# DevOps and Agile

DevOps Culture - the goal, challenges, benefits.

* [9 Key Benefits of DevOps - Business 2 Community](https://www.business2community.com/business-intelligence/9-key-benefits-of-devops-02391855)
* [10 Challenges to DevOps Adoption and How to Overcome Them | Contino | Global Transformation Consultancy](https://www.contino.io/insights/5-challenges-to-devops-adoption-and-how-to-overcome-them)
* [16 Challenges of DevOps in 2021 – From Adoption to Implementation to Scaling | (bunnyshell.com)](https://www.bunnyshell.com/blog/challenges-of-devops)

DevOps CALMS model

* [CALMS Framework | Atlassian](https://www.atlassian.com/devops/frameworks/calms-framework)

Agile teams - goals, common methodologies

* [Agile Project Management - A Beginner's Guide | Adobe Workfront](https://www.workfront.com/project-management/methodologies/agile#:~:text=The%20goal%20of%20Agile%20is,the%20client's%20needs%20more%20effectively.)
* [Agile Manifesto Values and Principles (scrumalliance.org)](https://resources.scrumalliance.org/Article/key-values-principles-agile-manifesto)

# Programming Concepts

What is an algorithm?

* [What is an Algorithm? - Definition from WhatIs.com (techtarget.com)](https://www.techtarget.com/whatis/definition/algorithm)

What is pseudo code? Why is it useful?

* [What is Pseudocode? How to Use Pseudocode to Solve Coding Problems (freecodecamp.org)](https://www.freecodecamp.org/news/what-is-pseudocode-in-programming/#:~:text=Pseudocode%20literally%20means%20'fake%20code,and%20therein%20lies%20the%20problem.)

What is the longest part of the software development lifecycle?

* [The Maintenance Stage of the SDLC | Eternal Sunshine of the IS Mind (wordpress.com)](https://eternalsunshineoftheismind.wordpress.com/2013/03/02/the-maintenance-stage-of-the-sdlc/)
* [6 Stages of Software Development Life Cycle | Code Creators Inc](https://www.codecreatorsinc.com/6-stages-of-software-development-life-cycle/)

How is source code translated into machine code?

* [www2.hawaii.edu/~takebaya/ics111/process\_of\_programming/process\_of\_programming.html#:~:text=A compiler takes the program,to create an executable file.](http://www2.hawaii.edu/~takebaya/ics111/process_of_programming/process_of_programming.html#:~:text=A%20compiler%20takes%20the%20program,to%20create%20an%20executable%20file.)
* [Software-2 Languages (stanford.edu)](https://web.stanford.edu/class/cs101/software-2.html)

# Linux

Linux vs. Unix (differences, similarities)

* [Unix Vs Linux: What is Difference Between UNIX and Linux (softwaretestinghelp.com)](https://www.softwaretestinghelp.com/unix-vs-linux/)

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

* kernel

Common Open Source Licenses

* [Common Open Source Licenses - Search (bing.com)](https://www.bing.com/search?q=Common+Open+Source+Licenses&form=ANNTH1&refig=98b977622f704c0784cb29d1a64d73ce)

Common Linux Distributions

* [Common Linux Distributions - Search (bing.com)](https://www.bing.com/search?q=Common+Linux+Distributions&form=ANNTH1&refig=d636d625a4554976aed5122b9012490b)

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

* File
* Use common operations like read/write

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

* [The Linux File System Structure Explained - myTechMint](https://www.mytechmint.com/the-linux-file-system-structure-explained/)

## 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?

* Man pages

Redirection vs. Pipes

* [Understanding Pipes and Redirection For the Linux Command Line (maketecheasier.com)](https://www.maketecheasier.com/pipes-redirection-for-linux-command-line/#:~:text=Understanding%20Pipes%20and%20Redirection%20For%20the%20Linux%20Command,experimenting%20with%20them%20and%20see%20how%20they%20interact.)

Hard Links vs. Soft Links

* [Difference between Hard link and Soft link - GeeksforGeeks](https://www.geeksforgeeks.org/difference-between-hard-link-and-soft-link/)

Inodes

* [inodes meaning - Search (bing.com)](https://www.bing.com/search?q=inodes+meaning&form=ANNTH1&refig=f0c3f8add79a4fb6be63cc66cfa12c88&sp=4&qs=LT&pq=inodes&sk=EP1LT1UT1&sc=8-6&cvid=f0c3f8add79a4fb6be63cc66cfa12c88)

Process Exit Status (success vs. error)

* [Process Exit Status (success vs. error) - Search (bing.com)](https://www.bing.com/search?q=Process+Exit+Status+%28success+vs.+error%29&form=ANNTH1&refig=dca9207ce2f04abe9aaf7321bde7dcae)

Regular Expressions

* [Regular expression - Wikipedia](https://en.wikipedia.org/wiki/Regular_expression)

File Permissions / Users / Groups

* [Linux File Permissions and Ownership Explained with Examples (linuxhandbook.com)](https://linuxhandbook.com/linux-file-permissions/)

## Bash Scripting

Flow Control (test, if, for, while)

* [Python Control Flow Statements and Loops – PYnative](https://pynative.com/python-control-flow-statements/)

# Python

Interpreted vs. Compiled Language

* [Difference Between Compiled and Interpreted Language | Difference Between](http://www.differencebetween.net/technology/difference-between-compiled-and-interpreted-language/#:~:text=Difference%20between%20Compiled%20and%20Interpreted%20Language%201%20Basics,4%20Debugging.%20...%205%20Compiled%20Language%20vs.%20)

Dynamic vs. Static Type systems

* [Type Systems in Programming Languages: Static vs. Dynamic and Strong vs. Weak | by Yong Cui | Better Programming](https://betterprogramming.pub/type-systems-in-programming-languages-static-vs-dynamic-and-strong-vs-weak-ed1bb542b06)

Strong Typing vs. Weak Typing

* [Type Systems in Programming Languages: Static vs. Dynamic and Strong vs. Weak | by Yong Cui | Better Programming](https://betterprogramming.pub/type-systems-in-programming-languages-static-vs-dynamic-and-strong-vs-weak-ed1bb542b06)

Basic Data Types - str, int, float, bool

* [Understanding Python 3 data types: string, int, float and boolean (able.bio)](https://able.bio/ZoranPandovski/primitive-data-types-in-python-3--57tqcfp)

Common Built-in functions

* [10 Python built-in functions you should know | by Amanda Iglesias Moreno | Towards Data Science](https://towardsdatascience.com/10-python-built-in-functions-you-should-know-f6beba1698bb)

Variables

* Variables are containers for storing data values.

Statements vs. Expressions

* [Difference between expression and statement in Python - MUDDOO](https://muddoo.com/tutorials/difference-between-expression-and-statement-in-python/)

Operators and Order of Operations

* [Operations In Python. Order of Operations in Python | by Thoa Shook | Medium](https://medium.com/@thoashook/operations-in-python-69bbbef781a4)

Boolean operators and logic

* [Boolean Operators | Quick Guide with Examples (scribbr.com)](https://www.scribbr.com/working-with-sources/boolean-operators/)

Conditionals - if, elif, else

* [Python If..Elif..Else Conditional – PythonTect](https://pythontect.com/python-if-elif-else-conditional/)

Looping - for, while, break/continue/else

Complex Types / Data Structures

list

Iterating, Indexing, Slicing, Splitting, Joining

dict

Functions

Modules

Objects / Classes

File I/O

Exceptions - try, except

# 

# REST

Uses HTTP Protocol Verbs

Deals with “resources”

Stateless

* [What is a REST API? (redhat.com)](https://www.redhat.com/en/topics/api/what-is-a-rest-api)

# Unit Testing

* [DevOps Unit Testing - Tutorial (vskills.in)](https://www.vskills.in/certification/tutorial/devops-unit-testing/)

Test smallest portions of an application (function, class)

Isolate from environment as much as possible

Not the only tests that should be run

Automated, fast

Test Driven Development

# Git

* [Git - Basic Branching and Merging (git-scm.com)](https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging)

Distributed vs. Centralized Version Control

Repositories

Working Directory vs. Index vs. Repository

Commits

Branches

Push / Pull

Merging vs. Rebasing

Common Workflows

# Ansible

* [Ansible Documentation — Ansible Documentation](https://docs.ansible.com/ansible/latest/index.html)

Infrastructure as Code

Ad-hoc Ansible

Inventories

Playbooks - Plays, Tasks, Modules

Variables

Loops and Conditionals

Handlers

Advanced Ansible - Roles, Playbook reuse / structure