IFB104 — Building IT Systems
Topic 6 — How to Interact with the User

School of Computer Science

Semester 2, 2024

Lecturer Update

- Assoc. Prof. Laurianne Sitbon is on leave
- Dr. Madison Klarkowski to deliver remaining lectures and course content
  - Any email threads you had with Laurianne – I will be taking over
  - But please prompt me if you don’t hear from me!

Assignment 1B

Completed and Ongoing Submissions

- You should have all submitted 1B by now (for those without extensions)
- Those with active extensions: please email these to ifb104.query@qut.edu.au
- Those who received errors submitting to Gradescope: you will still be marked
  - We can still see your code, and the most recent submission will be marked
  - You won't receive feedback as quickly as your peers

Gradescope

- Thank you for your patience as we piloted Gradescope this semester!
  - First time deploying it in IFB104
- A powerful tool that allows students to receive immediate feedback – received some very positive comments about this from students
- Nonetheless, we will be returning to standard Canvas submissions for the rest of semester
  - Concerns with technical issues
  - Some of the automated feedback confusing

gradescope

Assignment 2A Sneak Peek

- **Assignment 2A details** will go live **tomorrow**
- Those of you attending/watching the lecture get a **sneak peek** 👀

Assignment 2A Sneak Peek

Assignment 2A Overview

This assignment is Part A of a two-part project focusing on building a Graphical User Interface (GUI) in Python using Tkinter. You will display movie details using dummy data (already provided in your template file). In Part B, you will enhance your GUI by integrating it with a real database and adding text-cleaning functionality.

<u>Your main goals for Part A are:</u>

- Design and implement a user-friendly GUI with Tkinter.
- Display movie titles and details from the provided dummy data.
- Structure your code modularly, anticipating easy integration with future database functionality in Part B.
    - It may be helpful to structure your code following Model-View-Controller (MVC) principles to clearly separate data handling, user interface, and control logic.

Assignment 2A Sneak Peek

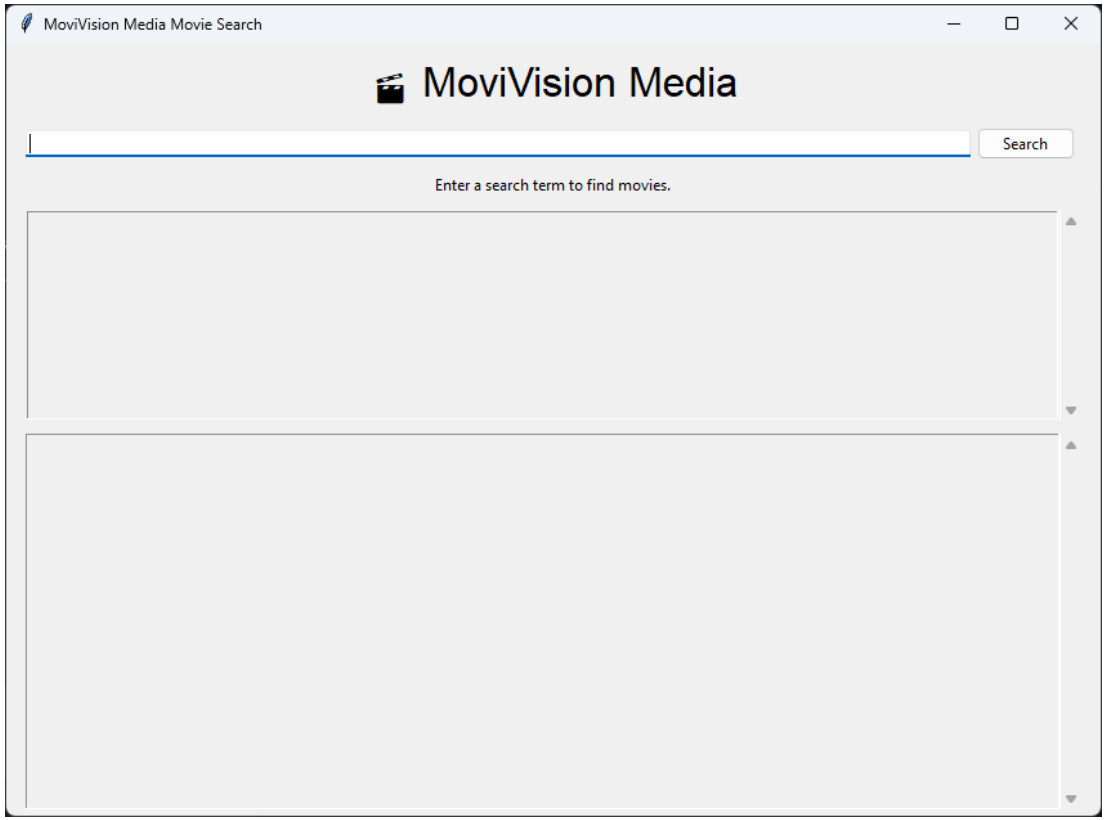
Assignment 2A Scenario

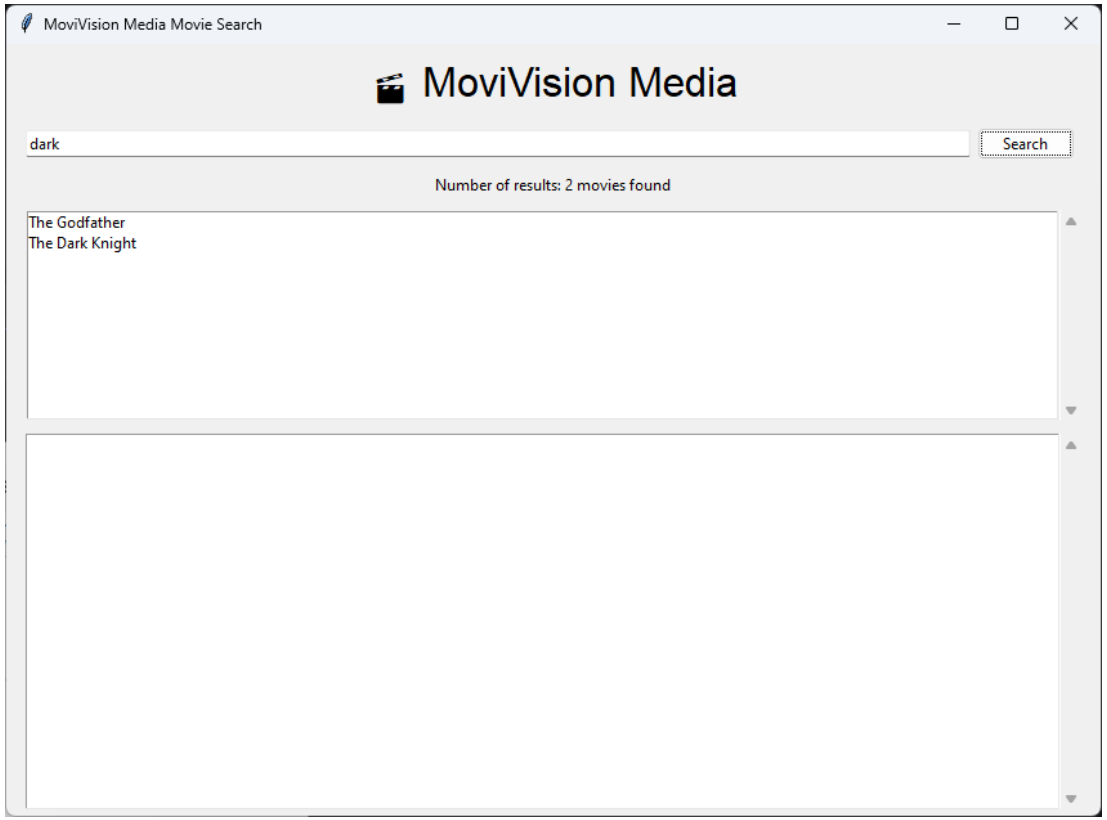
MoviVision Media's content writers regularly create articles and reviews based on movie information gathered from multiple third-party websites using automated web scrapers. Unfortunately, this automated scraping process often introduces unwanted HTML tags, special characters, and inconsistent formatting into the data, making it difficult for writers to read and use effectively.

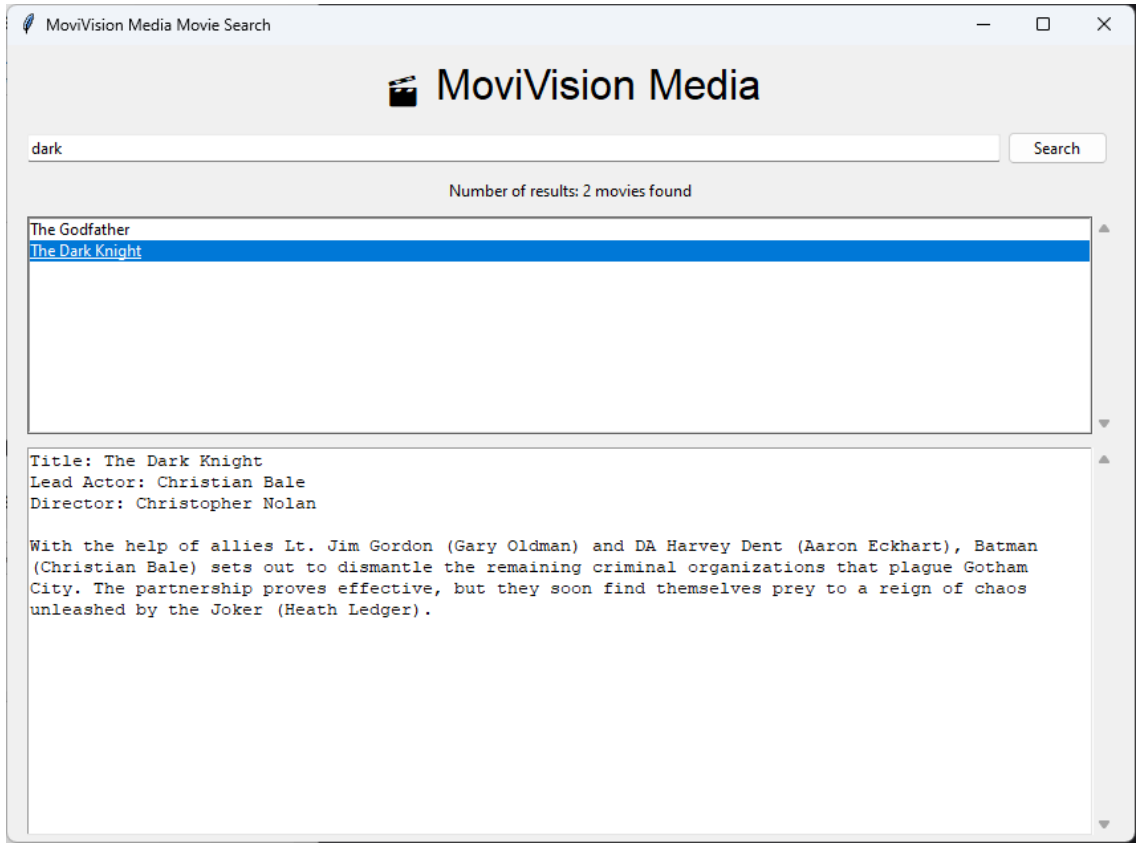
To address this issue, MoviVision Media requires a desktop application that clearly displays cleaned movie details, enabling writers to quickly review accurate movie information. While actual data cleaning and HTML tag removal will be handled in Part B, your GUI design in Part A must anticipate these future requirements.

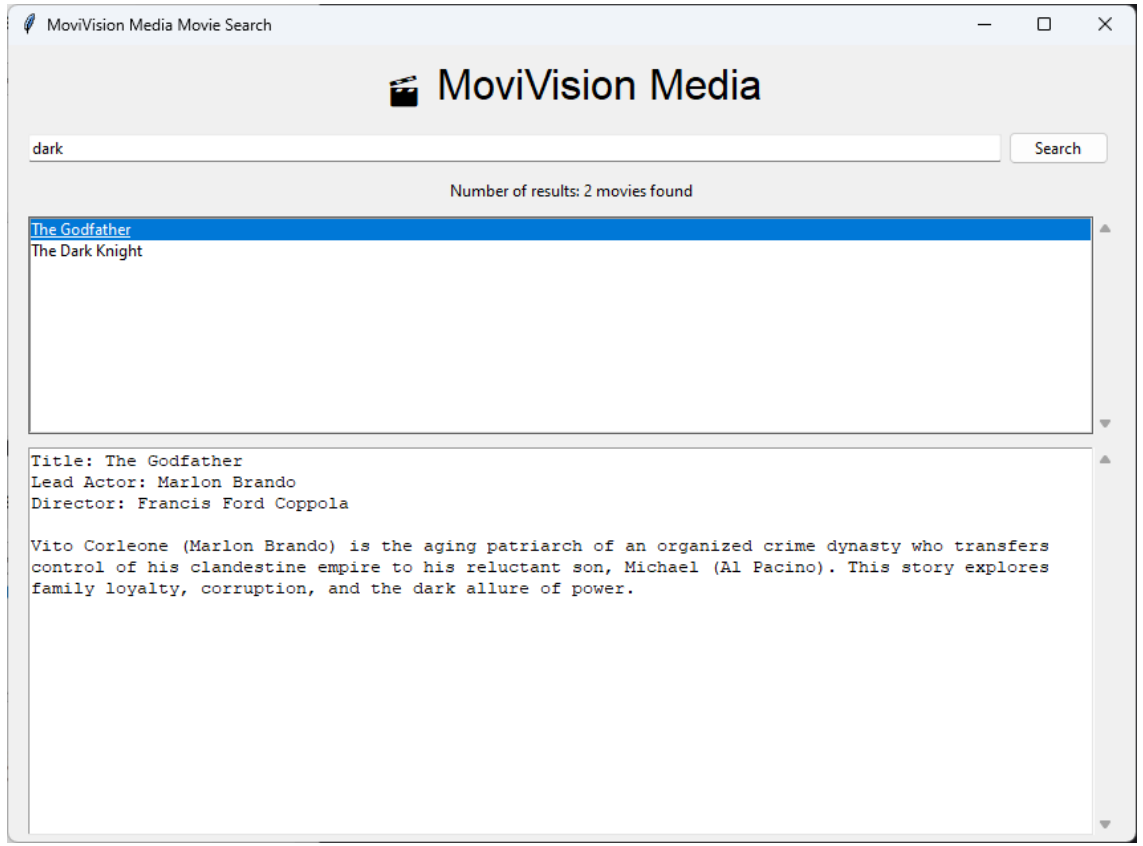
For Part A:

- No database interactions are required.
- Dummy data provided in your template file is clean and ready to display.
- Emphasis is on creating a clear and effective GUI.


MoviVision Media Movie Search
MoviVision Media
Search
Enter a search term to find movies.


MoviVision Media Movie Search
MoviVision Media
dark
Search
Number of results: 2 movies found
The Godfather
The Dark Knight


MoviVision Media Movie Search
MoviVision Media
dark
Search
Number of results: 2 movies found
The Godfather
The Dark Knight
Title: The Dark Knight
Lead Actor: Christian Bale
Director: Christopher Nolan
With the help of allies Lt. Jim Gordon (Gary Oldman) and DA Harvey Dent (Aaron Eckhart), Batman (Christian Bale) sets out to dismantle the remaining criminal organizations that plague Gotham City. The partnership proves effective, but they soon find themselves prey to a reign of chaos unleashed by the Joker (Heath Ledger).


MoviVision Media Movie Search
MoviVision Media
dark
Search
Number of results: 2 movies found
The Godfather
The Dark Knight
Title: The Godfather
Lead Actor: Marlon Brando
Director: Francis Ford Coppola
Vito Corleone (Marlon Brando) is the aging patriarch of an organized crime dynasty who transfers control of his clandestine empire to his reluctant son, Michael (Al Pacino). This story explores family loyalty, corruption, and the dark allure of power.

Assignment 2A Sneak Peek

- Full details (including Task Breakdown, CRA, and video) to be released tomorrow
- Useful to start thinking about this assignment during the lecture
- Workshops this week will cover Tkinter

Need help?

- *STIMulate* Peer Learning Facilitator consultation sessions
  - On-campus, Monday to Friday, 10am to 3pm – see the drop-in timetable at https://stimulate.qut.edu.au/ (choose “IFB104” as the unit)
  - Online – submit a request for peer support via https://qut.to/q826v

STIMulate

Facilitating real success in maths, science and IT across QUT

Aims of This Week's Lecture

- Python **modules**
- **Graphical User Interfaces** (GUIs)
  - Using Tkinter in Python
- Start thinking about **Human-Computer Interaction**
  - How to design good interfaces
- A quick overview of **Object-Oriented Programming**

- **Bonus content:** Model-View-Controller

Modules

- So far we have been looking at ‘core’ features of the Python language
- To extend the language’s capabilities we can import additional *modules* containing extra functions
  - Before using an external module we must *import* the functions we want to use
- A module that helps us create simple game-like programs is the `random` module for creating random numbers and choosing random values from lists
  - See the *Python Standard Library* manual under *Generate Pseudo-Random Numbers* for more detail

```
>>> # import two functions from the random module
>>> from random import randint, choice
>>> randint(3, 9) # choose a no. between 3 and 9 (incl.)
3
>>> randint(3, 9) # choose another
5
>>> randint(3, 9) # and another
9
>>> options = ['a', 'b', 'c'] # create a list
>>> choice(options) # choose a random value from the list
'c'
>>> choice(options) # choose another
'b'
>>> choice(options) # and another
'b'
>>>
```


Demonstration file: `random_numbers.py`

Modules

- In Python a *module* is any file containing one or more function definitions
- Given a file *m*`.py` containing function definitions *f*, *g*, *h*, etc we can import all or part of module *m* into our program as desired:

```
from m import f, h  # imports just specific functions f and h
from m import *  # imports all functions from module m
```

User Interfaces

- So far we've done simple text input and output in IDLE's shell window
- Text interfaces can already achieve a lot and have the advantage of simplicity
  - Command line execution
  - Chatbots (ChatGPT, Ms Copilot or Llama are all text user interfaces)
  - Text-based games.

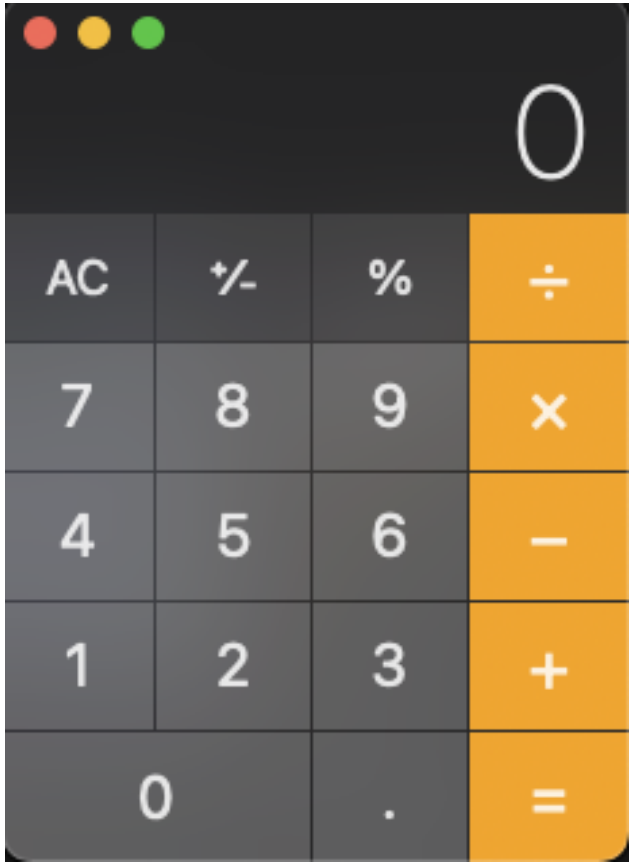
```
Opening the brown sack reveals:
A clove of garlic.
A lunch.

> eat the garlic

Thank you very much. It really hit the spot.
```

```python
# Get some input from the user
response = input('Please enter an expression: ')
# Echo it
print("You typed '" + response + "'")
# Display its value
print('Your expression equals', eval(response))
```

```
Please enter an expression: 6 * 7.2
You typed '6 * 7.2'
Your expression equals 43.2
```

User Interfaces

But we haven’t seen how to create our own user interfaces - with buttons to click or forms to fill!



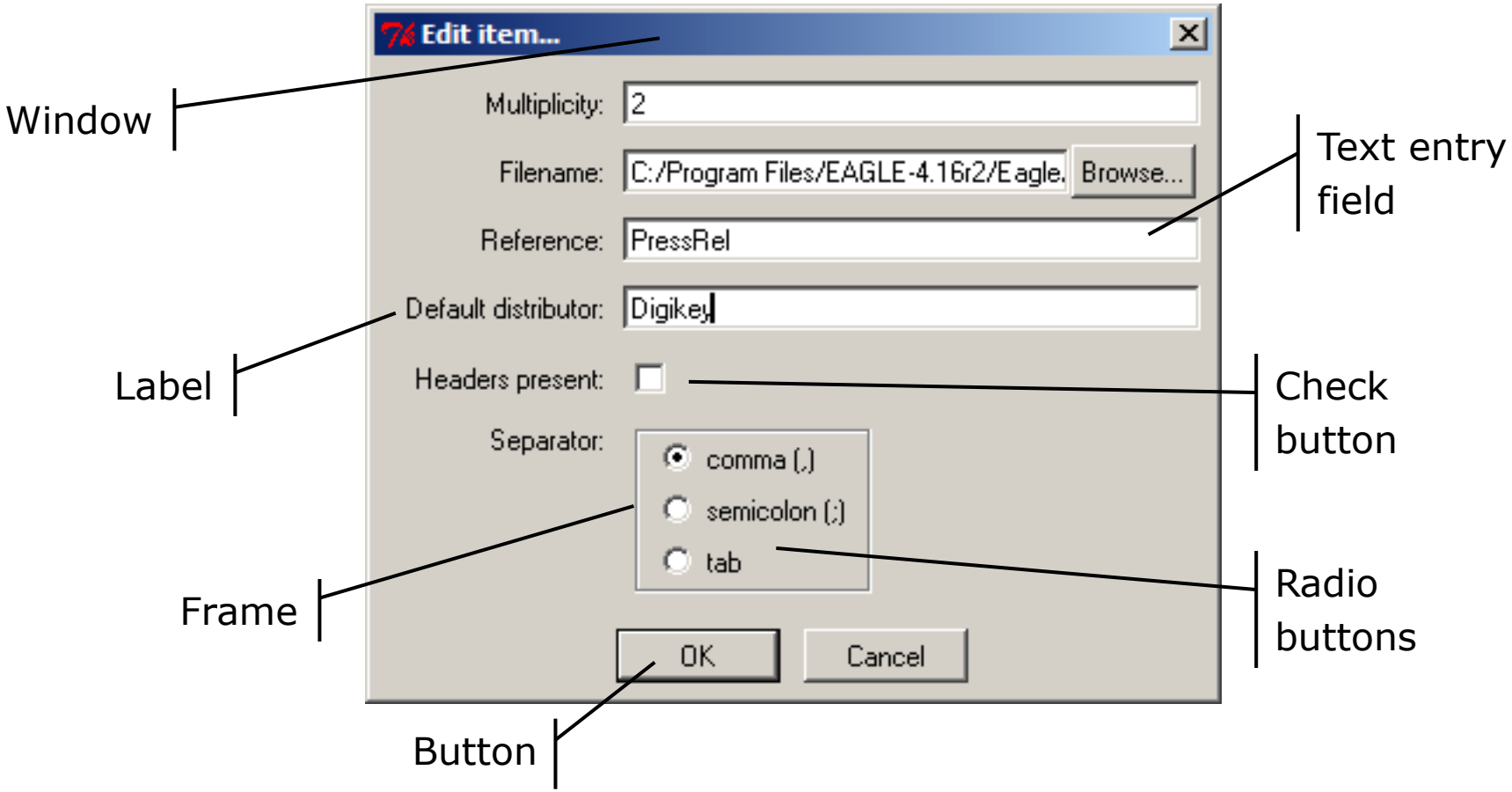
Tkinter

- Here we create our own graphical user interfaces using Python's *Tkinter* module
- Tkinter is an Application Programming Interface for accessing the "Tk" GUI functions supported by many programming languages
  - It provides functions for creating *windows* containing interactive *widgets*
  - Tkinter programs react to *events* initiated by the user by executing *commands*

```
# Get the Tkinter functions
from tkinter import *

# Create a window
my_window = Tk()

# Create a button for the window
my_button = Button(my_window,
                   text = 'Push me')

# Pack the button into the window
my_button.pack()

# Wait for the user to do something
my_window.mainloop()
```

Demos: `pop_up_windows.py`, `fraction_calculator.py`

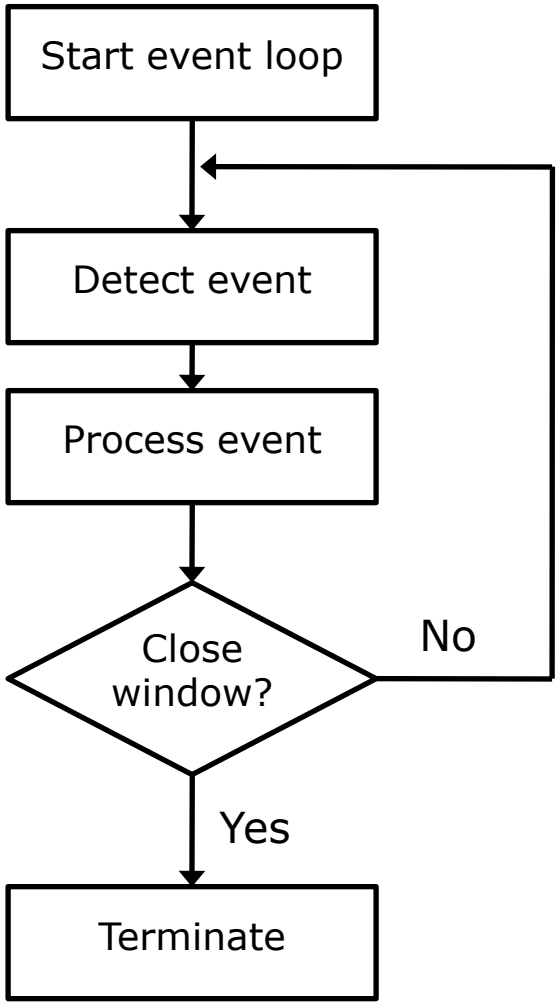
Some features of a GUI



Demo: `frustrating_button.py`

The event loop

- Unlike most of the programs we've developed so far, which run to completion when we start them, GUI programs are "reactive"
- They wait for the user to initiate some event via the interface before doing anything



Demos: `simple_button.py`, `button_with_memory.py`, `display_bits.py`

Some standard Tkinter widgets

- **Button** (to execute a command when pressed)
- **Canvas** (for drawing)
- **Checkbutton** (buttons that toggle; allows several options to be chosen simultaneously)
- **Entry** (text entry field)
- **Frame** (a widget containing other widgets)
- **Label** (to display text or an image)
- **Listbox** (list of options for the user to select)
- **Menu** (pull-down or popup menu at top of window)
- **Menubutton** (pull-down menu inside window)
- **Message** (text display with automatic wrapping)
- **Radiobuttons** (buttons that set a shared variable when pressed; allows one of several options to be chosen)
- **Text** (formatted text display)

Some standard Tkinter widget properties

- **fg** (foreground colour)
- **bg** (background colour)
- **text** (text to display)
- **command** (function to call)
- **font** (text font as name-size pair)
- **justify** (text alignment left, center or right)
- **cursor** (choose the cursor style)
- **variable** (Python variable to set)
- **value** (value of the given variable)

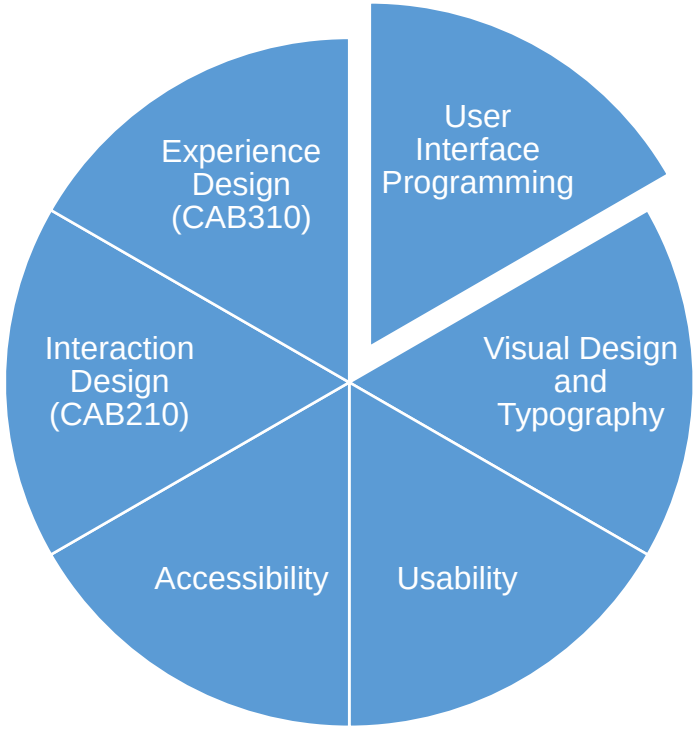
- To change a property of a widget after its creation:

  widget['*property*'] **=** *value*

Demo: `change_properties.py`, `listbox_demo.py`

User Interface Programming

- Only a single aspect of User Experience (UX)
- Visual Design and Typography is concerned with visually pleasing interfaces, e.g.
  - Harmonious positioning of elements on the screen
  - Colour scheme
- Interaction and Experience Design consider the person using the application first
  - Enquires about functionality
  - Offer different interfaces
  - Innovate



Accessibility and Usability

- Accessibility and Usability are concerned with making it possible and easy for everyone to use the interface
- Logical ordering of elements on the screen
- Compatibility with accessibility tools (text to speech applications, colour adapter, text simplification tools)
- Hardcoding text onto an image (for example a button) without providing an indication of the corresponding text in the application would prevent screen readers from telling a user what the buttons are about.

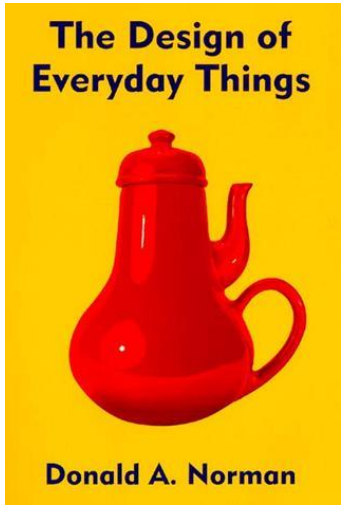
- https://www.w3.org/TR/WCAG21/

Web Content Accessibility Guidelines (WCAG) 2.1

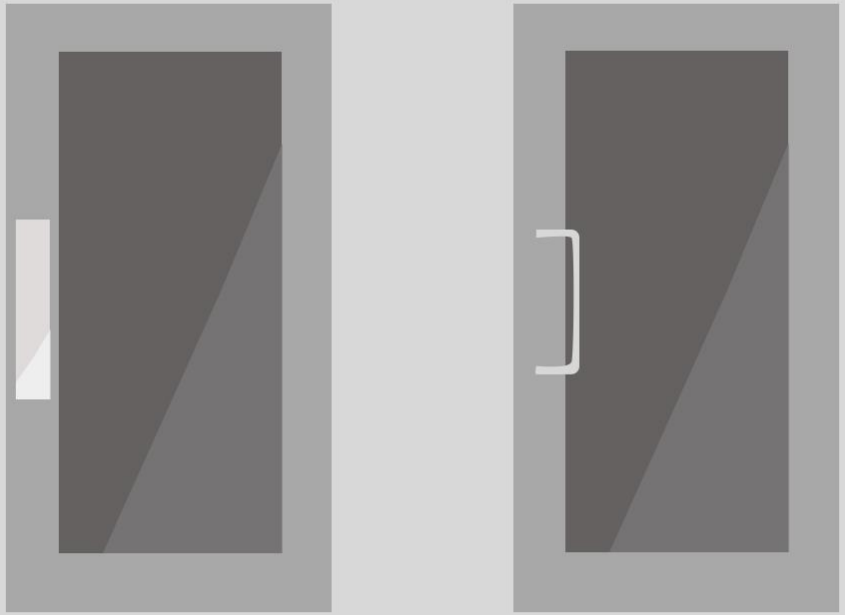
W3C Recommendation 21 September 2023

- Use libraries that enable accessibility, set accessibility parameters
  - We use Tkinter in the unit because it is easy to learn and understand…
  - but this does not create accessible interfaces!

Examples of poor Usability

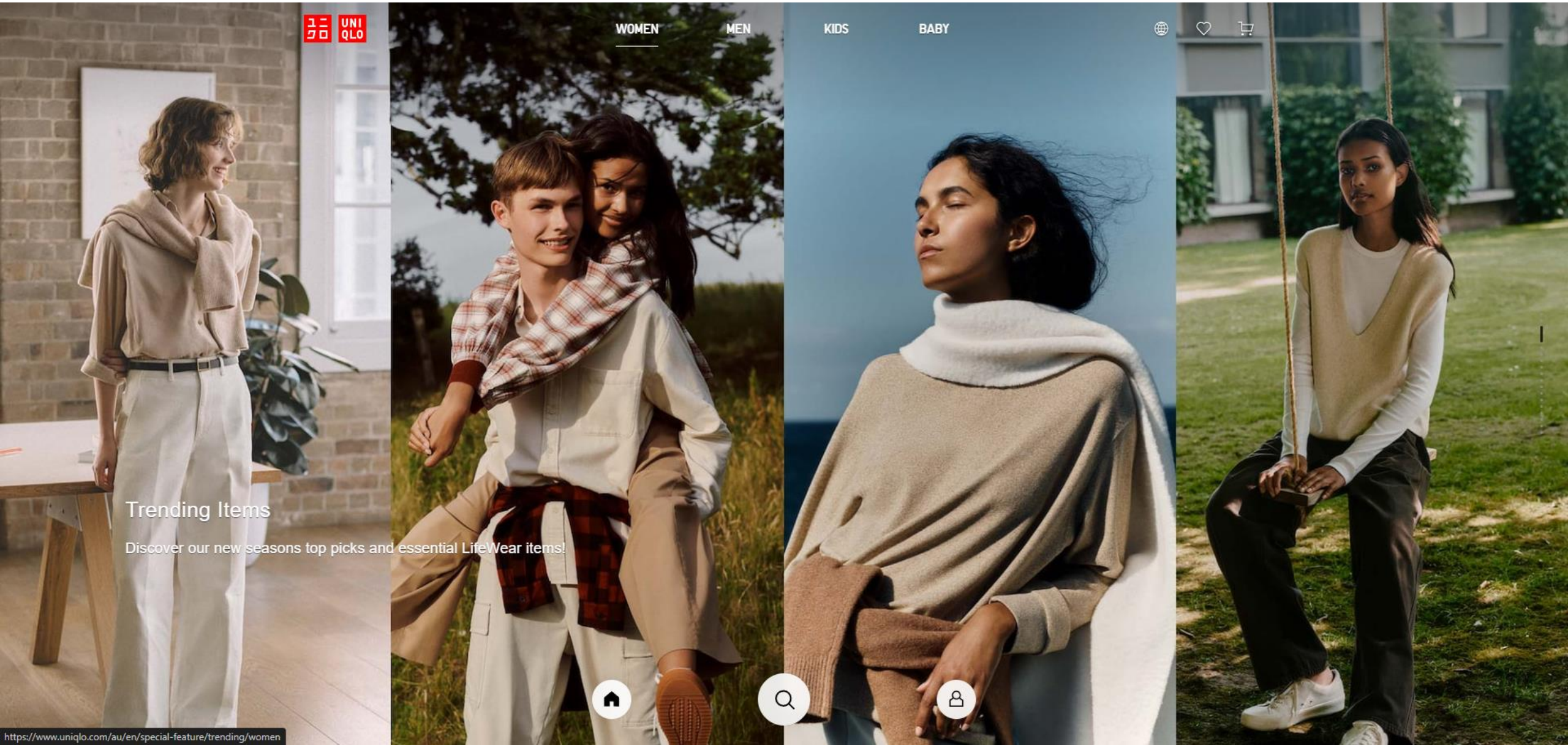


Push or Pull?

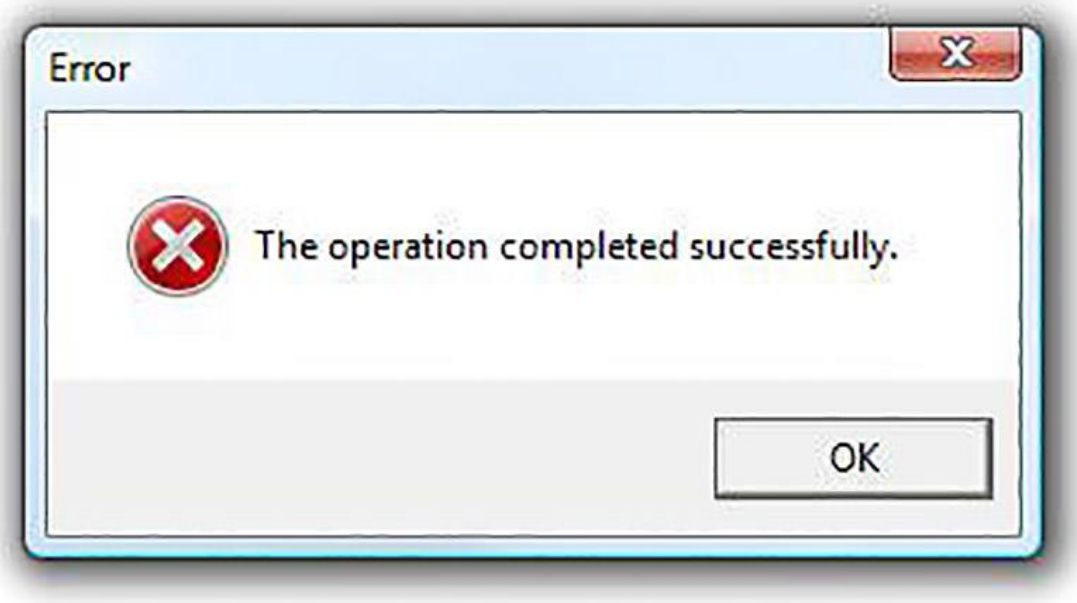


Norman's Doors

https://www.youtube.com/watch?v=yY96hTb8WgI


UNI QLO
WOMEN
MEN
KIDS
BABY
Trending Items
Discover our new seasons top picks and essential LifeWear items!
https://www.uniqlo.com/au/en/special-feature/trending/women


STOP TRYING TO
MAKE
HAPPEN
IT'S NOT GOING TO HAPPEN
memegen.link


Error
The operation completed successfully.
OK

Card Number

1111 1111 1111 1111 1

Name on Card

Smith William

Expiration Date

02/06/2022

Security Code

123

The data is incorrect. Please check your info

Pay Now

Card Number

1111 1111 1111 1111 1

Please enter a vaild card number

Name on Card

Smith William

Expiration Date

02/06/2022

Security Code

123

Pay Now

LOG IN

E-mail adress

Password

LOGIN ME

SIGN UP

FORGOT PASSWORD?

LOG IN

E-mail adress

Password

LOGIN ME

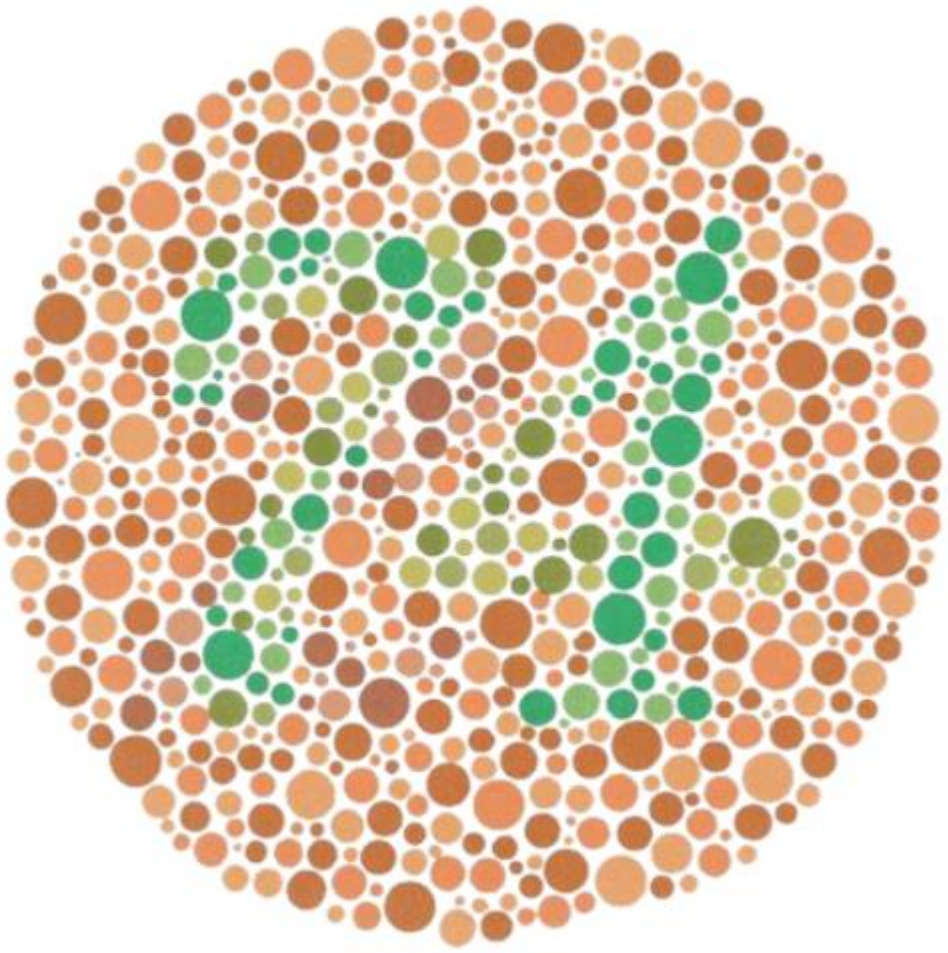
SIGN UP

Forgot Password?

@Alamoudiby

Thinking Carefully about Colour

Thinking Carefully about Colour



Bad UI Battle

Bad Phone Selector

Phone: (10) 000-000-000

Advanced Mode

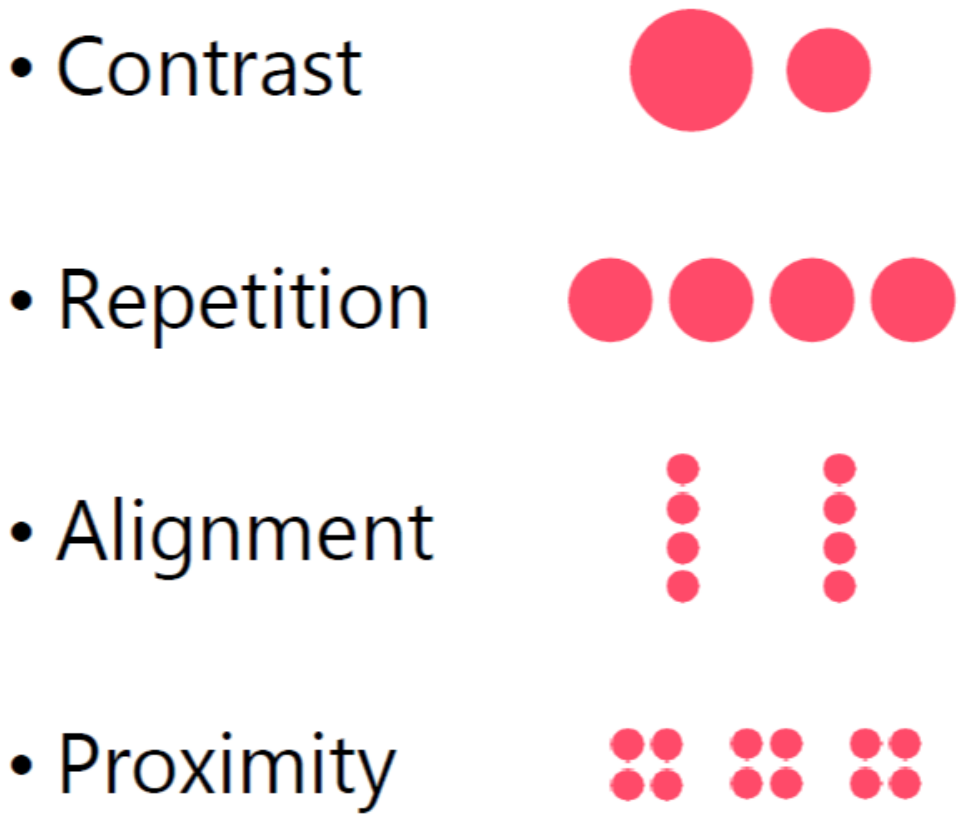
Submit

Available at: github.com/GoulartNogueira/BadUI

Made with ♥ by André G. N.

CRAP Design

- Useful framework to think about GUI design
- Contrast, Repetition, Alignment, and Proximity
  - It’s a poor choice of acronym ☺



CRAP Design - Proximity

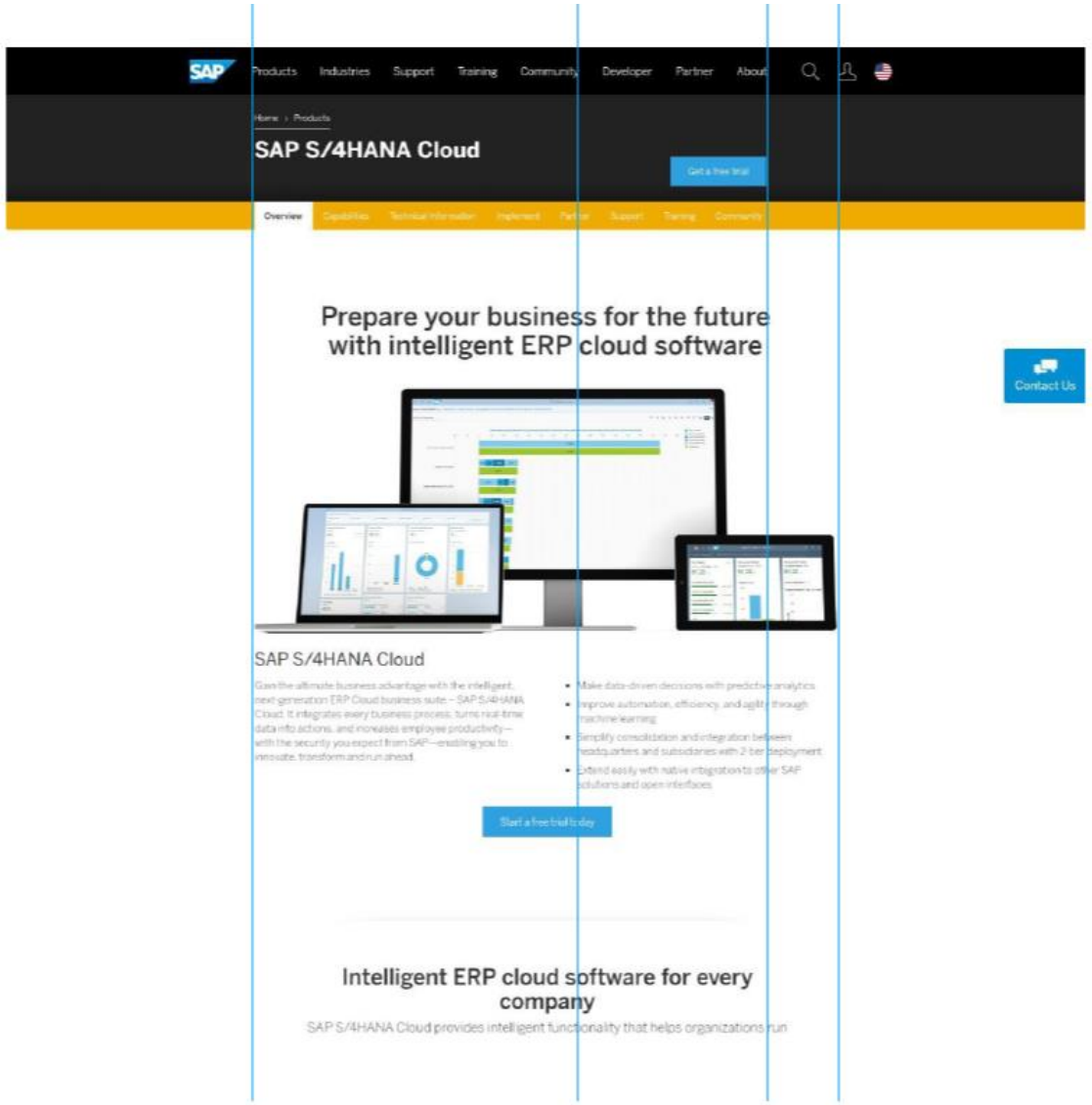
- Group related items together
- The basic purpose:
  - Organise
  - If information is organised, it is more readable and more memorable
  - Improves usability
- How to get it:
  - Squint your eyes and count the number of visual elements in an area by counting the number of times your eye stops
  - If more than 3 – 5, try to regroup some elements

Proximity - Examples

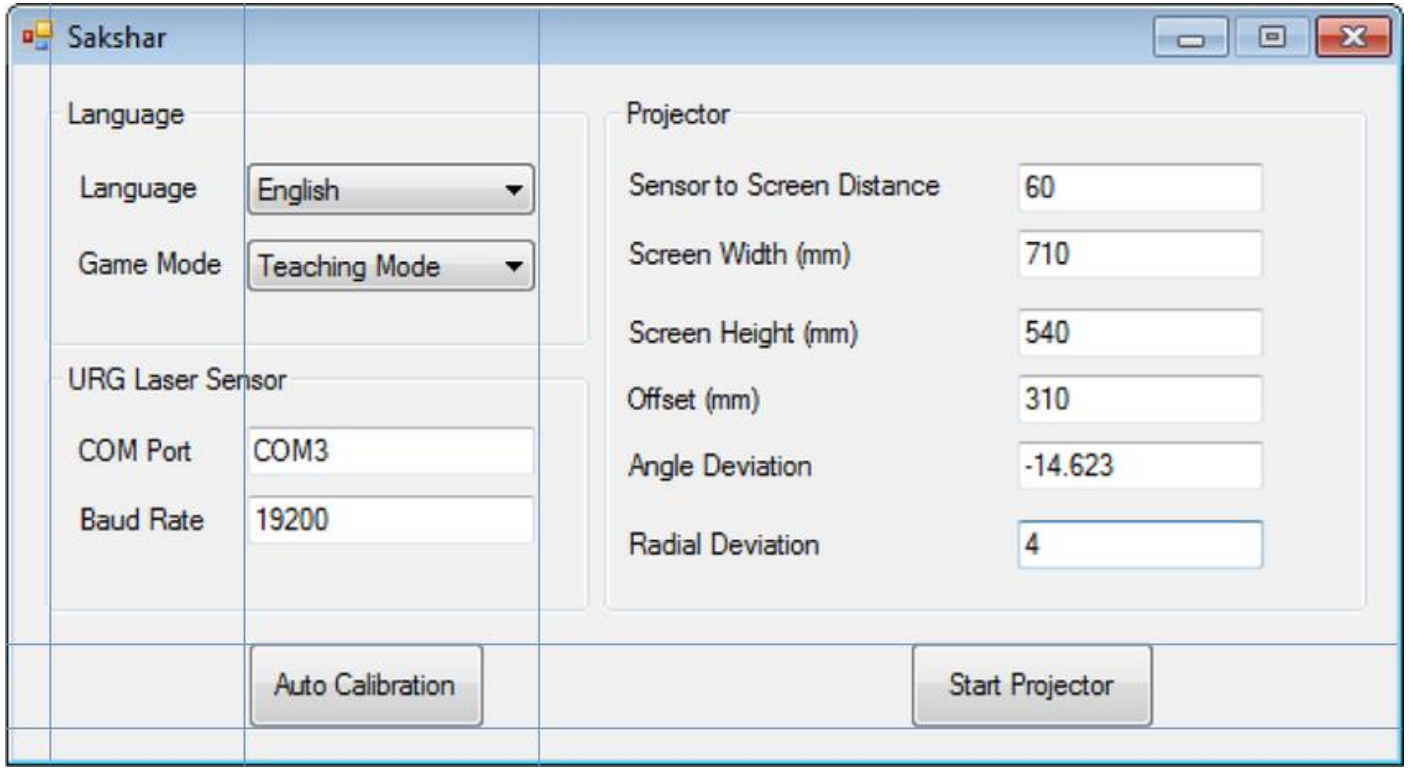
Proximity - Examples

CRAP Design - Alignment

- Nothing should be placed on the GUI arbitrarily
- The basic purpose: unify and organise
- How to get it:
  - Be conscious of where you place elements (buttons, text, etc)
  - Always find something else to align with, even if the two objects are far apart



Alignment - Example



CRAP Design - Repetition

- Repeat some aspect of the design throughout the entire GUI
- The basic purpose:
  - Unify and add visual interest
- How to get it:
  - Think of it as being consistent
  - Find existing repetition and strengthen it
- What to avoid:
  - Avoid repetition that becomes annoying or overwhelming
  - Be aware of contrast

Repetition - Examples

Laura Mathews
1953 Knolls Drive
Santa Rosa, California 95405
707.987.1254

Related Skills
Excellent working knowledge of laboratory tests and their significance in oncology care through working in a clinical laboratory, reinforced while providing patient care. Assisted with bone marrow biopsy and aspiration, lumbar puncture, paracentesis, thoracentesis, and intrathecal chemotherapy administration. Promoted self-care skills and adaptation of the client to their disease and particular treatment program.

Extensive experience with at-home care of AIDS and cancer patients, including IV line maintenance, pain management; understanding of medicare reimbursement and social service referrals.

Education
1990 Associate in Science Nursing, High Honors
Santa Rosa Junior College, Santa Rosa, California.

Experience
1992-present Registered Nurse for Home Health Plus, Visit Division. At-home care of patients with multiple health problems, AIDS, and cancer patients.

1990-present Registered Nurse for Memorial Hospital Oncology Unit, Santa Rosa, California. Managed the care of 4-5 oncology patients. Assumed lead nurse responsibilities. Assisted with new RN orientation. Assisted with procedures, administered chemotherapy, assessed for side effects of chemotherapy and disease process.

1985–1986 Nurse's Aide for Mendocino Coast District Hospital, Fort Bragg, California. Assisted with patient care in Med-Surg and Obstetrical settings.

1985–1986 Lab Assistant for Mendocino Coast District Hospital, Fort Bragg, California. Computer skills while inputting data, cultured lab specimens.

Personal Statement
Previous work experience in a fast-paced, high-stress environment has fine-tuned my organizational skills. My experiences have made me comfortable with oncology patients and their families. Supervisors value my organizational skills, eagerness to learn and assume responsibilities, and my dedication to my job.

Laura Mathews
1953 Knolls Drive
Santa Rosa, California 95405
707.987.1254

Related Skills
Excellent working knowledge of laboratory tests and their significance in oncology care through working in a clinical laboratory, reinforced while providing patient care. Assisted with bone marrow biopsy and aspiration, lumbar puncture, paracentesis, thoracentesis, and intrathecal chemotherapy administration. Promoted self-care skills and adaptation of the client to their disease and particular treatment program.

Extensive experience with at-home care of AIDS and cancer patients, including IV line maintenance, pain management; understanding of medicare reimbursement and social service referrals.

Education
1990 **Associate in Science Nursing,** High Honors
Santa Rosa Junior College, Santa Rosa, California.

Experience
1992-present **Registered Nurse** for Home Health Plus, Visit Division. At-home care of patients with multiple health problems, AIDS, and cancer patients.

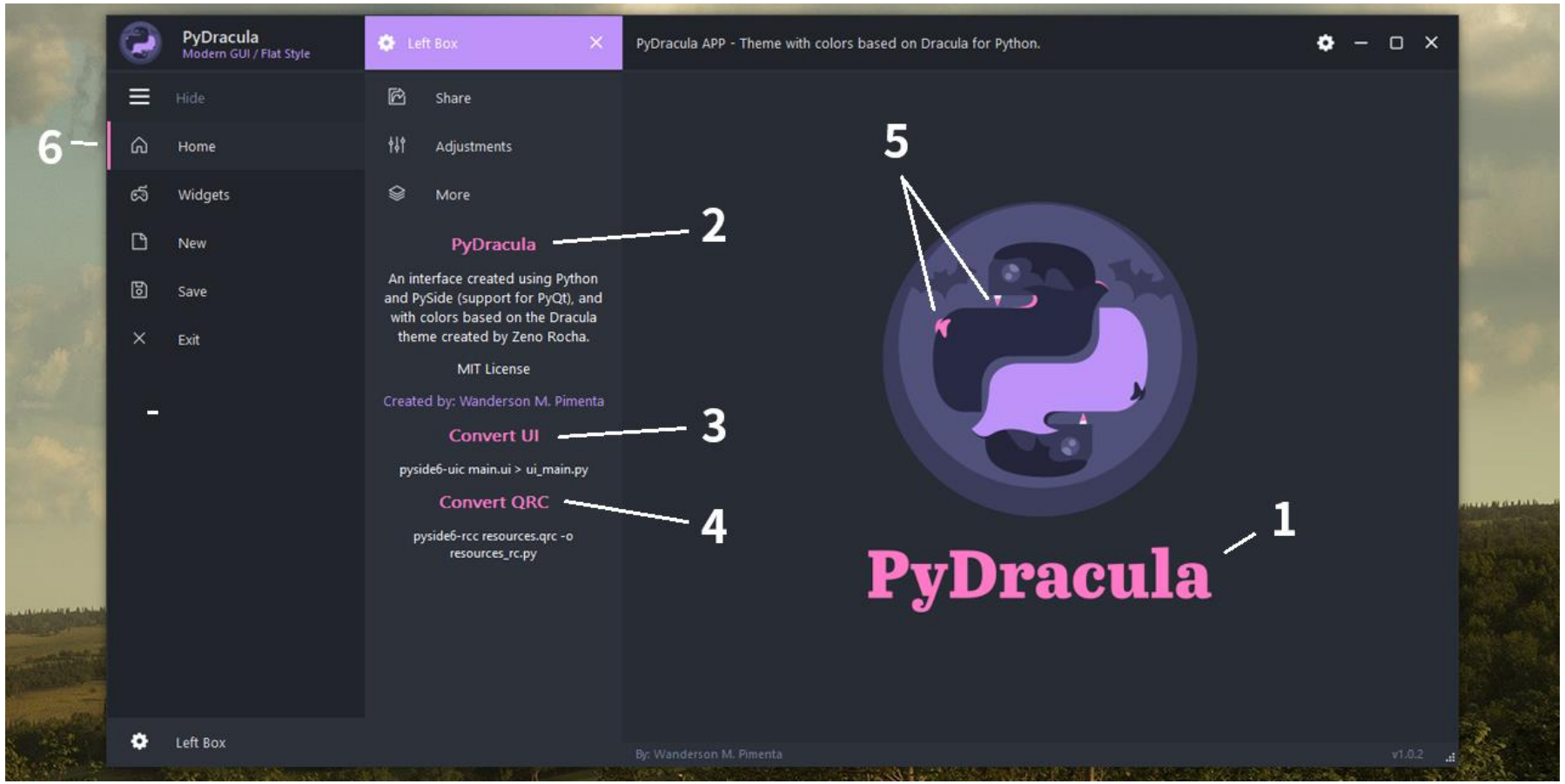
1990-present **Registered Nurse** for Memorial Hospital Oncology Unit, Santa Rosa, California. Managed the care of 4-5 oncology patients. Assumed lead nurse responsibilities. Assisted with new RN orientation. Assisted with procedures, administered chemotherapy, assessed for side effects of chemotherapy and disease process.

1985–1986 **Nurse's Aide** for Mendocino Coast District Hospital, Fort Bragg, California. Assisted with patient care in Med-Surg and Obstetrical settings.

1985–1986 **Lab Assistant** for Mendocino Coast District Hospital, Fort Bragg, California. Computer skills while inputting data, cultured lab specimens.

Personal Statement
Previous work experience in a fast-paced, high-stress environment has fine-tuned my organizational skills. My experiences have made me comfortable with oncology patients and their families. Supervisors value my organizational skills, eagerness to learn and assume responsibilities, and my dedication to my job.

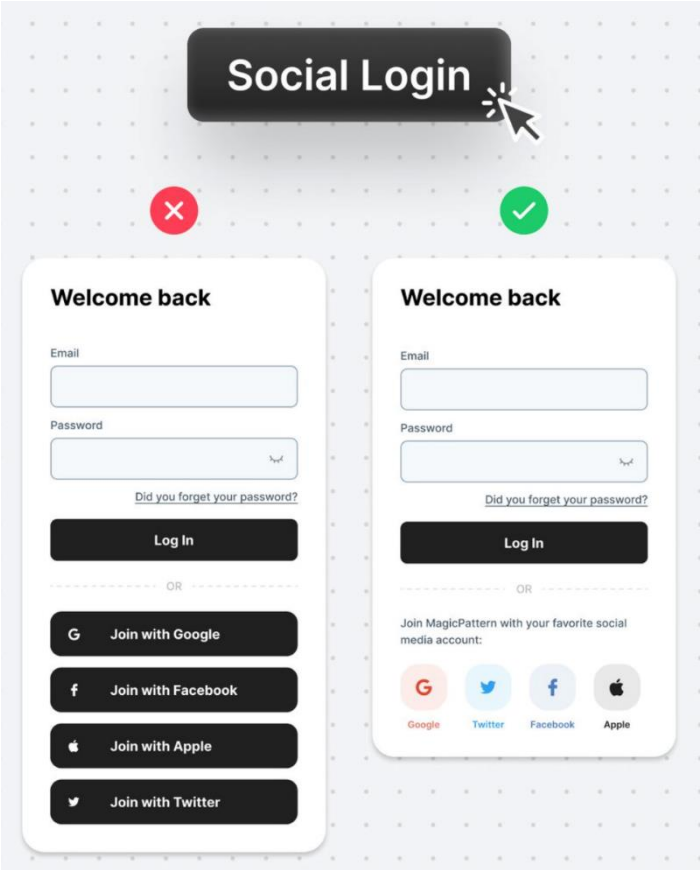
Repetition - Examples

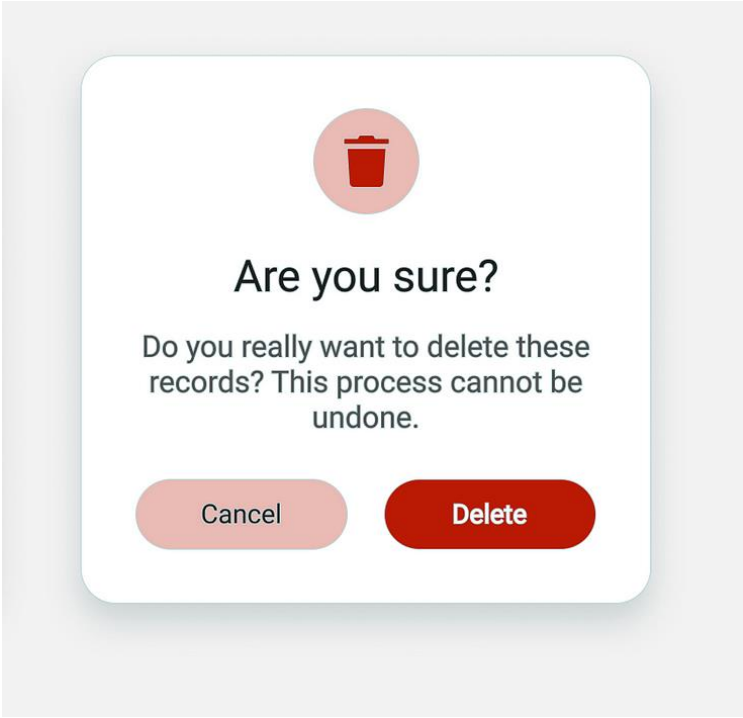


CRAP Design - Contrast

- Create contrast between two items
- The basic purpose:
  - Create interest in the GUI
  - Organise information – highlight important elements
- How to get it:
  - Through typeface, line thickness, colour, size, space
- What to avoid:
  - No subtle differences: make them distinct!

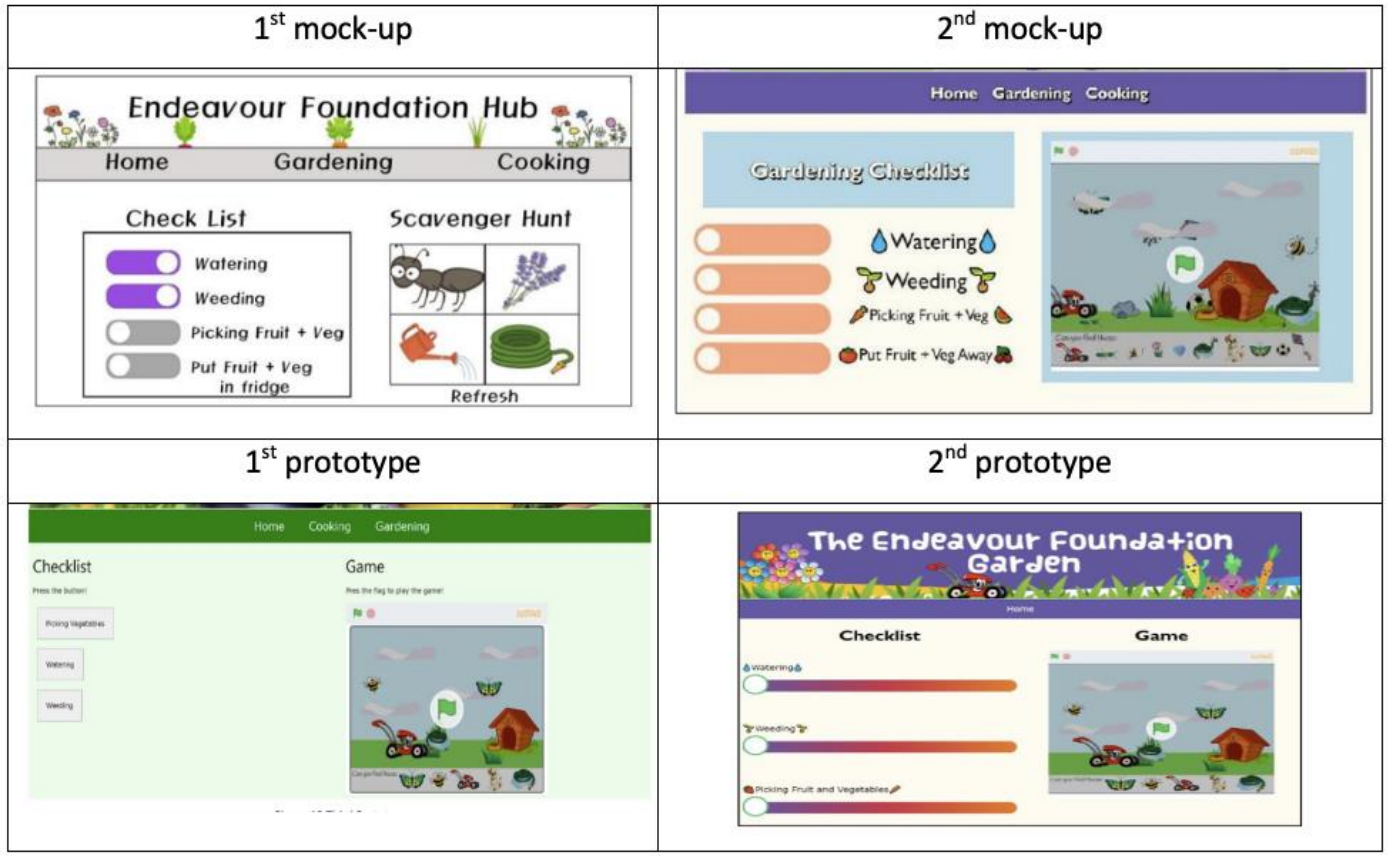
Contrast - Examples





Iterative Interface Design

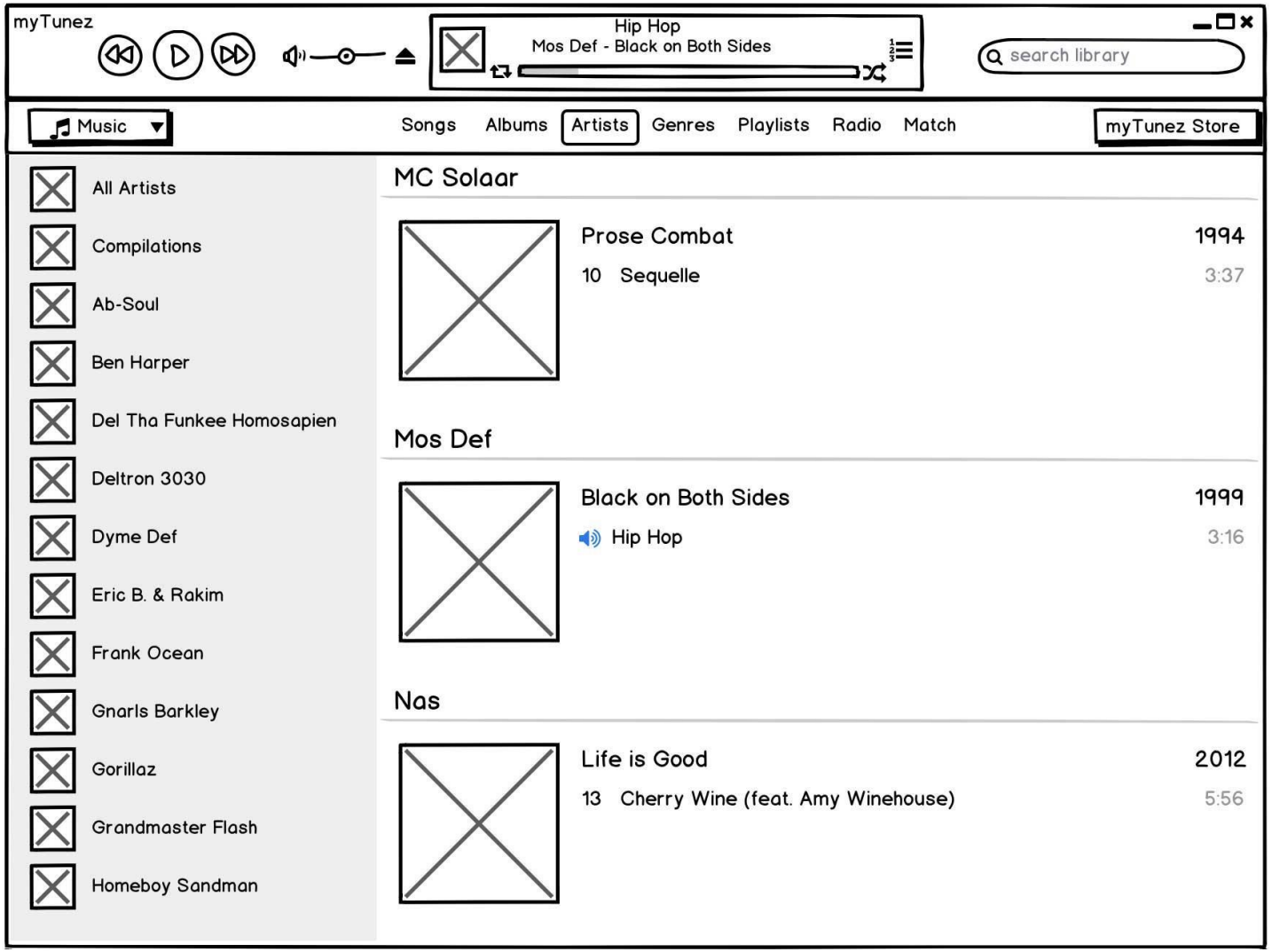
- It is rare to get the design right the first time
- It is a good idea to start with mock-ups first
- Then the interface can be tested and modified to improve usability



Balasuriya, Saminda Sundeepa & Sitbon, Laurianne (2023) Designing gardening applications to engage people with intellectual disability in gardening activities. In Bowen, Judy, Pantidi, Nadia, McKay, Dana, Ferreira, Jennifer, Soro, Alessandro, Blagojevic, Rachel, et al. (Eds.) *OzCHI '23: Proceedings of the 35th Australian Computer-Human Interaction Conference*, pp. 415-422.

Wireframing

- Balsamiq



Changing the user interface

- In the demos so far we have allowed the Tkinter module to choose the window's layout, but we usually want to control this ourselves
- Most widgets allow text fonts, foreground and background colours, and width and height properties to be set using optional parameters
- A Tkinter window's size can be specified using the geometry function:

  window`.geometry('200x50')`

```
# Create a window
my_window = Tk()

# Set the window's size
my_window.geometry('200x50')

# Change the window's background colour
my_window['bg'] = 'red'

# Create a button for the window with
# the same colour background
my_button = Button(my_window,
                   text = 'Push me',
                   highlightbackground = 'red')

# Pack the button into the window with
# some vertical space around it
my_button.pack(pady = 10)
```

Geometry management

- Arranging widgets in the window is done using a ‘geometry manager’:
  - The **pack manager** stacks widgets vertically or horizontally in the available space
  - The **grid manager** places widgets in the cells of an invisible grid, specified by row and column
  - The **place manager** places widgets in absolute *x*-*y* coordinate positions (with the origin at the top left)
- Margins can be added around widgets with the `padx` and `pady` options
- Frame widgets can be used to group widgets together in subwindows

Demos: `grid_demo.py`, `place_demo.py`

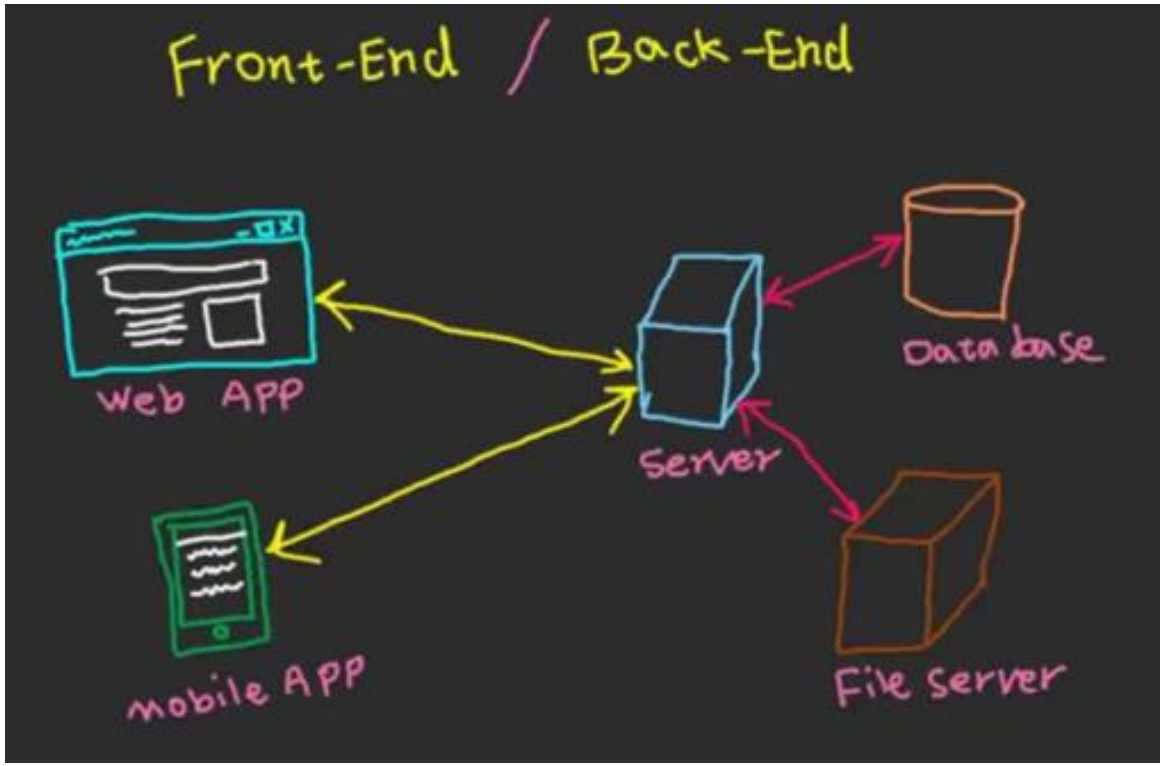
Communication between widgets

- Widgets can interact with each other in several ways:
  - Some widgets, e.g., `Button`s and `Spinbox`es, have a `command` property which can be linked to a parameterless function which gets called whenever the widget is activated
  - Some widgets, e.g., `Text` areas, have methods that can be used to directly change their state, e.g., `insert`
- Some widgets, e.g., `Checkbutton`s, can have a special variable associated with them via a `variable` property
  - These variables are declared as global Python variables using `StringVar`, `IntVar`, etc, constructors
  - The variable's values can be accessed and changed via `get` and `set` functions

Demo: `movie_type_chooser.py`

Front and back ends

- Non-trivial IT systems can usually be divided into a number of separate components:
  1. A storage back end (e.g., a set of files or a database)
  2. A central computational component (i.e., programmatic functions)
  3. A user-friendly front end (e.g., a GUI or Web browser interface)



Demos: `periodic_table.py`, `loan_calculator.py`

Case study: Connecting a GUI to a “database”

- Typically the back end will be a database, file server, etc.
- For this demo we will simply use a Comma-Separated Values text file as the data source

carId	make	model	seriesYear	price	engineSize	tankCapacity	bodyType	seatingCapacity	transmission
14891	DATSUN	FAIRLADY	1964	2530	1.5L	43L	2D ROADSTER	0	4M
17058	TOYOTA	LANDCRUISER	1966	2880	3.9L	70L	2D SOFTTOP	3	3M4x4
15190	CHRYSLER	VALIANT	1966	2490	3.7L	77L	4D SEDAN	0	3M
15196	CHRYSLER	VALIANT	1966	2128	3.7L	65L	UTILITY	3	3M
14988	ALFA ROMEO	SPIDER	1968	4695	1.8L	0L	2D ROADSTER	0	5M
14894	DATSUN	1000	1968	1899	1.0L	0L	4D SEDAN	0	3M
16935	HOLDEN	KINGSWOOD	1968	2924	5.0L	5L	4D SEDAN	0	2A
16107	FORD	CAPRI	1969	2950	1.6L	0L	2D SEDAN	0	4M
15053	AUSTIN	1800	1969	2726	1.8L	0L	4D SEDAN	0	3A
14892	DATSUN	1000	1969	1895	1.0L	0L	2D WAGON	0	3M
16934	HOLDEN	KINGSWOOD	1969	2476	3.0L	0L	4D SEDAN	0	3M
14963	VOLKSWAGEN	1500	1970	2059	1.5L	0L	2D SEDAN	0	4M
15289	FIAT	500	1970	1278	0.5L	0L	2D SEDAN	0	4M
17506	HOLDEN	PREMIER	1970	3305	3.0L	0L	4D SEDAN	0	3A
16635	FORD	FAIRLANE	1970	4528	5.8L	74L	4D SEDAN	0	3A

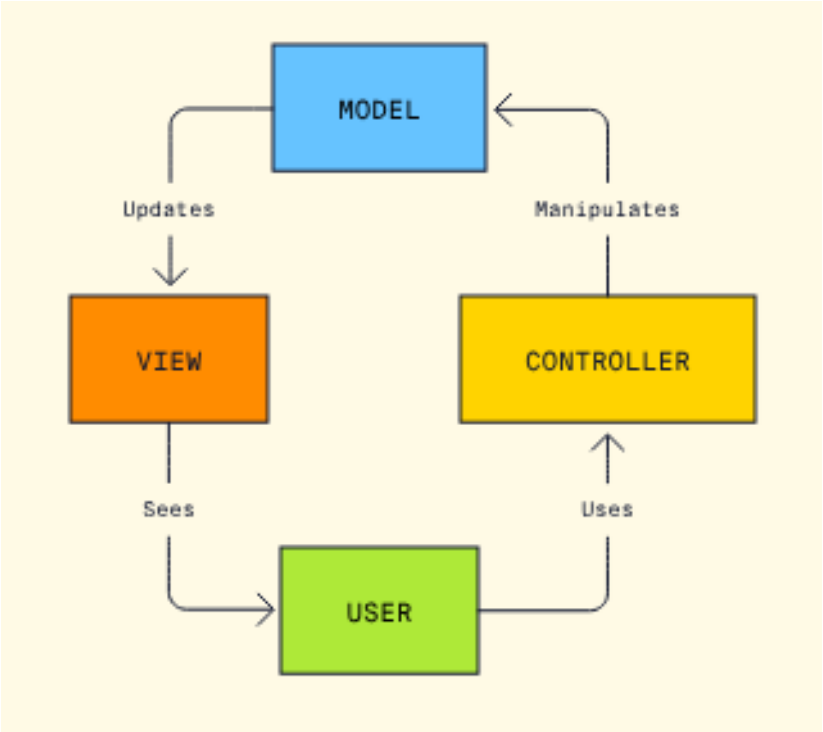
Demo: `used_cars.py`

Model-View-Controller (MVC)

You **won't be assessed** on this content. However, useful when considering how to maintain **organised, re-usable,** and **readable code.**

MVC is a **programmatic design pattern** that organises program logic into **three essential components:**

- Model
- View
- Controller

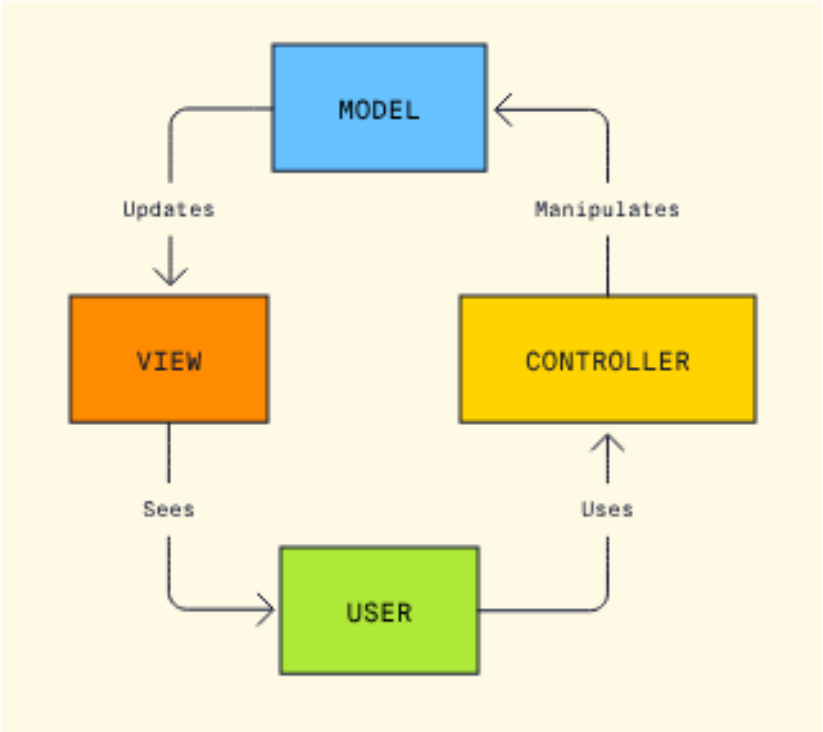


Model-View-Controller (MVC)

Model: The information, or data, of the program – the layer responsible for storing, maintaining, and retrieving data from the back-end.

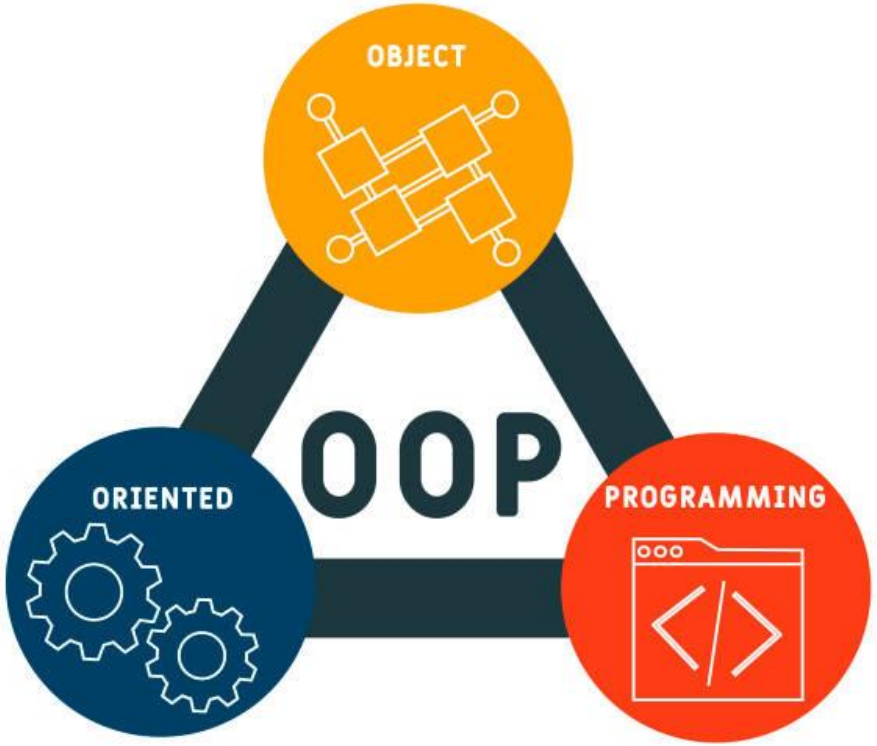
View: How the program is displayed – for example, the GUI. Includes components that enable users to interact with the data (e.g., widgets like buttons).

Controller: Updates the model in response to input from the users (converting input into program commands).



Object Oriented Programming (OOP)

- A very quick overview of OOP (may be a familiar term to some).



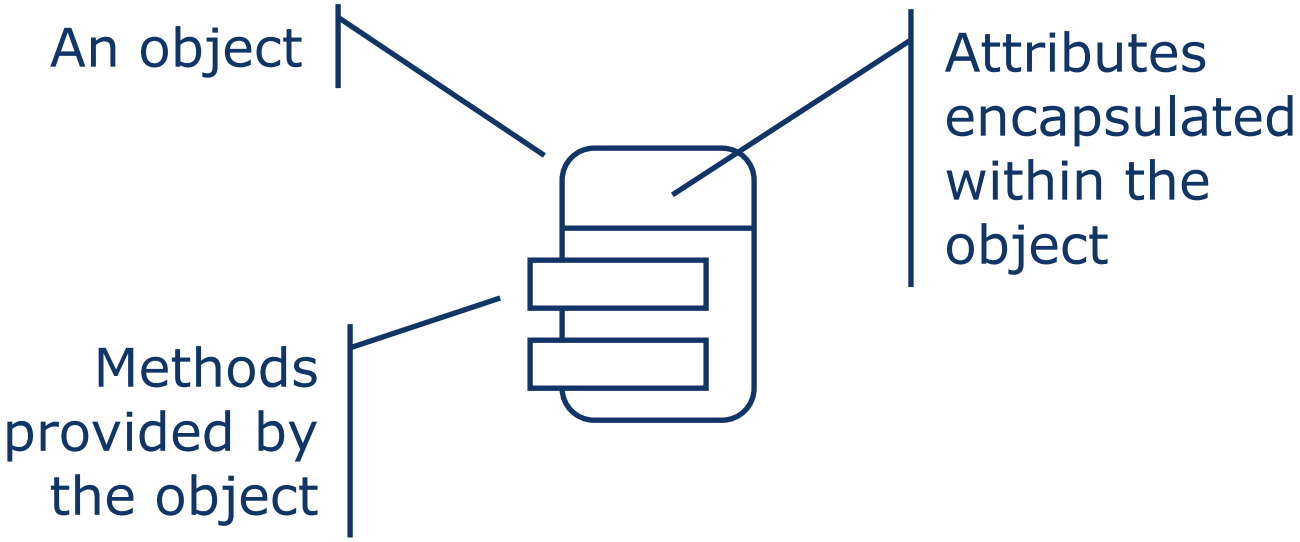
Object orientation

- Python is an “object-oriented” language, although we have avoided referring to this aspect of it as much as possible in IFB104
  - Object orientation is covered in later teaching units
- Object-oriented programming has been a major programming paradigm since the 1960s
  - Most modern programming and scripting languages directly support object-oriented programming: C++, Java, C#, Python, PHP, etc
- OOP is a way of structuring computer programs using “objects” that combine data (“attributes”) and actions (“methods”).
- These objects represent real-world things or concepts, and make code more organised and reusable.
- Each object has characteristics – like a car having a colour and model
- Each object can perform actions – like a car driving or braking.

Demonstration file: `busy_bees.py`

Objects know stuff and do things

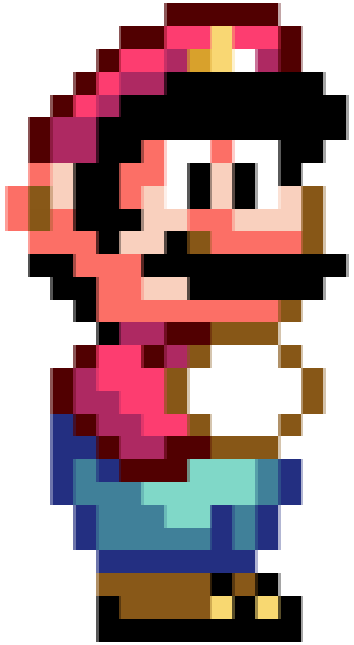
- An *object* contains data, or *attributes*, and provides functions, or *methods*, for accessing the data



Consider LEGO

- OOP is like playing with LEGO people.
- Each LEGO person is an **object.** They have **attributes** (like a name, a hat, or a job), and **things they can do** (like talk, walk, or wave).
- Instead of building everything from scratch every time, you make a template (called a **class**) like “Firefighter”…
  - Then you can make lots of Firefighter LEGO people, each with their own hat colour or name.
- Similar to this, OOP allows you to make these bundles of reusable info + actions.

Consider Mario.



Class: playerCharacter

Object: Mario

Attributes: red, moustachioed, player one

Methods: jump, double jump, run

What about Luigi?

Class: ?

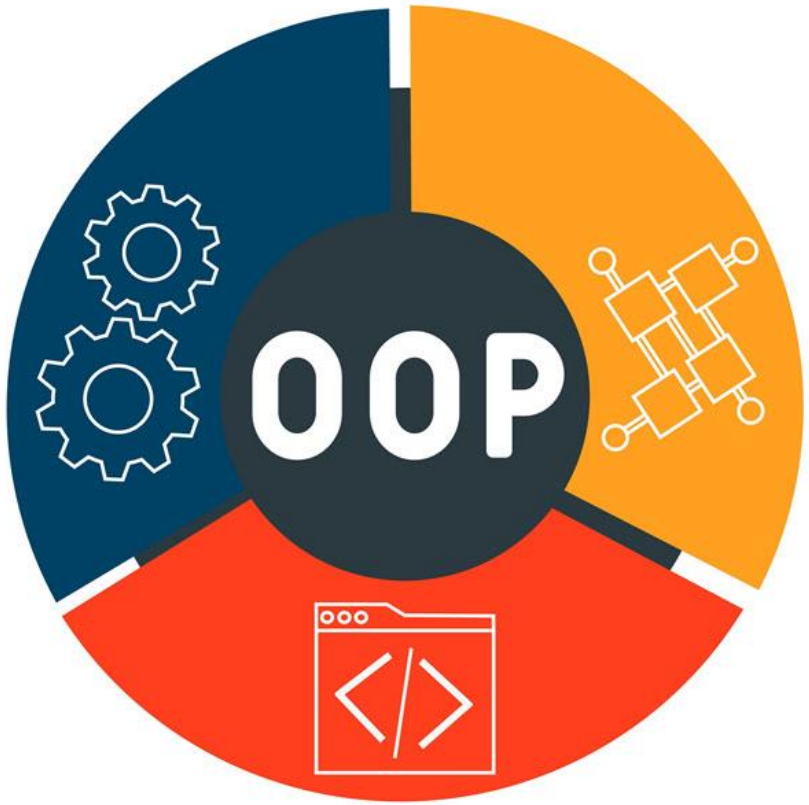
Object: ?

Attributes: ?

Methods: ?

For this unit, we don’t really need to know much more than that.

- You’ll do a much deeper dive on OOP in future units.
- Highly relevant to **game development, mobile app development, web development, data science and machine learning,** and **enterprise software development.**
- But broadly relevant to understanding user interface elements, like buttons!



Before next week ...

- Complete this week’s workshop exercises (these workshops introduce you to Tkinter)
- Familiarise yourself with 2A

Tkinter references

- Some general references:
  - Y. D. Liang, *Introduction to Programming Using Python*, Pearson 2013, Ch. 9, *GUI Programming Using Tkinter*
  - M. Dawson, Python *Programming for the Absolute Beginner*, Cengage 2010, Ch. 10, *GUI Development: The Mad Lib Program*

Tkinter references

- The Tkinter module is described rather briefly in the standard Python documentation:

  https://docs.python.org/3/library/tk.html

- The hardest part of Tk GUI programming is trying to remember which properties are applicable to each kind of widget – there are lots of widgets and lots of properties!

- The following tutorials appear to be a good introductions to Tk in Python:

  https://www.python-course.eu/python_tkinter.php

  http://www.tutorialspoint.com/python/python_gui_programming.htm

- And a previous IFB104 student recommended the following (thanks, Isaac):

  https://tkdocs.com/tutorial/