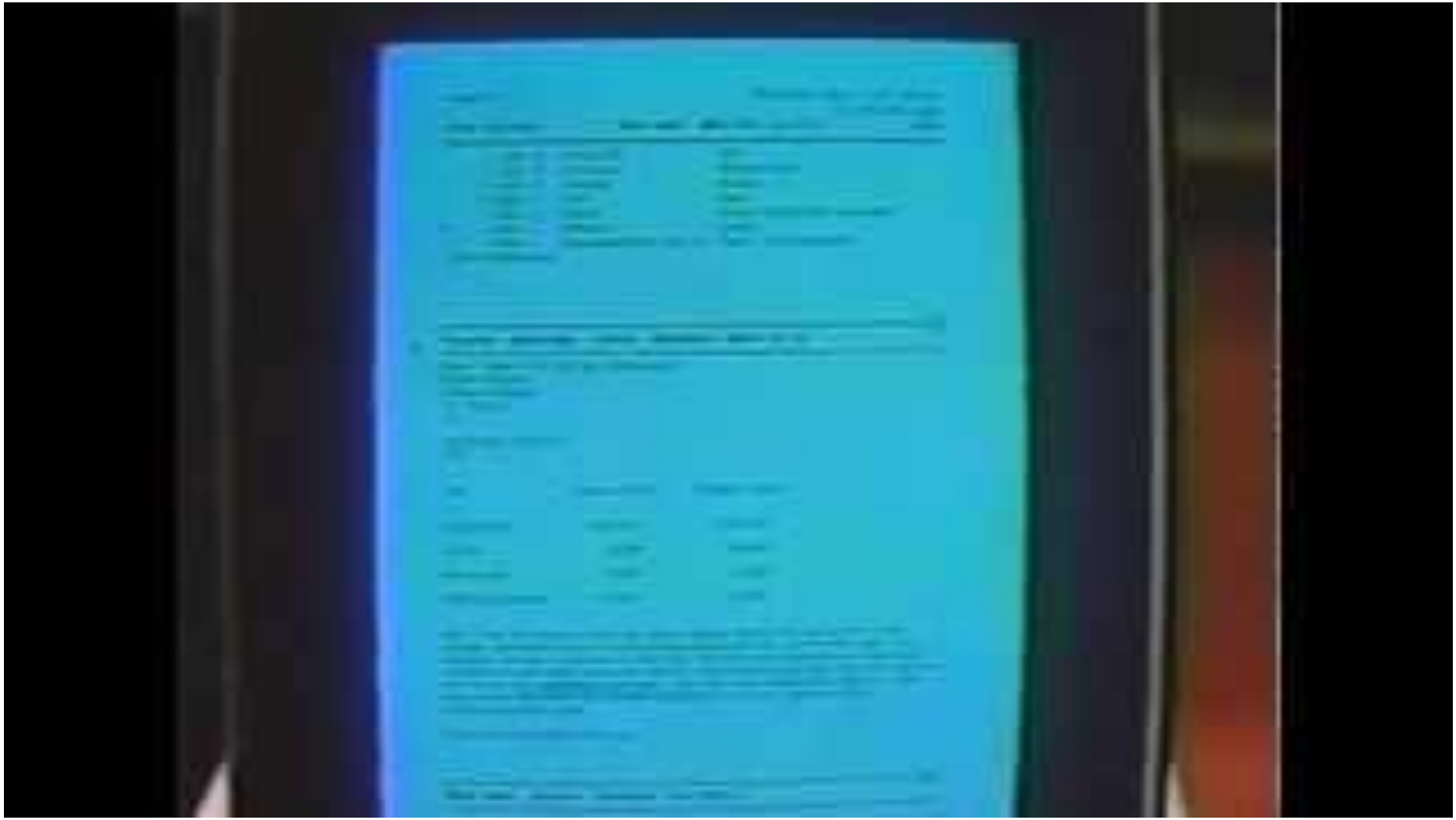
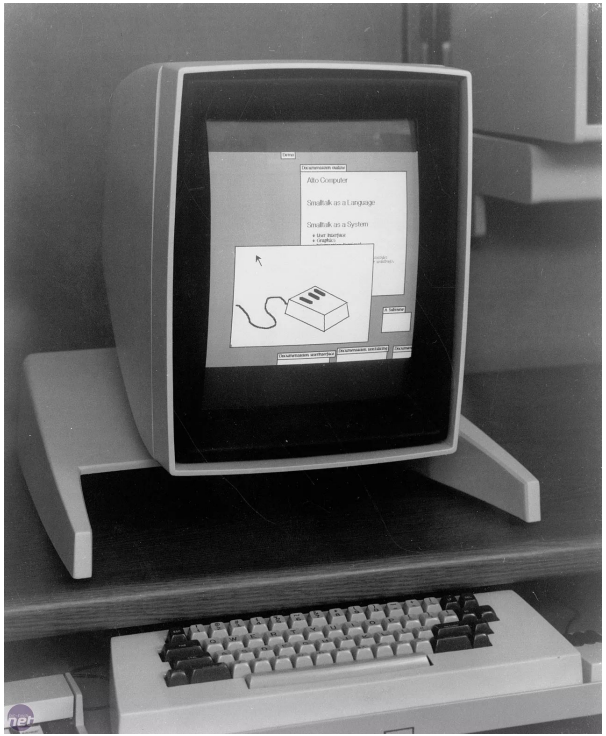
# Data Science Survival Skills

Graphical User Interfaces



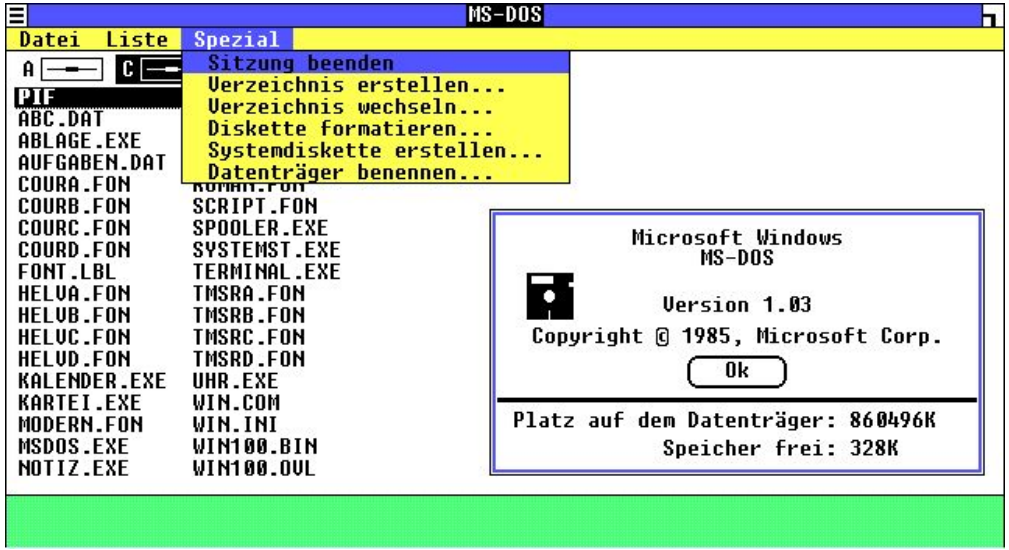
# History of GUIs

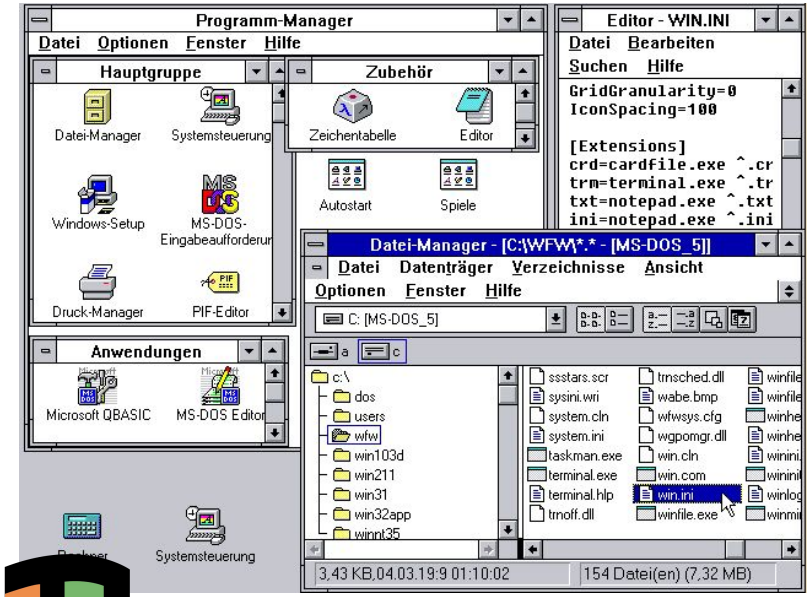



Xerox Alto (late 1970s)


Macintosh 128k (1984)
First PC with “modern” GUI

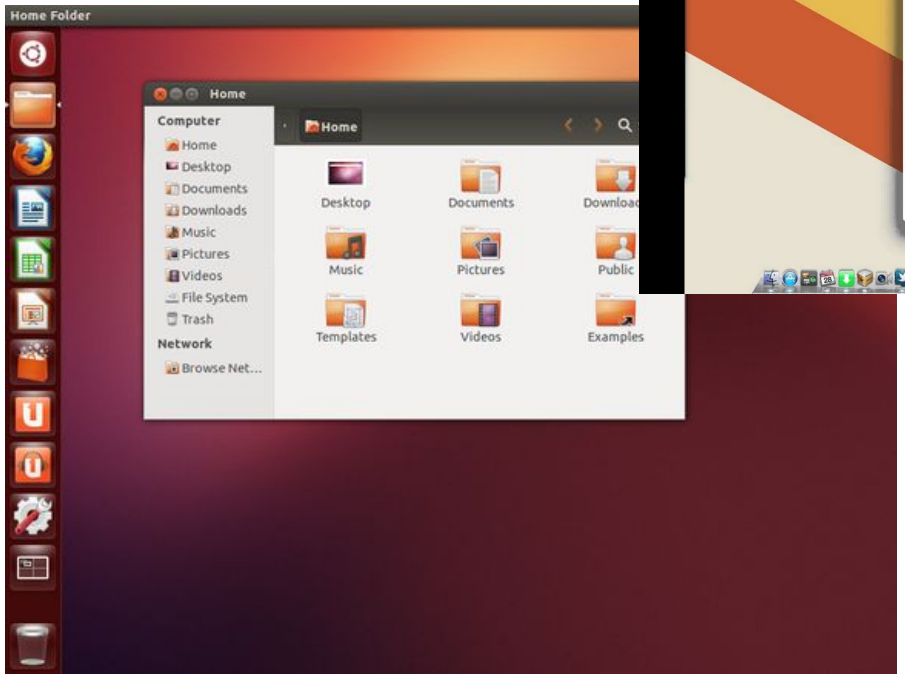
# Windows

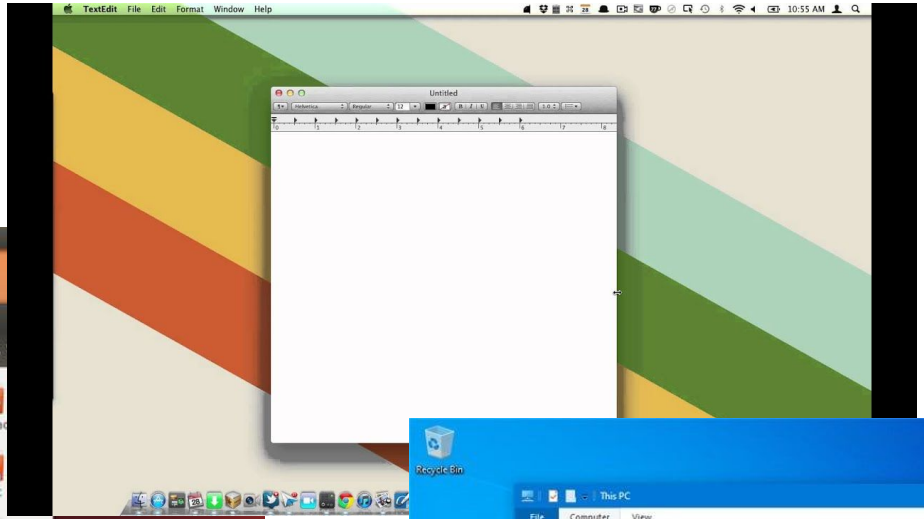


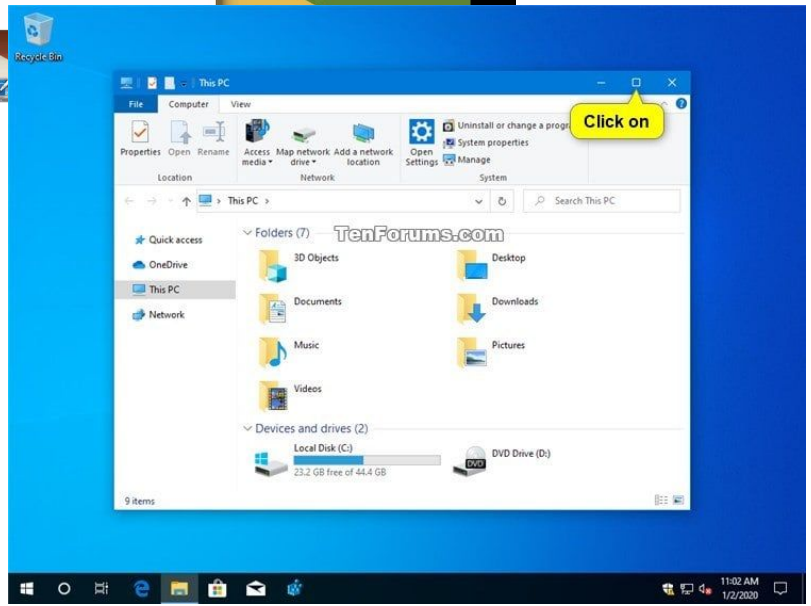




MICROSOFT® WINDOWS™

# Modern OS GUIs



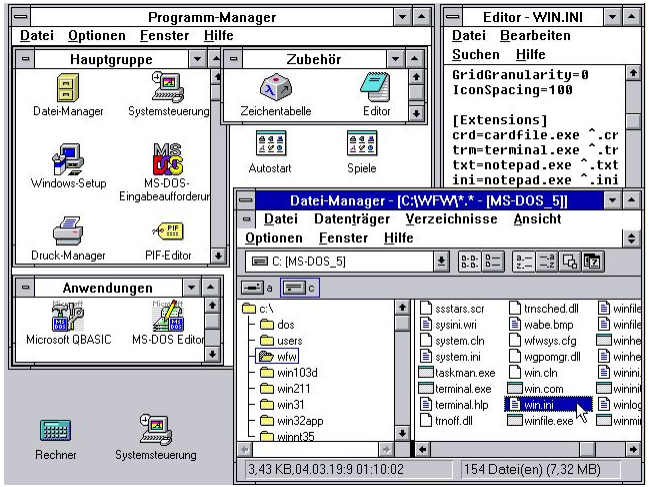
slido

# Which OS are you using most of the time?

ⓘ Click **Present with Slido** or install our Chrome extension to activate this poll while presenting.

# What are the key elements in a GUI vs. CLI?

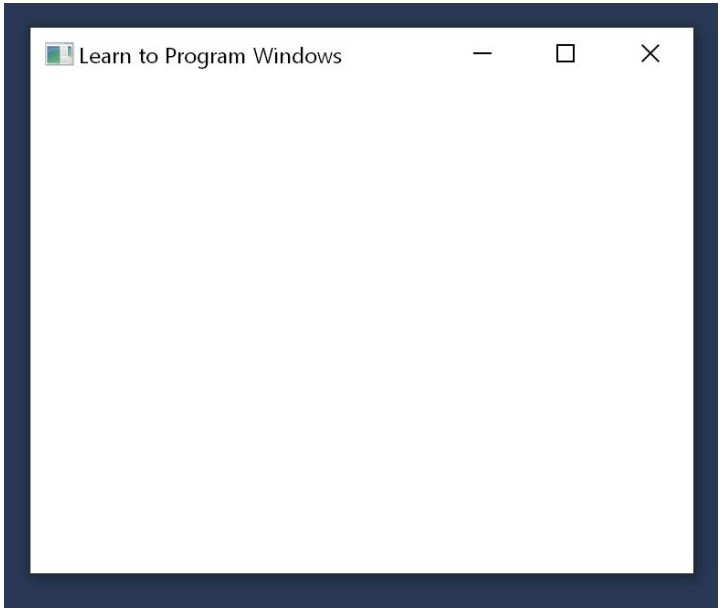
- **Windows**: Rectangular areas on the screen for displaying information and running programs.
- **Icons**: Graphic symbols representing programs, files, functions, etc.
- **Menus**: Lists of options or commands presented to the user.
- **Pointer**: A symbol (often a small arrow) that moves on the screen in response to user actions, used to select objects and text.



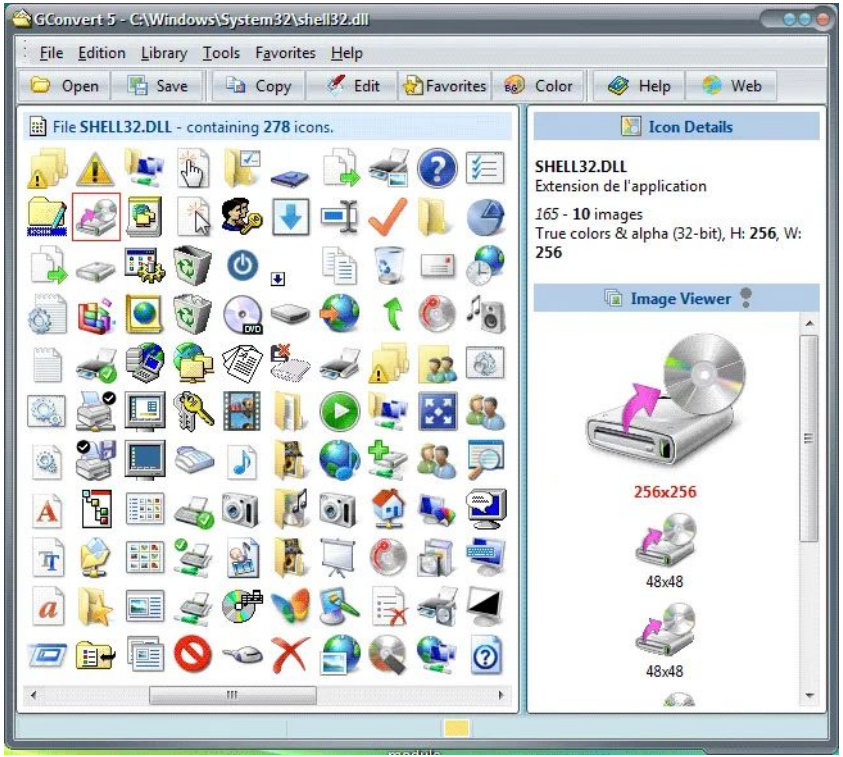
# Interacting with a window

Actions:

- Minimize
  (I don't want to interact for now)
- Maximize
  (I would like to focus)
- Close
  (I don't want to interact with this program anymore, free resources)



# Icons - OS/GUI



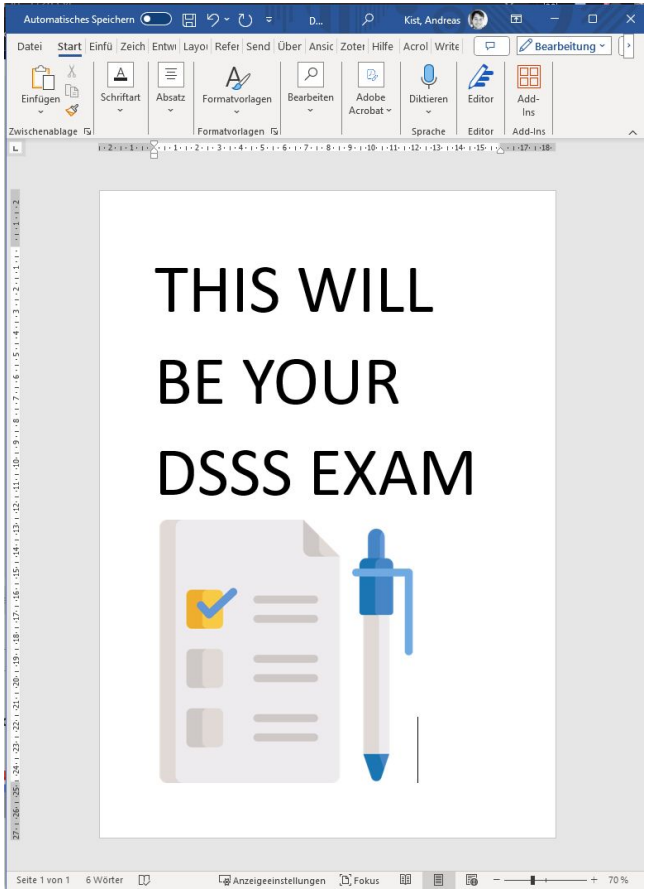

# Current action - Mouse

# Why GUI and not CLI?

- More user friendly with easy learning curve
- Do not remember commands
  (content of a folder is automatically shown, do not remember to write “ls” or “dir”)
- Symbols/Icons and written text navigate through the program

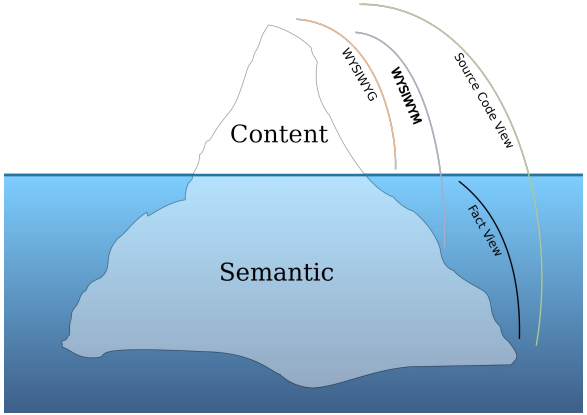
# WYSIWYG

What you see is what you get

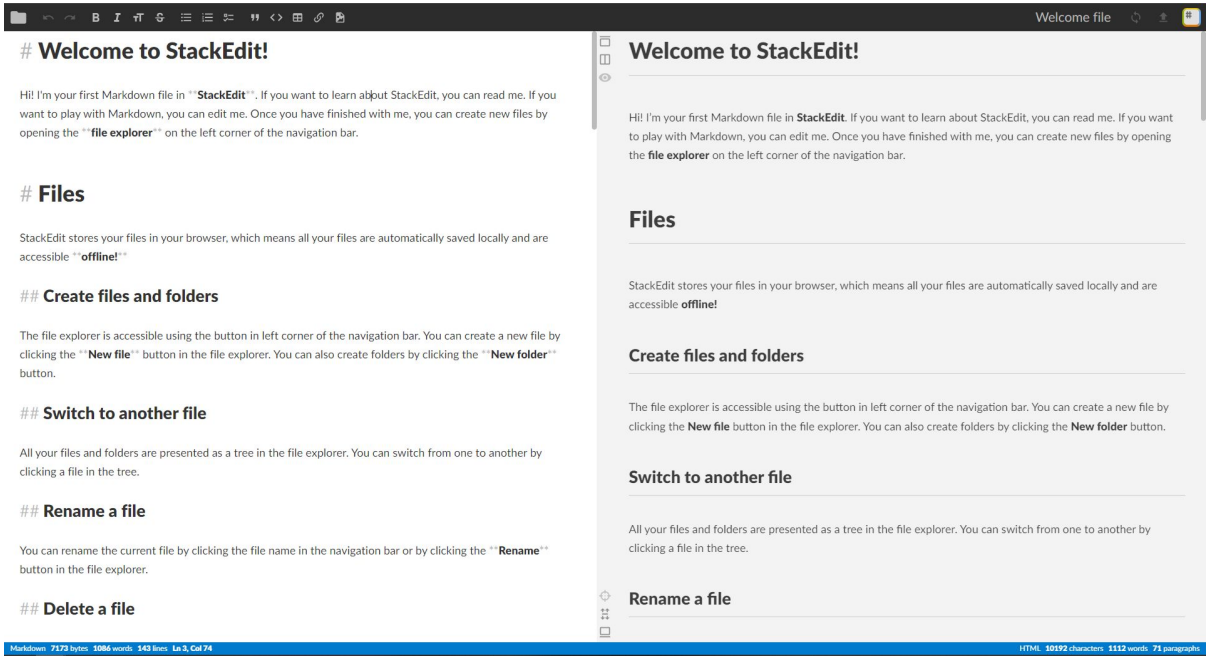


# WYSIWYM

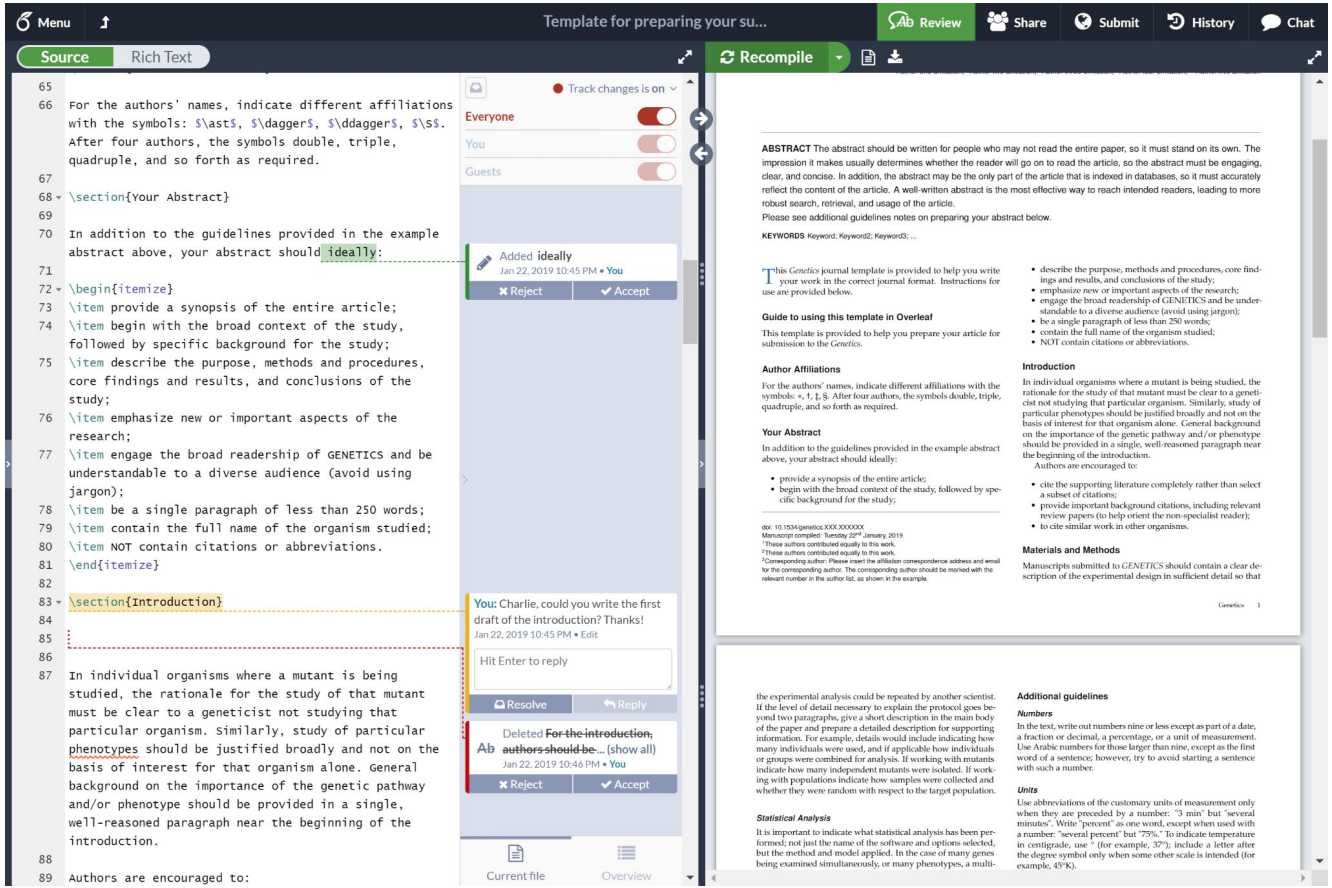
What you see is what you mean



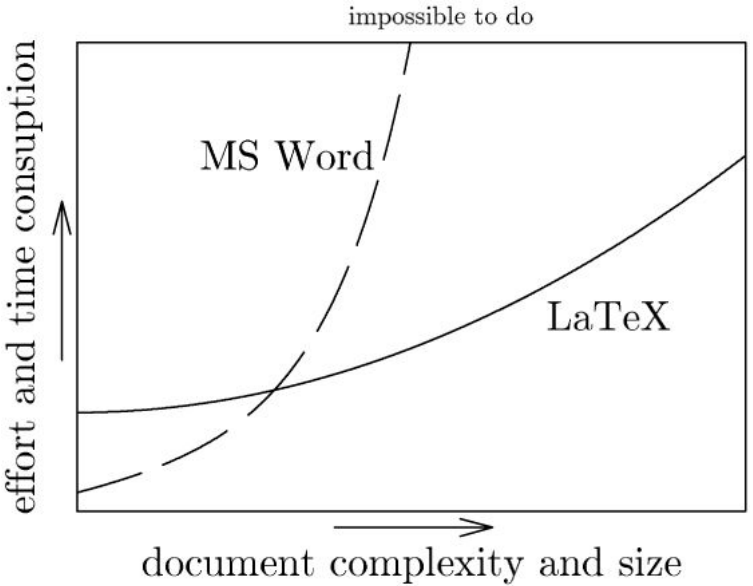

## E.g. Markdown editors

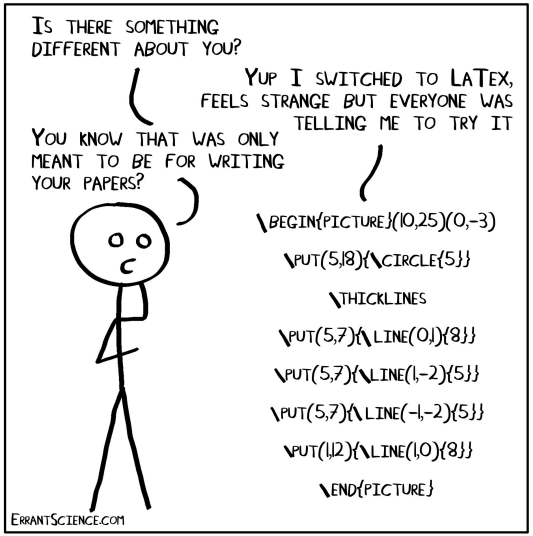


# LaTeX


Menu
Template for preparing your su...
Review
Share
Submit
History
Chat
Source
Rich Text
Recompile
Track changes is on
Everyone
You
Guests
For the authors' names, indicate different affiliations with the symbols: $\ast$, $\dagger$, $\ddagger$, $\S$. After four authors, the symbols double, triple, quadruple, and so forth as required.
\section{Your Abstract}
In addition to the guidelines provided in the example abstract above, your abstract should ideally:
\begin{itemize}
\item provide a synopsis of the entire article;
\item begin with the broad context of the study, followed by specific background for the study;
\item describe the purpose, methods and procedures, core findings and results, and conclusions of the study;
\item emphasize new or important aspects of the research;
\item engage the broad readership of GENETICS and be understandable to a diverse audience (avoid using jargon);
\item be a single paragraph of less than 250 words;
\item contain the full name of the organism studied;
\item NOT contain citations or abbreviations.
\end{itemize}
\section{Introduction}
In individual organisms where a mutant is being studied, the rationale for the study of that mutant must be clear to a geneticist not studying that particular organism. Similarly, study of particular phenotypes should be justified broadly and not on the basis of interest for that organism alone. General background on the importance of the genetic pathway and/or phenotype should be provided in a single, well-reasoned paragraph near the beginning of the introduction.
Authors are encouraged to:
Added ideally
Jan 22, 2019 10:45 PM • You
Reject
Accept
You: Charlie, could you write the first draft of the introduction? Thanks!
Jan 22, 2019 10:45 PM • Edit
Hit Enter to reply
Resolve
Reply
Deleted For the introduction, authors should be... (show all)
Jan 22, 2019 10:46 PM • You
Reject
Accept
Current file
Overview
ABSTRACT The abstract should be written for people who may not read the entire paper, so it must stand on its own. The impression it makes usually determines whether the reader will go on to read the article, so the abstract must be engaging, clear, and concise. In addition, the abstract may be the only part of the article that is indexed in databases, so it must accurately reflect the content of the article. A well-written abstract is the most effective way to reach intended readers, leading to more robust search, retrieval, and usage of the article.
Please see additional guidelines notes on preparing your abstract below.
KEYWORDS Keyword; Keyword2; Keyword3; ...
This Genetics journal template is provided to help you write your work in the correct journal format. Instructions for use are provided below.
Guide to using this template in Overleaf
This template is provided to help you prepare your article for submission to the Genetics.
Author Affiliations
For the authors' names, indicate different affiliations with the symbols: *, †, ‡, §. After four authors, the symbols double, triple, quadruple, and so forth as required.
Your Abstract
In addition to the guidelines provided in the example abstract above, your abstract should ideally:
provide a synopsis of the entire article;
begin with the broad context of the study, followed by specific background for the study;
describe the purpose, methods and procedures, core findings and results, and conclusions of the study;
emphasize new or important aspects of the research;
engage the broad readership of GENETICS and be understandable to a diverse audience (avoid using jargon);
be a single paragraph of less than 250 words;
contain the full name of the organism studied;
NOT contain citations or abbreviations.
Introduction
In individual organisms where a mutant is being studied, the rationale for the study of that mutant must be clear to a geneticist not studying that particular organism. Similarly, study of particular phenotypes should be justified broadly and not on the basis of interest for that organism alone. General background on the importance of the genetic pathway and/or phenotype should be provided in a single, well-reasoned paragraph near the beginning of the introduction.
Authors are encouraged to:
cite the supporting literature completely rather than select a subset of citations;
provide important background citations, including relevant review papers (to help orient the non-specialist reader);
to cite similar work in other organisms.
Materials and Methods
Manuscripts submitted to GENETICS should contain a clear description of the experimental design in sufficient detail so that
Genetics 1
the experimental analysis could be repeated by another scientist. If the level of detail necessary to explain the protocol goes beyond two paragraphs, give a short description in the main body of the paper and prepare a detailed description for supporting information. For example, details would include indicating how many individuals were used, and if applicable how individuals or groups were combined for analysis. If working with mutants indicate how many independent mutants were isolated. If working with populations indicate how samples were collected and whether they were random with respect to the target population.
Statistical Analysis
It is important to indicate what statistical analysis has been performed; not just the name of the software and options selected, but the method and model applied. In the case of many genes being examined simultaneously, or many phenotypes, a multi-
Additional guidelines
Numbers
In the text, write out numbers nine or less except as part of a date, a fraction or decimal, a percentage, or a unit of measurement. Use Arabic numbers for those larger than nine, except as the first word of a sentence; however, try to avoid starting a sentence with such a number.
Units
Use abbreviations of the customary units of measurement only when they are preceded by a number: "3 min" but "several minutes". Write "percent" as one word, except when used with a number: "several percent" but "75%." To indicate temperature in centigrade, use ° (for example, 37°); include a letter after the degree symbol only when some other scale is intended (for example, 45°K).

# LaTeX




https://www.quora.com/What-are-the-benefits-of-using-LaTeX-over-a-traditional-WYSIWYG-editor

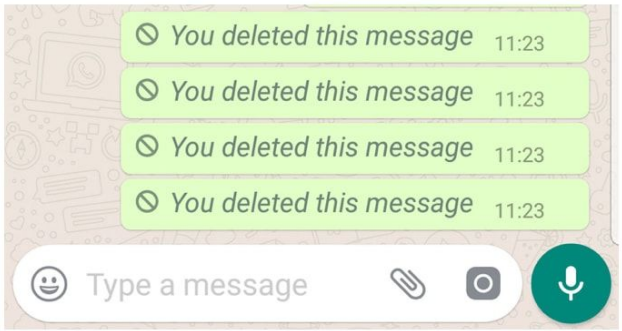
# Some terminology: UX and UI

- User Experience (UX)
  UX is how it works and feels

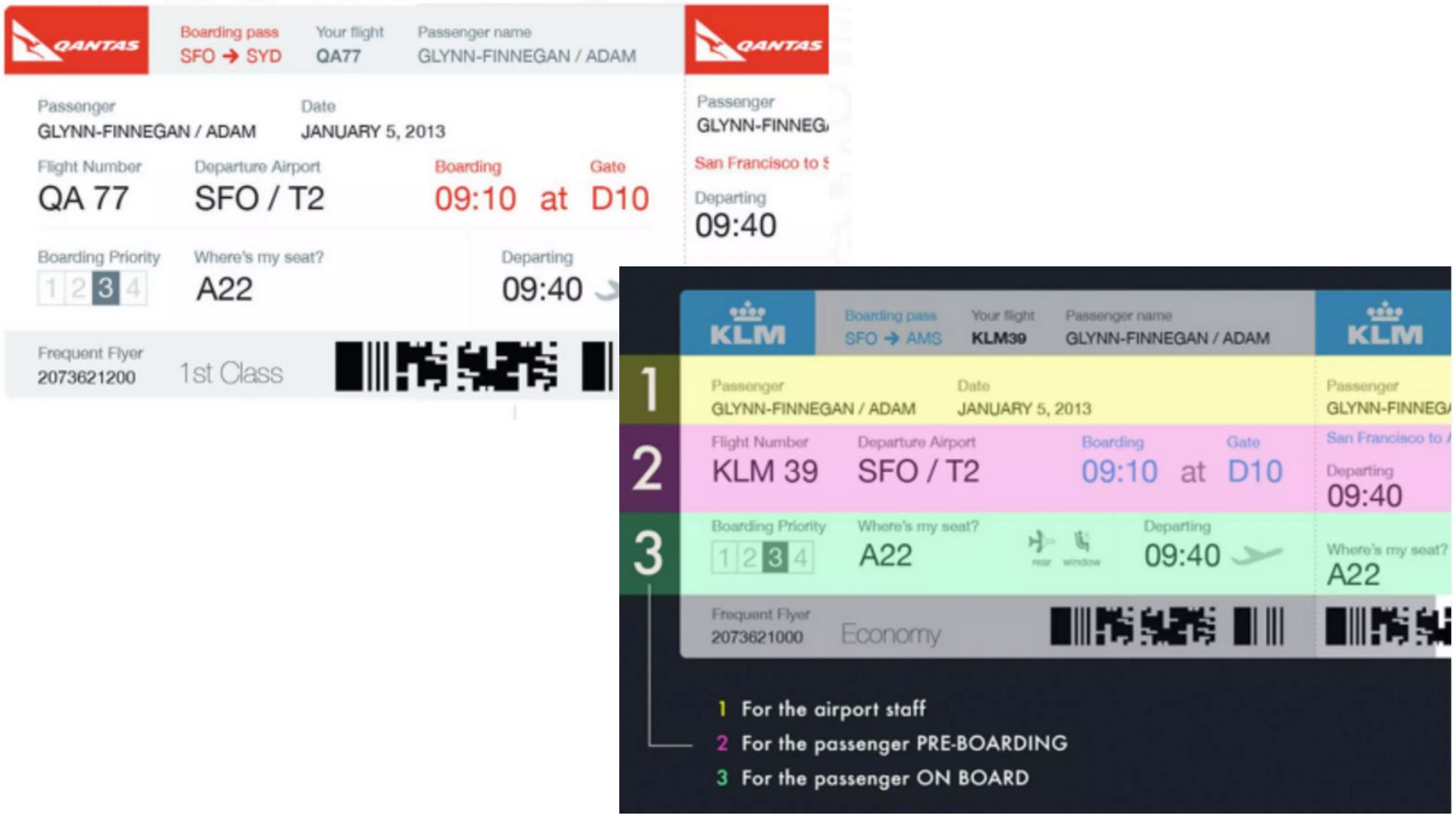
- User Interface (UI)
  UI is how it looks

  ⇒ UX and UI are related, UX != UI

# Bad UX/UI


You deleted this message 11:23
You deleted this message 11:23
You deleted this message 11:23
You deleted this message 11:23
Type a message


12:00


QANTAS
Boarding pass
SFO → SYD
Your flight
QA77
Passenger name
GLYNN-FINNEGAN / ADAM
Passenger
GLYNN-FINNEGAN / ADAM
Date
JANUARY 5, 2013
Flight Number
QA 77
Departure Airport
SFO / T2
Boarding
09:10
at
Gate
D10
Boarding Priority
1 2 3 4
Where's my seat?
A22
Departing
09:40
Frequent Flyer
2073621200
1st Class
Passenger
GLYNN-FINNEG
San Francisco to
Departing
09:40
KLM
Boarding pass
SFO → AMS
Your flight
KLM39
Passenger name
GLYNN-FINNEGAN / ADAM
1
2
3
Passenger
GLYNN-FINNEGAN / ADAM
Date
JANUARY 5, 2013
Flight Number
KLM 39
Departure Airport
SFO / T2
Boarding
09:10
at
Gate
D10
Boarding Priority
1 2 3 4
Where's my seat?
A22
rear
window
Departing
09:40
Frequent Flyer
2073621000
Economy
Passenger
GLYNN-FINNEG
San Francisco to
Departing
09:40
Where's my seat?
A22
1 For the airport staff
2 For the passenger PRE-BOARDING
3 For the passenger ON BOARD

# Purpose of UX

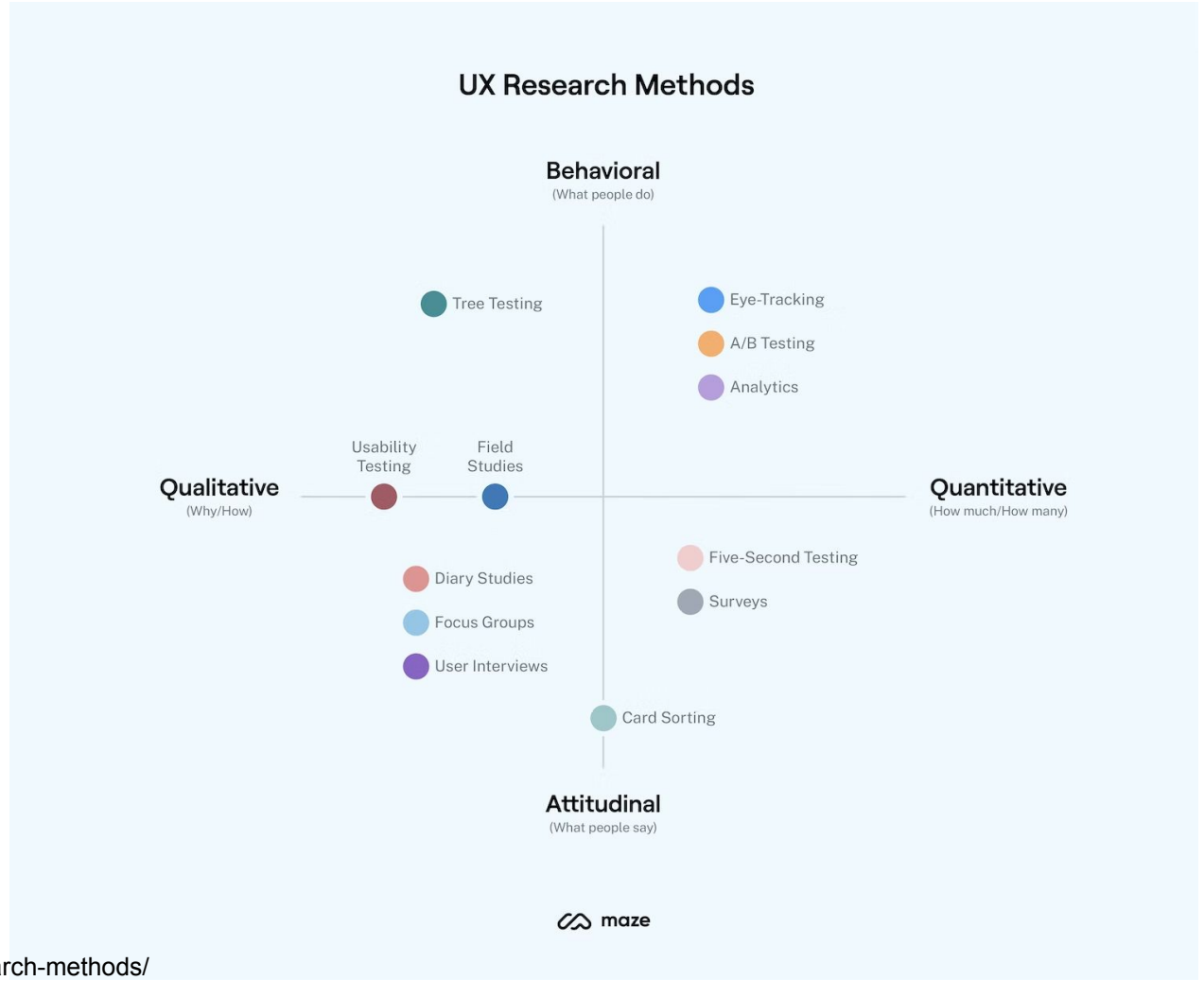
→ Fulfill business goals ←

1. USER ACQUISITION

2. USER ACTIVATION

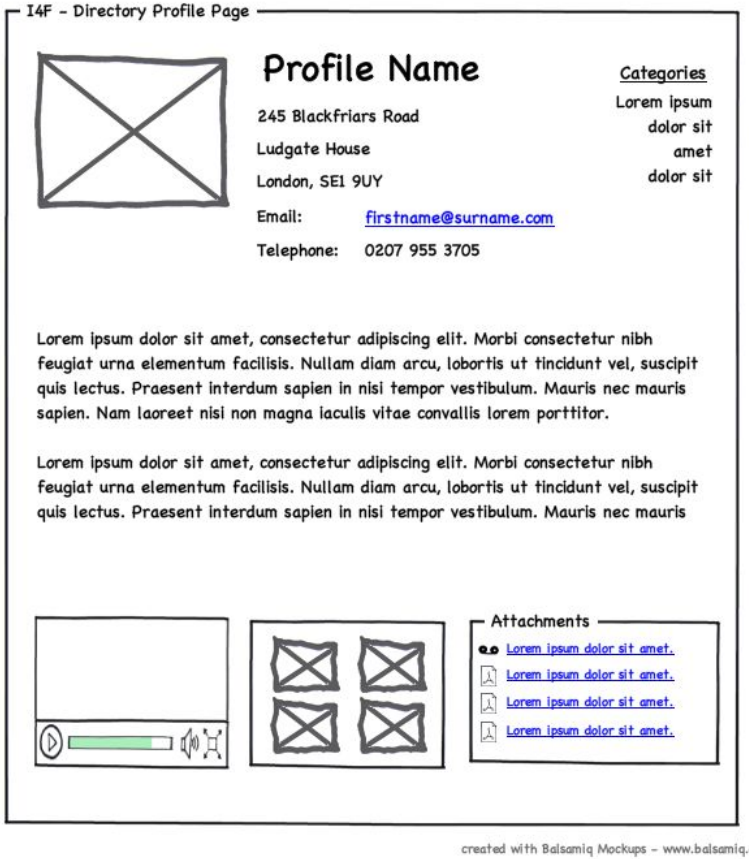
3. USER RETENTION

# User research




https://maze.co/guides/ux-research/ux-research-methods/

# Wireframes




Used by UX designers (show flow/navigation), by business analysts (how the brand works, how the product should be sold), UI designers (prototype, website)

**LOW FIDELITY**
First draft that depicts the coarse goal, to be negotiated with the customer (needs high imagination of the customer)

**HIGH FIDELITY**
Already very worked out wireframe, close to a mock-up (needs less imagination of the customer)

# Psych Design

Cognitive Psychology

➜ how people acquire, process and store information (science of mental processes)

Social Psychology

➜ how people behave in a social context / how thought, feeling and behavior are influenced by the ones around us

# Understanding the brain

**Reptilian** / Instinctual
- Common to all animals
- Basic Functions
- Risky vs Safe
- Sex
- Food

**Middle Brain** / Emotional
- Limbic system
- Ancient & Automatic
- Amygdala - fear, relevance, trust
- Thalamus - Happy, sad, disgusted
- VTA Dopamine - risk & reward

Emotions:
Pleasure, pain, empathy (“are you ok”)?

**New Brain** / Rational
- Unique to humans
- Higher cognitive functions
- Plane, organise, problem-solving
- Social learning & innovation
- Language, abstract thought

Understanding product benefits and product demonstrations, social context, authority

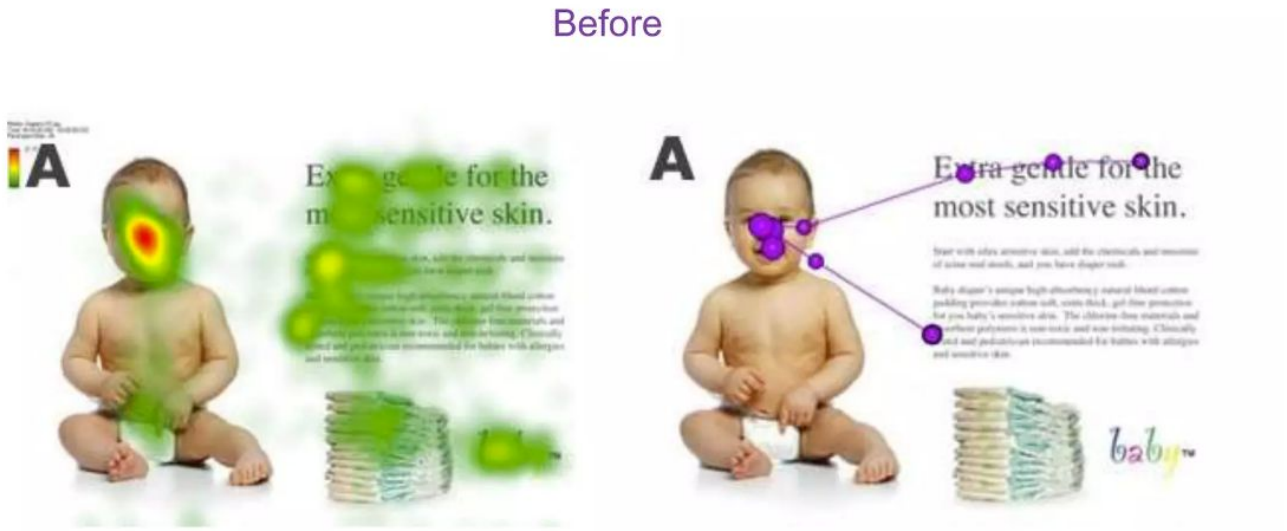
https://www.slideshare.net/maorshabbat/usable-psychology-for-uxui-designers

# Colors have different meaning depending on culture

| | Western/ American | Japanese | Hindu | Native American | Chinese | Asian | Eastern European | Muslim | African | South American |
|---|---|---|---|---|---|---|---|---|---|---|
| Holiness | | | | | | | | | | |
| Illness | | | | | | | | | | |
| Insight | | | | | | | | | | |
| Intelligence | | | | | | | | | | |
| Intuition | | | | | | | | | | |
| Religion | | | | | | | | | | |
| Jealousy | | | | | | | | | | |
| Joy | | | | | | | | | | |
| Learning | | | | | | | | | | |
| Life | | | | | | | | | | |
| Love | | | | | | | | | | |
| Loyalty | | | | | | | | | | |
| Luxury | | | | | | | | | | |
| Marriage | | | | | | | | | | |
| Modesty | | | | | | | | | | |
| Money | | | | | | | | | | |
| Mourning | | | | | | | | | | |
| Mystery | | | | | | | | | | |
| Nature | | | | | | | | | | |
| Passion | | | | | | | | | | |
| Peace | | | | | | | | | | |

https://blogs.sas.com/content/sastraining/2017/06/29/colors-in-different-cultures/

# We process faces first

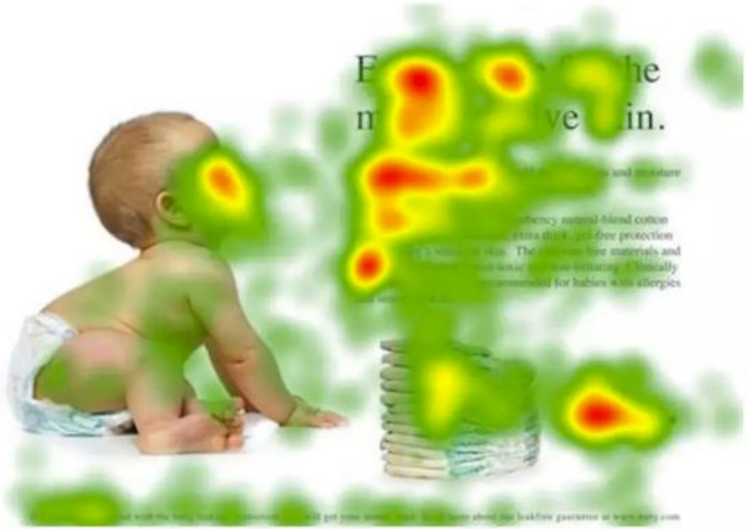


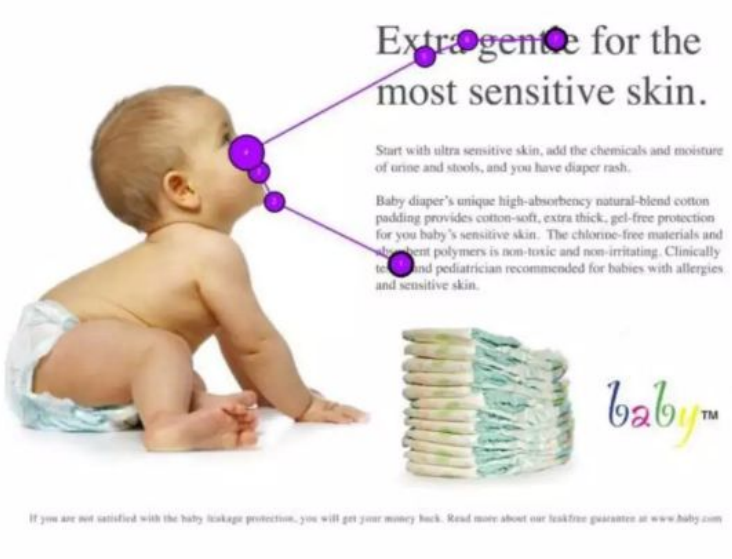

We process faces efficiently -> most of our time is spent identifying a person

https://www.slideshare.net/maorshabbat/usable-psychology-for-uxui-designers

# Adjusted photographs direct gaze

After

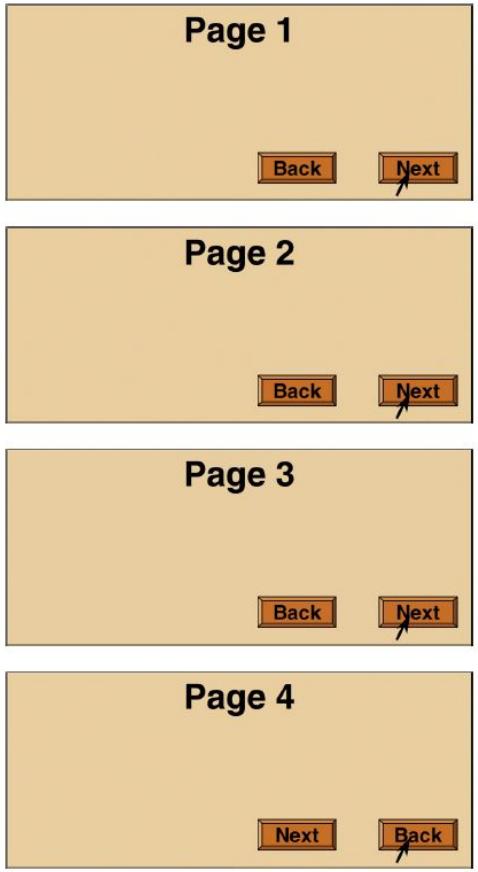




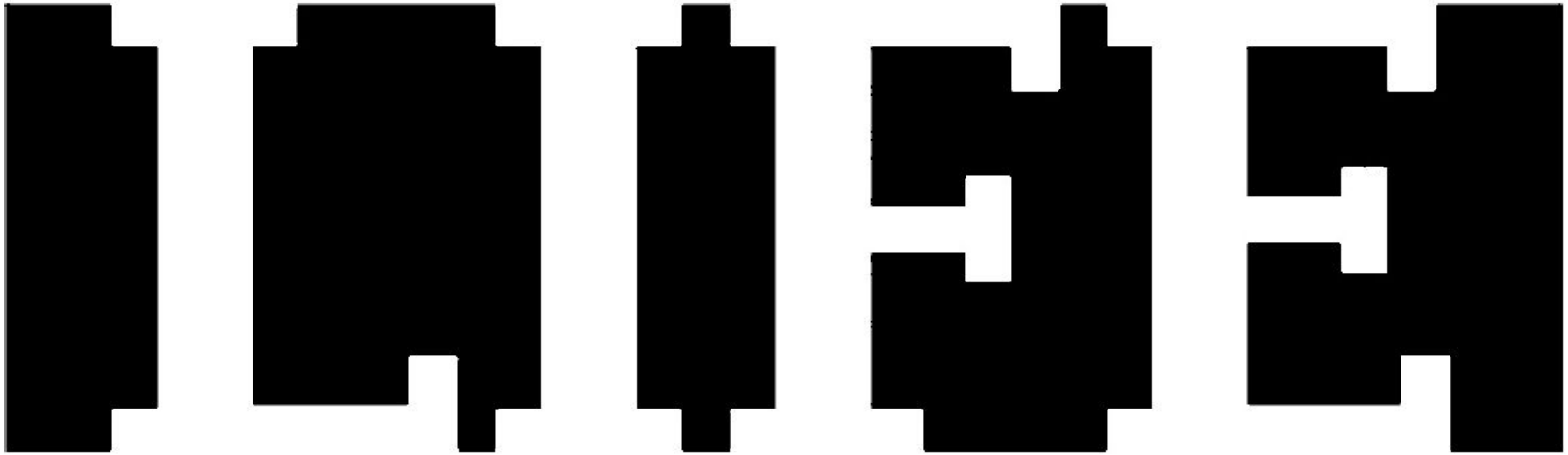

https://www.slideshare.net/maorshabbat/usable-psychology-for-uxui-designers

# Understanding human vision to guide interface development

# 1. Our perception is biased

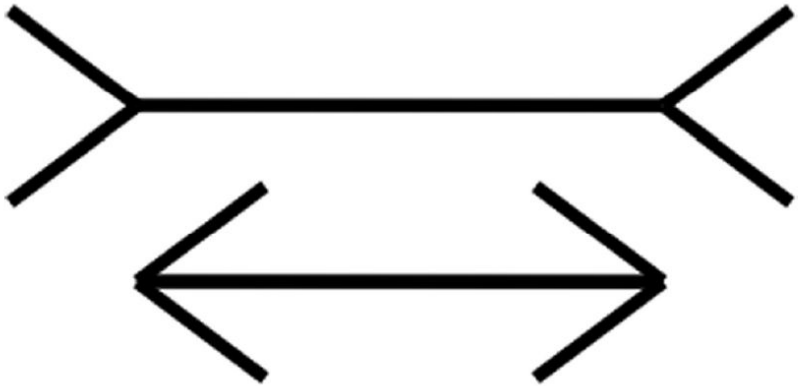


# You see what you are told to see



# Priming

# Also visual context is important

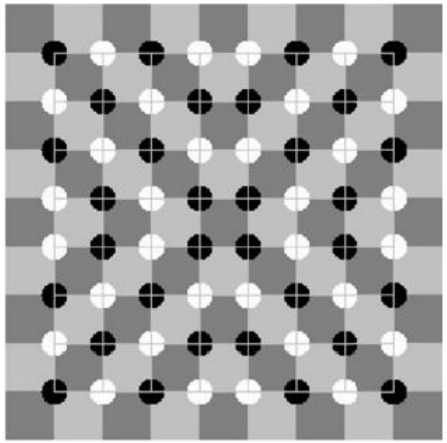


Müller-Lyer illusion

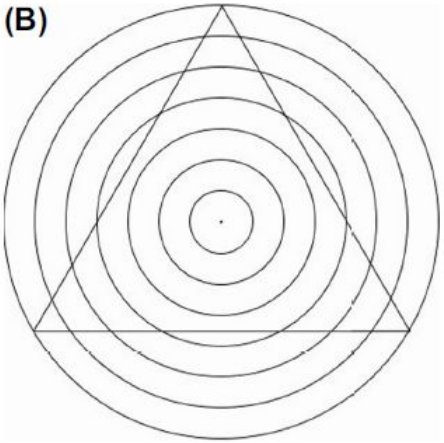
# More optical illusions

No bulge in the middle

(A)

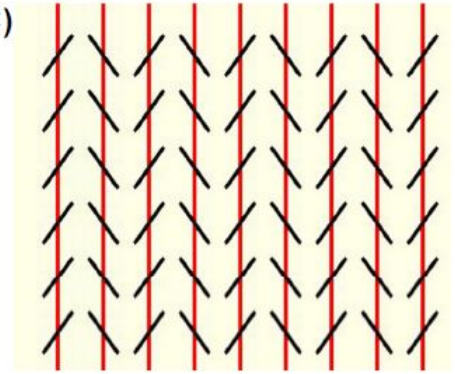


(B)



Triangle sides are not bend

(C)

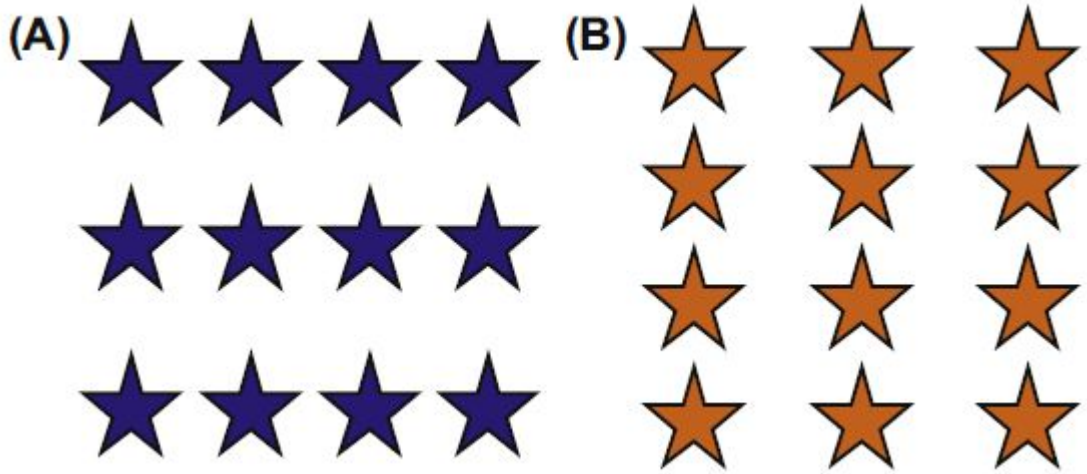


Red lines ARE parallel

# When designing, take biased perception into account...

- Avoid ambiguity
- Be consistent
- Understand the goals

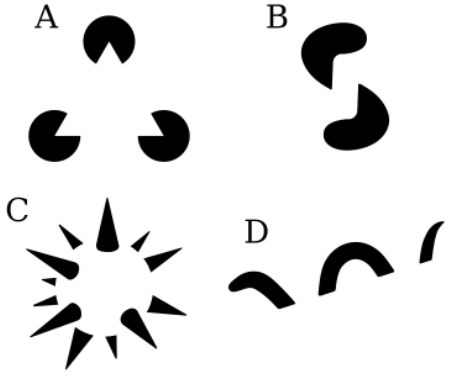
# Our vision is optimized to see STRUCTURE



# Gestalt principles

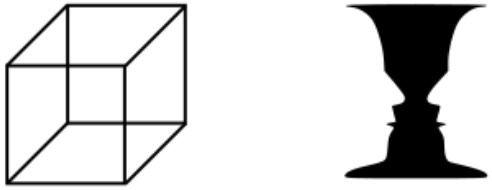
- Founded early 20th century in Austria/Germany
- introduced the Gestalt psychology and derived rules and laws

## Reification

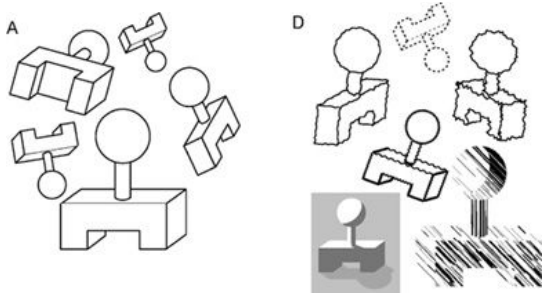



More explicit spatial information than the sensory information perceived

## Multistability



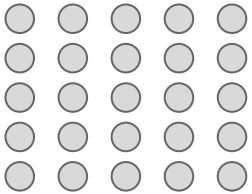
## Invariance



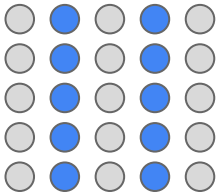

## Emergence

# Derived laws

Law of proximity

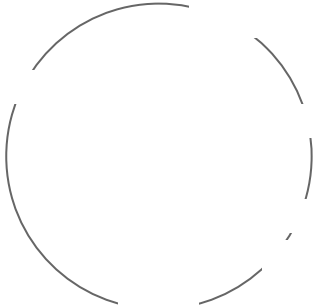


Law of similarity

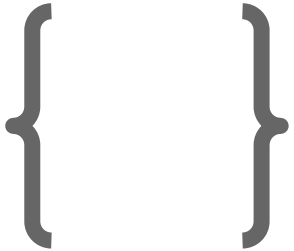


Law of past experience

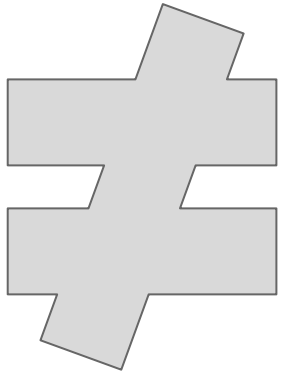
Law of closure



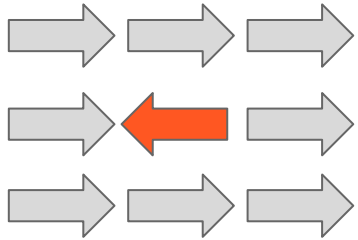
Law of symmetry



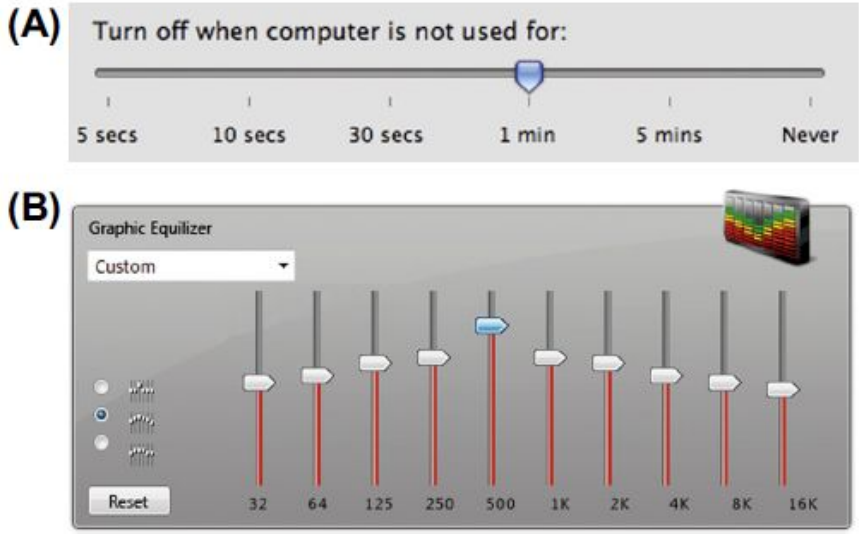
Law of continuity



Law of common fate



# Vision likes continuity



# Good and bad

**Renewals, Duplicates, and Information Changes for Driver Licenses and/or ID Cards**

- How to renew your driver license in person
- How to renew your driver license by mail
- How to renew your driver license by Internet
- How to renew your instruction permit
- How to apply for a duplicate driver license or identification (ID) card
- How to change your name on your driver license and/or identification (ID) card
- How to notify DMV of my change of address
- How to register for the organ donor gift of life program

| Licenses & ID Cards: | Renewals, Duplicates, Changes | | |
|---|---|---|---|
| • Renew license: | in person | by mail | by Internet |
| • Renew: | instruction permit | | |
| • Apply for duplicate: | license | ID card | |
| • Change of: | name | address | |
| • Register as: | organ donor | | |

# Another example

(A)

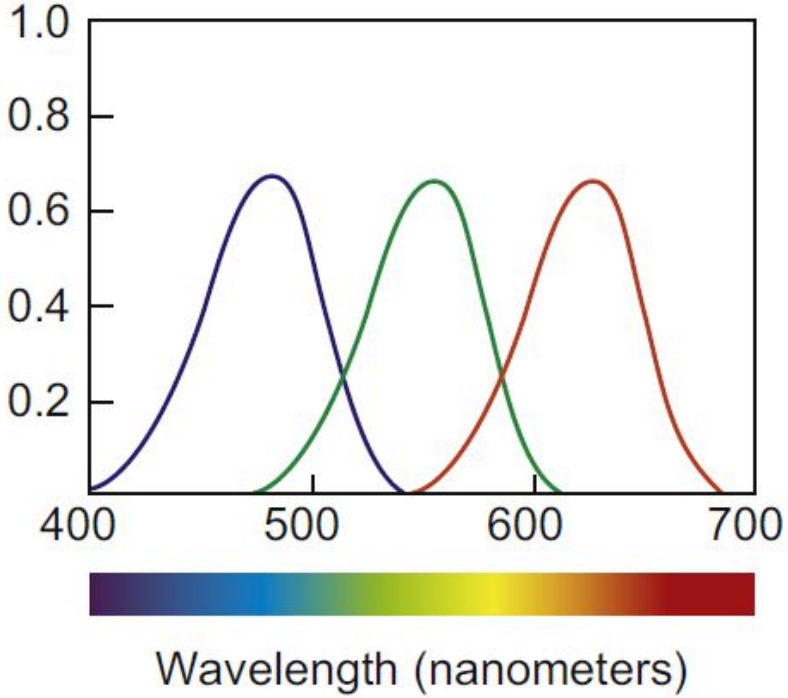
| Mortgage Summary | |
|---|---|
| $1,840.59 | $662,611.22 |
| Monthly Payment | Total of 360 Payments |
| $318,861.22 | Sep, 2037 |
| Total Interest Paid | Pay-off Date |
| $93,750.00 | $0.00 |
| Total Tax Paid | Total PMI Paid |

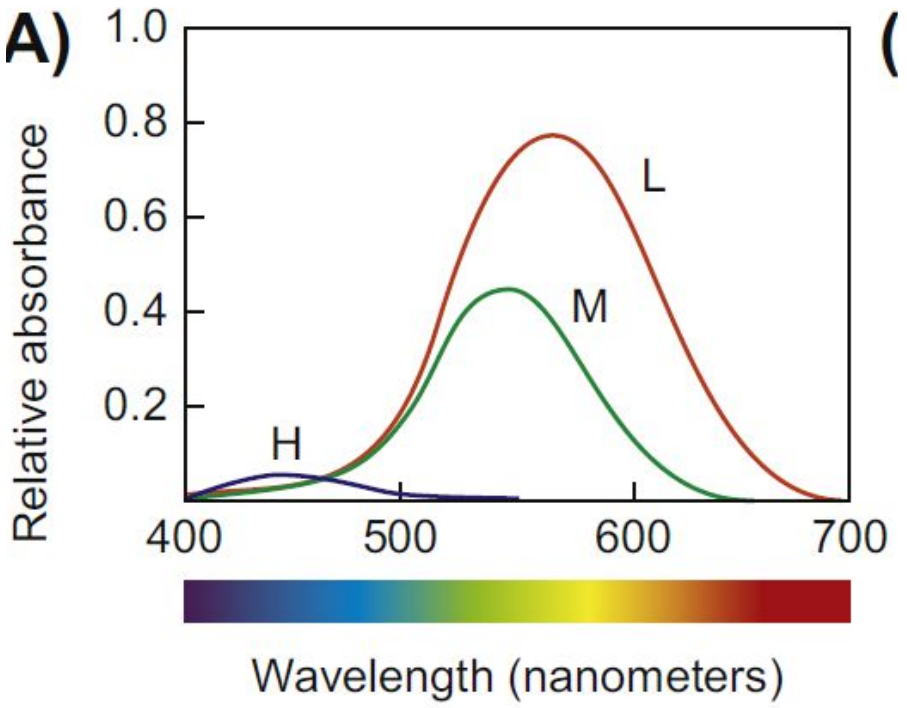
(B)

| Mortgage Summary | |
|---|---|
| *Monthly Payment* | $ 1,840.59 |
| *Number of Payments* | 360 |
| *Total of Payments* | $ 662,611.22 |
| *Interest Total* | $ 318,861.22 |
| *Tax Total* | $ 93,750.00 |
| *PMI Total* | $ 0.00 |
| *Pay-off Date* | Sep 2037 |

FIGURE 3.5

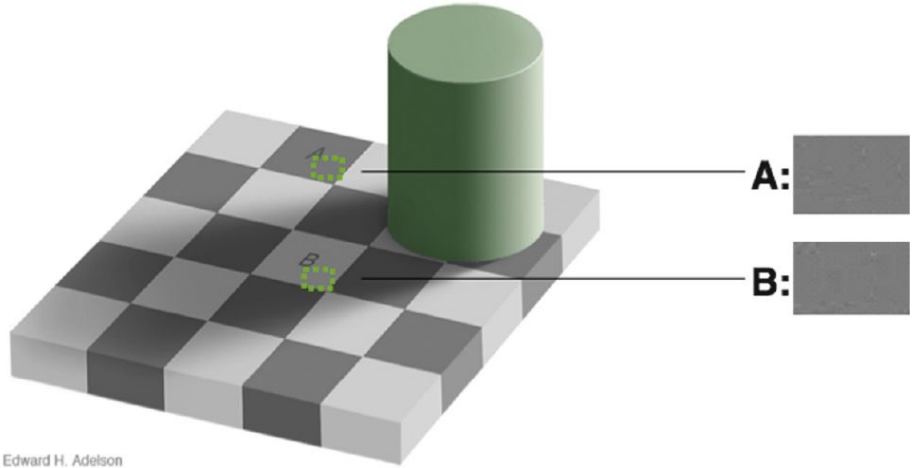
# Human color vision is highly limited



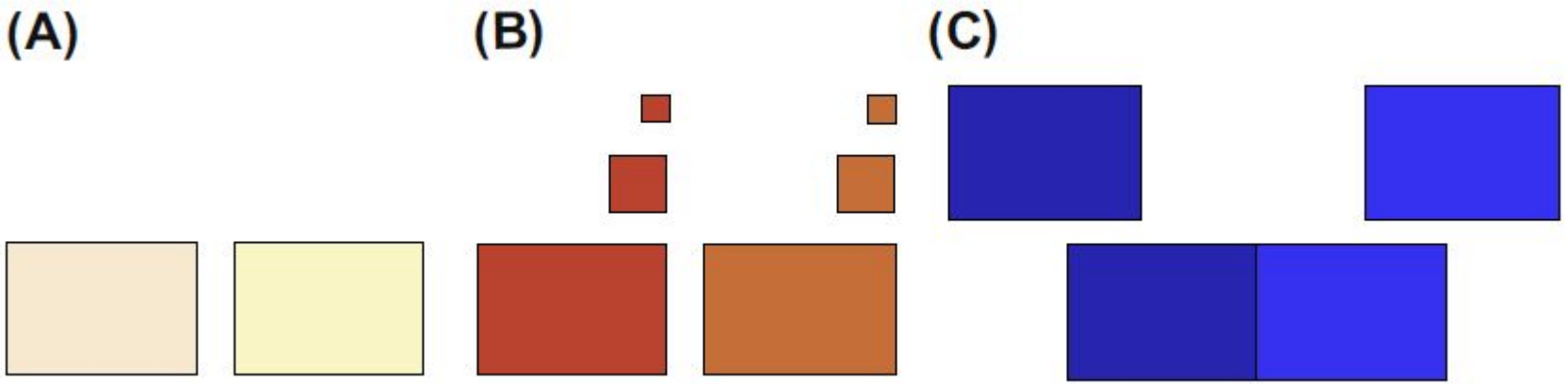
# 50 shades of gray

Inner gray bar has equal gray value EVERYWHERE!

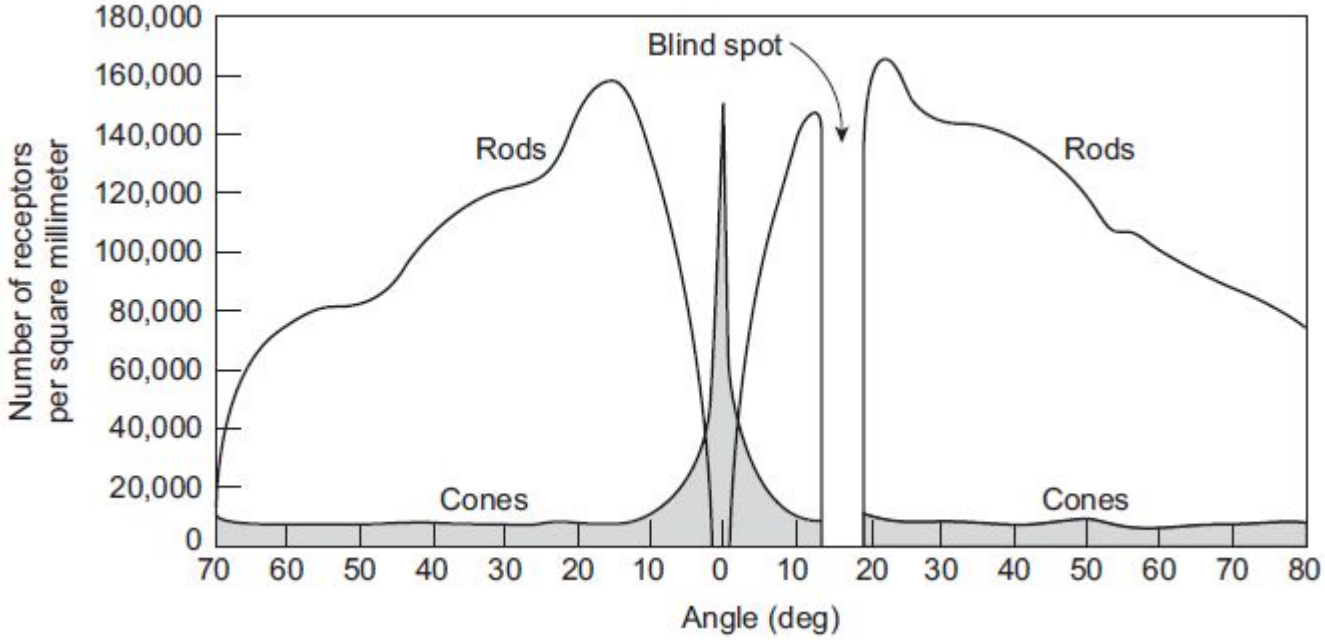


# Color separation is hard

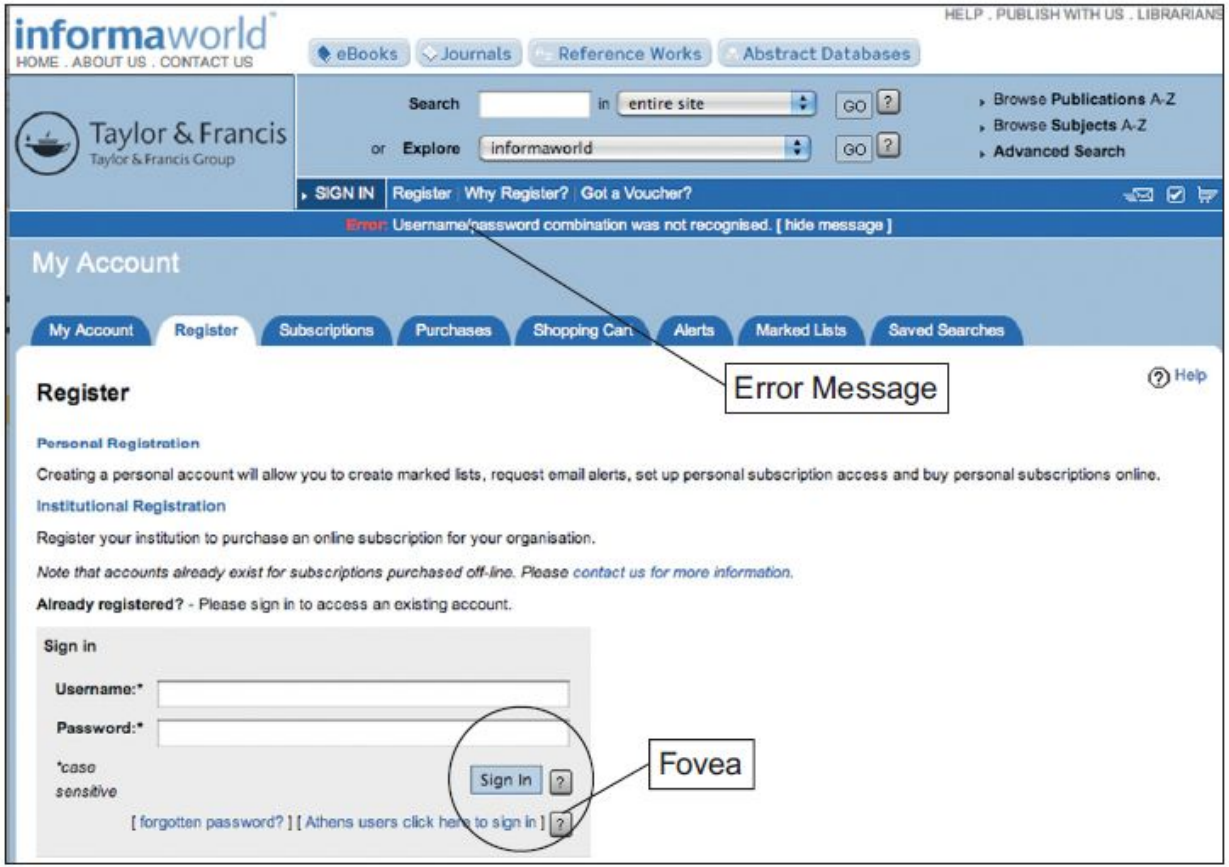
- Paleness (Saturation)
- Size
- Distance/Separation



# Peripheral color vision is poor



# Centralize important information




# Another example

RETURNING CUSTOMER LOGIN

Login ID not found.

Login ID: fooo

Password:

Remember my Login ID for faster logins.

LOGIN

# Common methods to make errors visible

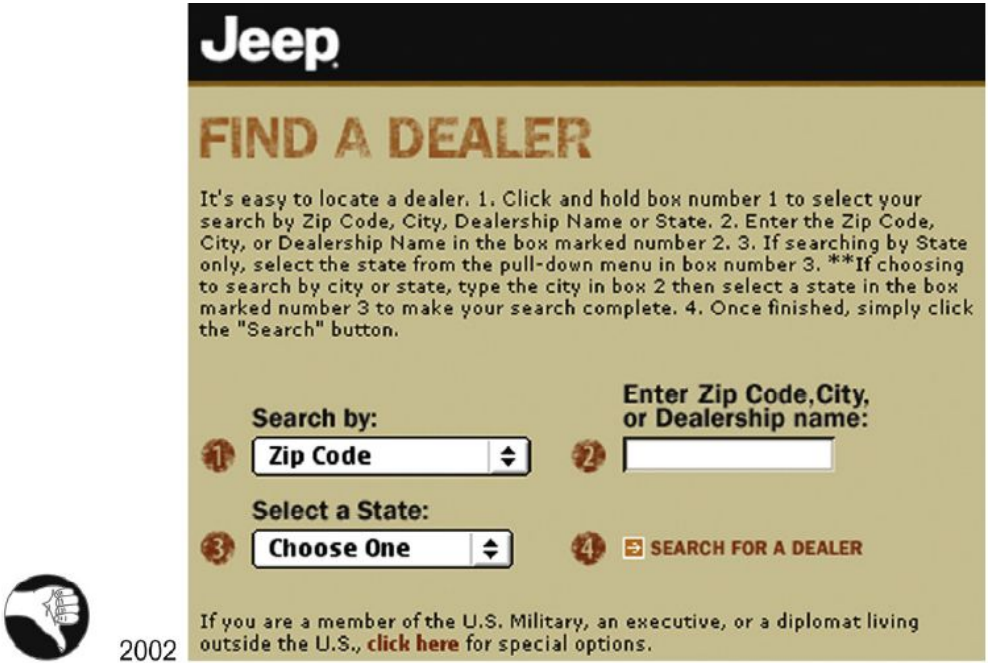
- Put it where users are looking at (peripheral vision!!)
- Mark the error
- Use an error symbol
- Reserve red for errors (and maybe green for success…)



# People are poor in reading

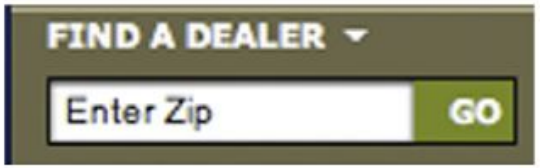
Avoid to much text and information.

Provide text on request/as needed.




2002


2003




2007

# Use a natural cycle

- GOAL
- EXECUTION
- EVALUATION

Example: Booking a flight/order some flowers

- ***Goal.*** Provide clear paths—including initial steps—for the user goals that the software is intended to support.
- ***Execute.*** Software concepts (objects and actions) should be based on the task rather than the implementation (see Chapter 11). Don't force users to figure out how the software's objects and actions map to those of the task. Provide a clear information scent at choice points to guide users to their goals. Don't make them choose actions that seem to take them away from their goal to achieve it.
- ***Evaluate.*** Provide feedback and status information to show users their progress toward the goal. Allow users to back out of tasks that didn't take them toward their goal.

# Recognition is easy, recall is hard

- When was America discovered?
- Capital of Brazil?
- Name of all 16 federal states of Germany?

# See and choose is easier than recall

*Remember and type:*

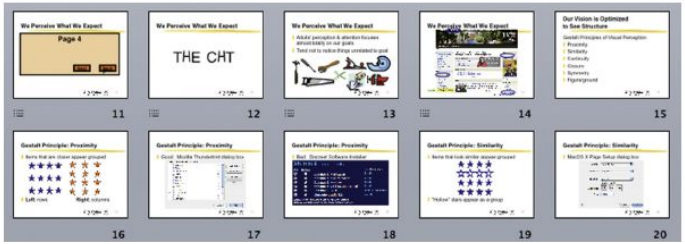
> copy doc1 doc2

> remove olddoc

*See and choose*



Analogy with physical items




Thumbnails for recognition, not recall


Wordpress uses Icon + Text

# Perform novel actions is hard

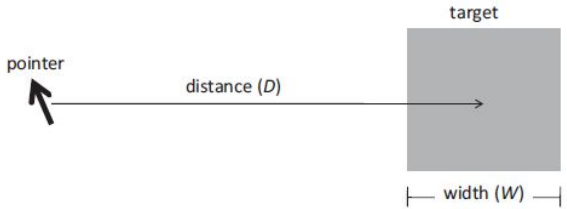
Example:

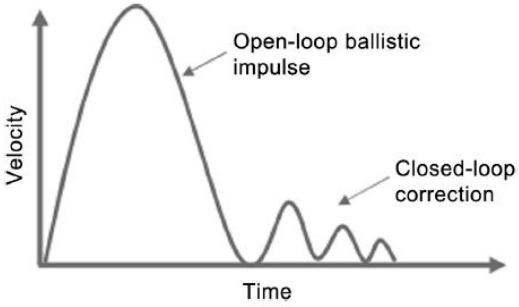
- Count from 10 to 0 (rocket launch)
- Count from 21 to 1 in steps of 2

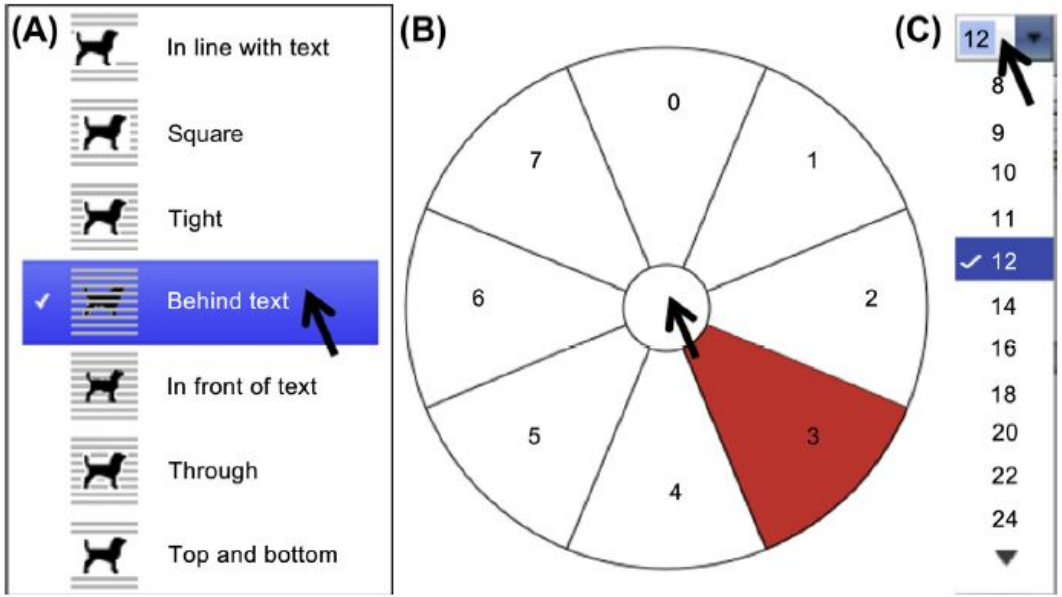
➜ Make interfaces similar to that people are used to!

# Hand-eye coordination follows laws!

- First ballistic
- Then correction

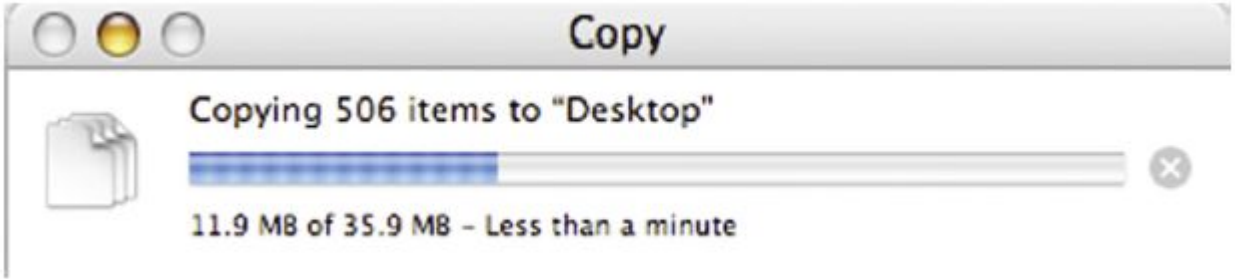


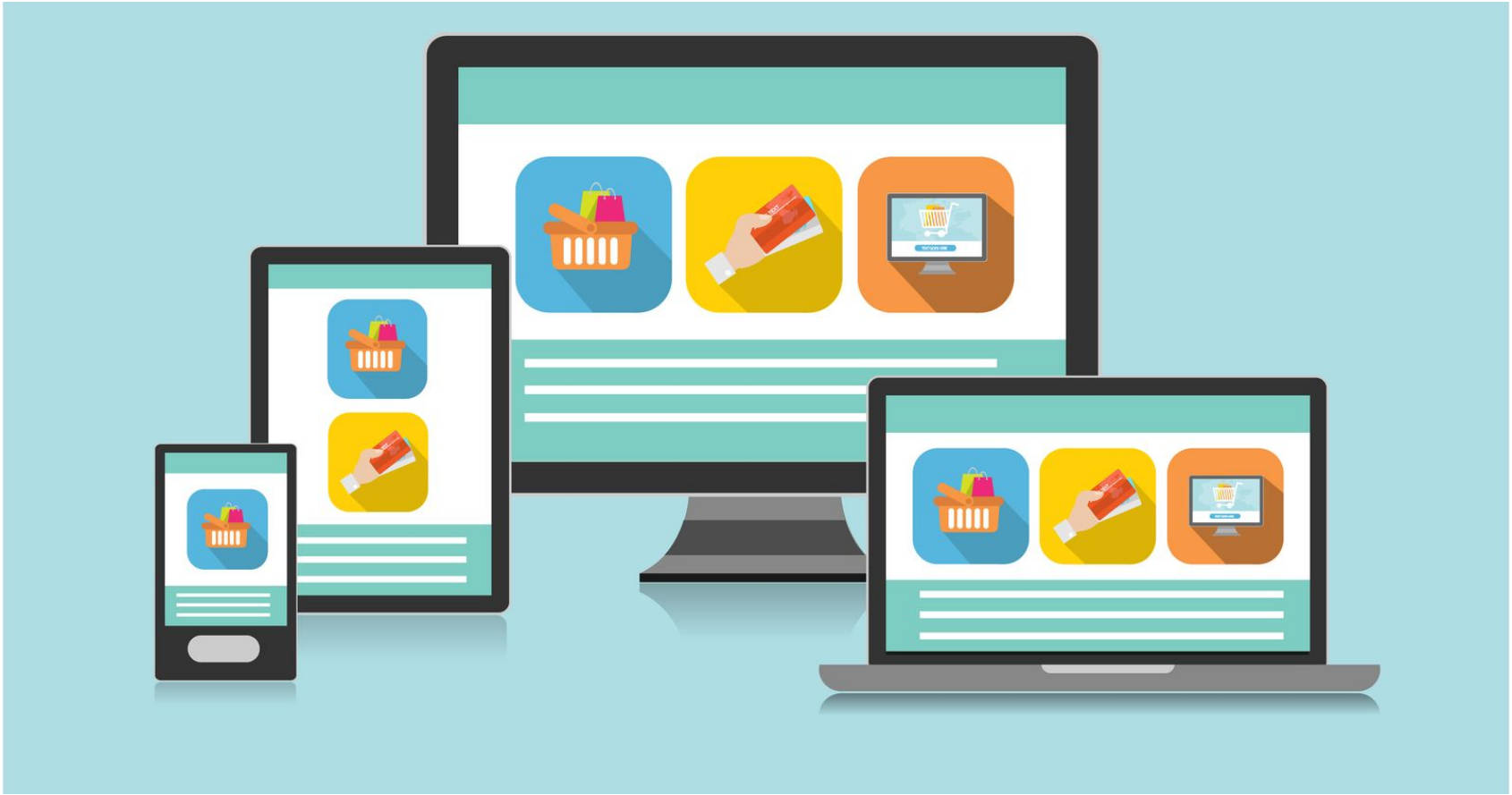




Prefer pop-up and pie menus rather than pull-down (C)

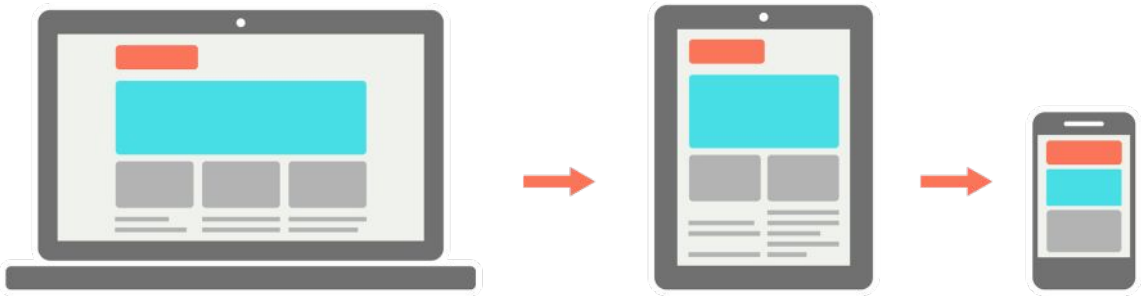
# People have time constraints

# Responsive design

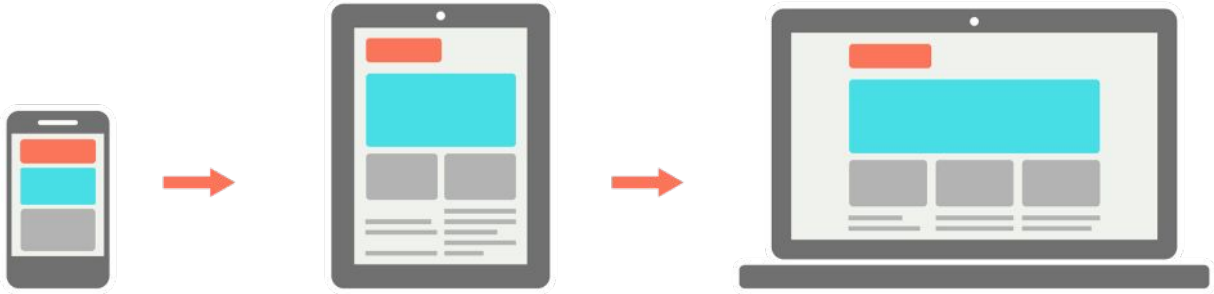


# Mobile first



Responsive Web Design

Mobile First Web Design



# Why is responsive design important?

- Allow access through various digital devices (phones, tablets, computers)
- Adapt information to different use cases
  - Mobile phone - directions, locations, important news
  - Desktop - research, information collection, investigation

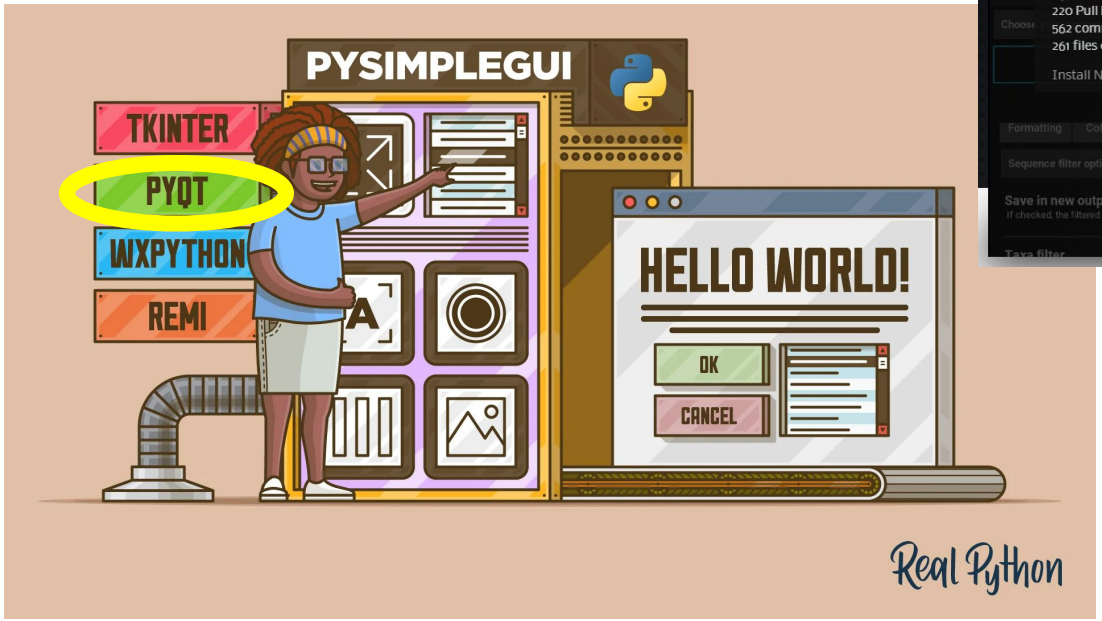
Challenges:

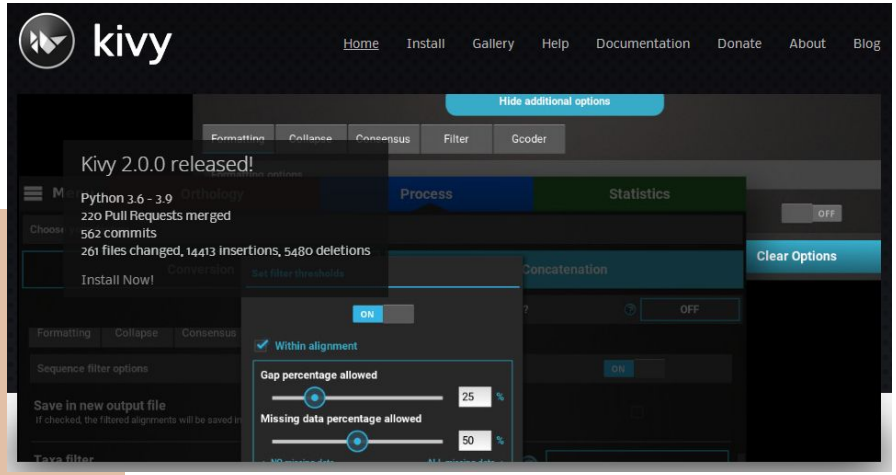
- Maintaining design integrity across different devices
- Load times
- Touch versus click interfaces.

# Changing gears

Now about Pythoooon

# Programming GUIs in Python



Qt

Qt

Qt 6.2 is o

FUTURE IS WRITTEN WITH Qt

# One framework. One codebase. Any platform.

Qt is the fastest and smartest way to produce industry-leading software that users love.

Browse Qt Tools | Browse Qt Features

PyQt

Riverbank
Easy going, GPL
license (commercial
ca. 500 USD)

vs

PySide


Qt for Python

# How to get PyQt5?

```
pip3 install PyQt5
```

- **QtCore** – Core non-GUI classes used by other modules
- **QtGui** – Graphical user interface components
- **QtMultimedia** – Classes for low-level multimedia programming
- **QtNetwork** – Classes for network programming
- **QtOpenGL** – OpenGL support classes
- **QtScript** – Classes for evaluating Qt Scripts
- **QtSql** – Classes for database integration using SQL
- **QtSvg** – Classes for displaying the contents of SVG files
- **QtWebKit** – Classes for rendering and editing HTML
- **QtXml** – Classes for handling XML
- **QtWidgets** – Classes for creating classic desktop-style UIs
- **QtDesigner** – Classes for extending Qt Designer

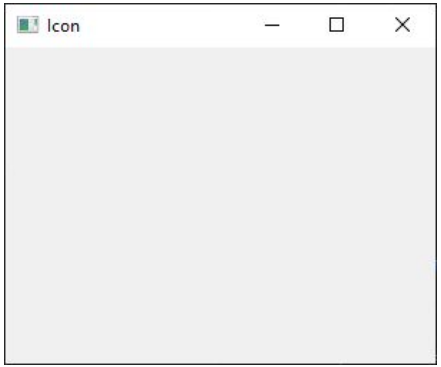
# How to create an application?

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget
from PyQt5.QtGui import QIcon


class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()


    def initUI(self):

        self.setGeometry(300, 300, 300, 220)
        self.setWindowTitle('Icon')
        self.setWindowIcon(QIcon('web.png'))

        self.show()


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

`QApplication`: The main loop

`QWidgets`: Windows

Start your application
in `__name__ == '__main__'`



# A button, and the signal/slot principle


```
widget.signal.connect(slot_function)
```

```
button.clicked.connect(slot_function)
```

```
b =  QPushButton("Button 1")
b.clicked.connect(self.clickedButton)
```

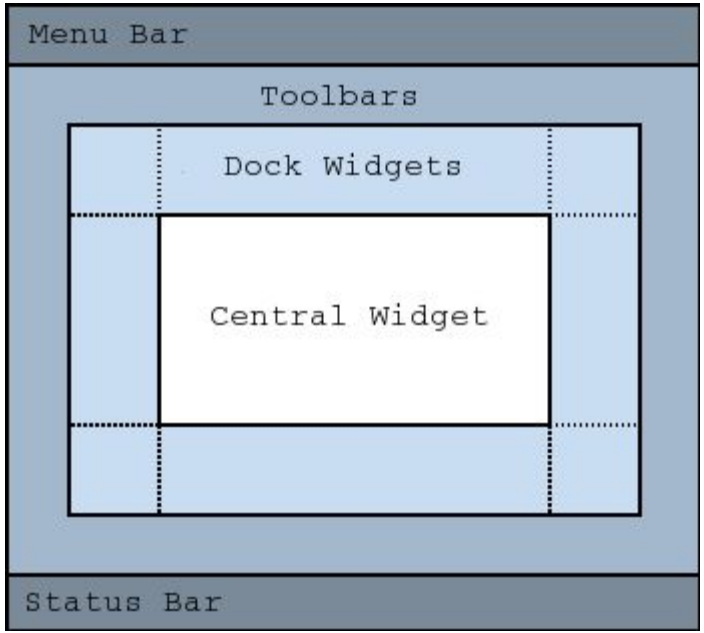
```
def clickedButton(self):
    print("Button pressed!")
```

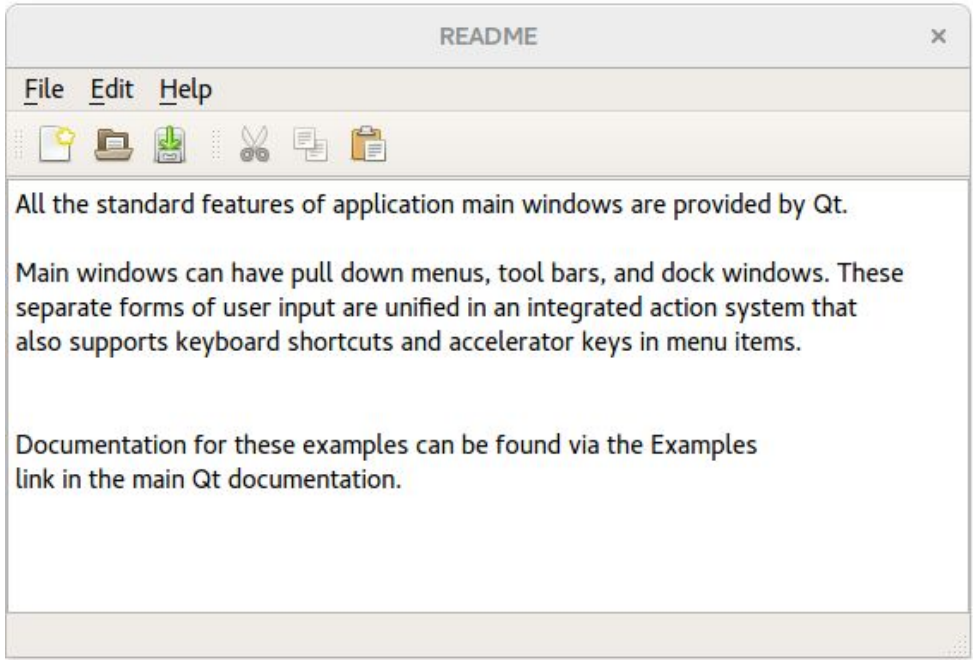
# Custom signals and slots

```
from PyQt5.QtCore import QObject, pyqtSignal

class Foo(QObject):

    # Define a new signal called 'trigger' that has no arguments.
    trigger = pyqtSignal()

    def connect_and_emit_trigger(self):
        # Connect the trigger signal to a slot.
        self.trigger.connect(self.handle_trigger)

        # Emit the signal.
        self.trigger.emit()

    def handle_trigger(self):
        # Show that the slot has been called.

        print "trigger signal received"
```

https://www.riverbankcomputing.com/static/Docs/PyQt5/signals_slots.html

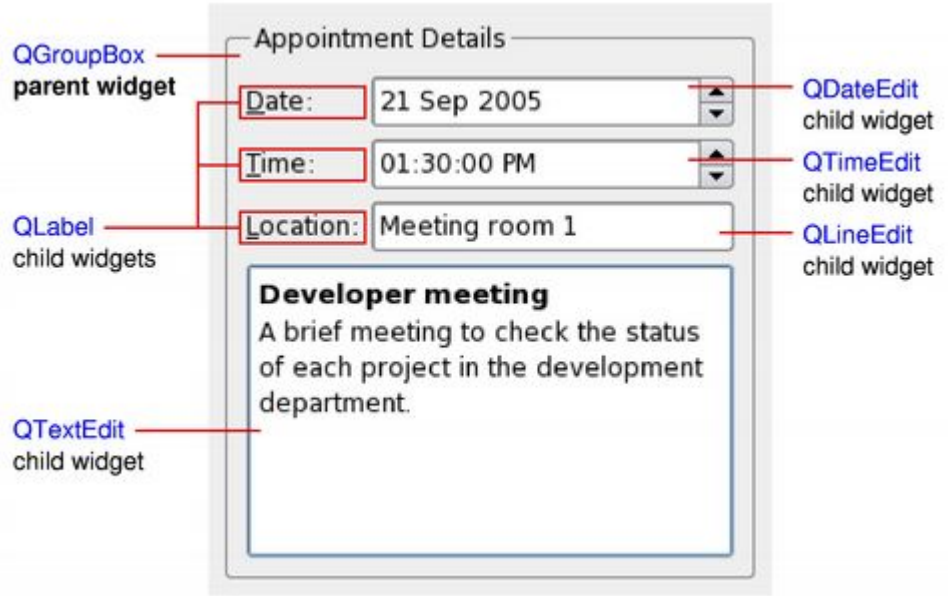
# QMainWindow



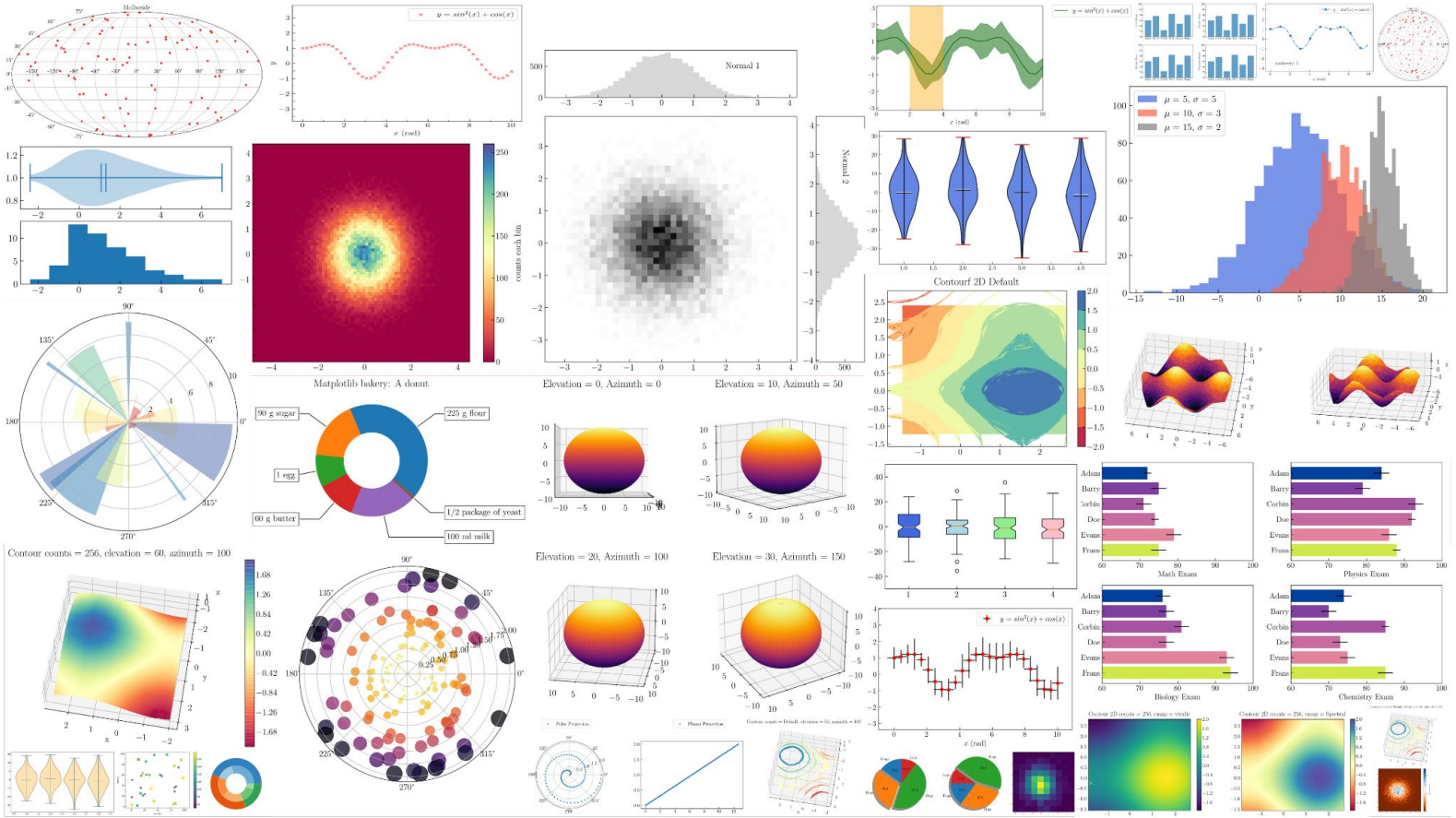

https://doc.qt.io/qtforpython-5/overviews/qtwidgets-mainwindows-application-example.html#application-example

# More widgets (parent/child)

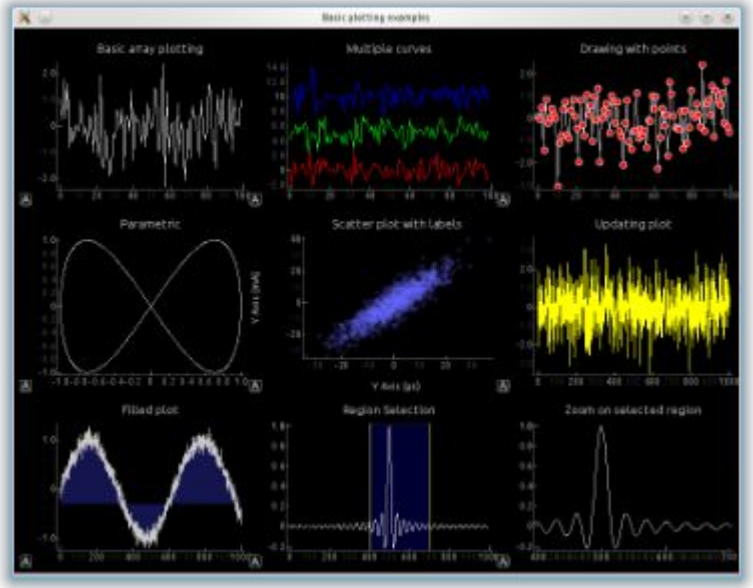


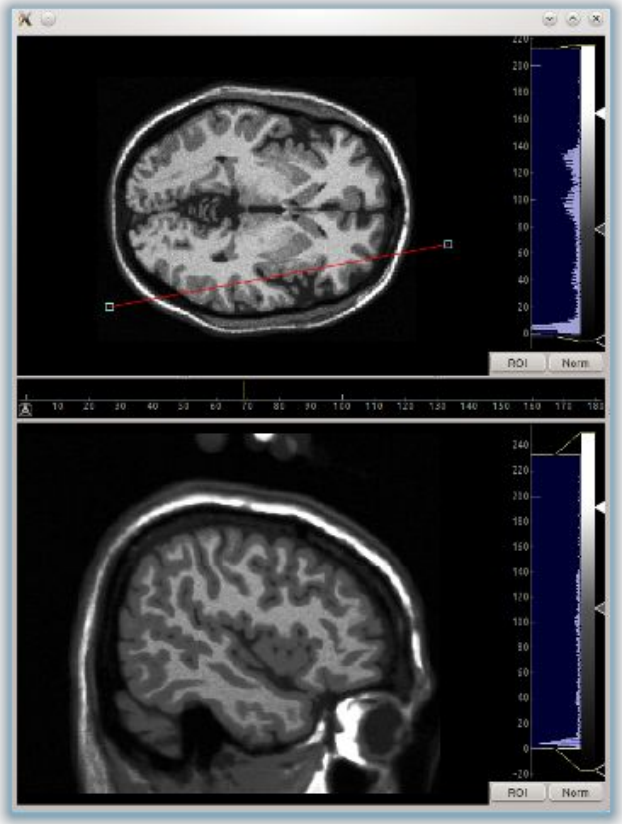

More widgets: https://doc.qt.io/qt-5/widget-classes.html#basic-widget-classes

# Showing graphs



# Showing graphs in GUI (fast)

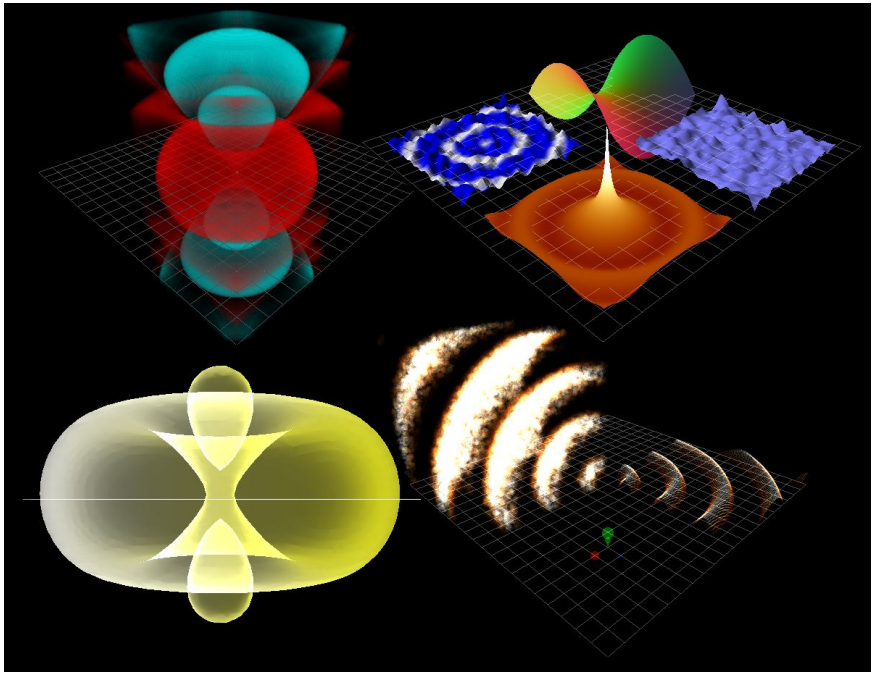


# pyqtgraph

```
import pyqtgraph as pg
pg.plot(data)   # data can be a list of values or a numpy array
```

➜ Creates a new window that plots data



# Pyqtgraph and PyQt

```
from PyQt5 import QtGui  # (the example applies equally well to PySide2)
import pyqtgraph as pg

## Always start by initializing Qt (only once per application)
app = QtGui.QApplication([])

## Define a top-level widget to hold everything
w = QtGui.QWidget()

## Create some widgets to be placed inside
btn = QtGui.QPushButton('press me')
text = QtGui.QLineEdit('enter text')
listw = QtGui.QListWidget()
plot = pg.PlotWidget()

## Create a grid layout to manage the widgets size and position
layout = QtGui.QGridLayout()
w.setLayout(layout)

## Add widgets to the layout in their proper positions
layout.addWidget(btn, 0, 0)   # button goes in upper-left
layout.addWidget(text, 1, 0)   # text edit goes in middle-left
layout.addWidget(listw, 2, 0)  # list widget goes in bottom-left
layout.addWidget(plot, 0, 1, 3, 1)  # plot goes on right side, spanning 3 rows

## Display the widget as a new window
w.show()

## Start the Qt event loop
app.exec_()
```

https://pyqtgraph.readthedocs.io/en/latest/qtcrashcourse.html

# Data Science Survival Skills

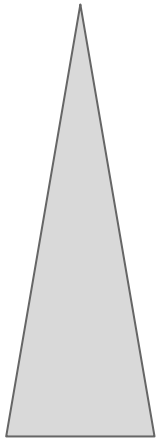
Code Deployment

# Sharing code with others

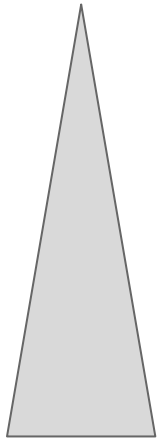
- Who is your audience?


- What is the purpose?

# Sharing code



- Raw code
- Library (*.dll, *.so)
- “EXE”
- Installer



User friendliness

Effort by developer

# Sharing raw code



Recap from earlier lectures:

- Code files
- Comments
- **Documentation**

⇒ Readthedocs and alike!

# Code

What do I need to run your code?

- OS (Windows, Mac OS X, Linux, ...)
- Additional hardware? GPU, Network access, ...
- Drivers
- Environment
- Libraries/dependencies
- Library version
- Code editor
- ...

We have a REPRODUCE RESEARCH RESULTS (RRR) project seminar to tackle exact this!

# Platform

- Windows 32 bit/64 bit (x86, x64)
- Mac OS X (10.5 runs 64-bit)
- Linux/Ubuntu
- ARM

# Sharing Python code

- Setup.py
- Requirements.txt
- Documentation!
- pip installable?

# ARM platform


ARM vs x86
Also 32/64 bit
RISC vs. CISC
(reduced instruction set computer vs. complex ...)

Advantages

- Lower TDP
- More efficient
- Low power requirements

Microsoft Surface with ARM
Macbook Air M
Android phones
iPhones
iPads...
Coffee machines

# Use tools for “compiling” Python files

- PyInstaller / fbs (this lecture)
- Py2exe
- Py2app (Mac OS)
- Nuitka
- Cython

# Use PyInstaller for packaging

1. Find all relevant packages

glabel > gui > main_window.py

```
1   #!/usr/bin/env python
2   from PyQt5 import QtGui
3   from PyQt5.QtWidgets import *
4   from PyQt5.QtGui import QColor, QScreen
5   from PyQt5.QtCore import Qt, QPointF, QThreadPool
6   from pyqtgraph import mkPen
7
8   import imageio as io
9   from skimage.io import imread
10  import cv2
11  import numpy as np
12
13  import h5py
14  import flammkuchen as fl
15  import json
16  import os
17  import re
18  import getpass
19  from datetime import datetime
20  import hashlib
21  import sys
22
```

# PyInstaller drawbacks

**Note:**

The output of *PyInstaller* is specific to the active operating system and the active version of Python. This means that to prepare a distribution for:

- a different OS
- a different version of Python
- a 32-bit or 64-bit OS

you run *PyInstaller* on that OS, under that version of Python. The Python interpreter that executes *PyInstaller* is part of the bundle, and it is specific to the OS and the word size.

# Poor man's distribution

## Bundling to One Folder

When you apply *PyInstaller* to `myscript.py` the default result is a single folder named `myscript`. This folder contains all your script's dependencies, and an executable file also named `myscript` (`myscript.exe` in Windows).

You compress the folder to `myscript.zip` and transmit it to your users. They install the program simply by unzipping it. A user runs your app by opening the folder and launching the `myscript` executable inside it.

# The –one-file option

## How the One-File Program Works

The bootloader is the heart of the one-file bundle also. When started it creates a temporary folder in the appropriate temp-folder location for this OS. The folder is named `_MEIxxxxxx`, where *xxxxxx* is a random number.

The one executable file contains an embedded archive of all the Python modules used by your script, as well as compressed copies of any non-Python support files (e.g. `.so` files). The bootloader uncompresses the support files and writes copies into the the temporary folder. This can take a little time. That is why a one-file app is a little slower to start than a one-folder app.

# Hidden imports

## Listing Hidden Imports

If Analysis thinks it has found all the imports, but the app fails with an import error, the problem is a hidden import; that is, an import that is not visible to the analysis phase.

Hidden imports can occur when the code is using `__import__()`, `importlib.import_module()` or perhaps `exec()` or `eval()`. Hidden imports can also occur when an extension module uses the Python/C API to do an import. When this occurs, Analysis can detect nothing. There will be no warnings, only an ImportError at run-time.

To find these hidden imports, build the app with the `--debug=imports` flag (see Getting Python's Verbose Imports above) and run it.

Once you know what modules are needed, you add the needed modules to the bundle using the `--hidden-import` command option, or by editing the spec file, or with a hook file (see Understanding PyInstaller Hooks below).

# What does an installer do?

**Tasks the installer may handle, are:**

- unpacking (often using exotic, high compression archivers)
- ensuring system hardware requirements
- ensuring sufficient hard-disk space
- ensuring software platform runtime requirements (e.g. 'redistributables')
- checking for newer software updates
- downloading the software from a remote repository
- creating and/or updating program files and folders
- create configuration files, registry entries or environment variables
- install sofware drivers, mount or unmount devices
- increase accessibility for everyday users, by explaining installation steps, creating links, shortcuts
- promote the own sofware through bookmarks, etc...
- create incentive for the user to actually startup the software, by presenting the keypoints of the software during the installation, slide by slide
- create additional revenue, through software-bundling
- configure kernel-modules and automatically running components (e.g. daemons, windows-services)
- automatic patching of the sofware
- setting folder, file and user permissions
- creating GUIDs to couple the software to a specific installation-instance and for instance, prevent portability

# Cross Platform Deployment

Diverse Operating Systems:

- Varied OS environments (Windows, macOS, Linux) lead to unique deployment challenges.
- System-specific features and limitations impact software behavior and compatibility.

Different System Architectures:

- Compatibility issues with varying architectures (e.g., x86, ARM).
- Performance optimization becomes a complex task across architectures.

Dependency Management Issues:

- Complexity in managing platform-specific dependencies.
- Risk of 'dependency hell' in multi-platform scenarios.

Increased Testing Complexity:

- Need for extensive testing across all platforms.
- Ensuring consistency and reliability becomes more challenging.

User Interface Consistency:

- Maintaining UI/UX consistency across different platforms is challenging.
- Ensuring a unified user experience is crucial for user satisfaction.

Diverse Tools and Technologies:

- Wide range of tools and technologies for cross-platform deployment.
- Challenge in selecting the most effective and compatible tools.

# Cross-Platform Tools and Strategies

Cross-compilers:

- Use of cross-compilers to generate executable code for multiple platforms
- Example: GCC (GNU Compiler Collection) for various OS and hardware combination

Platform agnostic languages

- Languages that run on multiple platforms without modification. Java (with JVM), Python, JavaScript (with Node.JS), ...

Continuous Integration and Deployment:

- Automate building and testing across platforms
- CD/CI tools like Jenkins, Travis CI, GitLab CI support multi-platform workflows

Cloud-based Dev. Environments

- Leverage cloud services for uniform development and deployment envs.
- Example: AWS, Azure, Google Cloud Platform

# Program cross-platform mobile apps

- Use Web-based apps
- Use Xamarin (C#)
  - Android: Java, Kotlin, C++
  - iOS: Objective-C, Swift
- Use Flutter (Dart)


React native, based on React Javascript Framework, published by Meta in 2015

| | Flutter | Xamarin |
|---|---|---|
| Owner | Google | Microsoft |
| First Introduced In | 2017 | 2011 |
| Performance | Native-like | Not as great as Flutter's |
| Heavy Load (for eg., graphics and animation, compared to native apps) | Moderate | Poor |
| Cross-platform features | Simple and common codebase for multiple platforms | Custom controls and renderers need to implement same advanced UI components on multiple platforms |
| Popularity | Quite High | Considerably lower than Flutter's |
| Development Language | Dart | .NET Languages like C#, F#, etc. |
| Learning Curve | Easy to learn | Easy to learn |
| Component Support | Wide range of components, plugins available. Integration is easy. | Good number of components are available but integration is not super smooth. |
| Community | Huge and growing | Good-sized but slowly reducing |
| Pricing | Free, open source | Free, open source |

# Examples

Netflix

- Server-side: Java and Python, ensuring compatibility across different platforms
- Using containers with Docker for consistent deployment

Microsoft Office Suite

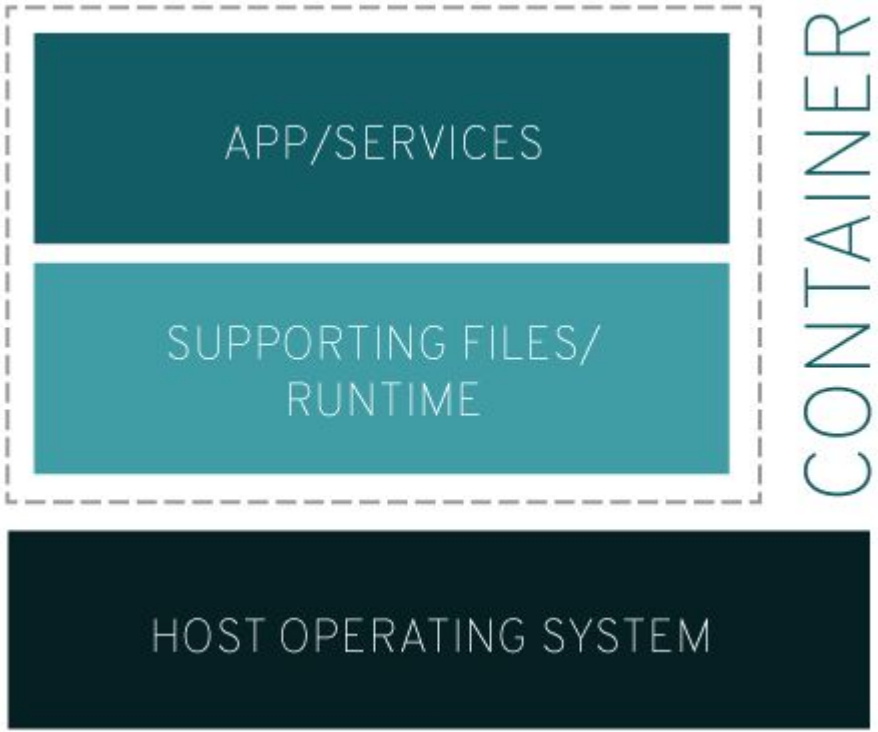
- Developed with unified codebase for Windows, macOS, iOS, Android
- Utilizes platform specific adaptations for optimal performance and user experience

Games

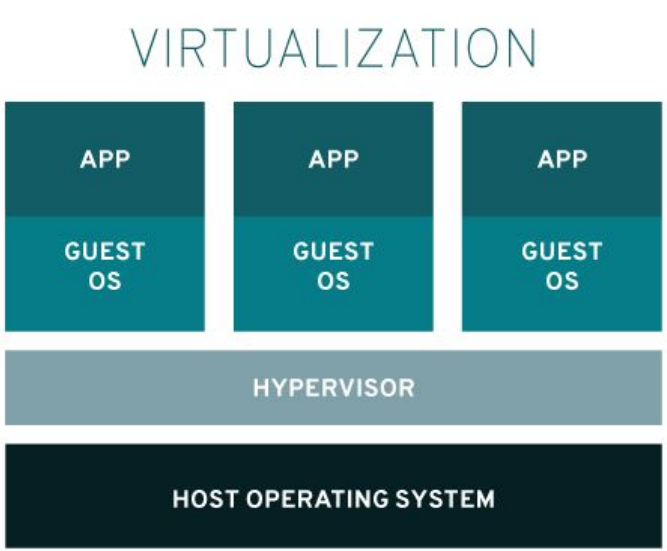
- Unity or Unreal Engine target multiple devices like PC, consoles, mobile devices
- Focus on consistent performance and user experience across hardware

# Shipping the whole box - Containers

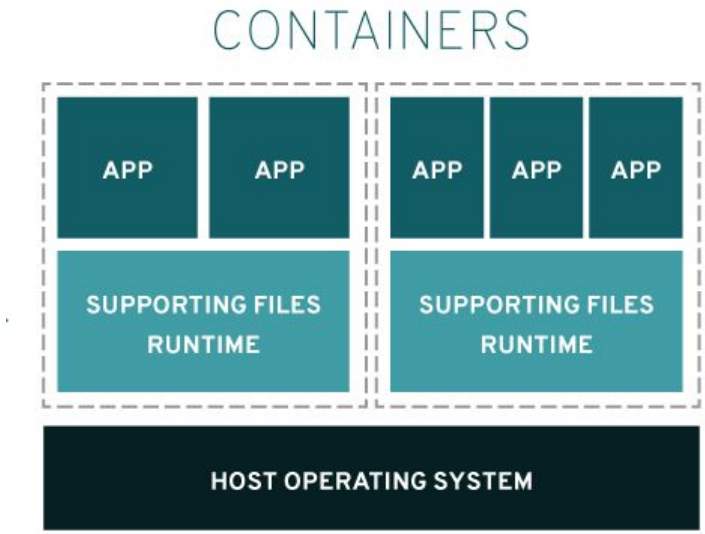
# Running code out of the box - container!



# Virtualization vs. container




**Hypervisor emulates hardware**, which allows multiple operating systems to run side by side

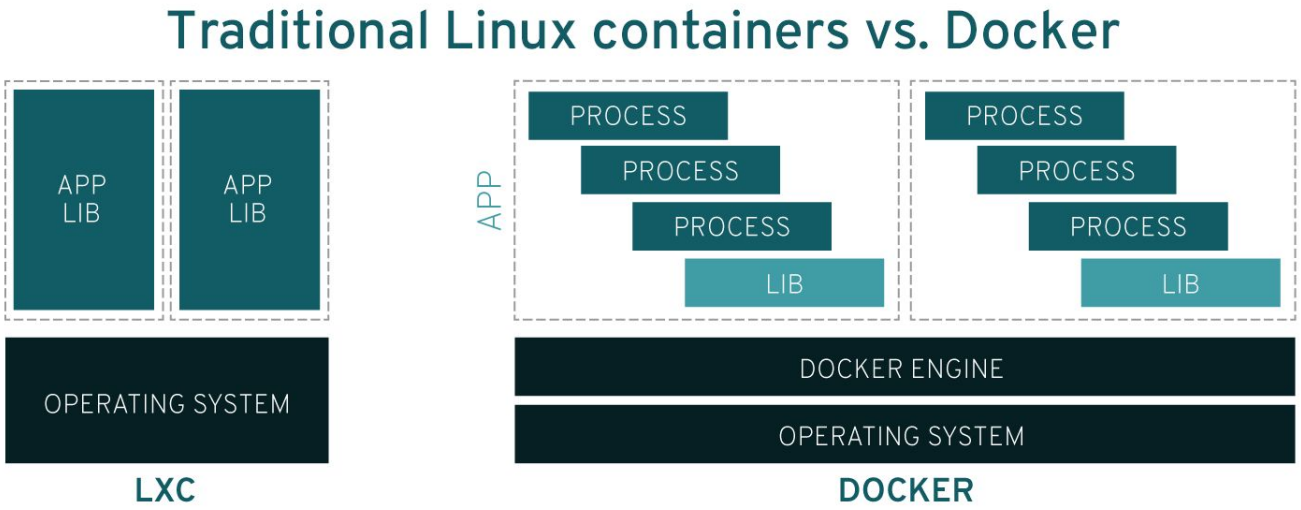



Containers are

- less resource-intensive
- Have a standard interface (start, stop, environment variables, etc.)
- retain application isolation
- are more easily managed as part of a larger application (multiple containers)

# Docker vs LXC

# LXD – Virtual Machine



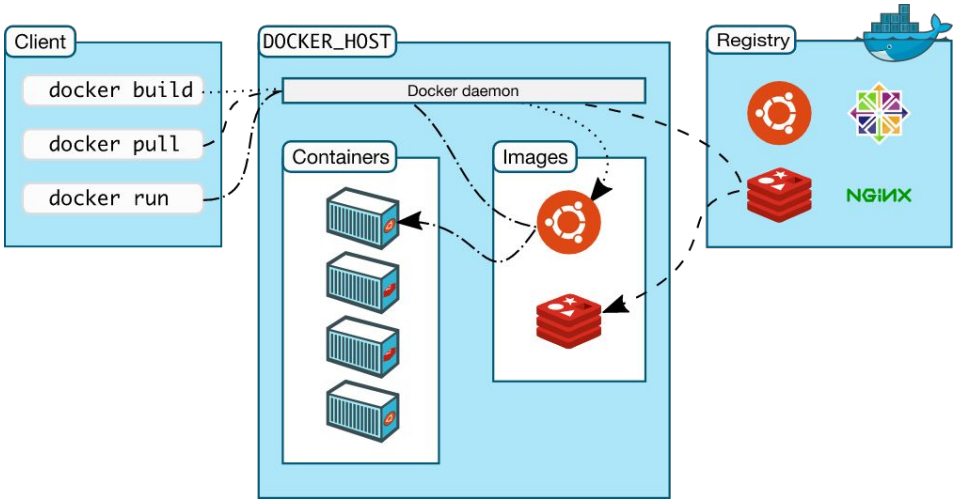

LXC is shipped with a full Linux environment: file system, networking and multiple applications. Docker relied on LXC, but now it has a Docker Engine being more lightweight.
Further, Docker promotes processes (broke down from Apps) instead of full Apps

# Docker and Kubernetes

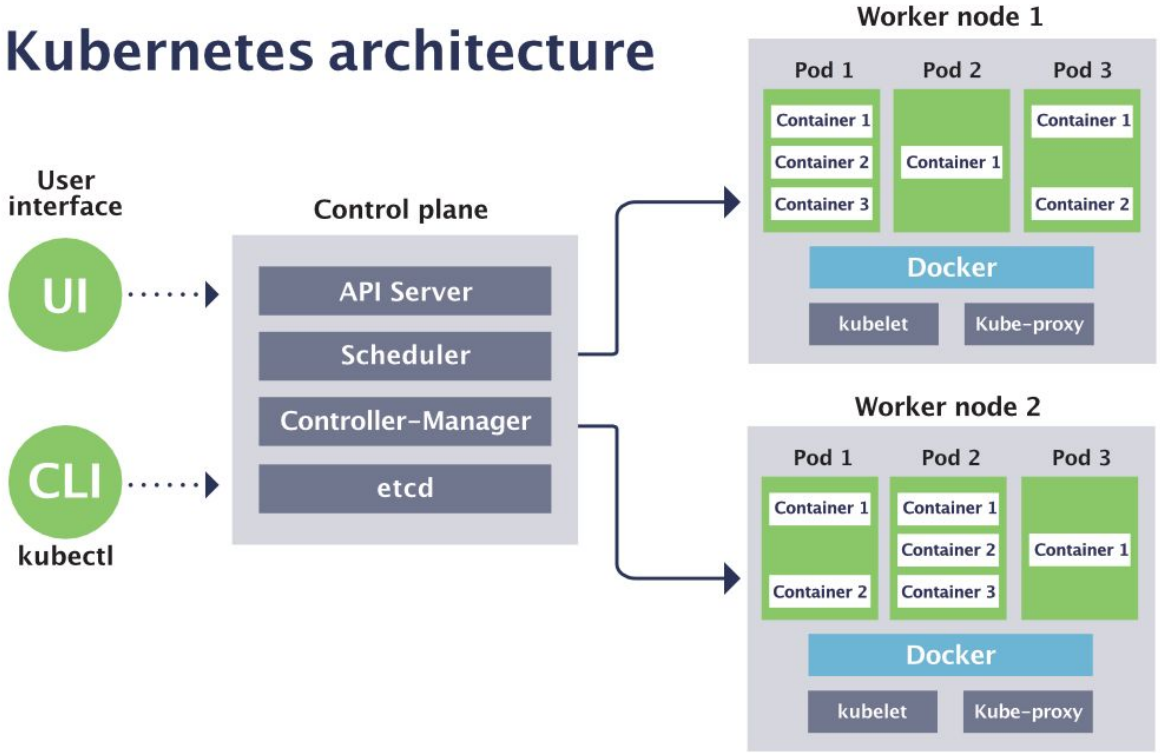
Docker uses CONTAINERS:

To summarize, a container:

- is a runnable instance of an image. You can create, start, stop, move, or delete a container using the DockerAPI or CLI.
- can be run on local machines, virtual machines or deployed to the cloud.
- is portable (can be run on any OS)
- Containers are isolated from each other and run their own software, binaries, and configurations.



# Kubernetes

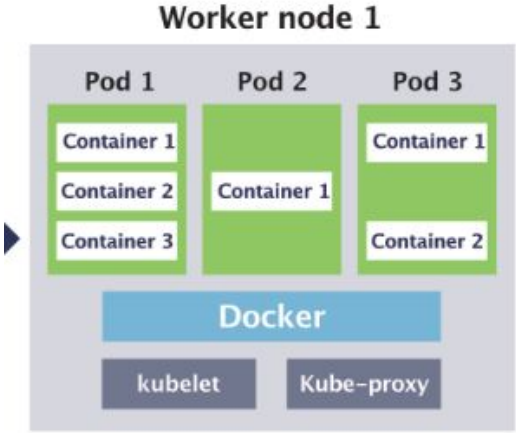



https://sensu.io/blog/how-kubernetes-works

# Kubernetes Pods

A Kubernetes pod is a group of containers and is the smallest unit that Kubernetes administers.

- Pods have a single IP address
  → applied to every container within the pod
- Containers in a pod share the same resources such as memory and storage. → containers inside a pod are treated collectively as a single application, as if all the containerized processes were running together on the same host in more traditional workloads.
- A pod can have a single container and more. Only one container is common, when the application or service is a single process that needs to run.




https://sensu.io/blog/how-kubernetes-works

# CI/CD

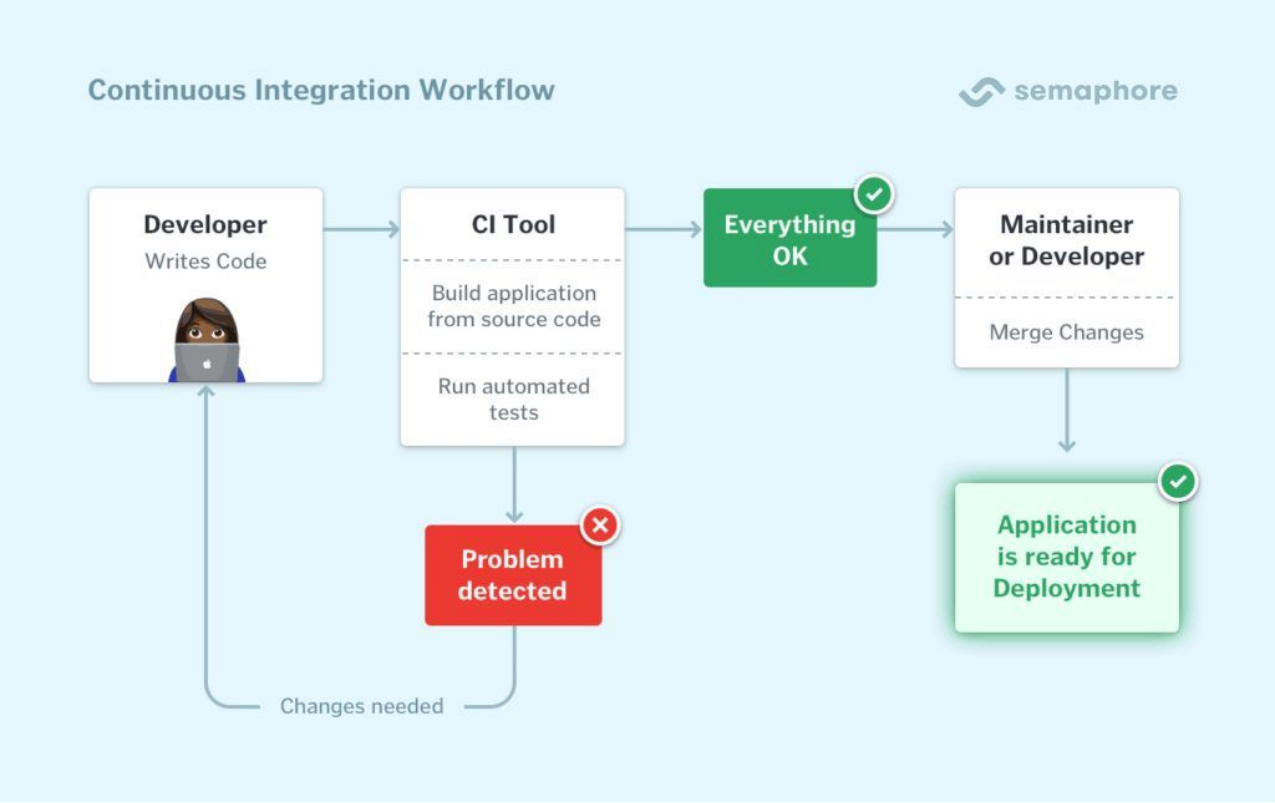
# Understanding Continuous Integration (CI)

Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day.

Each integration is verified by an **automated build and automated tests** to detect integration errors as quickly as possible.

- Maintain a single source of truth for the codebase
- Facilitate early detection of integration issues and conflicts
- Ensure that the software is in a deployable state **at all times**

# CI process




Tools:

- Jenkins
- Travis CI
- CircleCI
- GitLab CI

Travis CI

$64
per month

Starter
Ideal for hobby projects

# CI Workflow

CI Workflow Steps:

- Code Commit: Developers regularly push code to a shared repository.
- Automated Build: The CI server automatically builds the system and runs unit and integration tests.
- Test Results: If tests pass, the build is considered stable. If they fail, the team is alerted to fix issues immediately.

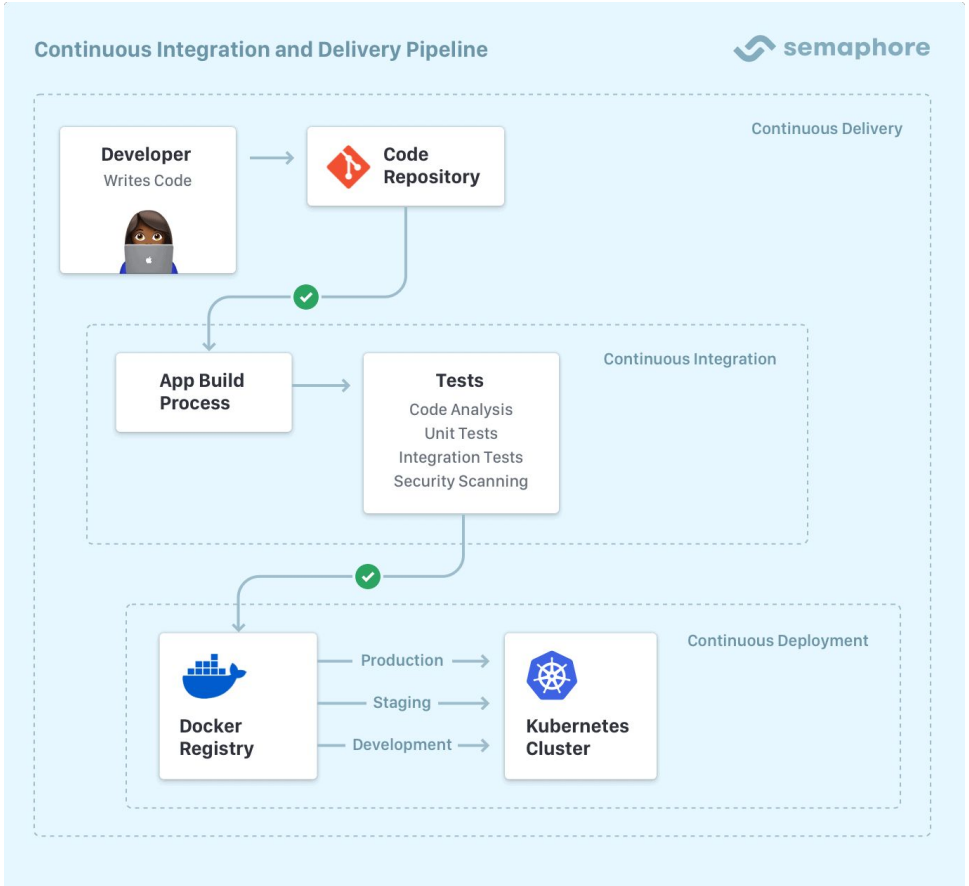
Automated Testing in CI:

- Integral part of CI to ensure code integrity after each commit.
- Types of tests: Unit, Integration, Functional, and Performance Tests.

CI Server Role:

- Monitors the repository and triggers a build upon new code commits.
- Manages the process of building, testing, and reporting.

# Continuous Deployment




Continuous Deployment is an advanced practice where every change that passes all stages of the production pipeline is released to customers automatically, **without explicit approval**.

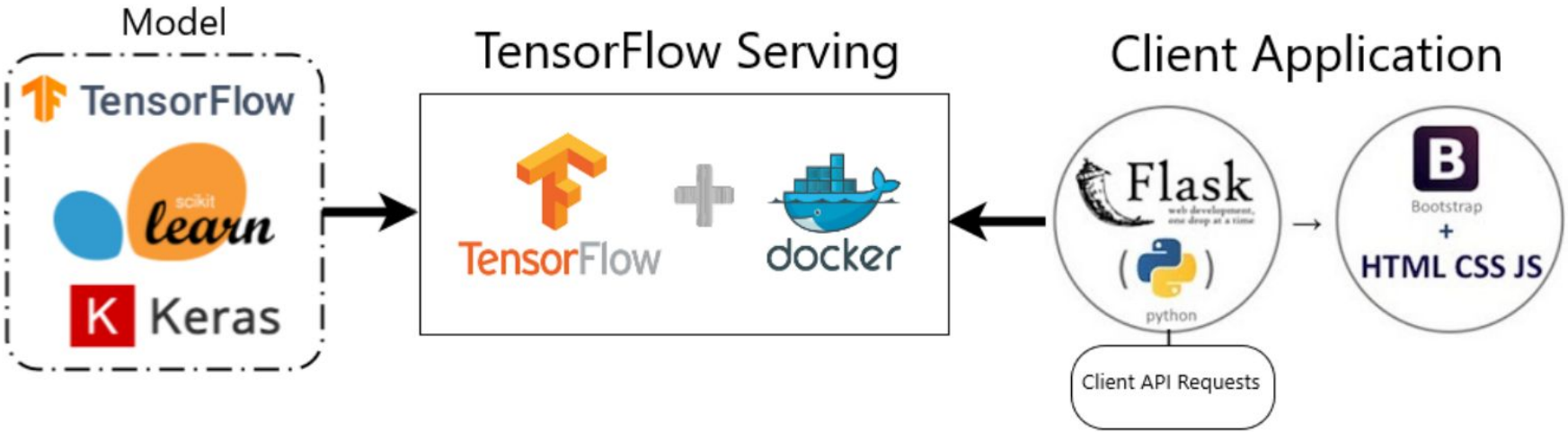
# CI vs CD

Goals of CD: To automate the software release process and enable rapid, reliable, and frequent deployments and to reduce the time taken to bring new features, fixes, and updates to the end-users.

CD vs CI: While CI focuses on integrating and testing code changes, CD automates the next steps of **deploying changes to production environments. ⇒ CD ensures a fully automated pipeline from code commit to deployment.**

# Deploy AI models

# Deploying TensorFlow models



# TFLITE

## How it works

**Pick a model**

Pick a new model or retrain an existing one.

Read the developer guide →

**Convert**

Convert a TensorFlow model into a compressed flat buffer with the TensorFlow Lite Converter.

Read the developer guide →

**Deploy**

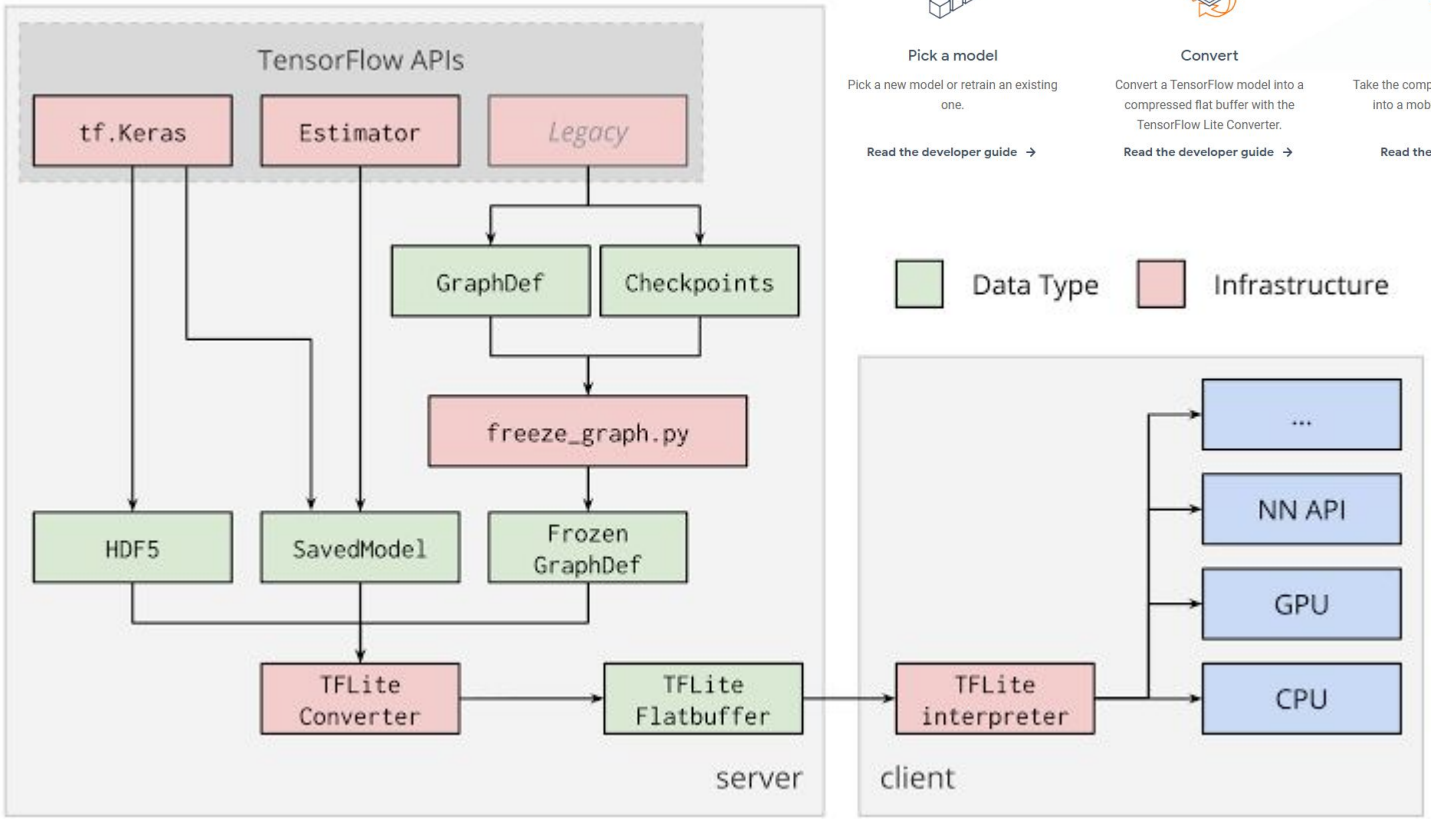
Take the compressed .tflite file and load it into a mobile or embedded device.

Read the developer guide →

**Optimize**

Quantize by converting 32-bit floats to more efficient 8-bit integers or run on GPU.

Read the developer guide →



# TFLITE on µCs

## TensorFlow Lite for Microcontrollers

TensorFlow Lite for Microcontrollers is designed to run machine learning models on microcontrollers and other devices with only few kilobytes of memory. The core runtime just fits in 16 KB on an Arm Cortex M3 and can run many basic models. It doesn't require operating system support, any standard C or C++ libraries, or dynamic memory allocation.

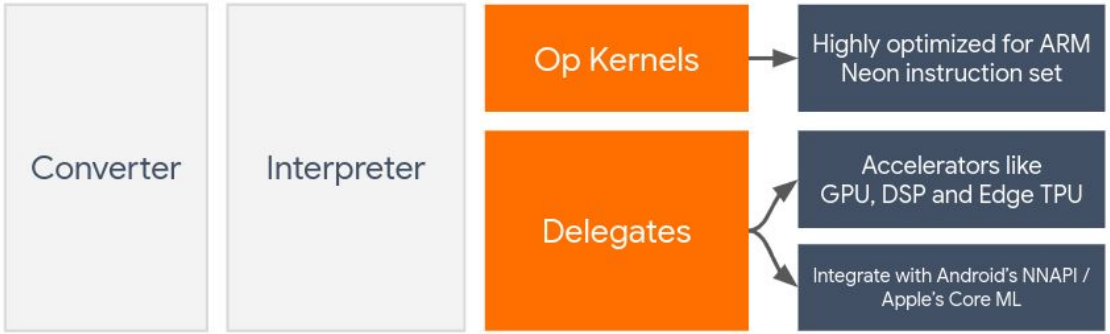
The following development boards are supported:

- Arduino Nano 33 BLE Sense
- SparkFun Edge
- STM32F746 Discovery kit
- Adafruit EdgeBadge
- Adafruit TensorFlow Lite for Microcontrollers Kit
- Adafruit Circuit Playground Bluefruit
- Espressif ESP32-DevKitC
- Espressif ESP-EYE
- Wio Terminal: ATSAMD51
- Himax WE-I Plus EVB Endpoint AI Development Board
- Synopsys DesignWare ARC EM Software Development Platform
- Sony Spresense

- Very limited memory
- Very limited throughput
- Limited operations
- Not easy to get to work

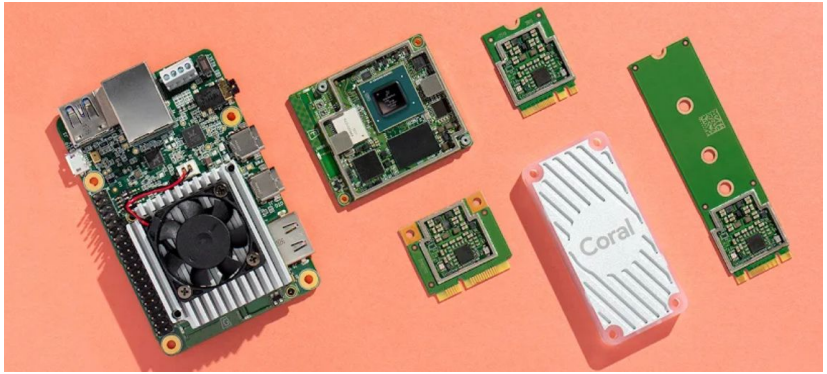
Fun stuff: https://experiments.withgoogle.com/collection/tfliteformicrocontrollers

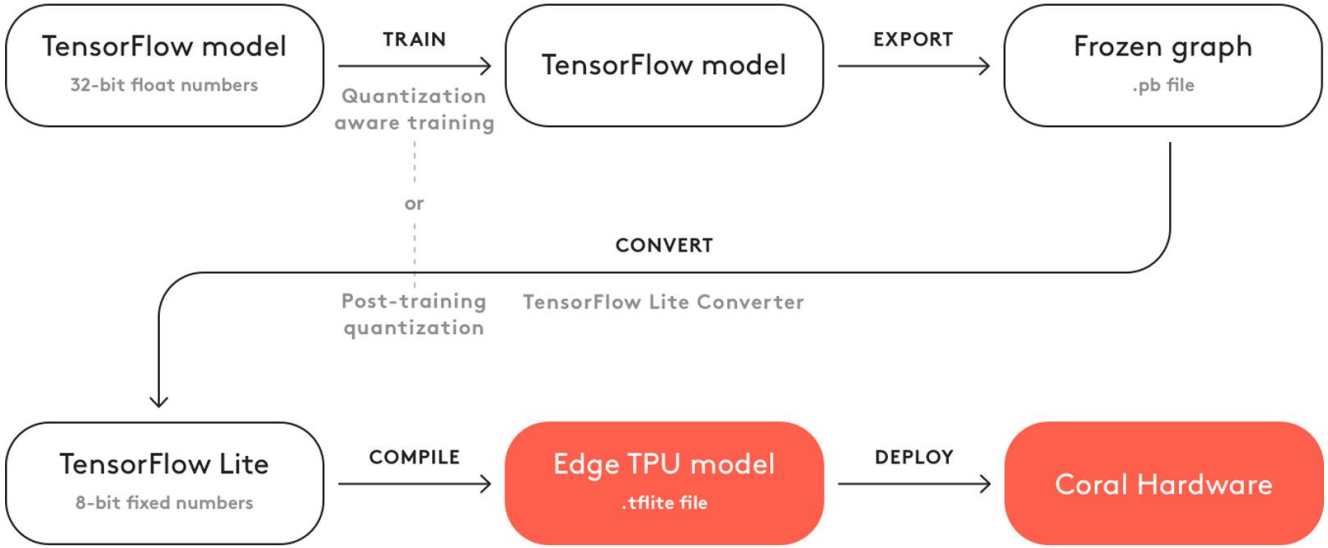
# TFLITE on Delegates (e.g. EdgeTPU)




| Model Type | GPU | NNAPI | Hexagon | CoreML |
|---|---|---|---|---|
| | | Android | | iOS |
| Floating-point (32 bit) | Yes | Yes | No | Yes |
| Post-training float16 quantization | Yes | No | No | Yes |
| Post-training dynamic range quantization | Yes | Yes | No | No |
| Post-training integer quantization | Yes | Yes | Yes | No |
| Quantization-aware training | Yes | Yes | Yes | No |

# Edge TPU


Coral


TensorFlow model
32-bit float numbers
TRAIN
Quantization aware training
TensorFlow model
EXPORT
Frozen graph
.pb file
or
CONVERT
Post-training quantization
TensorFlow Lite Converter
TensorFlow Lite
8-bit fixed numbers
COMPILE
Edge TPU model
.tflite file
DEPLOY
Coral Hardware

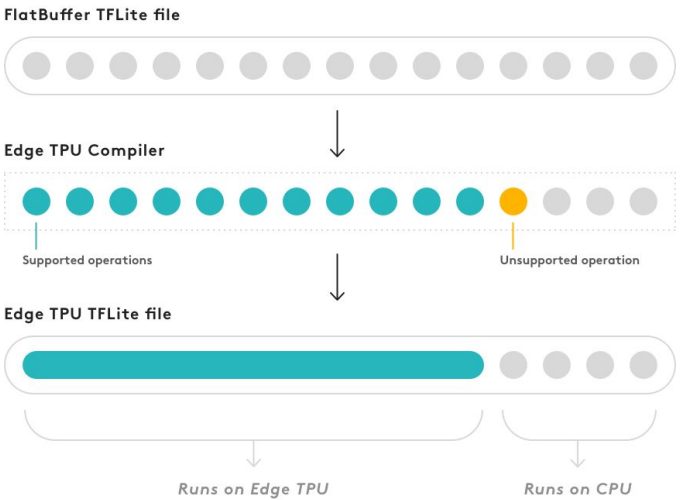
# Edge TPU

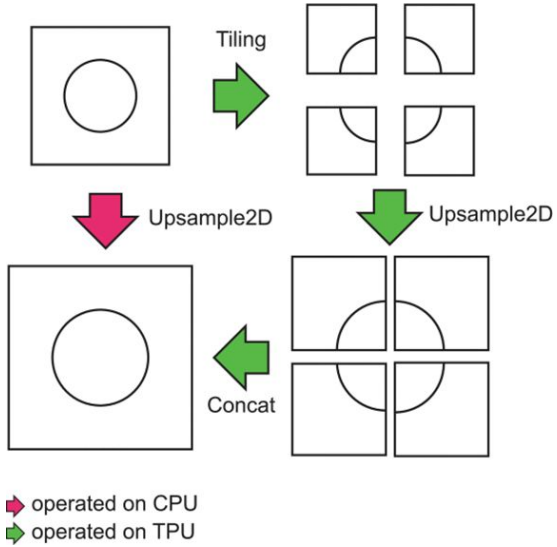
## Model requirements

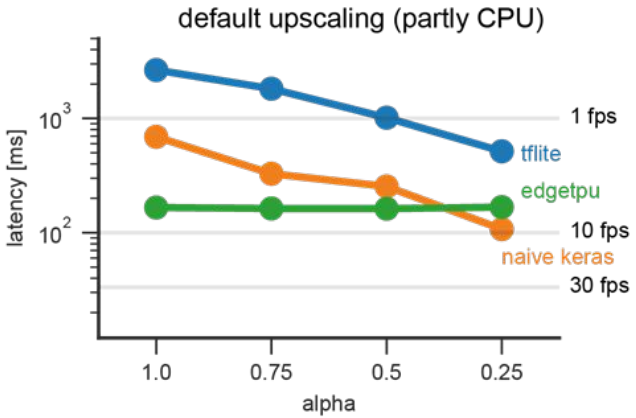
If you want to build a TensorFlow model that takes full advantage of the Edge TPU for accelerated inferencing, the model must meet these basic requirements:

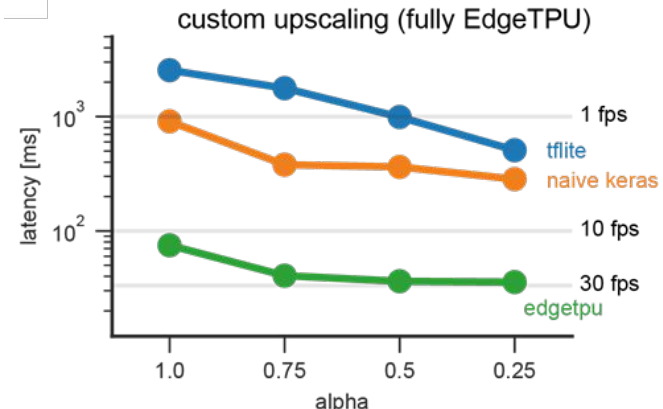
- **Tensor parameters are quantized** (8-bit fixed-point numbers; int8 or uint8).
- **Tensor sizes are constant at compile-time** (no dynamic sizes).
- **Model parameters (such as bias tensors) are constant at compile-time.**
- **Tensors are either 1-, 2-, or 3-dimensional.** If a tensor has more than 3 dimensions, then only the 3 innermost dimensions may have a size greater than 1.
- **The model uses only the operations supported by the Edge TPU** (see **table 1** below).

# Edge TPU




Kist and Döllinger, IEEE Access 2020