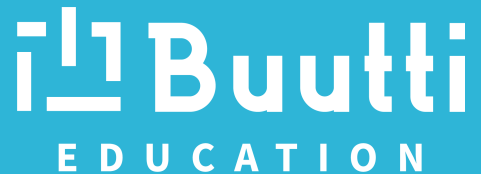


# Lecture 1

## Using a Computer



# Using Windows

# Windows 10/11

- Before jumping into the world of programming, we need to get acquainted with the tools we're using
- Namely, the Windows PC!
- To get everyone on the same page, let's go through some of the basics of navigating the Windows 10/11 operating system

# Start menu

- The most important UI element of a Windows PC is the start menu
  - You can pin applications to it as “live tiles”.
- Access it quickly with the **WINDOWS**, aka. **WIN** key.
- Then, start typing to search for files, installed applications, settings, etc.
- **Note:** Sometimes the search doesn't work! It's an “intelligent” search that tries to adapt to its users
  - search results may vary over time

# Applications

- The most important application we're using are:
  - **File Explorer**, for accessing the files inside the computer
  - **VS Code**, for creating & editing code
  - **Chrome**, and other browsers, for browsing the internet and running our web applications

# Administrator rights

- There are two ways to execute applications in Windows, as a “standard user” or as “administrator”
  - Administrator, or admin, can access some files standard users can't
- In Start Menu, right click on an application and select *More > Run as administrator*

# Applications

- In Windows, applications are installed into many different locations
  - C:\<user>\Appdata (Hidden folder!)
    - Apps installed for specific users
  - C:\Program files (64-bit applications)
  - C:\Program files (x86) (32-bit applications)
  - C:\ProgramData
    - (Stuff that can't have space on the directory path)

# App-specific keyboard shortcuts

- Apps can have their own shortcuts, but there are some standards most apps follow.
  - Check from the app's settings how shortcuts work
- Copy, Paste, Cut: **CTRL+C**, **CTRL+V**, **CTRL+X**
- Undo: **CTRL+Z**
- Redo: **CTRL+Y** or **CTRL+SHIFT+Z**
- Save: **CTRL+S**
- Save as: **CTRL+SHIFT+S** or **F12**
- Open: **CTRL+O** often



# Windows keyboard shortcuts

- **WIN+arrows**
  - Move apps to different positions on screen
- **ALT+TAB**
  - Toggle between open apps
- **ALT+F4**
  - Close app
- **Note about FnLock:**
  - Press **Fn+ESC** to change behavior of Fn (F1, F2, ...) keys.

# Task Manager

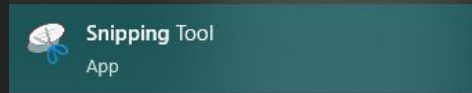
- Open
  - quickly with **CTRL+SHIFT+ESC**
  - slowly by right-clicking the Start menu button
- Use for:
  - Force closing an unresponsive application
    - *Right click > End task*
  - Seeing what is taking up resources

# Settings

- There are two places to look for settings in Windows 10
  - The new, modern *Settings app*
    - Bluetooth, Display settings...
  - The legacy *Control panel*
    - Programs & Features
    - Power options
    - Credential Manager
    - Sound Options
    - Some *hardware drivers* add their own settings here

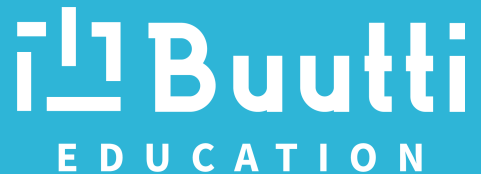
# Taking screenshots

- **Printscreen** key copies the whole screen into clipboard
- **WIN+SHIFT+S** lets you to **drag** a screenshot of a specific area
  - Neat, and just a little bit slower!
- you can then paste the copied screenshot into Paint or whatever™
- Alternatively, on Windows you can use Snipping Tool  
(Press WIN key and type "snipping tool")



# Assignment 1.1: Send a Screenshot

- Take a screenshot of something in your screen. Do not take a screenshot of the whole screen.
- Send the screenshot to the **#general** channel in Discord.

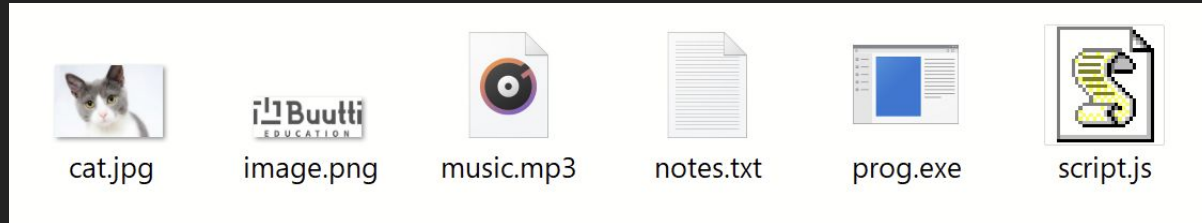


# File System

# Files and file system

*Files* are objects on a computer or other system. They might be pictures, videos, audio files, documents, code files, executable programs, or anything. *Everything* permanent in a computer is stored as a file.

A *file system* stores and manages *files* on a computer.



# File systems and mobile platforms

Many modern platforms, such as smartphones and tablets, typically hide the file system from the user because it is considered too complicated for the regular user. Instead of a file browser, you have, for example, a picture browsing application.

Even these platforms rely on traditional file systems, but they just don't expose their file system to the user.

Due to this, the file system is an unknown concept to many people, despite being a core part of operating systems.



# File system on desktop platforms

Traditional desktop platforms like Linux, Windows and macOS instead allow you to freely navigate and manipulate the file system.

In programming, it is necessary to understand how to utilize file systems. Programs often consist of a number of code files structured in different directories. Simple programs can do with only one code file, but more complex software has hundreds or even thousands of code files.

# Folders

A typical computer nowadays has hundreds of thousands of files. Even if you hadn't created that many, just the operating system and a few installed applications might be close to a million files.

If all these files were all in the same place, it would be very inconvenient for us to browse them or find a specific one!

Like with physical documents in real life, there's a solution: organizing the files into *folders* (also known as *directories*; although they technically have a different meaning, they are usually used interchangeably).

# Folders

A *folder* is a container for files. It can contain zero or more files, up to a practically unlimited number.

Folders can also hold other folders. These are called *subfolders* within the *parent folder*.

One folder can contain both files and subfolders.

Files are identified by their file names: Two files or subfolders within the same folder cannot share the same name!

# Folder structure example

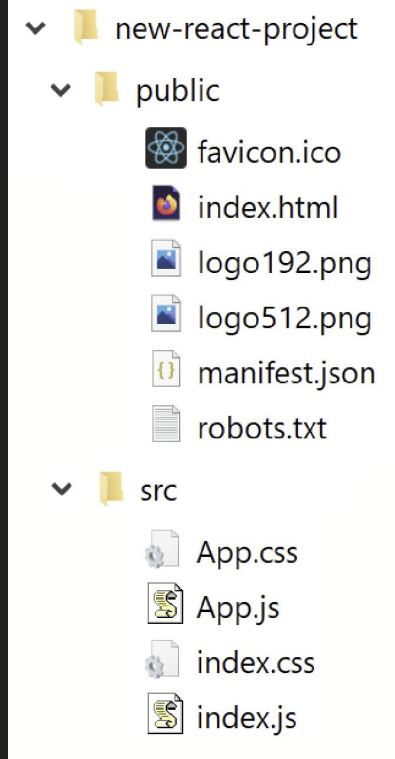
On the right you see a part of the folder structure of a new React project (you'll get familiar with it later on the course).

There is a `new-react-project` folder that has two subfolders: `public` and `src`.

The `public` subfolder has some image files (`.png` and `.ico`), a HTML file, a `.json` file and `robots.txt`.

The `src` subfolder has some CSS and JS files.

Don't get scared if you don't know all of the file types here - the point is to learn what a folder structure looks like.



# Folder structure example

Here is an alternative visualization of the folder structure on the previous slide.

This was generated with the `tree /f` command in Windows PowerShell.

```
new-react-project
├── public
│   ├── favicon.ico
│   ├── index.html
│   ├── logo192.png
│   ├── logo512.png
│   ├── manifest.json
│   └── robots.txt
└── src
    ├── App.css
    ├── App.js
    ├── index.css
    └── index.js
```

# File paths

To fetch a specific file from the file system, a *file path* is required.

Imagine that you have a `logo.png` image file in 5 different directories on your computer and you wish to find a specific one of them. The system couldn't give you that specific one if you told it to just fetch a file called `logo.png`. You have to specify which folder the one you're looking for belongs to.

# File paths

A file path can specify the exact location of the file for us. In this case, it might look something like `C:\MyCoolImages\logo.png`.

The backslash `\` here denotes a directory change. In this case,

- 1) `C:` would be our drive letter or *filesystem root*,
- 2) `MyCoolImages` would be a folder in the drive
- 3) `logo.png` would be a file in the `MyCoolImages` folder

# Absolute and relative paths

A file path can be either *absolute* or *relative*.

An *absolute path* starts from the root of the file system - on Windows usually from the drive letter (for example, C:) - and tells the absolute location of the file in our file system.

A *relative path* is given relative to some existing drive or folder.



# Absolute and relative paths: example

Imagine we want to fetch the `index.html` file from our React project folder structure shown earlier. It resides in a folder named `public`, which is inside our project folder called `new-react-project`.

If our project folder was on the `C:` drive, the *absolute path* to the file would be `C:\new-react-project\public\index.html`, where, again, `\` denotes a folder change.

On the other hand, a *relative path* to the file **from our project folder** would be `public\index.html`. A relative path **from the drive root** would be `new-react-project\public\index.html`.

# File path example

If we had our new React project folder on the C: drive, the file system tree would look like on the right.

(Don't run `tree /f` directly from the root of `C:` - it'll list all the hundreds of thousands of files on the whole drive.)

```
C:.\n├── new-react-project\n│   ├── public\n│   │   ├── favicon.ico\n│   │   ├── index.html\n│   │   ├── logo192.png\n│   │   ├── logo512.png\n│   │   ├── manifest.json\n│   │   └── robots.txt\n│   └── src\n│       ├── App.css\n│       ├── App.js\n│       ├── index.css\n│       └── index.js
```

# Dots (.) in file path

A lone dot (.) in a file path refers to the current folder. For example,

`C:\MyCoolImages\.\logo.png`

is the same as

`C:\MyCoolImages\logo.png`

Two lone dots (..) refer to the folder above. For example,

`C:\MyCoolImages\..\logo.png`

is the same as

`C:\logo.png`

This will be useful later on when we learn to use the command line.

# Platform-specific differences in file paths

There are some differences in file paths between operating systems.

On Windows, *absolute* paths start with the drive letter, which is usually `C :`, but can also be something else, especially if your computer has multiple drives. On Linux and Mac, this is not the case.

On Windows, the folder separator character is typically a backslash `\`. On Linux and Mac, it is a forward slash `/`.

Luckily, modern Windows versions also accept `/` as a folder separator, which makes programmers' life easier.

# File extensions

To make it easier to identify the type of a file, file names are usually split into two parts: a title, and an extension. The title is just an arbitrary name identifying the file, while the extension usually tells the type of the file.

For example, the `.jpg` in `cat.jpg` tells us that it's a JPG image file. Likewise, the `.js` in `script.js` tells us that it's (likely) a JavaScript file.



cat.jpg



image.png



music.mp3



notes.txt



prog.exe



script.js

# File extensions

File extensions are also considered a slightly advanced topic, which is why Windows **hides** file extensions by default. We'll see how to fix this later.

Note that while file extensions are helpful, they are just a convention. **There is nothing that prevents forging file extensions!** You could, for example, download a JPG image from the Internet and change its file extension to `.mp3` to trick it into looking like a music file. This doesn't change the actual contents or type of the file, iow. it still remains an image, but only makes it appear like a file of a different type to an unsuspecting user.

Or, the file extension can also be removed entirely.

# Case sensitivity

Case sensitivity means if the operating system makes any difference in UPPERCASE and lowercase letters in file and folder names.

Historically Windows has not been very case-sensitive. However unix-based systems are.

In Windows you **CAN NOT** have `hello.js` and `Hello.js` in the same folder because they are considered the same.

In unix-based systems you **CAN** have `hello.js` and `Hello.js` in the same folder since they are considered different files!

**Pay attention to case-sensitivity even if you are developing on Windows, since most programming languages are case-sensitive and your code may very likely run on some unix-based system later!**

# Characters to avoid in file names

In programming you should mostly stick to letters and numbers when naming files and folders. Things may break when moving from one operating system to another. Even Windows has some problems with some characters in file names.

Even spaces might cause problems. A good way to avoid problems with spaces is to use dash instead of a space.

`"my awesome program.js"`  `"my-awesome-program.js"`

Also avoid all special characters like `&`, `%`, `#` etc. whenever possible.



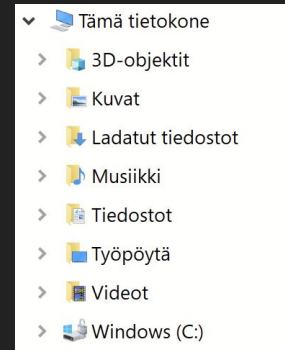
# File Explorer

You can access the file system in multiple ways, but the standard way in Windows is the **File Explorer** application (**Resurssienhallinta** in Finnish).



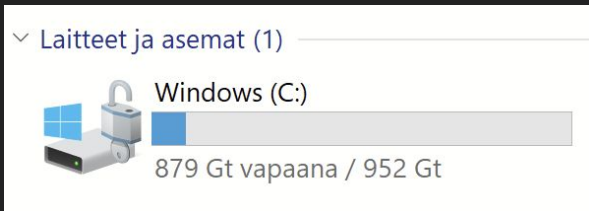
You can open it either by clicking this icon on the taskbar, or by opening Windows Search and typing *file explorer* there.

At first, it shows a view with some pinned folders and “recently used” files. From the left sidebar we can access these same pinned folders, as well as the raw file system by opening *This PC* (Tämä tietokone).



# Drives in File Explorer

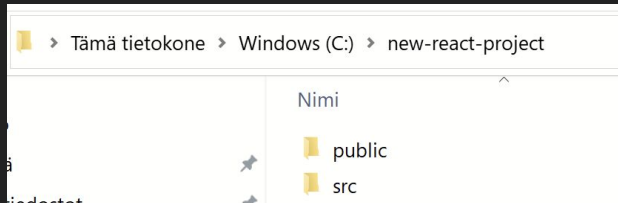
In This PC, you can - once again - see the familiar pinned folders at the top, but you can also access the drives from below them. Opening one of the drives by double-clicking allows us to view the files and folders inside the drive.



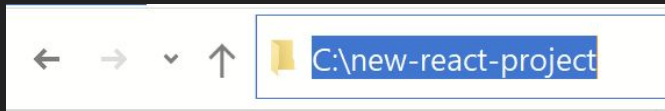
Many computers only have one drive (usually C:), but some have multiple.

# Breadcrumb navigator

Browse some folders inside your C: drive by double-clicking on them. At the top you can see a breadcrumb navigator that displays where you currently are.



By clicking on the breadcrumb navigator, you can view the absolute path to the folder you currently are in.



The buttons to the left of the path allow you to go back to the previous folder, to the “next” folder, open a recent path or go up a directory, respectively.

# Pinned folders

While Windows gives the different pinned folders a special appearance, they are technically also just regular folders in the file system and can be found by browsing the drive. These folder paths are customizable, but by default they point to the following locations:

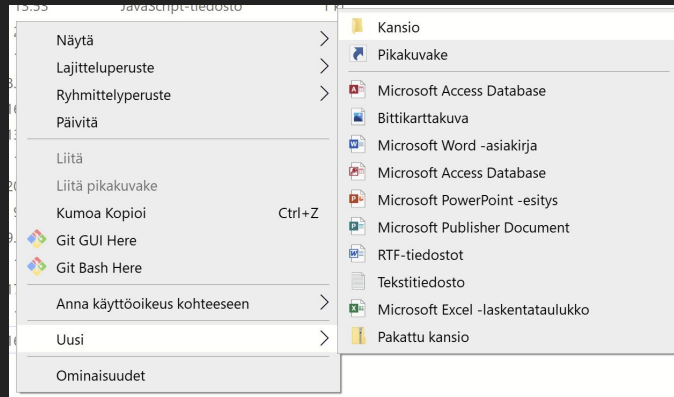
- Documents (Tiedostot) `C:\Users\<username>\Documents`
- Downloads (Ladatut tiedostot) `C:\Users\<username>\Downloads`
- Pictures (Kuvat) `C:\Users\<username>\Pictures`

...and so on, where `<username>` is a special folder Windows has created for your user in the PC. Try to find these folders by browsing to them from the C: drive!

# Creating new folders

You can create a new folder in File Explorer with the following steps:

- 1) browse to the drive or folder where you want to create the new folder
- 2) right-click **on empty space** (not on a file!) to open a context menu
- 3) navigate to New -> Folder and click on it



Windows will prompt you to name your new folder.

# Creating new files

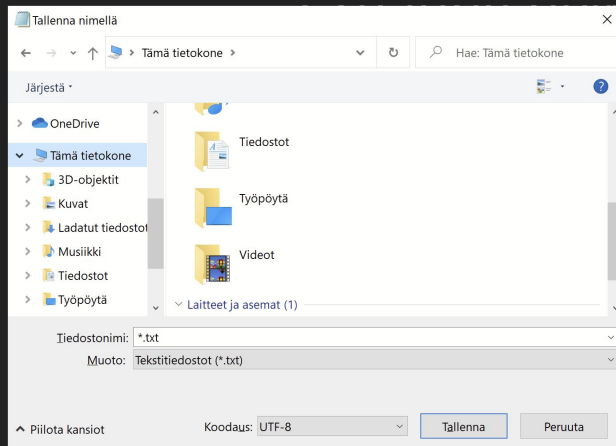
For the most part, File Explorer cannot be used to create new files, but it can be handy for creating text documents. This also works through the same right-click context menu as creating folders. For example, creating a new text document happens from *New -> Text Document*.

Instead, if you for example want to create a new image, you have to create it by using image editing software.

# Creating new files

The majority of software on Windows, like image and text editors, use the File Explorer or a similar interface that allows you to browse the file system and select the location where to save the file when you use them to create a new file and save it.

If you can manage File Explorer, you also know how to use these dialogs.

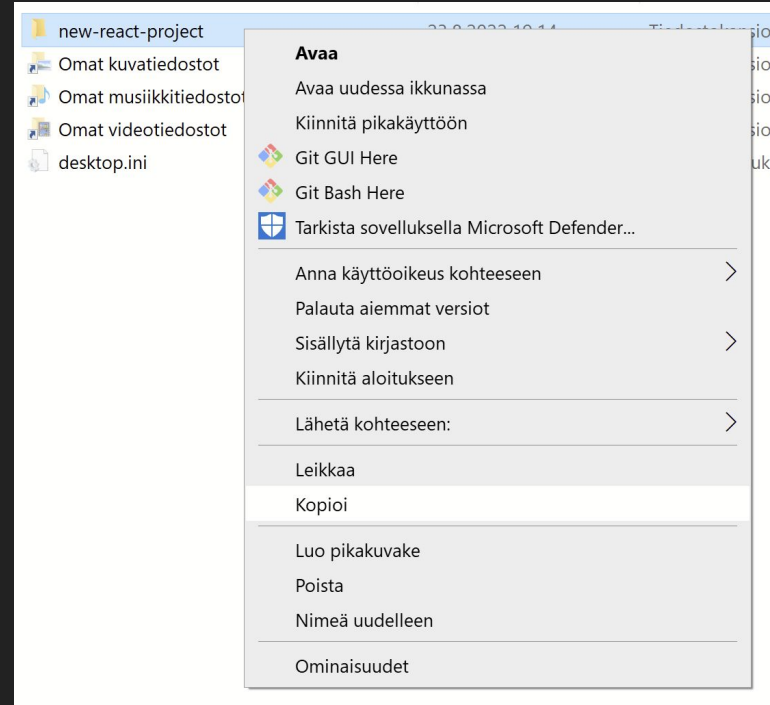


# Copying files and folders

File Explorer also allows you to copy files and folders through the same context menu.

Simply right-click on the file or folder and click “Copy” from the context menu. Afterwards, browse to the folder where you want to place the copy to, right-click on empty space while in that folder and click on “Paste”.

Cutting (Leikkaa) is used in the same way, but moves the file instead of copying it. In other words, cutting “removes” the original file and leaves only the copy.

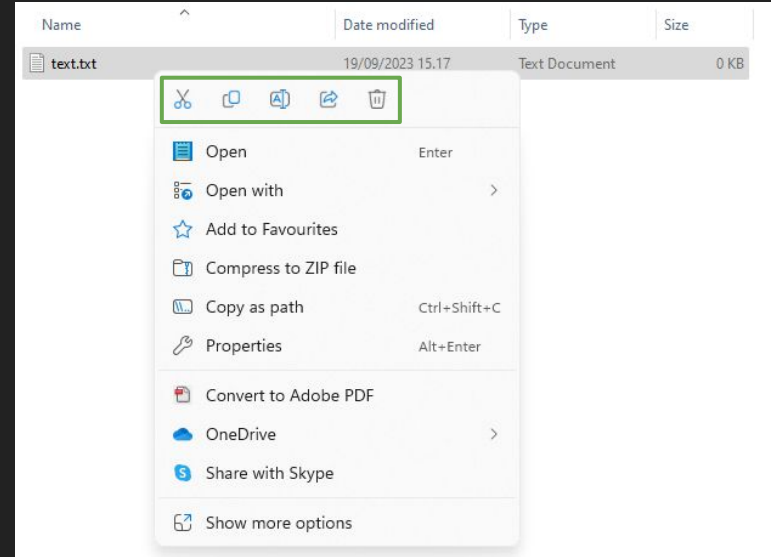




# Copying files and folders

In Windows 11 the copy, cut and rename functions are in the icons on the top row of the context menu.

You can also select “Show more options” to get the old windows 10 style context menu.



# Shortcuts

For copying files and folders, there's also the shortcuts *Ctrl*+*C* and *Ctrl*+*V* that are faster than using the context menu.

To copy files and folders, you can select one or many (by dragging with the mouse or by holding *Ctrl*/*Shift* while clicking the files), then hold *Ctrl* and press *C* while still holding *Ctrl*.

To paste the copied files, you can press *V* while holding *Ctrl*.

These shortcuts for copying and pasting are ubiquitous, regardless of program or domain (f. ex., copying text in text documents works the same).

# On permissions

Note that by default, you can't create new files and folders under certain protected directories, such as the Windows directory itself.

This is to prevent you from accidentally messing up your Windows installation or the installation of some programs.

# Assignment 1.2:

## File Explorer practice

Let's practice creating some directories and files.

- 1) Create a new directory named `lecture-1` under the C: drive.
- 2) Create a sub-directory named `assignment-1.2`
- 3) Inside that directory, create a new text document named `hello.txt`
- 4) Inside that directory, create a sub-directory and name it `images`
- 5) Find some nice image on the Internet and save it to the `images` directory
- 6) Copy `hello.txt` to the `images` directory.

# Displaying file extensions

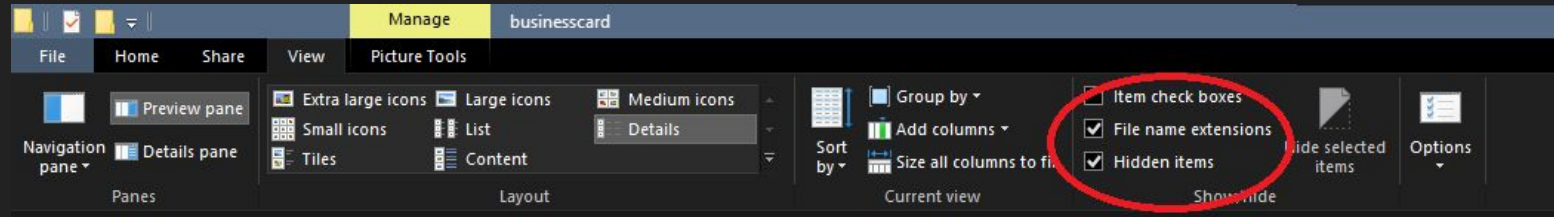
As we mentioned earlier, Windows hides file extensions by default. As programmers we really want to see them.

Instructions on enabling this differ by Windows version, so it's easiest to link to a guide, even if guide sites like this are often of questionable quality.

<https://www.howtogeek.com/205086/beginner-how-to-make-windows-show-file-extensions/>

# File extensions & hidden files

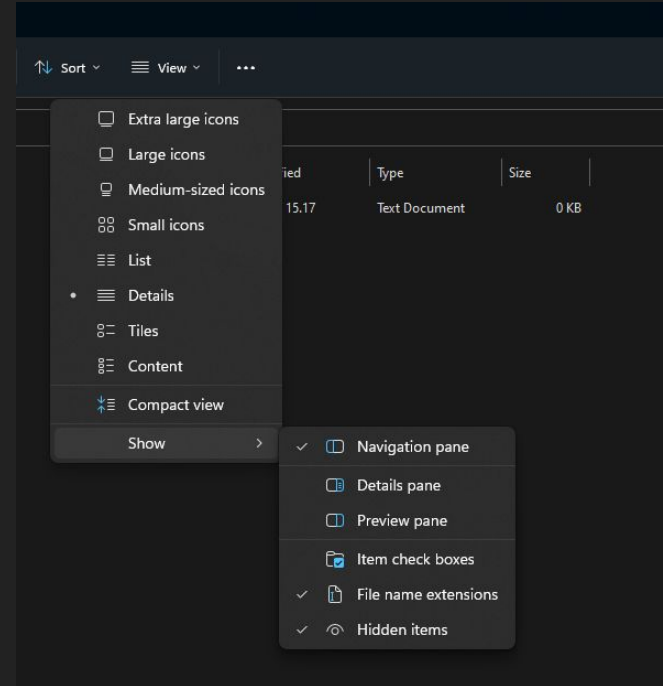
- Some files and folders are **hidden**.
- You can hide files and folders yourself as well
- Show hidden items by going to the View tab of the File explorer:



- Also: Windows hides file extensions by default
  - We REALLY want to see them, so this is where we can show them as well.

# File extensions & hidden files

Show/hide file extensions and hidden files in Windows 11



# File Explorer - advanced

There are some more advanced techniques for using File Explorer.

For example, you can open multiple File Explorer windows, browse them into different folders and then move files between folders by dragging the files from one window to the other.

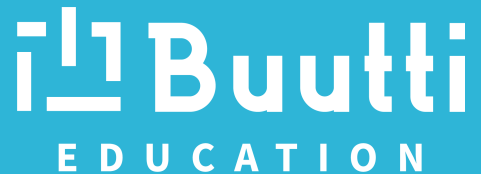
The desktop is also just a graphically fancy folder - if you right-click on your desktop, you'll see the same familiar options we're just learned. Copy/paste, drag-and-drop, creating subfolders etc. works identically on the desktop as on other folders.

Due to resource constraints we won't go over all the more advanced stuff that File Explorer has. You'll likely learn some during the course.



# Assignment 1.3: Hidden items and file extensions

**Make sure that you have configured your windows explorer so that it will display hidden items and file extensions.**



# Command line

# Command line

The command line is a text-based, less graphical way of using a computer and its file system compared to File Explorer or other functions of the operating system's graphical user interface.

It is typically regarded as less friendly to new users, but powerful and efficient once you've learned it.

In programming, using the command line is often more efficient than using graphical tools for the same job. In many special cases, it is even mandatory as no graphical version of a tool might exist.

# Shell

The command line is also often called the ***shell***. Some other alternative, often-used names are ***terminal*** and ***console***.

The graphical user interface of the operating system is often called the ***graphical shell*** to differentiate.

A *shell* is a program that, in simple terms, makes it possible to use the computer and access its resources. A command-line shell does this through text-based commands, while the graphical shell does this with, for example, clickable or tappable buttons or other elements.

# Different command-line shells

There are multiple different command-line shells depending on the operating system.

Windows comes with two shells: the newer *PowerShell*, and the older `cmd.exe` (*Command Prompt*). The older Command Prompt is derived from MS-DOS from the early 90s, so it is quite limited and antique.

The newer PowerShell is much more powerful and closer to shells of other operating systems, so using it is strongly preferred.

On Linux and Mac, the most common shell is *bash*, but there are variations (*tcsh*, *ksh*, *zsh* etc.) that usually behave similarly.

The latest Windows versions can also use *bash* through WSL (Windows Subsystem for Linux).

# Using the command line

The command line shell is started like any other program. On Windows, you can just type *powershell* to the search and launch it.

When started, the command line displays its current folder in the file system, usually the current user's home folder.

```
PS C:\Users\username>
```

From here, we can use text-based commands to navigate the file system, modify folders, launch programs etc.

# Command and parameters

When you enter a command, you have to provide the command's name and, optionally, one or more *parameters* that provide extra information to the command. Parameters are often also called *arguments*.

For example, in the following command

```
cd Pictures
```

The command identifier is `cd`, while `Pictures` is a parameter.

Later on the course, you will run into multi-parameter commands like

```
git branch -d my-custom-branch
```

# Navigating the file system

To change the directory you are in, use the `cd` command (change **d**irectory) with a parameter that tells it which directory to change into.

For some examples:

- 1) to change to a subfolder, type `cd foldername`
- 2) to go up a folder, type `cd ..`

To list all files in the directory, you can use the `ls` or `dir` commands.

Give them a try!



# Some useful PowerShell commands

Change directory: `cd directoryName`

List files in directory: `ls` or `dir` or `Get-ChildItem`

List files in directory as a filesystem tree: `tree`, recursively `tree /f`

Create directory: `mkdir directoryName`

Delete directory: `rmdir directoryName`

Create file: `New-Item fileName`

Set file content (text): `Set-Content fileName content`

Copy file: `cp sourcePath destinationPath`

Move file: `mv sourcePath destinationPath`

Delete file: `rm filePath`

Delete folder and all its contents: `rm -rf folderPath`

Execute program: `programName` (for example, `notepad`)

Execute program from our current directory: `./programName.exe`

File paths are always either absolute or relative to the current directory.

# Command line pro tips

- As programmers, we like to be **as lazy as possible**.
- If you feel like some mundane task takes awfully lot of effort, there's a good chance someone has created a faster way to accomplish it
- For example: **Tab completion**
  - You don't have to write commands (and especially file names) completely
  - The command line can guess what you mean from the first few letters
    - `cd P -> TAB -> cd Pictures`
  - If the guess is wrong, press **TAB** again to browse for other suggestions
    - Press **SHIFT+TAB** to browse in the opposite direction
- Another example: **Up and down arrows**
  - Press up & down to examine command history
  - You can run past commands easily this way.

# Assignment 1.4:

## Command-line practice

Using the command line, do the following:

- 1) In your previously created `lecture-1` folder, create a new subfolder `assignment-1.4`
- 2) Go to that folder
- 3) Print the contents of the directory (iow. list files in the directory)
- 4) Inside the folder, create a new file named `test.txt`
- 5) List the files of the directory again
- 6) Delete the `test.txt` file
- 7) Step outside of the directory
- 8) Delete the directory

EXTRA: Create a directory with files and figure out how to delete the directory while it still has files inside of it

# Opening PowerShell through File Explorer

You can hold Shift while right-clicking on empty space in a folder in File Explorer to display some additional context menu options intended for advanced users.

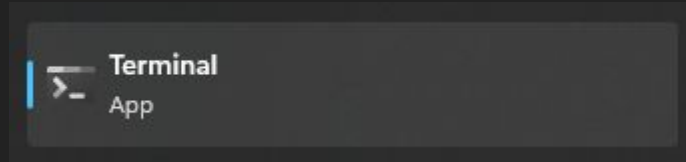
One of these options is very handy for programmers:  
*Open PowerShell Here.*



# Windows Terminal

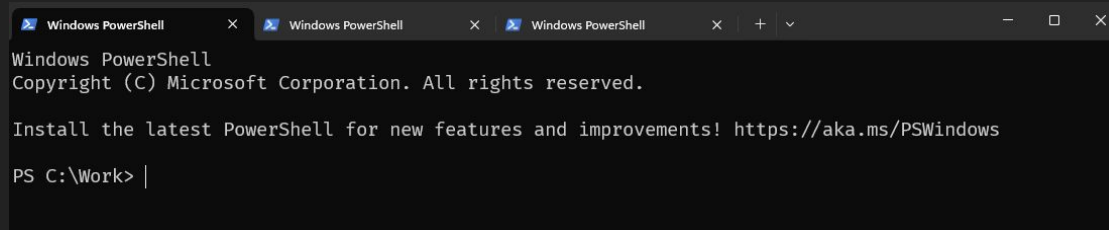
Windows Terminal is a program that comes with Windows. It can be used to run any shell you wish. It works with cmd, PowerShell, Git Bash etc.

To run Windows Terminal just hit the **WIN** key and write terminal



# Windows Terminal

Windows Terminal very configurable. You can change the appearance of the terminal, add background image, etc. You can also have multiple shells in tabs in the same window! You might want to give it a shot.

A screenshot of the Windows Terminal application. The window has a dark theme and contains three tabs, each labeled 'Windows PowerShell'. The active tab shows the following text: 'Windows PowerShell', 'Copyright (C) Microsoft Corporation. All rights reserved.', 'Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>', and the prompt 'PS C:\Work> |' with a cursor.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Work> |
```

# Command line wrap-up

There are way too many shell commands and techniques for us to cover them all. If you're ever in doubt on how to do something with the shell, search for it on your favourite search engine.

It can be hard to memorize all the different commands, but you'll eventually memorize all the commonly used ones. The rest you can google.

Note that you should **always read through any larger block of shell code that you find on the Internet before executing it**, as shell code can be malicious!

# Command line wrap-up

The command line is a common way to use the computer. It is sometimes more efficient than the graphical user interface. For example, the `git` version control system is often most efficiently used from the command line.

Many programming tutorials on the Internet directly tell you what kind of commands you should run, so knowing how to use the command line is necessary for following those tutorials.

There are also many tools that can only be used from the command line. One example is a tool known as `create-react-app` which is used for creating new React applications; it has no graphical interface.