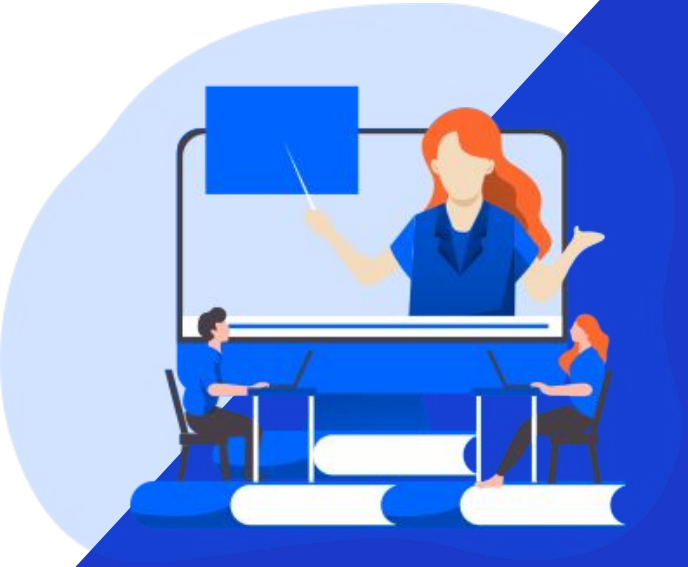
Python Backend Course

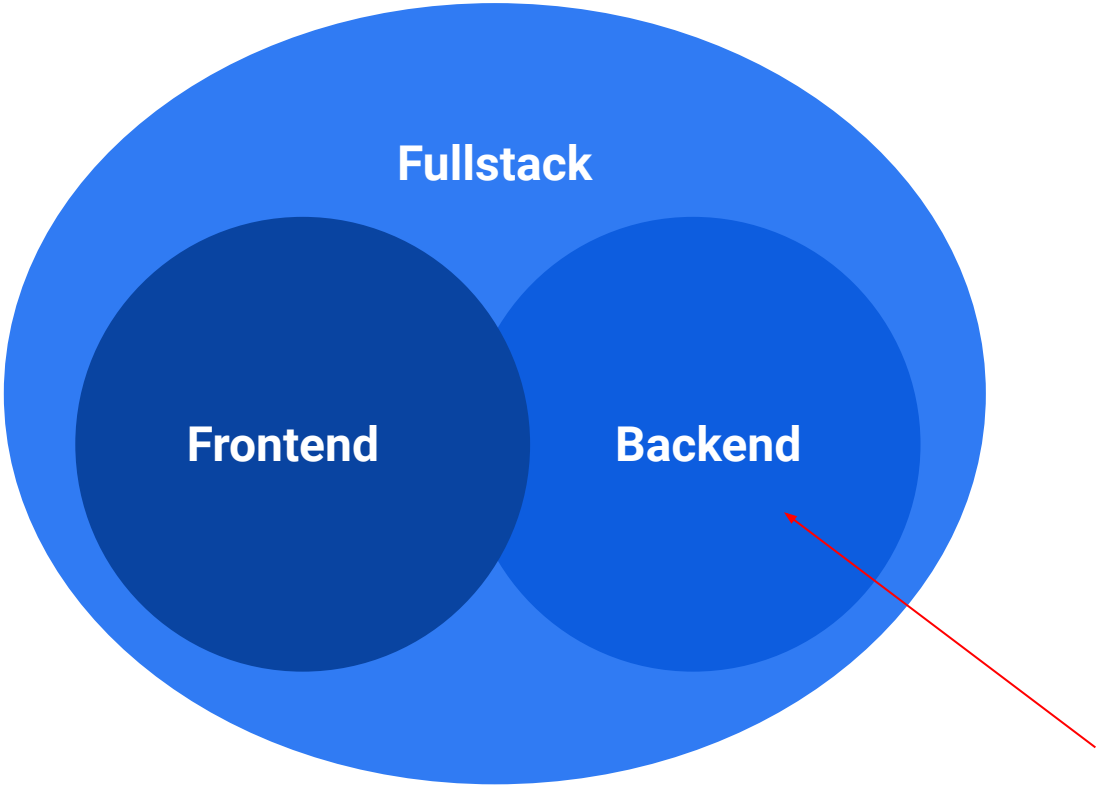


June 2024

Today's Agenda

- Course Introduction Presentation
- Python Curriculum Overview
- Introduction: Markup Languages & Programming Languages
- Technical Setup of Laptops
- Introduction to Linux

Developer Roles



Language Landscape

	Markup and Styling Languages (HTML, CSS)	**Interpreted** (JS, Python)	**Compiled** (Java, C++)
Frontend	⭐⭐	⭐	
Backend		⭐⭐	⭐⭐
Fullstack	⭐⭐	⭐⭐	⭐⭐

Course Highlights

Python

Fundamentals (13 days)	Python Basics (62 days)	Databases (23 days)	Frameworks (11 days)
- Terminal, Git, GitHub - HTML, CSS	- Python programming skills - Test Driven Development	- Relational Database Management System (RDBMS) - PostgreSQL	- Django Framework

APIs (11 days)	Cloud Services (14 days)	Final Project (21 days)
- REST API	- Cloud Computing & Servers - Services in GCP (Google Cloud Platform)	- Application of learned material by collaboratively building a real-world application.

Job in Details

- **Python is Everywhere!**
  - Web Applications
  - Machine Learning
  - Artificial Intelligence
  - Data Science
  - Desktop Applications
  - Game Applications

Job in Details

Software Development Lifecycle

Phase 1: *Requirement Analysis*

Phase 2: *Feasibility Study*

Phase 3: *Architectural Design*

Phase 4: *Software Development*

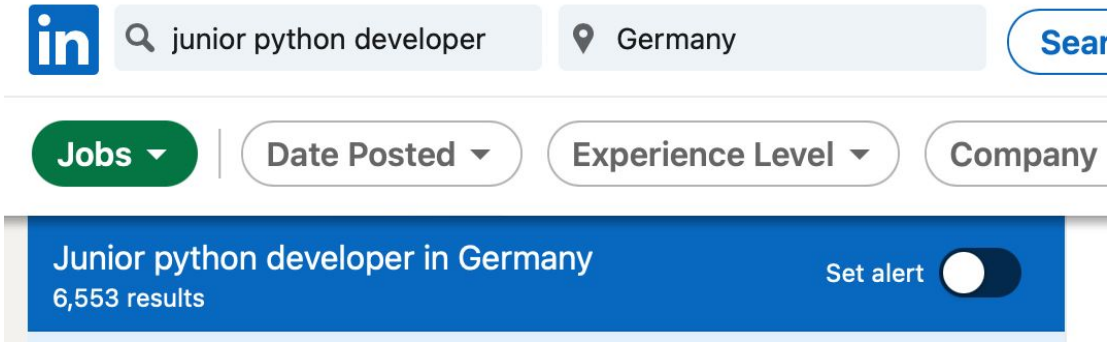
Phase 5: *Testing*

Phase 6: *Deployment*

Phase 7: *Maintenance*

Job Opportunity

Germany



*Right now, there are more than **6.000+** open positions for a **Junior Python Developer** in **Germany**.*

Job Roles

Python

- Varies depending on company and specific position
- A **Junior (entry level) position** does not require much knowledge, but a fundamental understanding of Python is required
- Responsible to write, analyze, test, and debug code
- Part of teams that have mentors and leaders whom they have to report to

Needed Skills

Python

Essential Skills for a Junior Developer:

- Passion for learning!
- Python syntax;
- Tools for coding (IDEs), version-control systems and services (GitHub, GitLab);
- Frameworks for building web projects (Django, Flask);
- Object-relational mapping;
- Building automation tools;

Needed Skills

Python

Essential Skills for a Junior Developer:

- Tools for unit testing;
- Basic knowledge of other common programming languages, like JavaScript, and technologies such as HTML5/CSS3;
- Basic practical experience in programming and code writing;
- Knowledge of databases and operating systems;
- Ability to **learn new software platforms and technologies quickly** (quite an important skill for any Junior coder);
- Ability to **follow instructions and work in a team environment** (another skill that in no way should be underestimated, even though it often does).

Course Modules

FUNDAMENTALS

Course Intro	Versioning & Collaboration	Introduction to the Internet	Introduction to Web Pages	Programming	Preparing the Environment
- Presentation - Linux (Ubuntu) - Terminal usage - Manipulating files - Installing packages - Markdown	- Intro to VCS - Git - GitHub	- DNS - Protocols - Requests & Responses	- HTML Basics - CSS Basics	- Programming Languages - Algorithms - Frontend vs. Backend - JSON	- Setting up Python - Setting up the IDE - Plugins

Course Modules

PYTHON BASICS

Introduction	Texts	Dates	Functions	Statements & Loops	Logical Thinking
- Primitive types - Variables - Operators - Commenting	- Strings - Common Manipulation - Encoding	- Date and time types - Formatting - Manipulation	- Parameters - Returns - Scoping - Lambdas	- if/else & switch - while / for loops - break & continue	- Logical operators - Truth table

Collections	Algorithmic Thinking	Debugging	OOP Concepts	OOP in Practice	Testing
- Lists - Arrays - Dictionaries	- Concept of Algorithm - Sorting problems - Complexity - Different sorting algorithms	- Debugging using the IDE - Breakpoints - Read variables - Debugging using CLI - Advanced debugging	- Class vs. Objects - Constructors, methods and variables - Inheritance - Visibility - Encapsulation	- Abstract classes & Interfaces - Design patterns - Polymorphism - Casting	- Concepts and types - Advantages of testing - TDD

Course Modules

PYTHON BASICS

Coding Standards	I/O	Basic Tools & Libraries	Exceptions
- Coding style - Linting	- Manipulating files	- pyenv - Including libraries - 3rd party libraries	- Principle of exceptions - Syntax - Stacktrace

Course Modules

DATABASES

Overview	Basic Usage	Usage in Python	Advanced SQL	Consistency	Basic Performance
- Intro to DMBS - Relational models	- Introduction to SQL - Basic statements - Data types - PostgreSQL	- Connect to PostgreSQL using psycopg2 - Manipulate data	- Joins - Advanced data types - Subqueries	- Transactions & ACID - Rollback & Locking	- Reading from disk - Caching - Indexes

Course Modules

FRAMEWORK

Django Framework	MVC	ORM
- Intro to Django - MVC - django-admin - Testing	- Request lifecycle - Render pages - Form processing - Sessions & cookies	- Django migrations - Models

Course Modules

APIs

Basics	Consuming REST APIs	Creating REST APIs
- Concepts of an API - Different types of APIs - REST APIs	- Postman - Client Libraries - Error handling	- Endpoints - Swagger - Designs - Validation

Cloud Services

Intro to Cloud	GCP Overview
- What is Cloud	- GCP Console - CloudStorage - AppEngine - BigQuery

Final Project

Final Project
- Group Project Work

Course Highlights

Lectures
Done by a professional teacher

Exercises
Variety of exercises for each module

Individual Projects
A project that evolves during the course

Group Projects
Experience working with other developers

Soft Skills
Develop a growth mindset

Industry Lectures
Hear from successful businesses

Language Classes
Learn your level of German

GitHub Education
Access to modern and professional softwares

Final Project
Following Agile Methodologies
(similar to a company environment)

Developer Career Path

01. Software Engineer

0-3 years experience

Build softwares, launch and debug applications

Skills:

Programming languages, operating systems, algorithms and databases

02. Senior Software Engineer

3-6 years experience

Oversee software development and coach engineers

Skills:

Basic architecture, advanced code design, coaching & training

03. Tech Lead

6-9 years experience

Manage a team responsible for complex software solutions

Skills:

Advanced architecture, system design, project management

04. Engineering Manager / VP of Engineering

9-13 years experience

Owns processes, product thinking and technical leadership

Skills:

Advanced architecture and system design

05. Chief Technology Offices

13+ years experience

Responsible for organization's technological needs and R&D

Skills:

Hiring, people skills, strategic thinking

Digital Career Institute

Intro: Markup Languages & Programming Languages


Scripting Language
Programming Language
Markup Language

Programming languages

C
C++
C#
Java

Scripting languages

Python
JavaScript
Php
Perl
VBScript

Markup Languages

HTML
CSS
XML

Digital Career Institute

Introduction to Linux

What is Linux?

Linux is an operating system, like macOS or Windows. Linux is derived from UNIX.

1. It is also the most popular **Open Source and free**, as in freedom, operating system.

2. **The Linux "core" (called kernel)** was born in 1991 in Finland, and it went a really long way from its humble beginnings.

3. **Linux is the ultimate freedom.** It is developed by volunteers and there's no single commercial company that can dictate what goes into Linux, or the project priorities.

4. **No one dictates which apps you run**, and you don't have pre-installed apps that track you, your location, etc.

How Linux works?

What is Terminal?

The Linux terminal is also known as the command-line, console, or shell.

1. It is a text interface for our computer.
2. We can interpret the commands and also write our scripts with this system program.
3. It might look difficult at first but once we get familiar with it, it will be easy to use.
4. Terminal commands are the instructions that you type into the terminal to execute a specific task.

Linux Commands

Terminal commands are the instructions that you type into the terminal to execute a specific task.

- man
- pwd
- cd
- ls
- ls – a
- ls – l
- ls – al
- mkdir
- rmdir
- mv
- cp
- touch
- gzip
- gunzip
- tail
- cat
- less
- nano
- help

How to launch the Terminal?

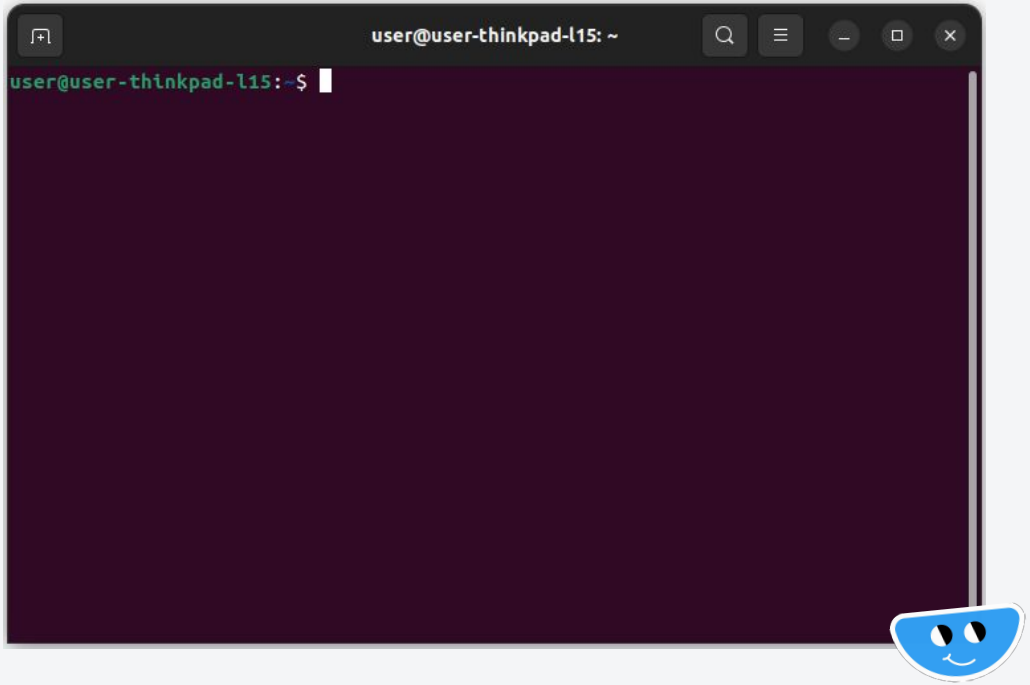
Linux: Ctrl + Alt + T

Windows:

1. windows + R
2. type cmd
3. Enter

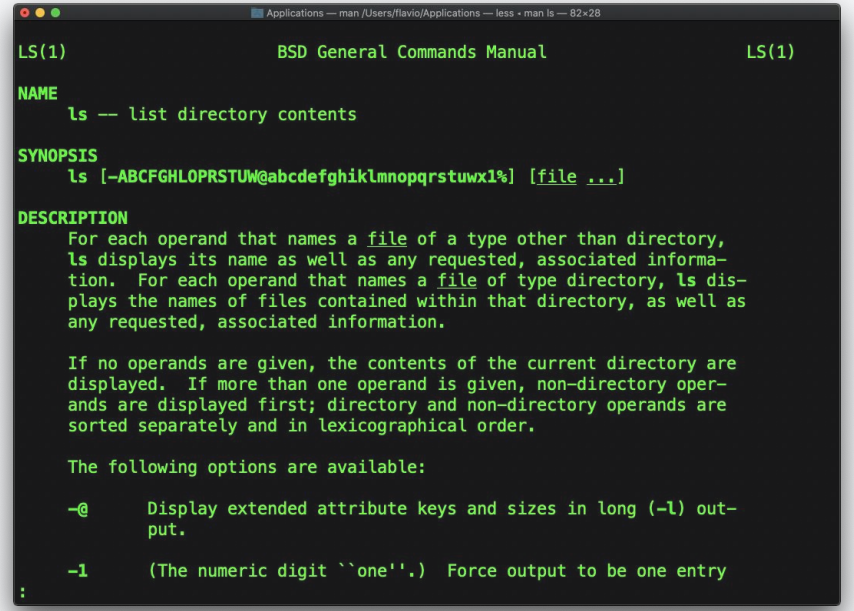
Mac:

1. Command + Space bar
2. Terminal
3. Click on the app



man

- The first command I want to introduce is a command that will help you understand all the other commands.
- Every time I don't know how to use a command, I type `man <command>` to get the manual.
- This is a man (from manual) page. Man pages are an essential tool to learn, as a developer.
- They contain so much information that sometimes it's almost too much.

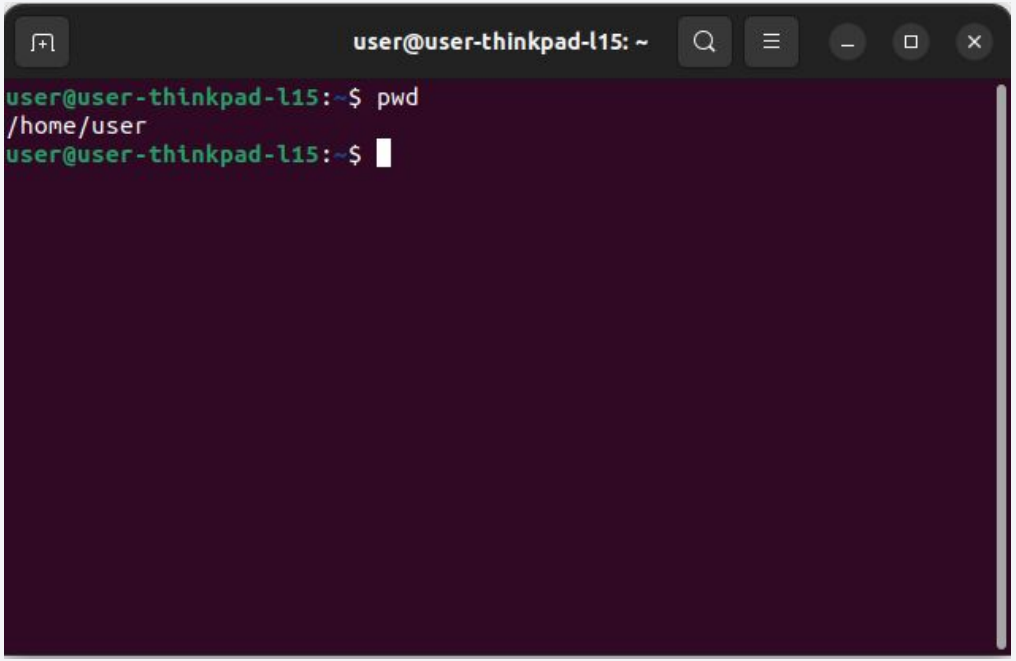


pwd

- Whenever you feel lost in the filesystem, call the command to know where you are:

  pwd

- It will print the current folder path.



cd

- Once you have a folder, you can move into it using the `cd` command. `cd` means change directory.

- You invoke it specifying a folder to move into. You can specify a folder name, or an entire path.

  Example:

  ```
  mkdir fruits
  cd fruits
  ```

  Now you are into the `fruits` folder.
- You can use the `..` special path to indicate the parent folder:

  ```
  cd .. #back to the home folder
  ```

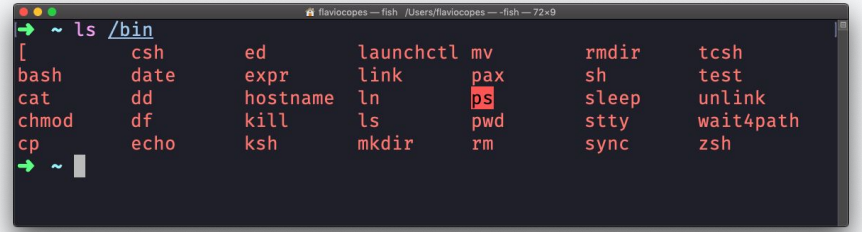
ls

- Inside a folder you can list all the files that the folder contains using the ls command:
  `ls`

- If you add a folder name or path, it will print that folder contents:
  `ls /navigation_demo`

- ls accepts a lot of options. One of my favorite options combinations is `-al`.

- Try:
  `ls -al /navigation_demo`

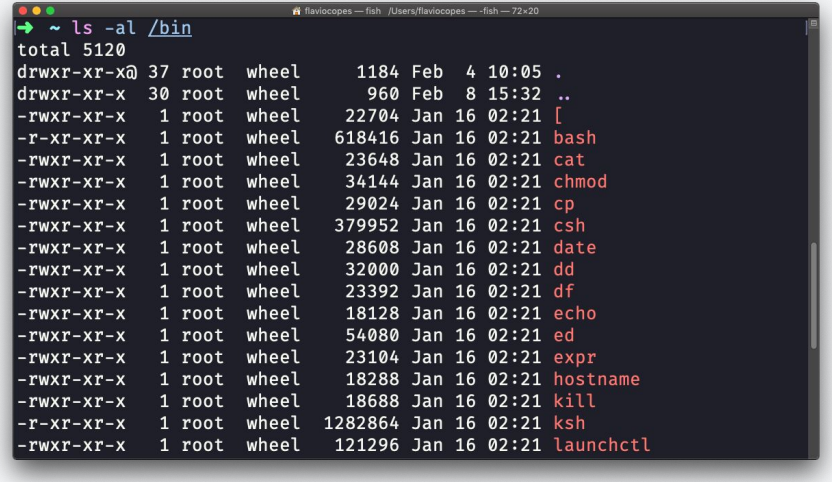


`ls -al`

Compared to the plain `ls` , `ls -al` this returns much more information.

You have, from left to right:

- the file permissions (and if your system supports ACLs, you get an ACL flag as well)
- the number of links to that file
- the owner of the file
- the group of the file
- the file size in bytes
- the file modified datetime
- the file name



Exercise

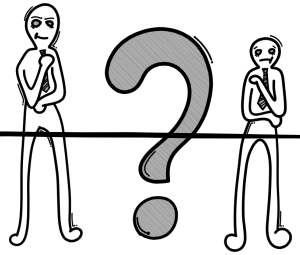
Objective:
Get familiar with man, pwd, cd, ls, ls -al.

Scenario

- Create a folder called Navigation_Demo in your home directory
- In Navigation_Demo folder create 5 sub_dir folders (sub_dir_1, sub_dir_2, sub_dir_3, sub_dir_4, sub_dir_5)
- In Navigation_Demo folder create 3 text files, named text_file_1, text_file_2, text_file_3.
- In Navigation_Demo folder create 3 hidden text files names .hidden_file_1, .hidden_file_2, .hidden_file_3

Your task:

- Print your current directory
- Change your directory to Navigation_Demo
- Show the files and folders within Navigation_Demo
- Show the hidden files within Navigation_Demo
- Change your directory to sub_dir_1
- From sub_dir_1 change directory to Navigation_Demo



mkdir

- You create folders using the mkdir command:

  ```
  mkdir fruits
  ```

- You can create multiple folders with one command:

  ```
  mkdir dogs cars
  ```

- You can also create multiple nested folders by adding the -p option:

  ```
  mkdir -p fruits/apples
  ```

- Options in UNIX commands commonly take this form. You add them right after the command name, and they change how the command behaves. You can often combine multiple options, too.

rmdir

- Just as you can create a folder using mkdir , you can delete a folder using rmdir :

```
mkdir fruits
rmdir fruits
```

- You can also delete multiple folders at once:

```
mkdir fruits cars
rmdir fruits cars
```

- The folder you delete must be empty.

rmdir

- To delete folders with files in them, we'll use the more generic command which deletes files and folders, using the -rf options:

  ```
  rm -rf fruits cars
  ```

- Be careful as this command does not ask for confirmation and it will immediately remove anything you ask it to remove.

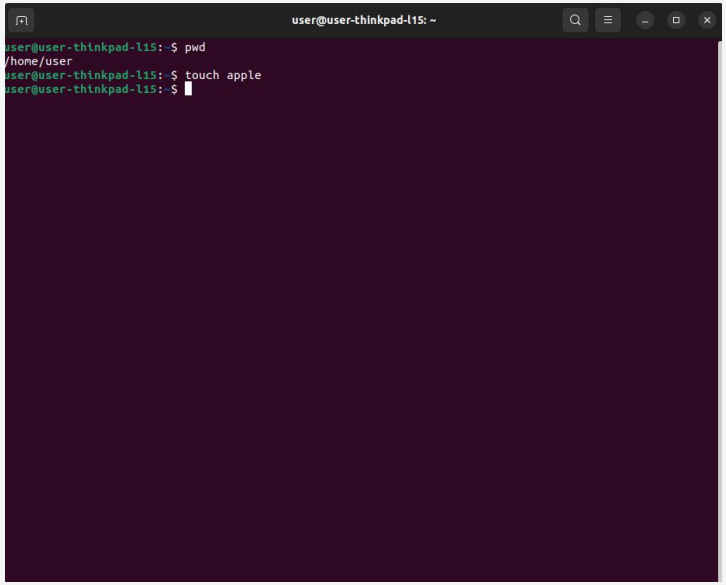
- There is no bin when removing files from the command line, and recovering lost files can be hard.

touch

- You can create an empty file using the touch command:

  ```
  touch apple
  ```

- If the file already exists, it opens the file in write mode, and the timestamp of the file is updated.



mv

- Once you have a file, you can move it around using the mv command. You specify the file current path, and its new path:

```
touch test
mv pear new_pear
```

- The `pear` file is now moved to `new_pear` . This is how you **rename** files and folders.

- If the last parameter is a folder, the file located at the first parameter path is going to be moved into that folder. In this case, you can specify a list of files and they will all be moved in the folder path identified by the last parameter:

```
touch pear
touch apple
mkdir fruits
mv pear apple fruits #pear and apple moved to the fr
```

cp

- You can copy a file using the cp command:

```
touch apple
cp apple another_apple
```

- To copy folders you need to add the -r option to recursively copy the whole folder contents:

```
mkdir fruits
cp -r fruits cars
```

open

The open command lets you open a file using this syntax:

```
open <filename>
```

You can also open a directory, which on macOS opens the Finder app with the current directory open:

```
open <directory name>
```

I use it all the time to open the current directory:

```
open .
```

The special `.` symbol points to the current directory, as `..` points to the parent directory
The same command can also be be used to run an application:

```
open <application name>
```

gzip

- You can compress a file using the gzip compression protocol named LZ77 using the gzip command.

  ```
  gzip filename
  ```

- This will compress the file, and append a .gz extension to it. The original file is deleted.

- To prevent this, you can use the -c option and use output redirection to write the output to the filename.gz file:
  ```
  gzip -c filename > filename.gz
  ```

- The `-c` option specifies that output will go to the standard output stream, leaving the original file intact

- `gzip` can also be used to decompress a file, using the `-d` option:
  ```
  gzip -d filename.gz
  ```

gunzip

- The gunzip command is basically equivalent to the gzip command, except the -d option is always enabled by default.
- The command can be invoked in this way:

```
gunzip filename.gz
```

- This will gunzip and will remove the .gz extension, putting the result in the filename file. If that file exists, it will overwrite that.
- You can extract to a different filename using output redirection using the -c option:

```
gunzip -c filename.gz > anotherfilename
```

tail

- Tail command writes the file specified by the file parameter beginning with a specified point.
- The best use case of tail in my opinion is when called with the `-f` option. It opens the file at the end, and watches for file changes. Any time there is new content in the file, it is printed in the window.

- This is great for watching log files, for example:
  `tail -f log-file`

- To exit, press `ctrl-C` .

- You can print the last 10 lines in a file:
  `tail -n 10 <filename>`

- You can print the whole file content starting from a specific line using + before the line number:
  `tail -n +10 <filename>`

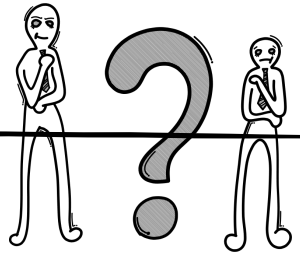
Exercise

Objective:
Get familiar with mkdir, rmdir, touch, mv, cp.

Hint:
Use only the above mentioned commands

Your task:

- Using terminal, in the home directory create one new folder called fruits
- Using terminal, in the home directory create three new folders with one command
- Using terminal, in the home directory create nested folders vegetable/salad
- Using terminal, delete the fruits folder
- Using terminal, in the home directory create 2 new empty files, a new folder and move the empty files within the new folder
- Using terminal, in the home directory create a new empty file and copy it into another file



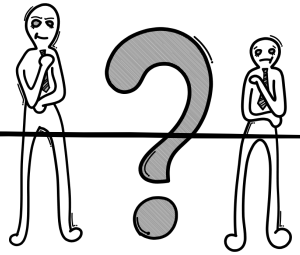
Exercise

Objective:
Get familiar with open, gzip, gunzip, tail.

Hint:
To perform all the tasks you need to use more commands that you learned earlier.

Your task:

- Using terminal, in the home directory create one file called doc
- Using terminal, open the doc file
- Using terminal, in the home directory create another file called test, compress the test file, leaving the original file intact
- Using terminal, decompress the test file
- Using terminal, in the home directory create one file called python, compress it and then extract it to a different file using output redirection
- Using file explorer, create a file called names, and write 10 names on it. Show content of this file using the terminal.



cat

- Similar to tail in some way, we have cat . Except cat can also adds content to a file, and this makes it super powerful.
- In its simplest usage, cat prints a file's content to the standard output:

  ```
  cat file
  ```

- You can print the content of multiple files:

  ```
  cat file1 file2
  ```

- and using the output redirection operator `>` you can concatenate the content of multiple files into a new file:

  ```
  cat file1 file2 > file3
  ```

cat

- Using `>>` you can append the content of multiple files into a new file, creating it if it does not exist:

  ```
  cat file1 file2 >> file3
  ```

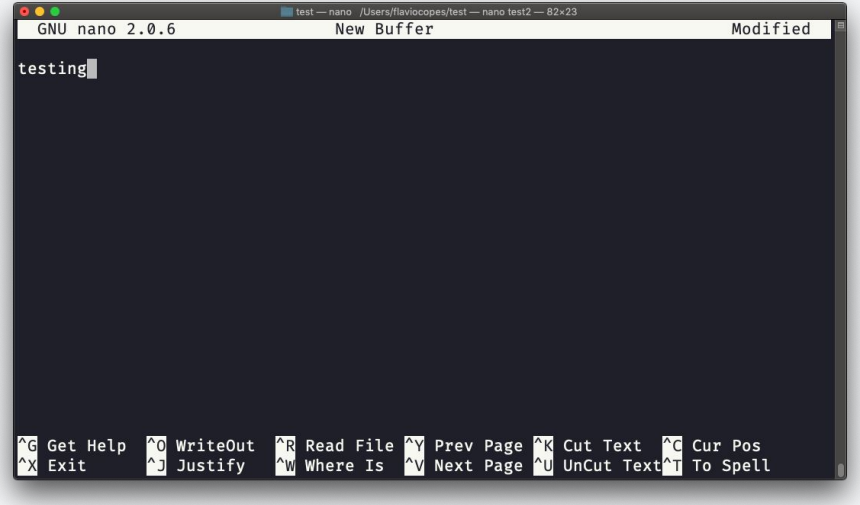
- When watching source code files it's great to see the line numbers, and you can have cat print them using the -n option:

  ```
  cat -n file1
  ```

- You can also remove all the multiple empty lines using `-s` .

nano

- nano is a beginner friendly editor.
- Run it using `nano <filename>` .
- You can directly type characters into the file without worrying about modes.
- ctrl-O saves the content. You can quit without editing using ctrl-X .
- If you edited the file buffer, the editor will ask you for confirmation and you can save the edits, or discard them.

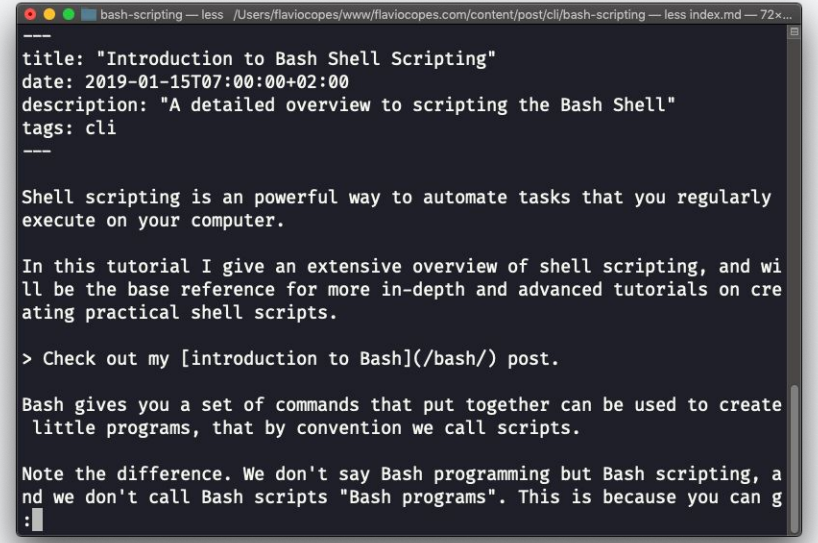


less

- The less command is one I use a lot. It shows you the content stored inside a file, in a nice and interactive UI.

```
less <filename>
```

- Once you are inside a less session, you can quit by pressing q .



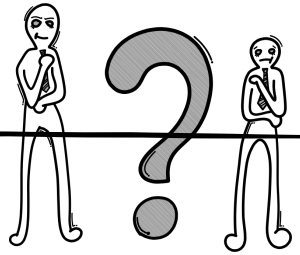
Exercise

Objective:
Get familiar with cat, nano, less

Hint:
Only use the above mentioned commands.

Your task:

- Using file explorer, create three files called file1, file2 and file3, write 5 lines of text in each one.
- Using terminal, print the content of the three files with one command
- Using terminal, concentrate the content of file1 and file2 in file3
- Using terminal, add the line numbers to file1
- Using terminal, add more lines of text in file1
- Using terminal, show the content stored inside file1



At the end of this lesson, you should be able to manipulate files from the terminal using the commands:

- man
- pwd
- cd
- ls
- ls - al
- mkdir
- rmdir
- mv
- cp
- touch
- gzip
- gunzip
- tail
- cat
- less

Digital Career Institute

Thank you!

Have a great day ahead!