# DevOps and Agile

DevOps Culture - the goal, challenges, benefits.

DevOps CALMS model

Culture, Automation, LEAN, Measurement, Sharing

Agile teams - goals, common methodologies

scrum, kanban

# Programming Concepts

What is an algorithm?

Series of steps

What is pseudo code? Why is it useful?

Logical layout of code

What is the longest part of the software development lifecycle?

maintenance

How is source code translated into machine code?

interpreter

# Linux

Linux vs. Unix (differences, similarities)

diff – Linux was designed from start as open source

same – fs, toolset (gnu)

The core of Linux that manages resources / coordinates everything is the: \_\_\_\_\_\_\_\_

kernel

Common Open Source Licenses

gnu public license, free BSD, MIT

Common Linux Distributions

ubuntu, redhat

Everything in Linux is treated as a \_\_\_\_\_. Which usually means we can use what for everything we do in Linux?

File, read/edit/change, etc

Common Linux filesystem structure (/etc, /home, /usr)

## Common Command Line Tools / Shell Builtins

ls

cd

mkdir / rmdir

cp

mv

rm

echo

date

grep

find

ps

top

How can you get help for command in Linux?

Help, man pages

Redirection vs. Pipes

stdin, stdout, stderr

Hard Links vs. Soft Links

Inodes

Process Exit Status (success vs. error)

0 vs everything else

Regular Expressions

File Permissions / Users / Groups

## Bash Scripting

Flow Control (test, if, for, while)

allows to choose different paths or branches

# Python

Interpreted vs. Compiled Language

little of both

Dynamic vs. Static Type systems

dynamic – don’t have to declare variable type before using

Strong Typing vs. Weak Typing

strong – type (int, str, float)

Basic Data Types - str, int, float, bool

just need to declare variable

Common Built-in functions

print, range , list

Variables

object, container for data

Statements vs. Expressions

expression returns a value (statements = if, loop, etc)

Operators and Order of Operations

parens go first

Boolean operators and logic

true, false, and, or

Conditionals - if, elif, else

flow control

Looping - for, while, break/continue/else

Complex Types / Data Structures

list

[ ] usually same type

Iterating, Indexing, Slicing, Splitting, Joining

dict

key, value pair

Functions

objects

Modules

.py file

Objects / Classes

classes allow creation of own built-in types

File I/O

with (open and close file after use) – open, close, read, write

Exceptions - try, except

error handling – raise expection, will get trace back if not caught

# 

# REST

Uses HTTP Protocol Verbs – post (create), put (update) , get (retrieve), delete (delete)

Deals with “resources” - anything related to the data being accessed

Stateless – client request must contain all info like authenication. Server handles each request from client as a one-off, unrelated to any other request

# Unit Testing

Test smallest portions of an application (function, class) – like microservices – test small, easier to find errors

Isolate from environment as much as possible - make sure tests run the same no matter where they are run (use mock)

Not the only tests that should be run – run integration, acceptance, smoke, etc

Automated, fast – long tests are likely to be skipped

Test Driven Development – write failing tests that then pass to ensure code does what is expected

# Git

Distributed vs. Centralized Version Control – git is distributed

Repositories – contains code and history

Working Directory vs. Index vs. Repository – working dir contains local files, index is staging area, repo is staged files pushed up

Commits – clear messages indicating work that was done

Branches – alias for a commit (doesn’t copy files, just creates logical link to them)

Push / Pull – all new commits

Merging vs. Rebasing – rebase changes history (creates new commit numbers)

Common Workflows – small, frequent commits to avoid conflicts; use feature branching (never commit directly to master/main)

# Ansible

Infrastructure as Code – repeatable, automated

Ad-hoc Ansible – connects via ssh

Inventories – hosts -> groups of single hosts, groups of other groups

Playbooks - Plays, Tasks, Modules – written in yaml; plabooks contain plays with use tasks run by modules

Variables – most specific wins (ie playbook vs command line – command line wins)

Loops and Conditionals

Handlers – event handler, run only if certain conditions are true

Advanced Ansible - Roles, Playbook reuse / structure – roles are a list of tasks