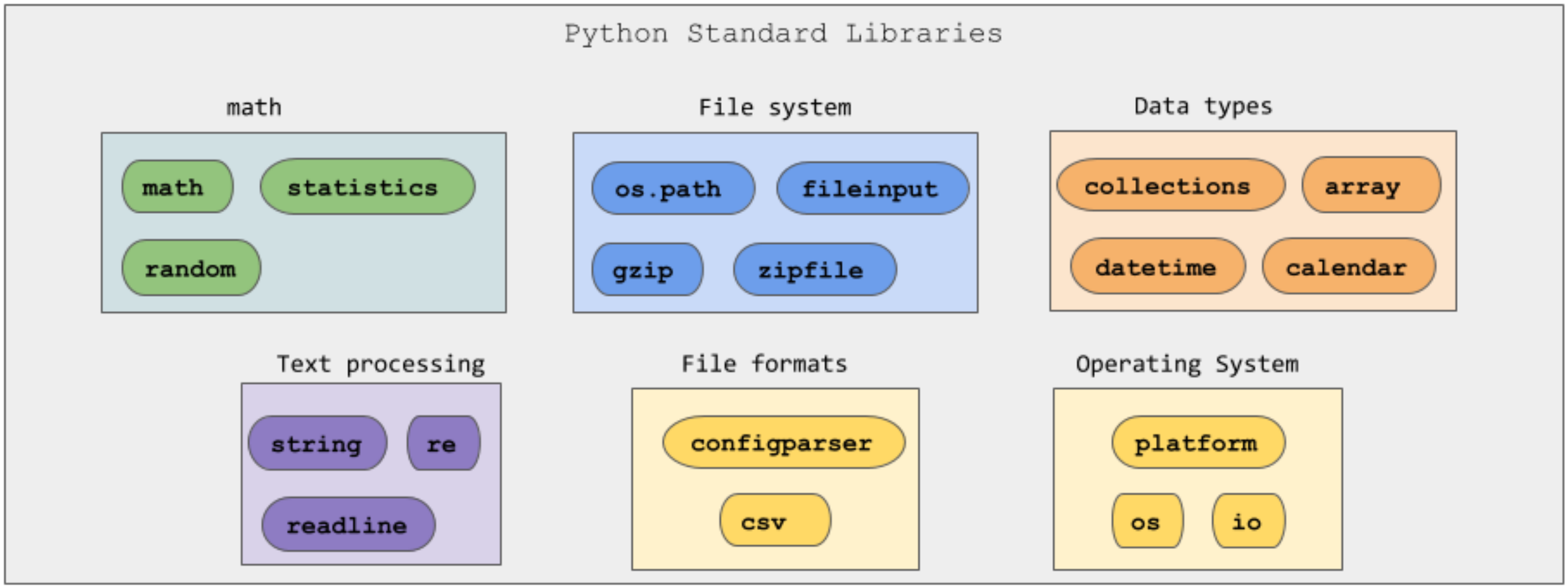
PYTHON LIBRARIES AS DEPENDENCIES

```python
import os
import socket
import warnings
from functools import update_wrapper
from threading import RLock

import werkzeug.utils
from werkzeug.exceptions import NotFound
from werkzeug.routing import BuildError
from werkzeug.urls import url_quote

from .globals import _app_ctx_stack
from .globals import _request_ctx_stack
from .globals import current_app
from .globals import request
from .globals import session
from .signals import message_flashed

```

LIBRARIES AND YOU



Python


Python Standard Libraries
math
math
statistics
random
File system
os.path
fileinput
gzip
zipfile
Data types
collections
array
datetime
calendar
Text processing
string
re
readline
File formats
configparser
csv
Operating System
platform
os
io

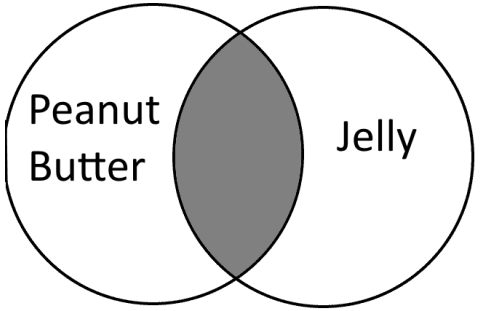
SOFTWARE BILL OF MATERIALS (SBOM)

PROGRAM FACETS

BOOLEAN

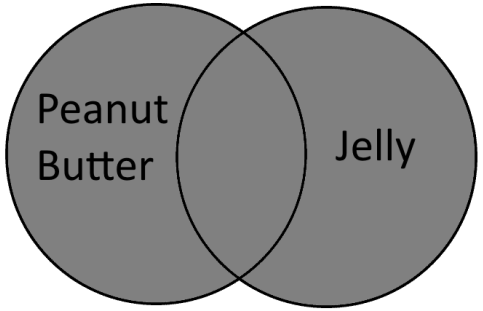

True
False

LOGICAL OPERATIONS



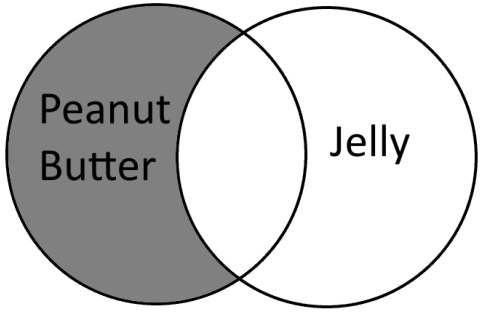
AND

Using AND, this search would only retrieve results with Peanut Butter and Jelly.



OR

Using OR, this search would retrieve results with peanut butter, with jelly, and with both.



NOT

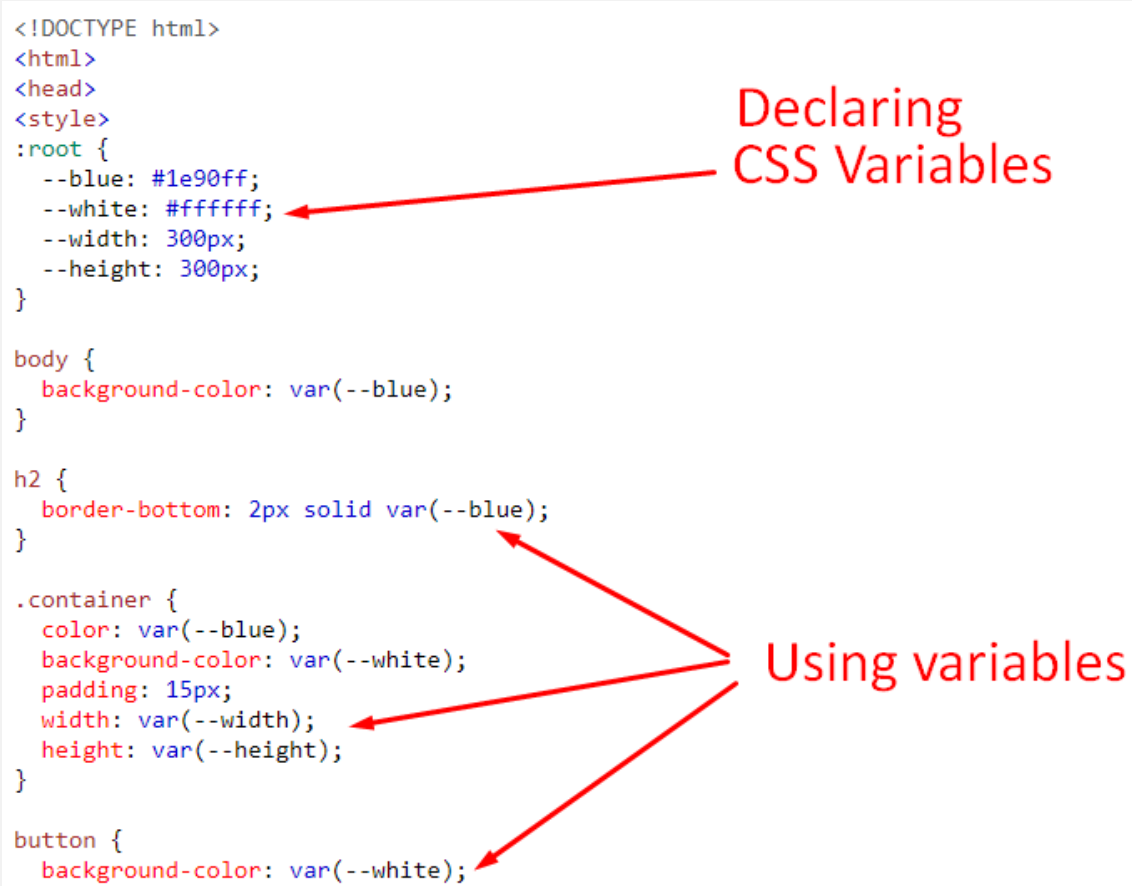
Using NOT, this search would retrieve results with peanut butter, and exclude those with jelly or PB with jelly.

Operator	Description
&&	AND
| |	OR
!	NOT
!=	NOT EQUAL TO
&	BITWISE AND
|	BITWISE OR
^	BITWISE XOR
&=	AND EQUAL
|=	OR EQUAL
^=	XOR EQUAL

10 MINUTE BREAK

VARIABLES

```
<!DOCTYPE html>
<html>
<head>
<style>
:root {
  --blue: #1e90ff;
  --white: #ffffff;
  --width: 300px;
  --height: 300px;
}

body {
  background-color: var(--blue);
}

h2 {
  border-bottom: 2px solid var(--blue);
}

.container {
  color: var(--blue);
  background-color: var(--white);
  padding: 15px;
  width: var(--width);
  height: var(--height);
}

button {
  background-color: var(--white);
  color: var(--blue);
  border: 1px solid var(--blue);
  padding: 5px;
}
</style>
</head>
<body>
```



NAMING VARIABLES

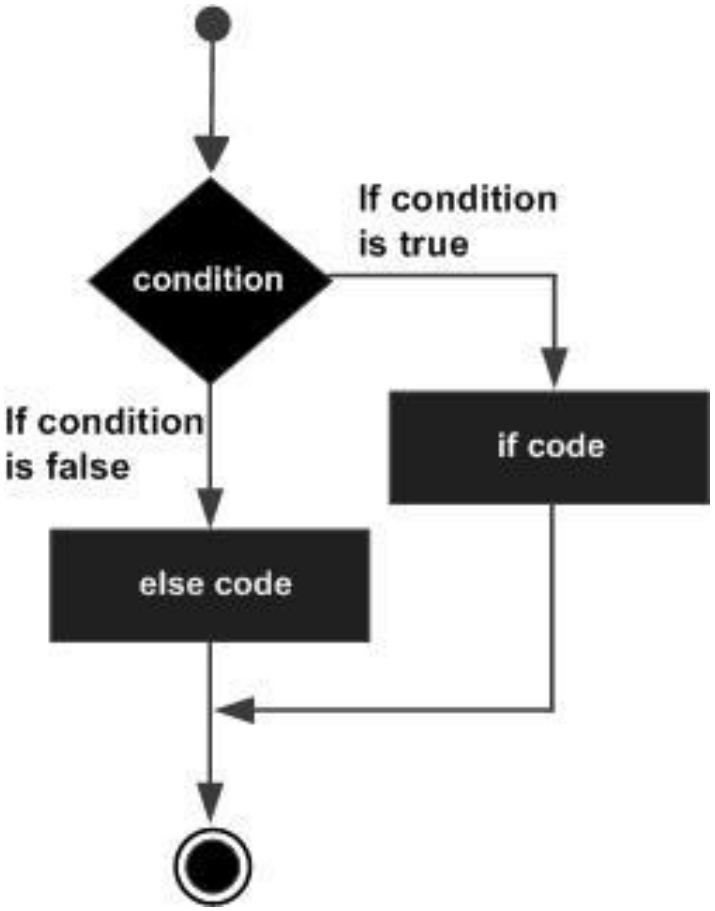
Giving them meaningful names, according to their use.

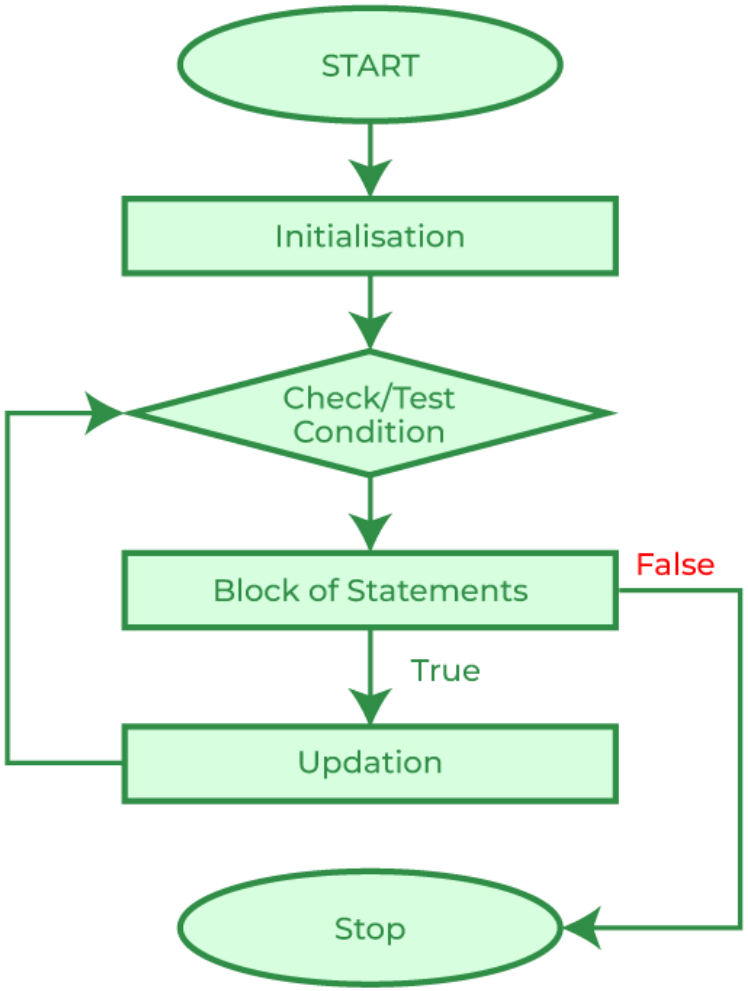
Giving them the most compact names possible, for less storage usage.

Giving them random names like "ahshjdn" or "yeetus".

LOOPS

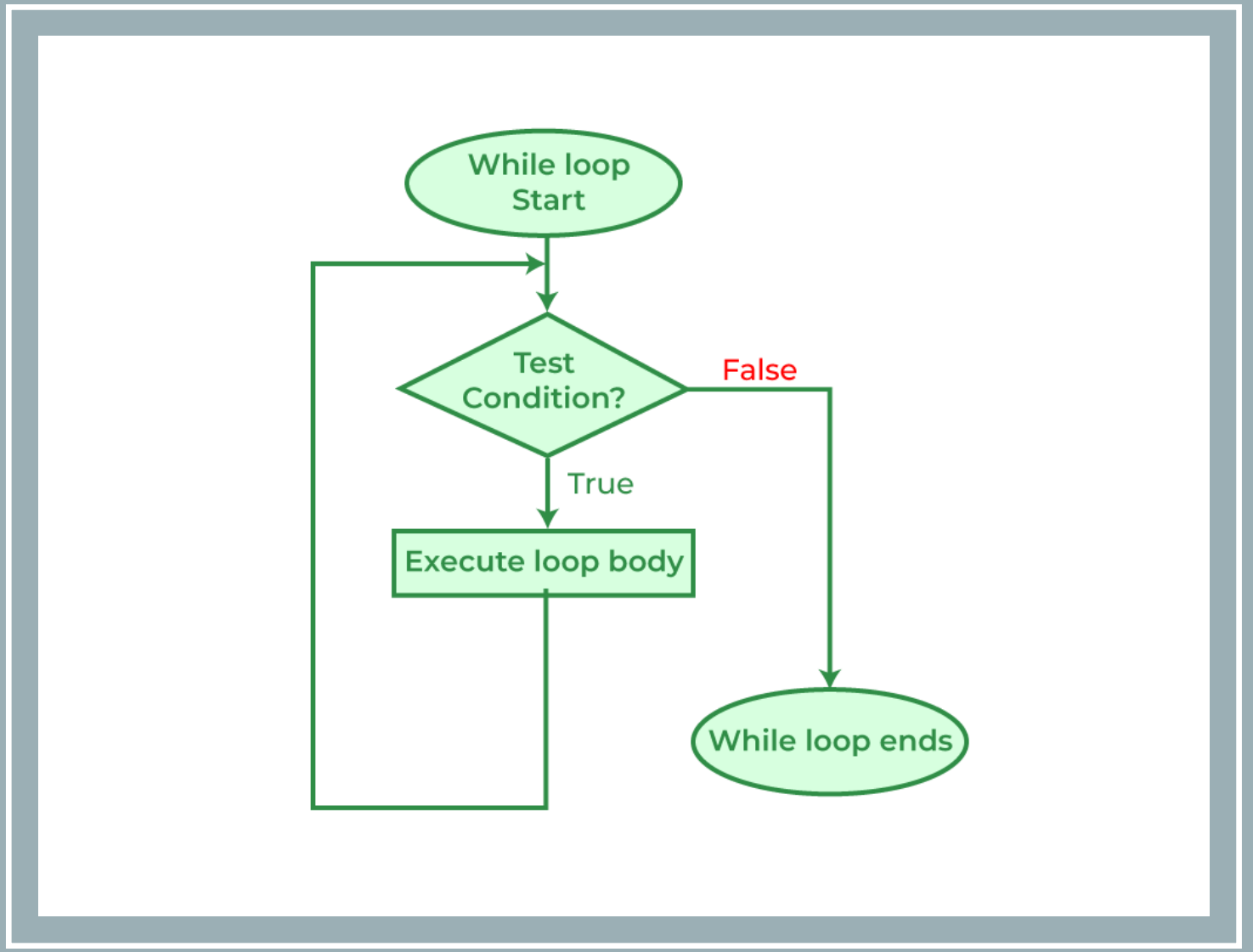
IF/ELSE LOOP


condition
If condition is true
If condition is false
if code
else code


START
Initialisation
Check/Test Condition
Block of Statements
False
True
Updation
Stop

FOR LOOP

WHILE LOOP


While loop Start
Test Condition?
False
True
Execute loop body
While loop ends

NESTED LOOPS

```
for i = 1 to 9
  for j = 1 to 9
    matrix(i,j) = i*j;
  end
end

// The result of this code is that a matrix variable with
// have 9 rows and 9 column (81 buckets), and every row,col
// will contain its multiplication value, such as:

//    | 1  2  3  4  5  6  7  8  9
// --+----------------------------
// 1 |  1  2  3  4  5  6  7  8  9
// 2 |  2  4  6  8 10 12 14 16 18
// 3 |  3  6  9 12 15 18 21 24 27
// 4 |  4  8 12 16 20 24 28 32 36
// 5 |  5 10 15 20 25 30 35 40 45
// 6 |  6 12 18 24 30 36 42 48 54
// 7 |  7 14 21 28 35 42 49 56 63
// 8 |  8 16 24 32 40 48 56 64 72
// 9 |  9 18 27 36 45 54 63 72 81
```

CASE STATEMENTS

```
43  # Set flags.
44  while getopts "ahk:c:u:l:" FLAG
45  do
46          case $FLAG in
47                  a)
48                          AWS=1
49                          ;;
50                  k)
51                          KEY="$OPTARG"
52                          ;;
53                  c)
54                          HOST=$OPTARG
55                          ;;
56                  u)
57                          USERC2=$OPTARG
58                          ;;
59                  l)
60                          USERLOCAL=$OPTARG
61                          ;;
62                  h)      echo "$USAGE"
63                          exit
64                          ;;
65                  *)
66                          echo "$USAGE"
67                          exit
68                          ;;
69          esac
70  done
71
```

LANGUAGES

MODERN SOFTWARE DEVELOPMENT CONCEPTS


WHY WOULD WE MAKE AN APP?

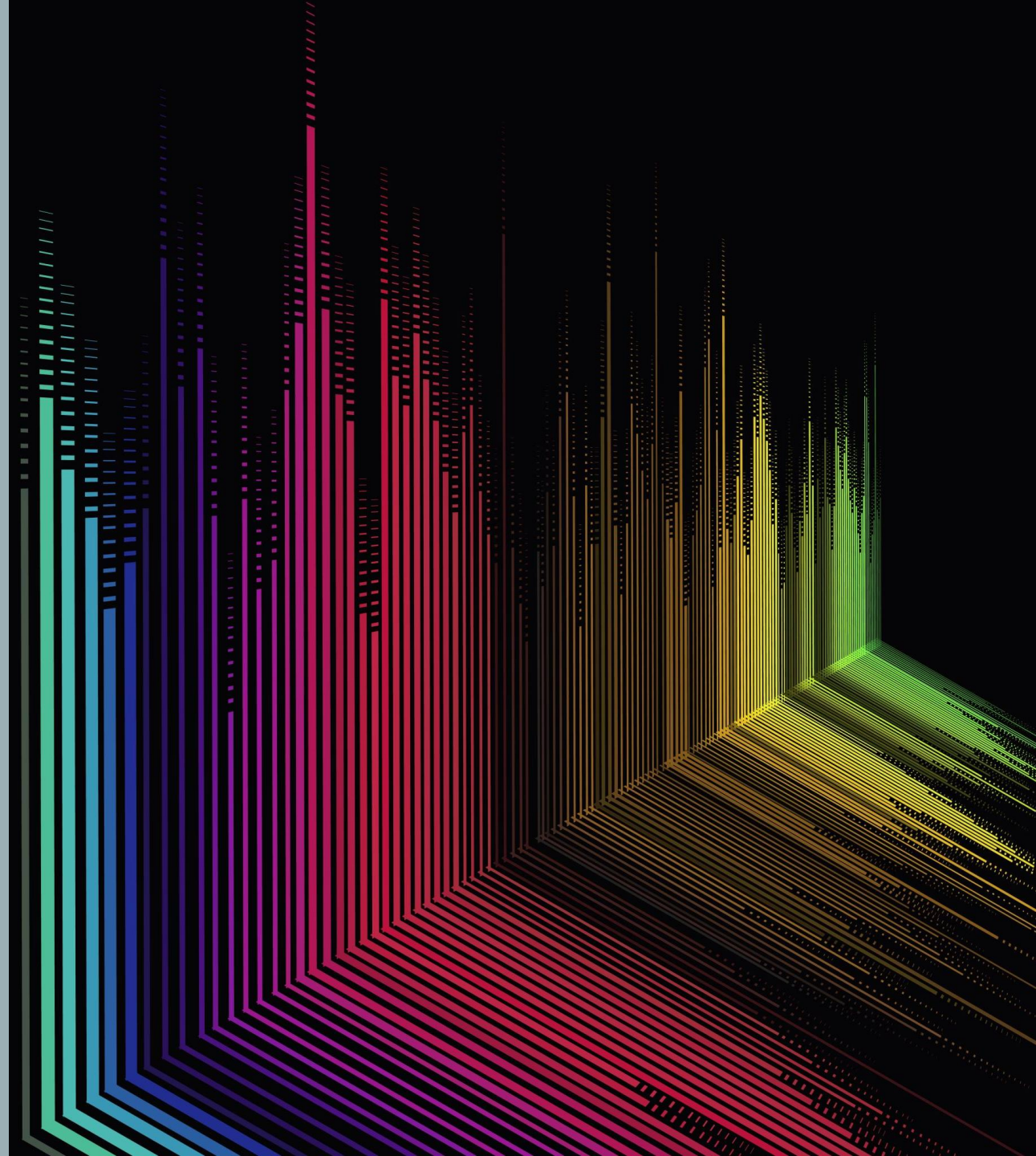
WHAT IS THE PROBLEM?

- What problems does this app solve?
- How does it solve these problems?

HOW DO WE BUILD A QUALITY APP?

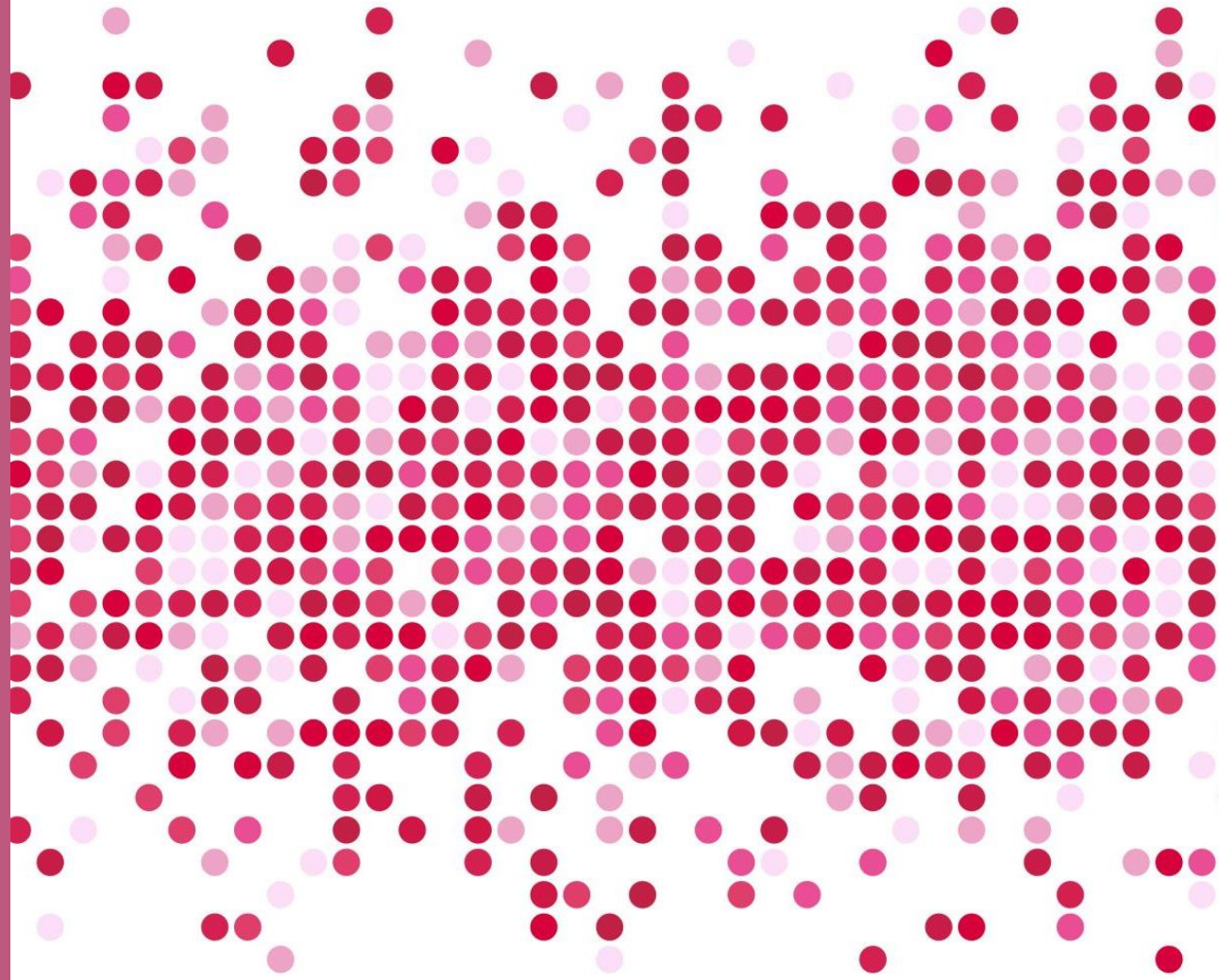
SYNTHESZING A SOLUTION

PROGRAMMING WITH TEAMS



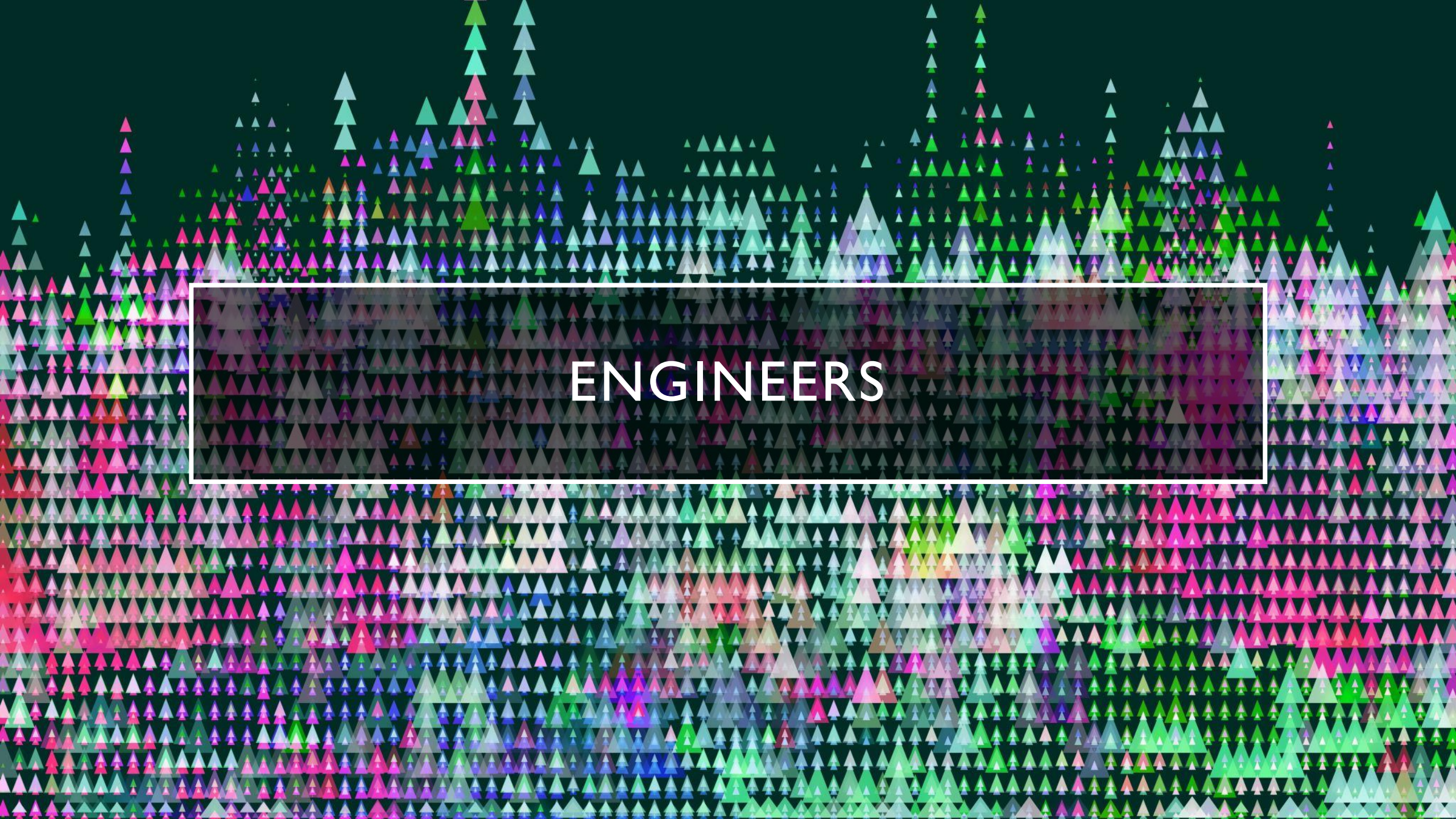
LEADERS & MANAGERS

GENERATE REQUIREMENTS



DEVELOPERS

DEVELOPERS WORK THE CODE


ENGINEERS

ENGINEERS MAKE STUFF WORK

SEPARATION OF DUTIES

10 MINUTE BREAK

TESTING

Does the code work?

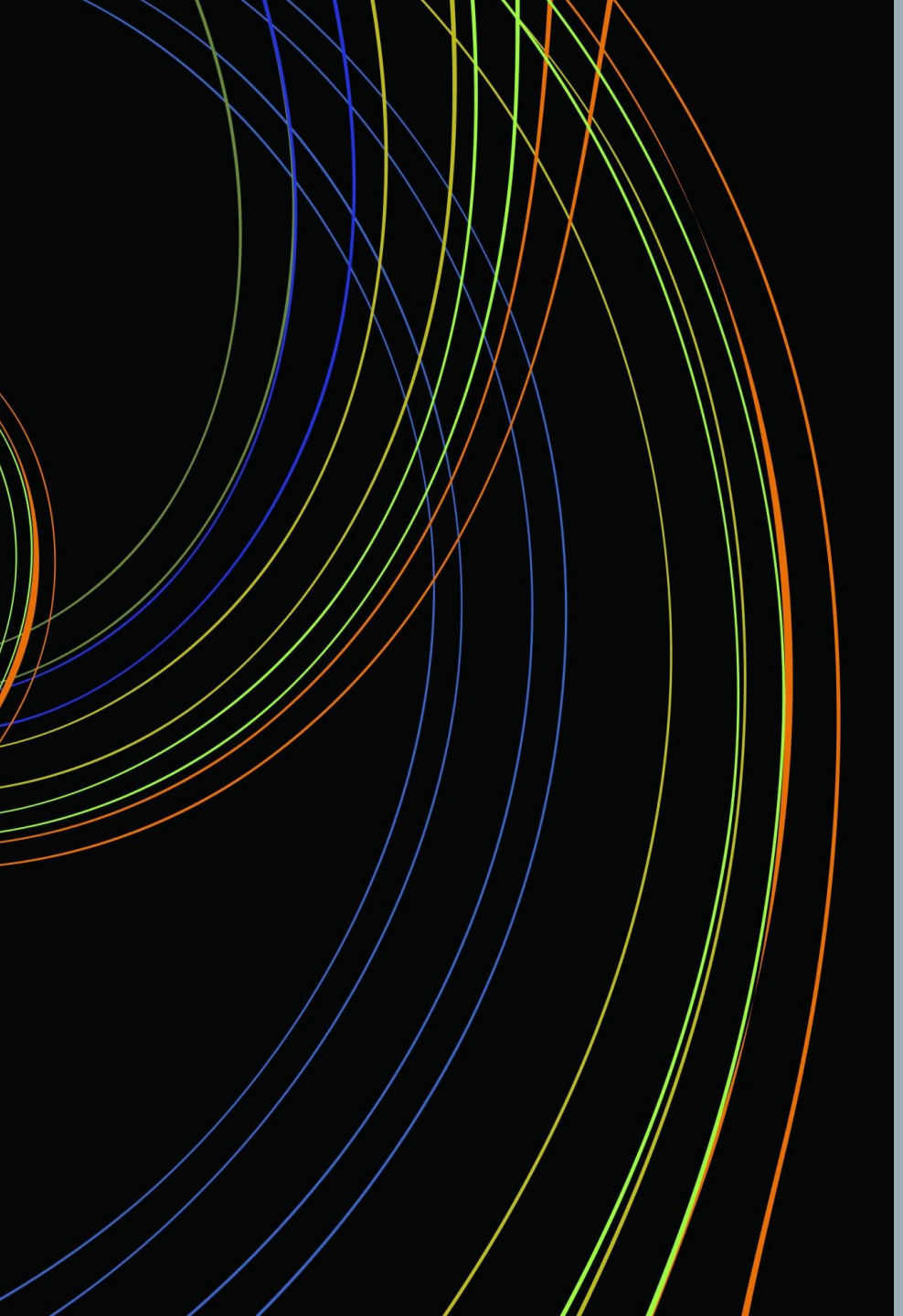
Does the code work *as expected*?

Does the code solve the problem we identified?

What does the code work on?

Does the code function with all of our supported systems?

Do we even know what possible systems exist in the field?



OUR CODE WORKS!

Now what?

CODE DISTRIBUTION

PROBLEM DISCOVERY

WHO ARE OUR USERS?

Do we know who they are?

What kind of sensitive information might we be processing?

Personally identifiable information?

HIPAA information?

Legally sensitive?

SUPPORTING THOSE CUSTOMERS

How do our customers contact us?

How do we contact them?

Do we have field support?

What is our process for addressing customer concerns/complains?

What is our process for addressing customer feature requests?

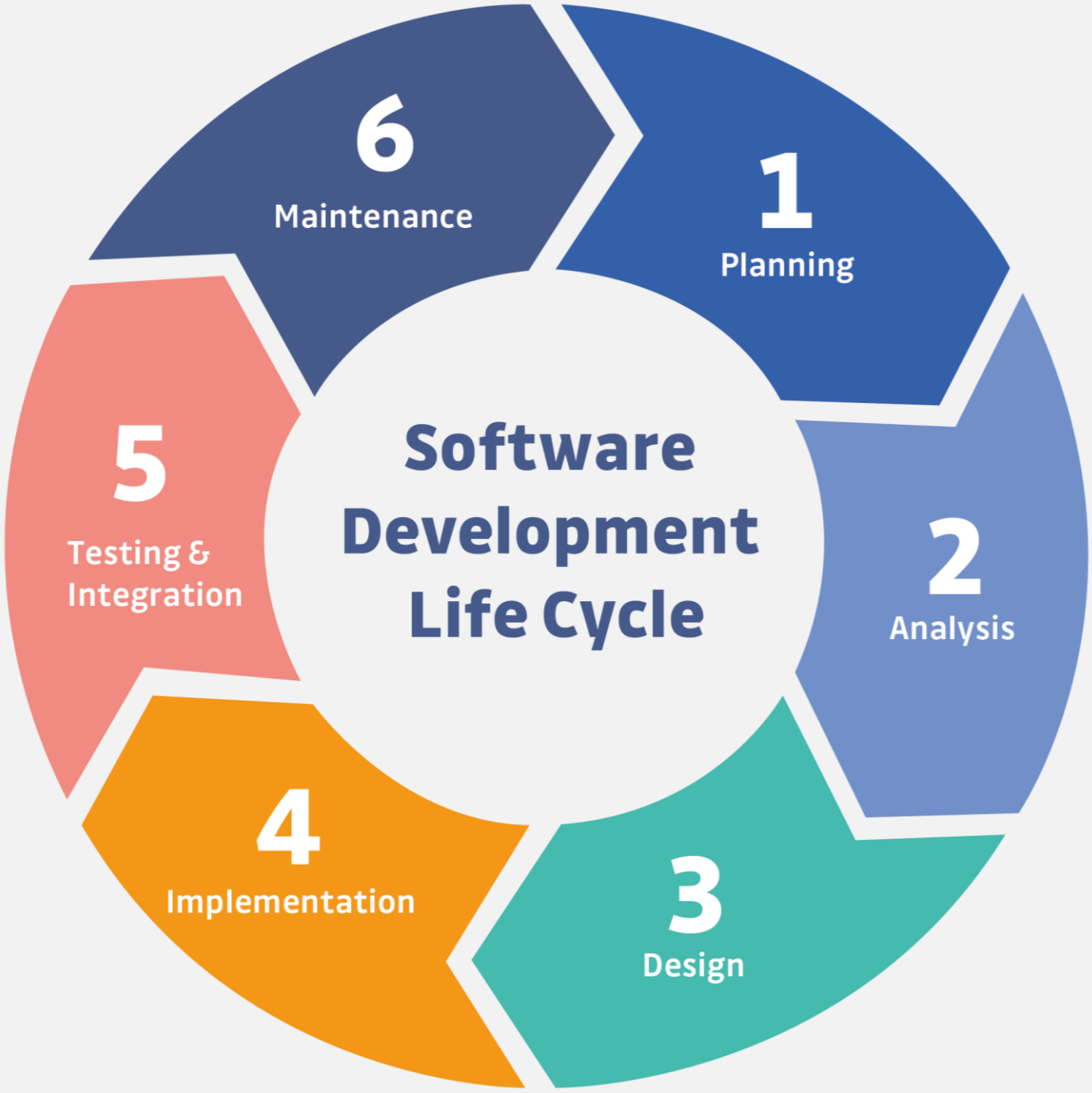
- Ticketing system?

SOFTWARE DEVELOPMENT LIFE CYCLE

WHAT IS THE SOFTWARE DEVELOPMENT LIFE CYCLE?

- The SDLC helps us organize the process of building software
- Includes steps for building, revising, and deploying software
- Includes roles associated with each step along the building process

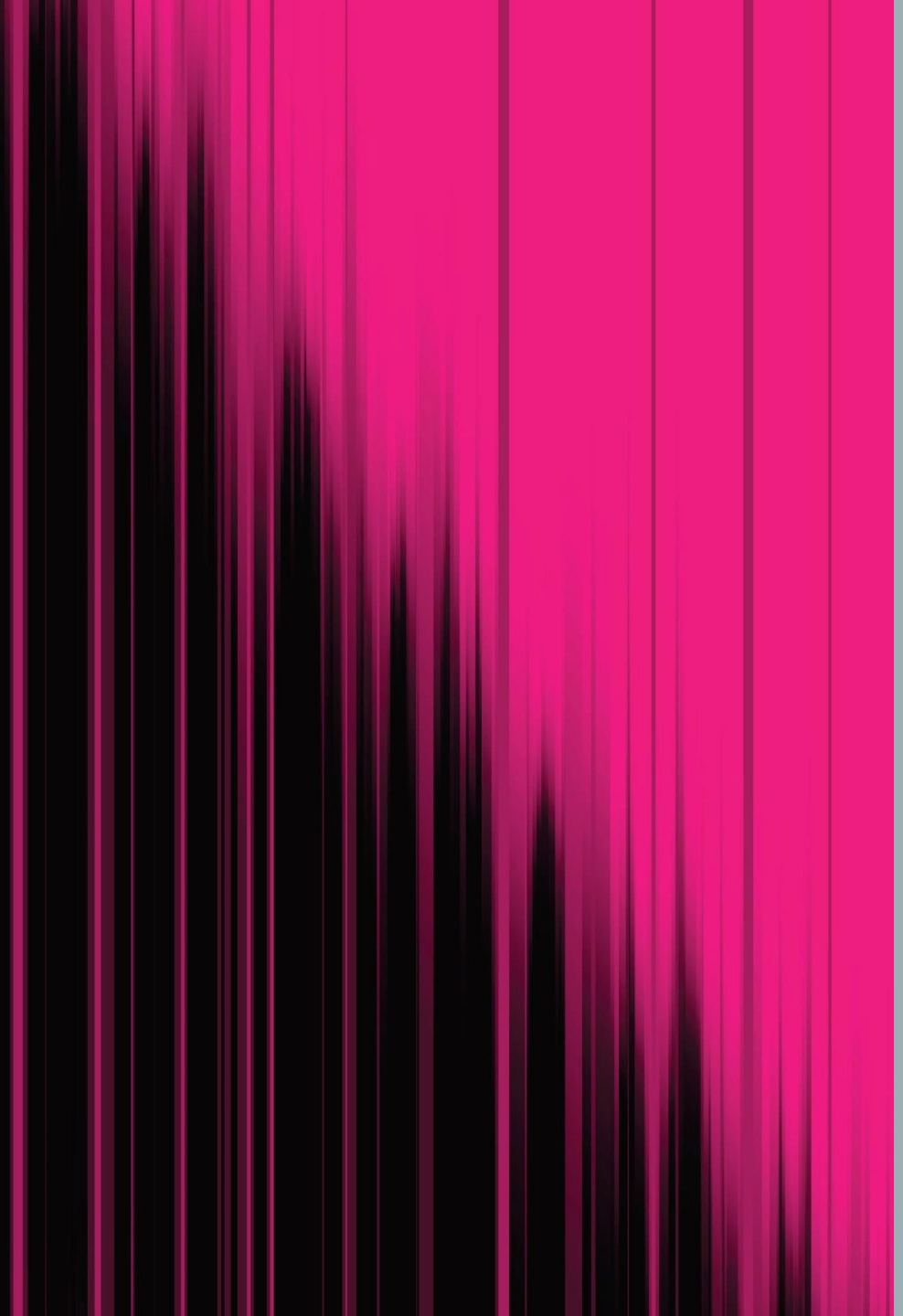
WHY DO WE NEED AN SDLC?


Software Development Life Cycle
1
Planning
2
Analysis
3
Design
4
Implementation
5
Testing & Integration
6
Maintenance

LET'S MAKE A GITHUB ACCOUNT

- https://github.com

QUESTIONS?



SESSION REVIEW