

# CS4043 Advanced Programming

Omar Usman Khan  
omar.khan@nu.edu.pk

Department of Computer Science  
National University of Computer & Emerging Sciences, Peshawar

July 22, 2024



# Syllabus

## 1 Course Overview

- Premise
- Note on Programming Languages
- Job Market

## 2 UI/UX

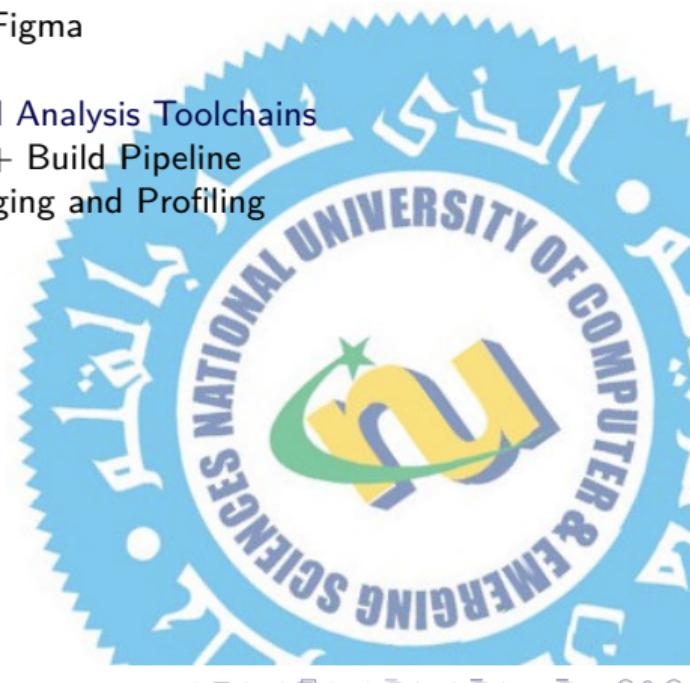
- Overview

### • UI

- GUI - Tkinter
- GUI - Figma
- UX

## 3 Build and Analysis Toolchains

- C/C++ Build Pipeline
- Debugging and Profiling



## 1 Course Overview

- Premise
- Note on Programming Languages
- Job Market

## 2 UI/UX

- Overview

- UI
- GUI - Tkinter
- GUI - Figma
- UX

## 3 Build and Analysis Toolchains

- C/C++ Build Pipeline
- Debugging and Profiling

# Course Overview

## Marks Breakdown

- Mid-Term: 30 %
- Final Examination: 40%
- Assignments: 30%

## Assignment Notes

- Use Linux based tools
- If C/C++ (Use GCC)
- Marking Scheme (Typical):
- No assignment submission through email
- I May run through similarity index.

# Complex Software

## Production Process

- ① Customers & Line of Business
- ② Software Development Lifecycles (Design → Develop → Testing → Deploy)

## Deployment Platforms

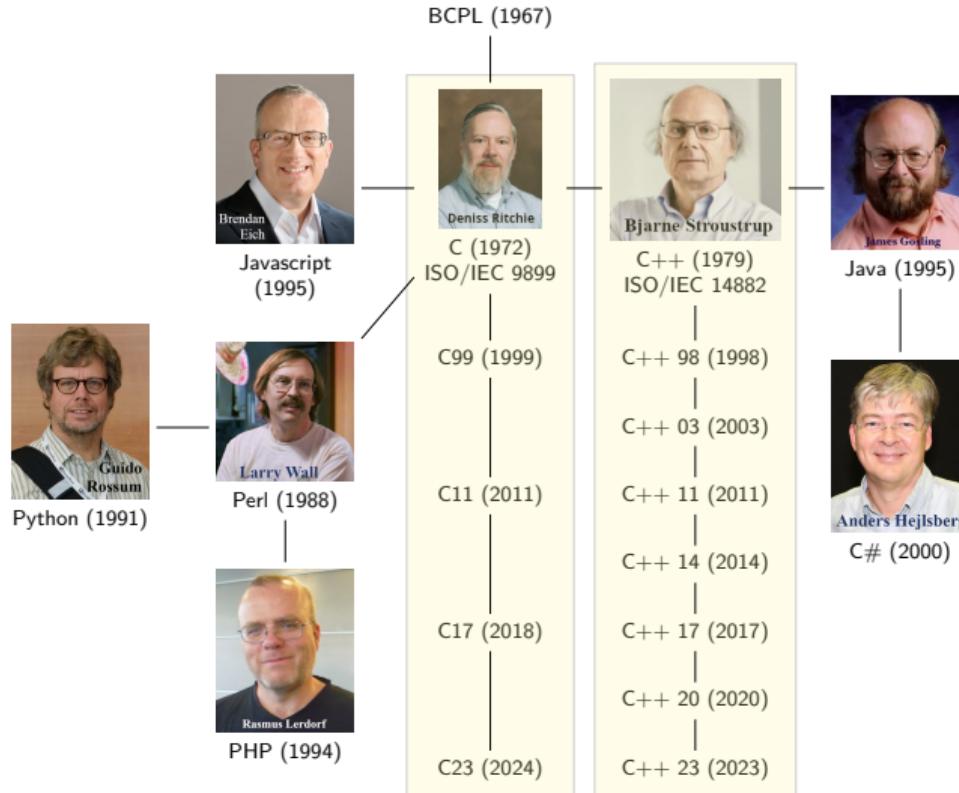
- Standalone Systems
  - Mobile Apps
  - Distributed Software
  - Big Data & Cloud
  - Internet of Things
  - ... Metaverse
- 
- Software Complex than ever (Programming Languages, Data Sources, Integrated Data Types, Software Design, Algorithms, Performance, Group based Development, Security, ...)
  - Solution: Embrace innovation in Programming Languages, Data Handling Methods, Undertake high level design choices, Software Development Frameworks, Development Methodologies, Execution Environments, ... All qualities of a **Software Architect** (job role appears with slight variations in names)

# Complex Software (cont.)



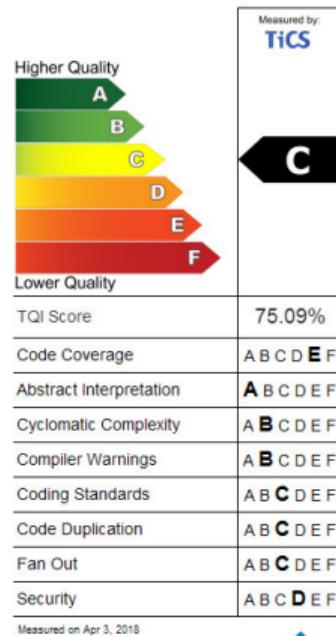
# Programming Languages

# Programming Languages (cont.)

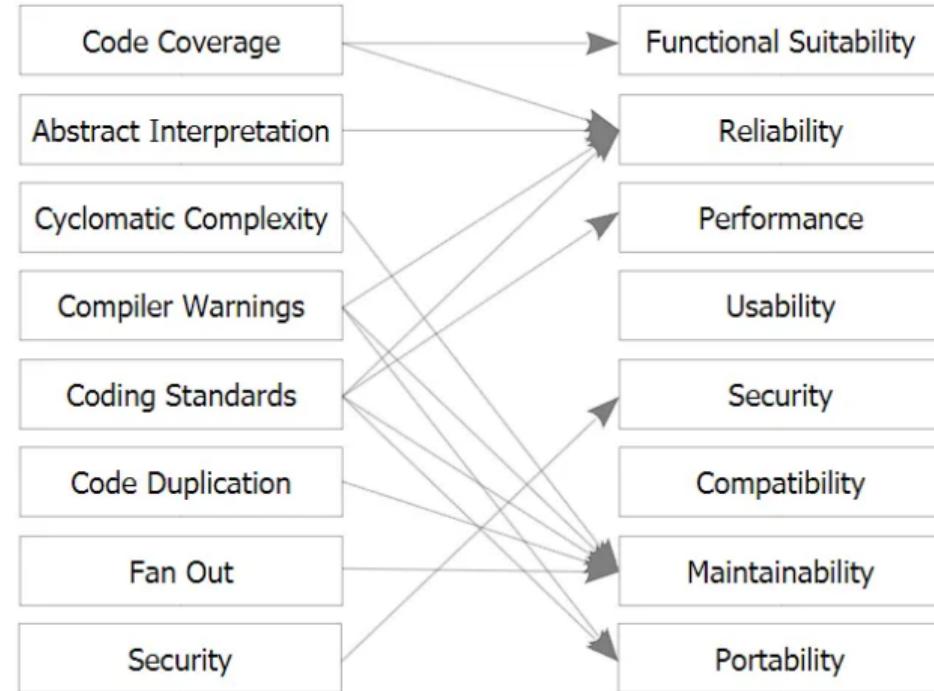


# TIOBE Index (ISO 25010)

## TIOBE Quality Indicator



This product has been tested with the utmost care  
against the TIOBE Quality Indicator 4.0.



# TIOBE Index (ISO 25010) (cont.)

Feb 2022	Feb 2021	Change	Programming Language	Ratings	Change
1	3	▲	Python	15.33%	+4.47%
2	1	▼	C	14.08%	-2.26%
3	2	▼	Java	12.13%	+0.84%
4	4		C++	8.01%	+1.13%
5	5		C#	5.37%	+0.93%
6	6		Visual Basic	5.23%	+0.90%
7	7		JavaScript	1.83%	-0.45%
8	8		PHP	1.79%	+0.04%
9	10	▲	Assembly language	1.60%	-0.06%
10	9	▼	SQL	1.55%	-0.18%
11	13	▲	Go	1.23%	-0.05%
12	15	▲	Swift	1.18%	+0.04%
13	11	▼	R	1.11%	-0.45%
14	16	▲	MATLAB	1.03%	-0.03%
15	17	▲	Delphi/Object Pascal	0.90%	-0.12%
16	14	▼	Ruby	0.89%	-0.35%
17	18	▲	Classic Visual Basic	0.83%	-0.18%
18	20	▲	Objective-C	0.81%	-0.08%
19	19		Perl	0.79%	-0.13%
20	12	▼	Groovy	0.74%	-0.76%

Jun 2024	Jun 2023	Change	Programming Language	Ratings	Change
1	1		Python	15.39%	+2.83%
2	3	▲	C++	10.03%	-1.33%
3	2	▼	C	9.23%	-3.14%
4	4		Java	8.40%	-2.88%
5	5		C#	6.65%	-0.06%
6	7	▲	JavaScript	3.32%	+0.51%
7	14	▲	Go	1.93%	+0.93%
8	9	▲	SQL	1.75%	+0.28%
9	6	▼	Visual Basic	1.66%	-1.67%
10	15	▲	Fortran	1.53%	+0.53%
11	11		Delphi/Object Pascal	1.52%	+0.27%
12	19	▲	Swift	1.27%	+0.33%
13	10	▼	Assembly language	1.26%	-0.03%
14	12	▼	MATLAB	1.26%	+0.14%
15	8	▼	PHP	1.22%	-0.52%
16	13	▼	Scratch	1.17%	+0.15%
17	20	▲	Rust	1.17%	+0.26%
18	18		Ruby	1.11%	+0.17%
19	29	▲	Kotlin	1.01%	+0.50%
20	22	▲	COBOL	0.96%	+0.22%

By the Way ...



## Frameworks vs Libraries

- Both provide reusable abstractions of code, wrapped in well defined API
- Libraries: Flow of control dictated by caller (software calls library)
- Frameworks: Flow of control dictated by Framework (Framework drives software)

### Framework Examples

- General Software: .NET, Android SDK, ...
- GUI Based: Gnome, QT, ...
- Web Based: ASP.Net, ...
- Concurrency: Hadoop Map/Reduce, ...
- ...

## Jobs (Pakistan) 2022-2023 Source: P@sha

 1 JavaScript Fullstack (MEAN/MERN)	 2 .NET	 3 React Native (Hybrid)	 4 Android - JAVA	 5 PHP (Laravel/CodeIgnitor/Yii/Zend/Drupal)	 6 Python	 7 Flutter (Hybrid)
 7 iOS - Objective C & SWIFT	 9 AWS Developer- Associate	 10 Java	 11 MICROSOFT SQL	 12 Unity	 13 Selenium	 13 Microsoft Certified: Azure Fundamentals
 15 Postgres	 16 Redis/ElasticSearch AWS	 16 Agile Certified Practitioner PMI-ACP	 18 Professional Scrum Master™ PSM	 19 Ruby on Rails	 20 Salesforce	

Figure 1: P@sha Top In-Demand Technologies &amp; Tools

# Jobs (Pakistan) 2022-2023 Source: P@sha (cont.)

## Hiring Resources (Next 6-8 Months)

Following are future top 20 in-demand technologies & tools nationwide keeping in mind the current technology trends.

A total of **19,451 resources** will be hired in next 6-8 months for the **42 technology tracks** mentioned in the survey.

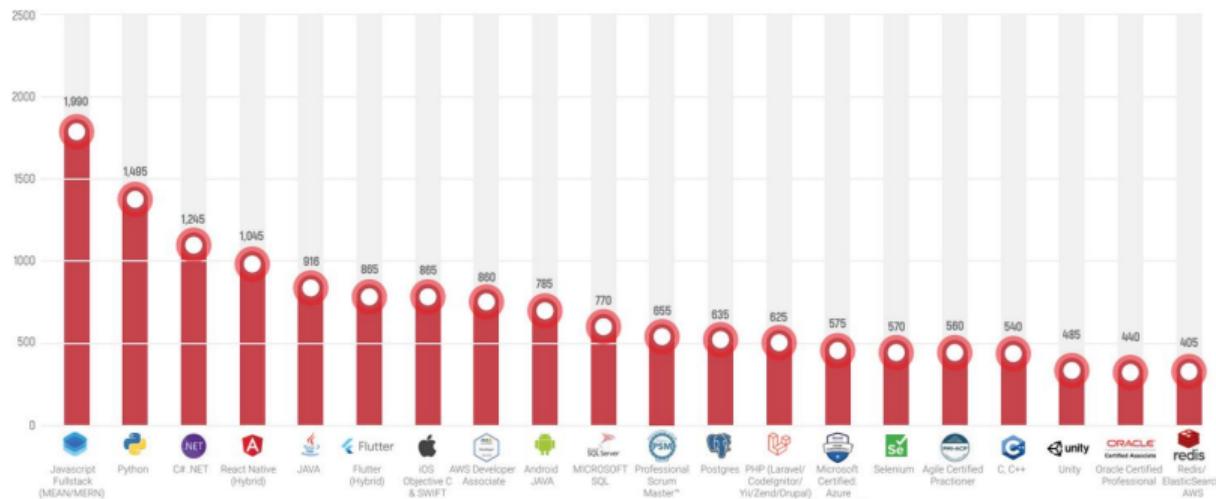


Figure 2: Next 6-8 Months Job Market

## Jobs (Pakistan) 2022-2023 Source: P@sha (cont.)


Figure 3: Participating Companies List A

## Jobs (Pakistan) 2022-2023 Source: P@sha (cont.)

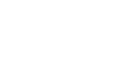
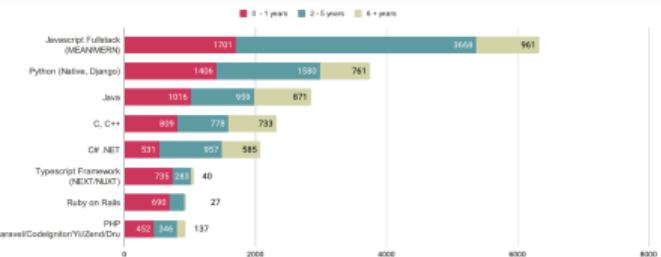
								
								
								
								
								
								
								

Figure 4: Participating Companies List B

# Jobs (Pakistan) 2023-2024 Source: P@sha

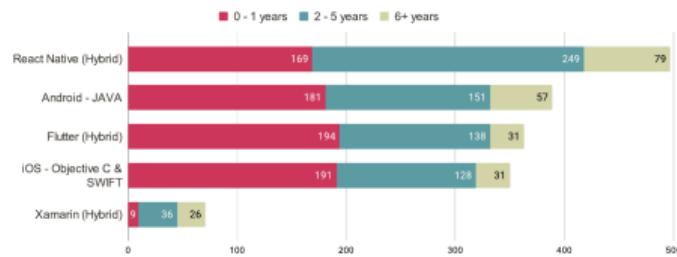
## Resources Required Web Engineering Job Roles

A total of **20,250 resources** are required in the 'Web Engineering' roles with the highest demand for **Javascript Fullstack (MEAN/MERN)**.



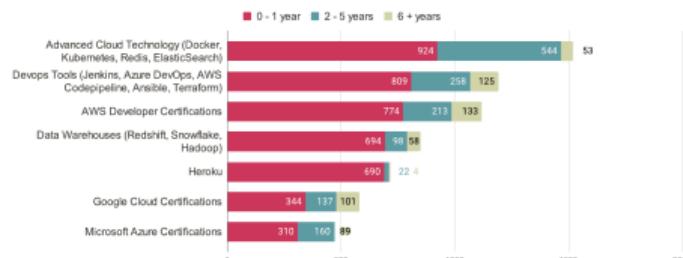
## Resources Required Mobile Development Job Roles

A total of **1,670 resources** are required in the 'Mobile Development' roles with the highest demand for **React Native (Hybrid)**.



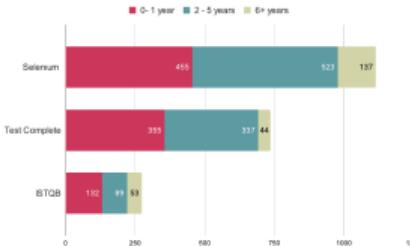
## Resources Required Cloud Infrastructure - Tool / Certification

A total of **6,540 resources** are required in the 'Cloud Infrastructure - Tool / Certification' roles with the highest demand for **Advanced Cloud Technology (Docker, Kubernetes, Redis, ElasticSearch)**.



## Resources Required Automation & Software Testing - Tool / Certification

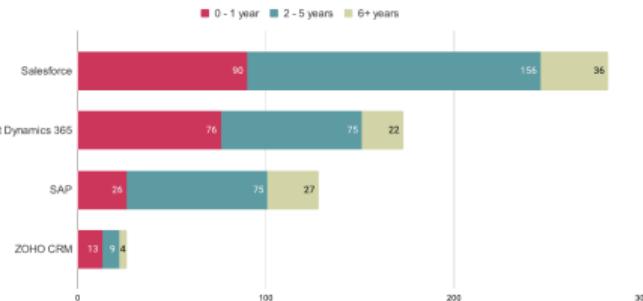
A total of **2125 resources** are required in the 'Automation & Software Testing - Tool/Certification' roles with the highest demand for **Selenium**.



# Jobs (Pakistan) 2023-2024 Source: P@sha (cont.)

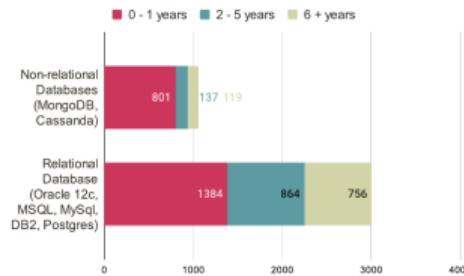
## Resources Required CRM Tool

A total of **609 resources** are required in the 'CRM Tool' roles with the highest demand for **Salesforce**.



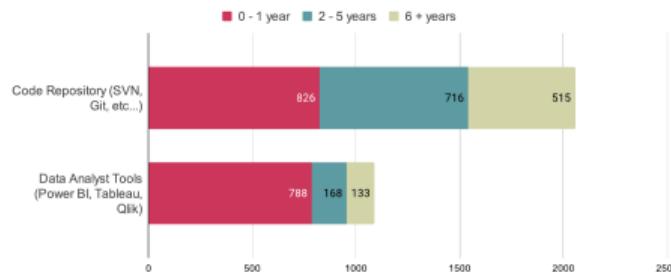
## Resources Required Database Job Roles

A total of **4061 resources** are required in the 'Database' roles with the highest demand for **Relational Database**.



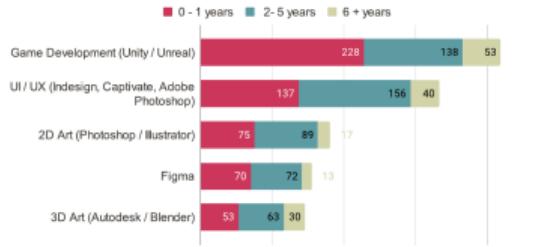
## Resources Required Programming and Data Analyst

A total of **3146 resources** are required in the programming and data analyst domain with the highest demand for **Code Repository (SVN, Git, etc...)**.



## Resources Required Animation, Graphic, Games – 3D Engine/Tool

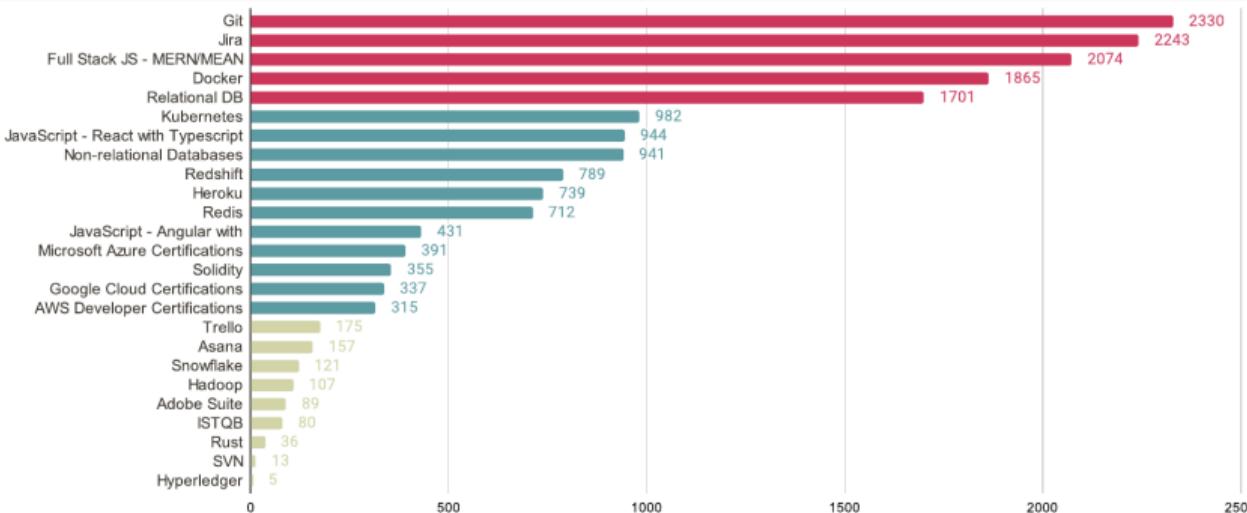
A total of **1234 resources** are required in the 'Animation, Graphic, Games – 3D Engine/Tool' with the highest demand for **Game Development (Unity/Unreal)**.



## Jobs (Pakistan) 2023-2024 Source: P@sha (cont.)

## Resources Required Major tools & technologies for the Upskilling of Employees

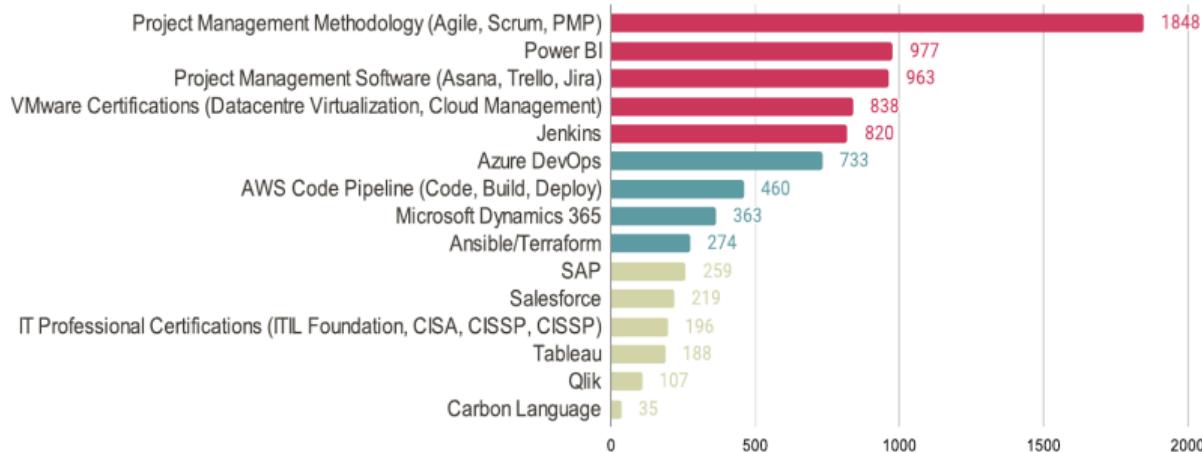
A total of **17,932 resources** are required in following tools & technologies for upskilling of employees with highest demand for **Git**.



# Jobs (Pakistan) 2023-2024 Source: P@sha (cont.)

## Resources Required - Major tools & technologies for the Reskilling of Employees

A total of **8,280** resources are required in following tools & technologies for reskilling of employees with highest demand for '**Project Management Methodology (Agile, Scrum, PMP)**'



# Homework Task 1

## Task Juggler

Use Task Juggler to create a Financial Proposal, and Project Gantt Chart for your Final Year Project

## 1 Course Overview

- Premise
- Note on Programming Languages
- Job Market

## 2 UI/UX

- Overview

- UI
- GUI - Tkinter
- GUI - Figma
- UX

## 3 Build and Analysis Toolchains

- C/C++ Build Pipeline
- Debugging and Profiling

# UI/UX Overview

## UI

- UID: Design of **UI's** for machines and softwares, with focus on maximizing **usability** and **user experience**
- UI: Visual elements of a product, adaptable to different views (mobile, tablet, computer, ...)
- *Makes the interface beautiful*
- UI Designer: Typography, Colors, Styles, Branding, Spacing/Layouts, Number of items, Icons, Images, ...

## UX

- UXD: Study user behaviour (usability, usefulness) and understanding user motivations (desirability) in the interaction with a product, with goal of maximizing digital experiences.
- *Makes the UI useful*
- UX Designer: Interaction design, Wireframes/Prototypes, Information architect, user research, personas, scenarios, ...

## UI/UX Overview (cont.)



# UI/UX Overview (cont.)

## Don Norman

American Researcher

Overview

Books

Videos



N Nielsen Norman Group

Don Norman, co-founder and board member of Nielsen ...

Don Norman, Ph.D., is co-founder and board member of Nielsen Norman Group: User Experience Research, Training, and...

Age

88 years

25 Dec 1935

Known for

The Design of Everyday Things; Cognitive er-...



# UI/UX Overview (cont.)

## Jakob Nielsen

Danish consultant

Overview

Books

Videos



NN Nielsen Norman Group

Jakob Nielsen, Ph.D. and Principal at Nielsen Norman Group

Biography: Jakob Nielsen, Ph.D., is a usability advocate and retired principal of the Nielsen Norman Group: User...

Age

66 years  
5 Oct 1957

Parents >  
Gerhard Nielsen, Helle Höpfner Nielsen

YouTube • NNgroup

The NN/g UX Podcast

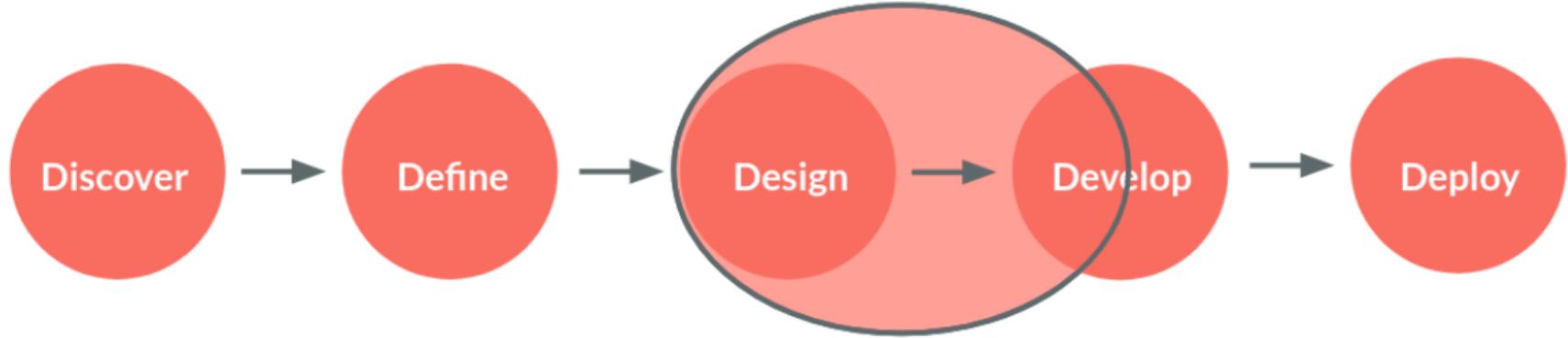
EPISODE 1

What is UX,  
anyway?

16:53



# UI/UX Design Process



## Design

- Iterate (Information Architecture: Personas & User Research, Storyboards, Wireframes)
- Prototype (Visual Design & Layout, Interaction Design, Accessibility)
- Test (Usability Testing, A/B Testing)

# UI/UX Design Process (cont.)

## Develop (Front-end)

- Responsive Layouts
- Behavior (React/Angular)
- ...

**Sophie March**

**Bio:** Sophie March is the most successful businesswoman in the world. She has never been any kind of office worker she just always had her own ideas and she's always been a leader. She's got a lot of money and she's got a lot of power. She's got a lot of influence and she's got a lot of authority.

**Motivations:** Curiosity, Achievement, Curiosity, Curiosity, Curiosity, Curiosity.

**Personality:** Intelligent, Ambitious, Hardworking, Loyal, Positive, Ambitious.

**Pain Points / Concerns:**

- Sophie is a little bit of a workaholic. That's why she's been working all day and all night. She's got a lot of stress and she's got a lot of pressure. She's got a lot of responsibility and she's got a lot of responsibility.
- The environment is different from what she's used to. She's used to a quiet office environment where she can do her work. Now she's in a noisy office environment where she can't concentrate.
- Sophie is worried about her new job. She's worried about whether she'll be able to do it well. She's worried about whether she'll be able to succeed.

**Scenario:** Sophie has been invited to a dinner party at a friend's house. She's been invited to a dinner party at a friend's house. She's been invited to a dinner party at a friend's house. She's been invited to a dinner party at a friend's house. She's been invited to a dinner party at a friend's house.

**Brands:** McDonald's, Ray-Ban

**Kind - Achiever - Charming**

**Age 19**  
Occupation Business Student  
Family Single  
Location Beijing - China  
Achievement The Intelligent

**Personas 3**

**Person A:** Person A is a person who likes to work alone. They have a lot of responsibilities and they're always trying to get things done. They're very organized and they like to keep things in order. They're also very detail-oriented and they like to make sure everything is perfect.

**Person B:** Person B is a person who likes to work in teams. They have a lot of responsibilities and they're always trying to get things done. They're very organized and they like to keep things in order. They're also very detail-oriented and they like to make sure everything is perfect.

### Top of Form Validation

- High cognitive load on memory  
 Longer time to correct errors

### Inline Validation

- Low cognitive load on memory  
 Shorter time to correct errors



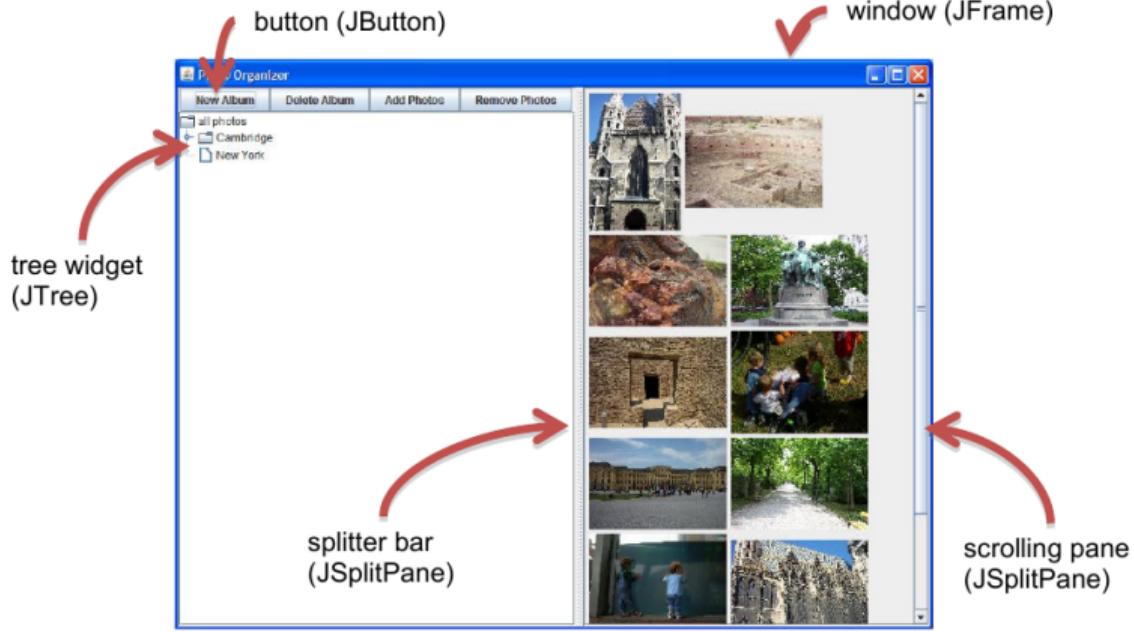
# UI/UX Design Process (cont.)

## UI Frameworks

- Python: Tkinter, PyQt, PyGTK, wxPython, ...
- C/C++: TCL/TK, GTK (Gnome), QT, WxWidgets, NCurses, Flutter (Google), ...
- Java: Swing, JavaFX, ...

## Visual Elements in a Tree (View Tree)

- GUI's composed of View Objects; Each occupying certain portion of screen (covering a rectangular area = bounding box)
- View Objects arranged into a hierarchy of Containment (views inside other views)
  - Examples of Containers: Window, Panel, Toolbar, etc.
  - Child views are nested inside parent's bounding box
- Primitive Views (Do not contain other Views)
  - Buttons, Trees, Textbox, Thumnail, etc.
- View Objectives can be composited / grouped / joined together



## Visual Elements in a Tree (View Tree) (cont.)

```
1  <?xml version="1.0" encoding="UTF-8"?>LF
2  <ui version="4.0">LF
3  <class>Dialog</class>LF
4  <widget class="QDialog" name="Dialog">LF
69 <resources/>LF
70 <connections>LF
71 <connection>LF
72   <sender>buttonBox</sender>LF
73   <signal>accepted()</signal>LF
74   <receiver>Dialog</receiver>LF
75   <slot>accept()</slot>LF
76   <hints>LF
77     </connection>LF
78   <connection>LF
79     <sender>buttonBox</sender>LF
80     <signal>rejected()</signal>LF
81     <receiver>Dialog</receiver>LF
82     <slot>reject()</slot>LF
83     <hints>LF
84       <hint type="sourcelabel">LF
85         <hint type="destinationlabel">LF
86           <x>286</x>LF
87           <y>274</y>LF
88         </hint>LF
89       </hints>LF
90     </connection>LF
91   </connections>LF
92 </ui>
```

Figure 5: QT GUI as XML Tree

## Visual Elements in a Tree (View Tree) (cont.)

### View Object Properties (Layout)

- Arrangement of bounding boxes of each view object arranged on the basis of a layout.
- May involves information about position and size of view objects

### View Object Properties (Output)

- Each View Object responsible for displaying itself
- Associated with a thread
- If Tree changes, so will its child nodes

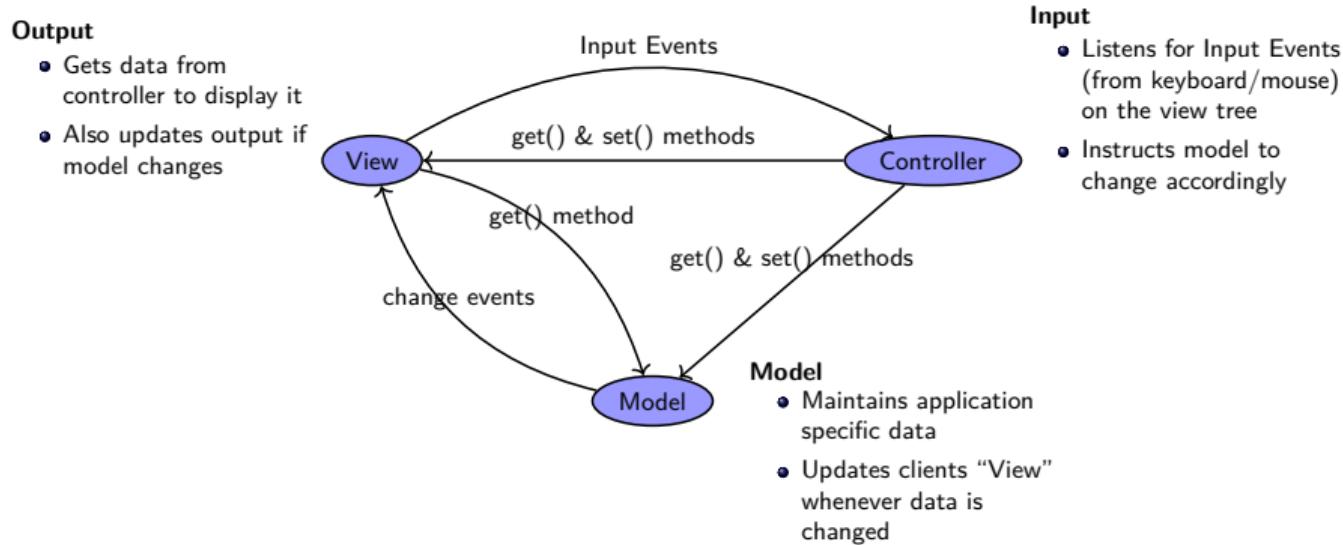
## Visual Elements in a Tree (View Tree) (cont.)

### View Object Properties (Input)

- Enabled View Objects can listen for input from Keyboard and Mouse using additional threads
  - Event Object** Contains information about an event
  - Event Source** The component where an event is generated
  - Event Listener** A listener is created which records an interest to the particular stream of events. It then listens for the event.
  - Event Handler** What to do when an event occurs (User Defined) A function is called whenever a new event occurs.

# Model View Controller (MVC)

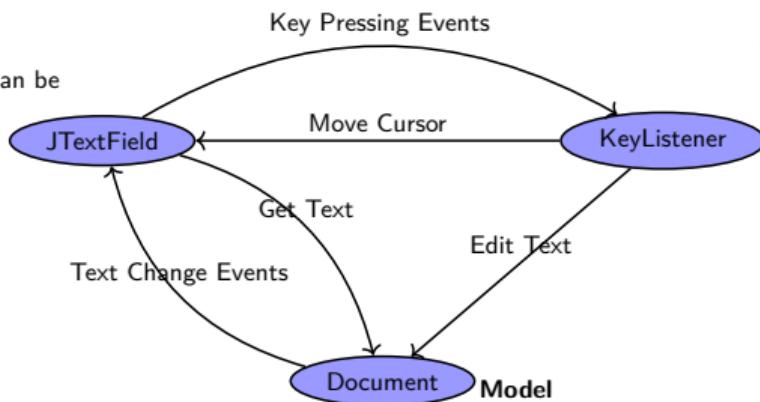
- A software architecture pattern for GUI's (initially) and Web Services
- Introduced in 1970's
- Separates GUI front-end from application back-end
  - Back-end code = Model
  - Front-end code = View, Controller



## Model View Controller (MVC) (cont.)

### Output

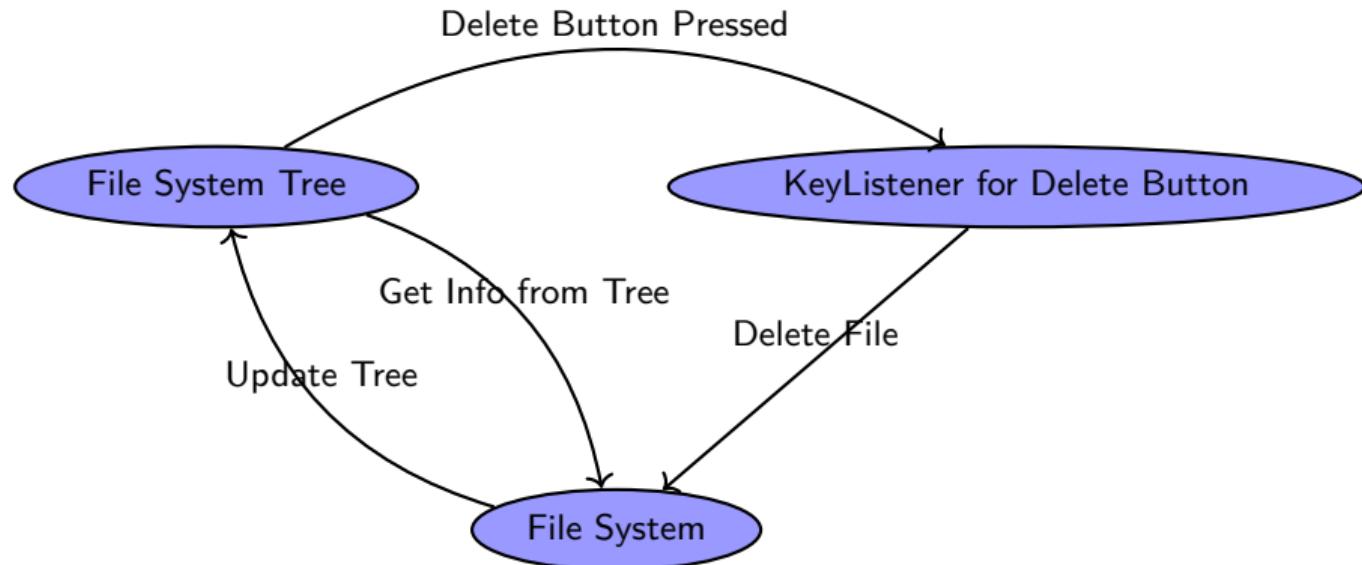
- JTextField is a Jcomponent that can be added to the view hierarchy



### Input

- KeyListener is a listener for Keyboard Events

## Model View Controller (MVC) (cont.)



## Model View Controller (MVC) (cont.)

### Benefits of Separating Model from View

- Many views of same application data possible
- Allows for creating user interface toolkits (view toolkits, completely separate from the application data)

# Tkinter

- Python interface to TK (GUI tool for TCL/TK)
- Standard GUI Library in Python (bundled)
- Object Oriented Approach
- Elements represent MOTIF components (non-native components)



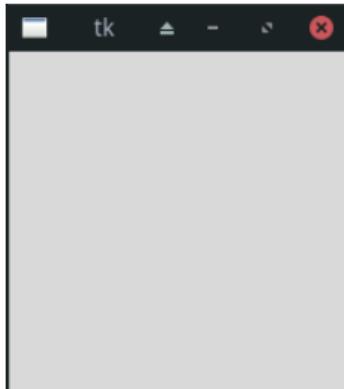
# Hello World

```
import tkinter

root = tkinter.Tk()

# Widget Codes

root.mainloop()
```



## Bypassing WM

```
screen_width = str(root.winfo_screenwidth())
screen_height = str(root.winfo_screenheight())

root.geometry(screen_width+"x"+screen_height+"+0+0")
print(root.winfo_geometry())
print(root.winfo_width(), root.winfo_height())

root.overrideredirect(True)
```

## Top Level

```
t1 = Toplevel(root)

t2 = Toplevel(root)
t2.transient(root)
```

## Frames

```
for relief in [RAISED, SUNKEN, FLAT, RIDGE, GROOVE, SOLID]:
    f = Frame(root, borderwidth=2, relief=relief)
    Label(f, text=relief, width=10).pack(side=LEFT)
    f.pack(side=LEFT, padx=5, pady=5)
```



## Common Widgets

```
Label(root, text="Label Text", wraplength=300,
      justify=LEFT).pack()

Button(root, text="Go1", command=go1func).pack()
Button(root, text="Go2", command=lambda: go2func()).pack()

e = StringVar()
Entry(root, width=40, textvariable=e).pack()
```

# Hello World (cont.)

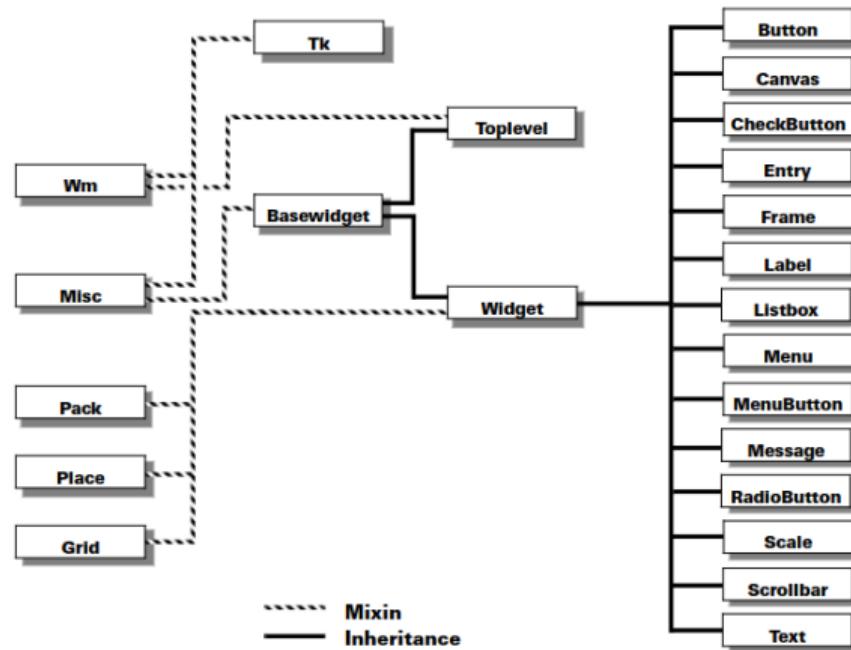
```

print(dir(root))
'''['_Misc__winfo_getint', '_Misc__winfo_parseitem', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattr__', '__getattribute__', '__getstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '__bind__', '__configure__', '_displayof', '_getboolean', '_getconfigure', '_getconfigure1', '_getdoubles', '_getints', '_grid_configure', '_gridconvalue', '_last_child_ids', '_loadtk', '_nametowidget', '_noarg_', '_options', '_register', '_report_exception', '_root', '_subst_format', '_subst_format_str', '_substitute', '_tclCommands', '_tkloaded', '_unbind', '_w', '_windowingsystem', 'after', 'after_cancel', 'after_idle', 'anchor', 'aspect', 'attributes', 'bbox', 'bell', 'bind', 'bind_all', 'bind_class', 'bindtags', 'cget', 'children', 'client', 'clipboard_append', 'clipboard_clear', 'clipboard_get', 'colormapwindows', 'columnconfigure', 'command', 'config', 'configure', 'deiconify', 'deletecommand', 'destroy', 'event_add', 'event_delete', 'event_generate', 'event_info', 'focus', 'focus_displayof', 'focus_force', 'focus_get', 'focus_lastfor', 'focus_set', 'focusmodel', 'forget', 'frame', 'geometry', 'getboolean', 'getdouble', 'getint', 'getvar', 'grab_current', 'grab_release', 'grab_set', 'grab_set_global', 'grab_status', 'grid', 'grid_anchor', 'grid_bbox', 'grid_columnconfigure', 'grid_location', 'grid_propagate', 'grid_rowconfigure', 'grid_size', 'grid_slaves', 'group', 'iconbitmap', 'iconify', 'iconname', 'iconphoto', 'iconposition', 'iconwindow', 'image_names', 'image_types', 'info_patchlevel', 'keys', 'lift', 'loadtk', 'lower', 'mainloop', 'manage', 'master', 'mazzize', 'minsize', 'nametowidget', 'option_add', 'option_clear', 'option_get', 'option_readfile', 'overrideredirect', 'pack_propagate', 'pack_slaves', 'place_slaves', 'positionfrom', 'propagate', 'protocol', 'quit', 'readprofile', 'register', 'report_callback_exception', 'resizable', 'rowconfigure', 'selection_clear', 'selection_get', 'selection_handle', 'selection_own', 'selection_own_get', 'send', 'setvar', 'size', 'sizefrom', 'slaves', 'state', 'title', 'tk', 'tk_bisque', 'tk_focusFollowsMouse', 'tk_focusNext', 'tk_focusPrev', 'tk_setPalette', 'tk_strictMotif', 'tkraise', 'transient', 'unbind', 'unbind_all', 'unbind_class', 'update', 'update_idletasks', 'wait_variable', 'wait_visibility', 'wait_window', 'waitvar', 'winfo_atom', 'winfo_atomname', 'winfo_cells', 'winfo_children', 'winfo_class', 'winfo_colormapfull', 'winfo_containing', 'winfo_depth', 'winfo_exists', 'winfo_fpixels', 'winfo_geometry', 'winfo_height', 'winfo_id', 'winfo_interps', 'winfo_ismapped', 'winfo_manager', 'winfo_name', 'winfo_parent', 'winfo_pathname', 'winfo_pixels', 'winfo_pointerx', 'winfo_pointery', 'winfo_pointerx', 'winfo_reqheight', 'winfo_reqwidth', 'winfo_rgb', 'winfo_rootx', 'winfo_rooty', 'winfo_screen', 'winfo_screencells', 'winfo_screendepth', 'winfo_screenheight', 'winfo_screenmmheight', 'winfo_screenmmwidth', 'winfo_screenvisual', 'winfo_screenwidth', 'winfo_server', 'winfo_toplevel', 'winfo_viewable', 'winfo_visual', 'winfo_visualid', 'winfo_visualsavailable', 'winfo_vrootheight', 'winfo_vrootwidth', 'winfo_vrootx', 'winfo_vrooty', 'winfo_width', 'winfo_x', 'winfo_y', 'withdraw', 'wm_aspect', 'wm_attributes', 'wm_client', 'wm_colormapwindows', 'wm_command', 'wm_deiconify', 'wm_focusmodel', 'wm_forget', 'wm_frame', 'wm_geometry', 'wm_grid', 'wm_group', 'wm_iconbitmap', 'wm_iconify', 'wm_iconmask', 'wm_iconname', 'wm_iconphoto', 'wm_icomposition', 'wm_iconwindow', 'wm_manage', 'wm_maxsize', 'wm_minsize', 'wm_overrideredirect', 'wm_positionfrom', 'wm_protocol', 'wm_resizable', 'wm_sizefrom', 'wm_state', 'wm_title', 'wm_transient', 'wm_withdraw']'''

print(root.winfo_children())                                # [<tkinter.Label object .!label>, <tkinter.Button object .!button>]

```

# Library Structure



## Class Based GUI

```
class GUI:  
    def __init__(self, master):  
        self.root = master  
        self.root.title("Class based version")  
  
root = tkinter.Tk()  
my_gui = GUI(root)  
root.mainloop()
```

## Radio Buttons

```
var = IntVar()  
for text, value in [('Passion fruit', 0), ('Loganberries', 1), ('Mangoes in syrup', 2)]:  
    Radiobutton(root, text=text, value=value, variable=var, command=radiogo).pack()  
  
def radiogo():  
    r = var.get()
```

## Check Buttons

```
var = [IntVar() for i in range(3)]
for text, value in [('Passion fruit', 0), ('Loganberries', 1), ('Mangoes in syrup', 2)]:
    Checkbutton(root, text=text, variable=var[value], command=checkgo).pack()

def checkgo():
    r1 = var[0].get()
    r2 = var[1].get()
    r3 = var[2].get()
```

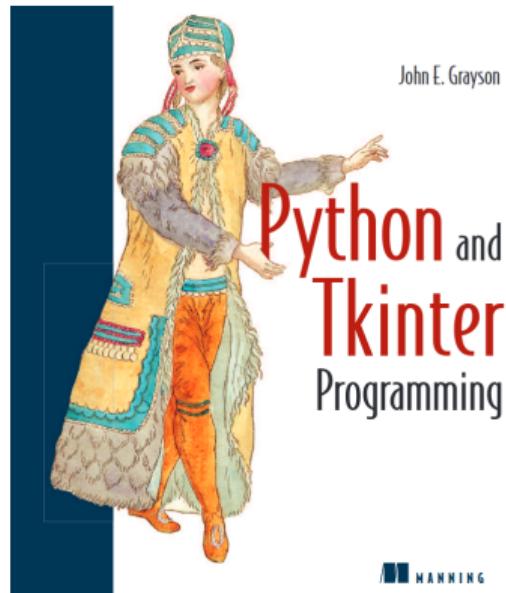
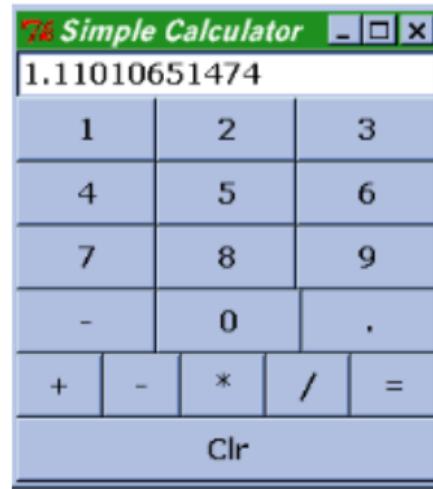


Figure 6: Reading List: Refer to Section 4.1



# Homework Task 2

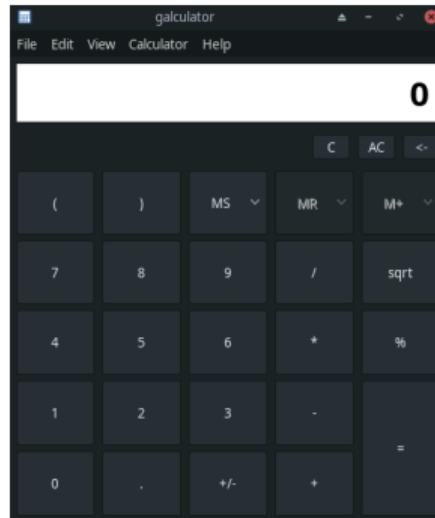


Figure 7: Modify the provided calc.py code to create this layout

## Overview



- Collaborative (Cloud + Local Cache)
- Flavours: Web ([figma.com](https://figma.com)) or Desktop
- Web: Uses google fonts
- Desktop: Uses system fonts
- Most tools at no cost (Free, 12\$/month, 45\$/month)
- Able to generate CSS, iOS, Android code

## Overview (cont.)

### Alternates

- Sketch
- Adobe XD
- Canva
- GPT (?)
- ...

### Welcome Screen

- Design File (Graphic / UI Designs)
- Figjam Board (Collaborative Whiteboard)
- Slidedeck (Presentations)

# Figjam

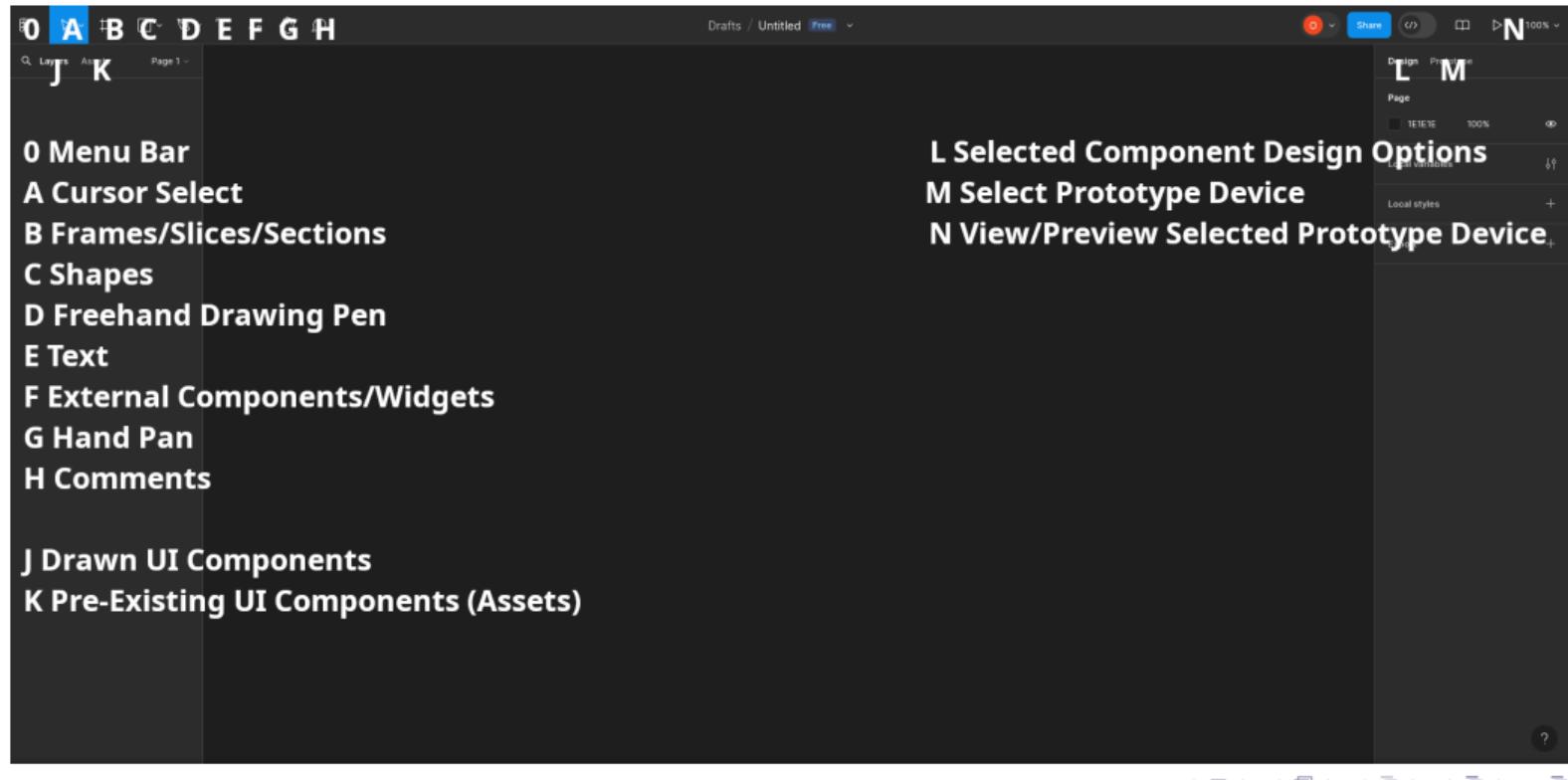


- Navigation
- Freehand Drawing Marker
- Shapes & Connectors
- Sticky Notes (for comments)
- Text Components
- Sections
- Tables
- Stamps
- Libraries / Templates



Research (Mission Statement) → Competitor Analysis → Moodboard → User Personas → Golden Path

# Design File



# Design File (cont.)

## Frames

- Used as Artboard + Container (different from Groups)

## Options

- Create Own UI components
- Download Resources (free) from web

# Figma - TkInter Interfacing

## TkInter Export

- <https://github.com/ParthJadhav/Tkinter-Designer>
- Specify API Token (Accessible from User Settings)
- Specify Figma Project Folder (Accessible from Share Option)
- Specify Local Directory for Export
- Launch using Python

# Wireframes

- Draft of a raw UI without any style, detail, or even color
- Helps teams align requirements, keeping UX design conversations focused and constructive.
- Covers:

- Screen Layouts
- Navigation Bars
- Components of UI/UX
- Interactive Elements

- Blueprint / Skeleton
- Use pen and paper if you want
- Plenty of Figma Wireframe Kits

# Wireframes (cont.)

Google search results for "wireframes figma kit".

Search bar: wireframes figma kit

Filter: Images

Results:

- Landing page wireframe
- Dashboard wireframe
- Template
- Hand drawn wireframe
- Lo fi
- Low fidelity wireframes
- Mobile app wireframe

Grid of wireframe UI kits:

- Wireframing Starter Kit | Figma
- Paper Wireframe Kit | Figma
- FREE WIREFRAMES KIT | Figma
- Website Wireframes UI Kit | Figma

- Helio - Wireframe Kit | Figma
- Website Wireframes UI Kit Vol. 1 | Figma
- Wireframe Kit | Figma
- Minimal Wireframe Kit | Figma
- Website Wireframes UI Kit | BRIX A...

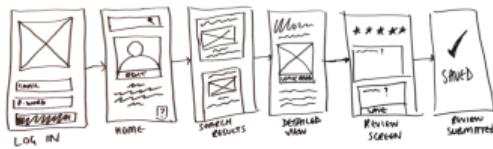
# Wireframes (cont.)

## Is It Needed?

- Working in an established/embedded design system
- Explored design needs to copy best practices (competitor lookalike)

### Low Fidelity

Basic wireframes focused on layout, basic navigation, and information architecture. Show what the interface will look like. Illustrate user flows with key UI design elements.



### Mid Fidelity

Helps iterate and shape the final design. Maps out core functionality and key interactions to designed pathways

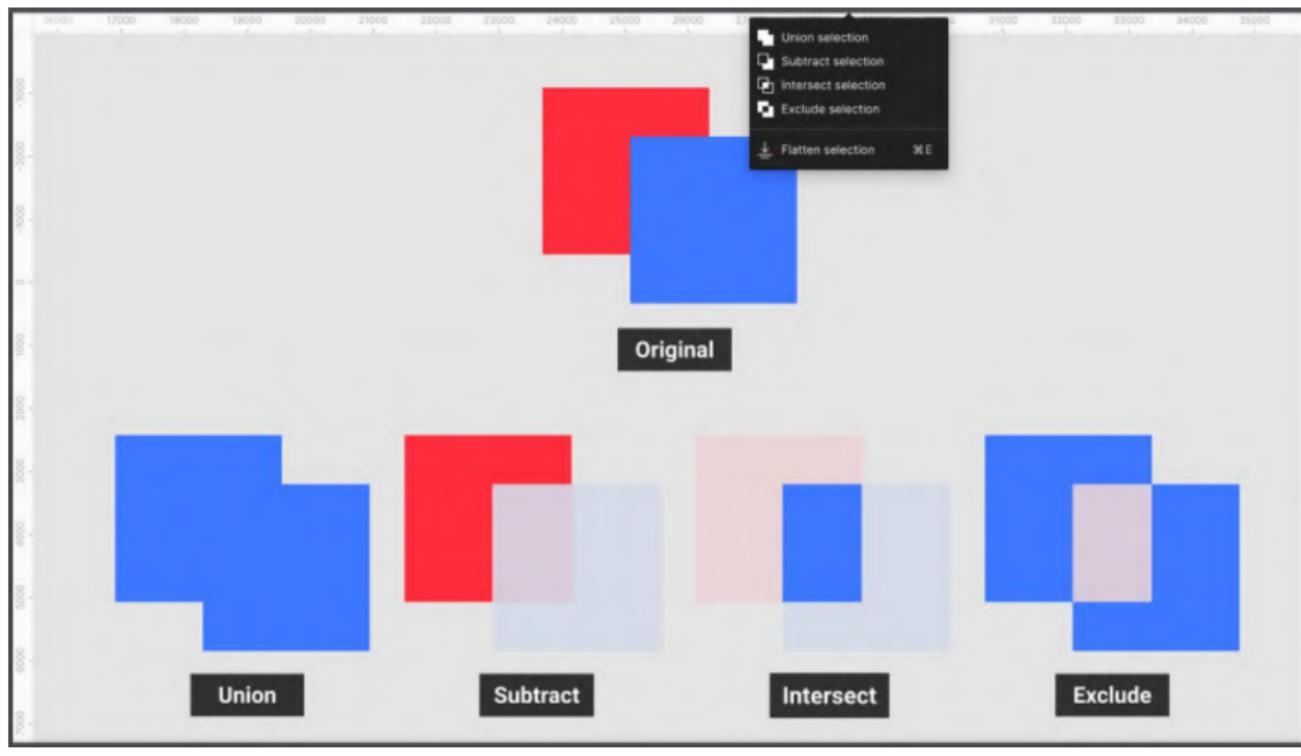


### High Fidelity

Looks like early product mockups, without the functionality of low-fidelity prototypes. Layouts, Navigation, Architectures much refined. Includes brand elements like fonts, logos.



# Preparing Complex Designs



# Preparing Complex Designs (cont.)

## Components

- Helps prepare Reusable UI Objects (Parent Child Relationship, Parent Change Can Change All Children, Child Change effects Concerned Child Only)
- Parent Component can be stored in non-visible Area
- Helpful for creating Variants

## Design Principles (Usability)

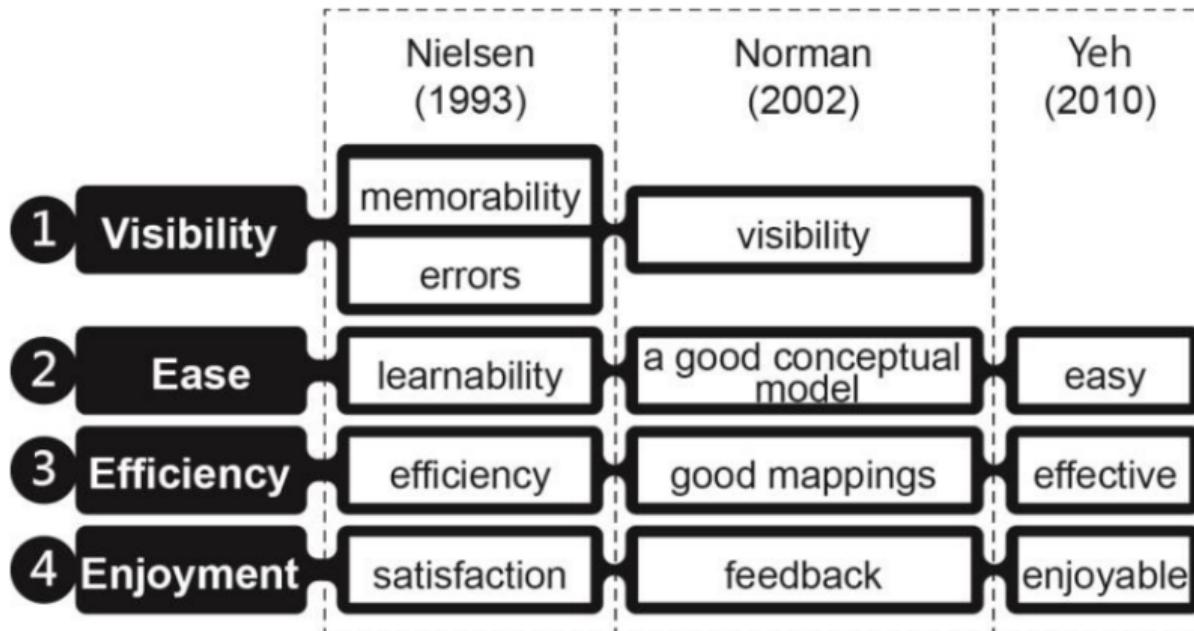


Figure 8: Wang, C.M., Huang, C.H., A Study of Usability Principles and Interface Design for Mobile e-Books, Ergonomics, 2015

# A/B Testing

## A/B Testing

Broader objective to compare two (or more) versions of a UI to find which is better? Quite flexible in choice of comparison technique.

## Preparation

- ① Prepare UI Pathways
- ② Prepare UI(s) for each node in Pathway
- ③ Collect Data for UI(s) that is linked to Usability Design Principles (e.g. Response Time, Pathway Goals, number of components, memory usage, etc.)
- ④ Normalize Data (scale or 0-1)
- ⑤ Find better score (simple comparison vs statistical testing)

## 1 Course Overview

- Premise
- Note on Programming Languages
- Job Market

## 2 UI/UX

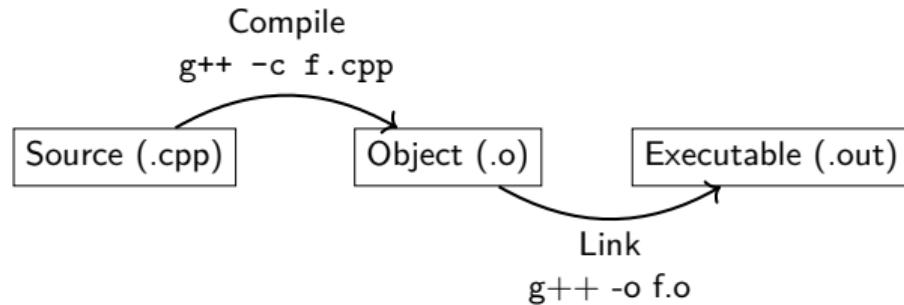
- Overview

- UI
- GUI - Tkinter
- GUI - Figma
- UX

## 3 Build and Analysis Toolchains

- C/C++ Build Pipeline
- Debugging and Profiling

# Build Pipelines



`g++ [options] source.cpp`

`clang++ [options] source.cpp`

`icpc [options] source.cpp`

`gcc [options] source.c`

`clang [options] source.c`

`icc [options] source.c`

# Sources

## Integrated Development Environment

- CodeBlocks, Vim, Sublime Text, IntelliJ, Eclipse, Xcode, Android Studio, Geany, pyCharm, ... <sup>a</sup>
- Whichever you are comfortable with.
- Desirable Features:
  - Auto Complete features.
  - Auto Compilation features.
  - Debugger integration.
  - Code folding.
  - One IDE for all (development, office work)
  - Cross-Sectional Editing.
  - Remote development.
  - Integration with versioning systems
  - ...

---

<sup>a</sup>[pypl.github.io](https://pypl.github.io) for ranking

# Sources (cont.)

## Naming and Style Conventions

- `cpp`, `hpp` in C++, and `c`, `h` in C.
- `followCamelCaseNamingConvention`
- Folder conventions: `src` for source code, `build` or `bin` for binaries, `lib` for libraries, `doc` for documentation, `test` for unit testing.
- Suffixes: `g_` for global, `s_` for static, `c_` for constant variables.
- ... Much more on [google.github.io/styleguide](https://google.github.io/styleguide) (Note: Company specific Guides)
- Can be enforced at commit time, or using static code checkers (e.g. `cpplint`)
- Indentation can be forced using `indent` with args `gnu` (for Stallman), `kr` (for Kernighan & Ritchie), `linux` (for Torvalds), `orig` (for Berkeley). Check man pages for details.

# Sources (cont.)

## Comments

- Governed by style guidelines.
- Documentation generators (Latex, HTML, XML, PDF, Man): Doxygen, Sphinx
- Doxygen Step 1: Fix settings
- Doxygen Step 2: Write comments (E.g. below)

```
/*
 * Search whether a given directed edge exists in a graph. Edges are specified as
 * head-tail pairs and must be specified in order.
 * @param G Graph object
 * @param head First incident vertex on edge
 * @param tail Second incident vertex on edge
 * @return Not Found = 0, Found = 1
 */
int directedEdgeExists(struct nGraph *G, int head, int tail) { }
```

- Doxygen Step 3: Generate documentations

# Pre-Processing

- Source code transformation by Pre-Processor
- Pre-processor behavior controlled by directives
- File Inclusions:

```
#include <iostream>
#include <boost/tokenizer.hpp>
#include "my_header_file.hpp"
#include "some_directory/my_header_file.hpp"
```

- Macro definitions

```
#define DEBUG_LEVEL 10
#define myMacro(param1, param2) param1+param2
```

- Conditional Compilation Directives

```
#if, #ifdef, or #ifndef directive
#elif
#else
#endif
```

# Compilations

## Common Arguments

- -c Compile only
- -g Debugging symbols
- -p Performance symbols
- -O*n* Optimization level to *n* (0; none, 3: full)
- -std Version (e.g. -std=c++20)
- -I Include directory
- -L Library path
- -l*txt* Linking with library (lib*txt*) file
- -Wall enable most warnings
- -Wextra enable extra warnings
- -Werror treat warnings as errors

## Compilations (cont.)

### ELF64

- Executable and Linkable Format (for all executables, object code, libraries)
- Described in `elf.h` of Linux kernel
- Learning Benefits: Digital forensics, OS internals, Malware research
- Viewable using `readelf`

#### ELF File Structure

- ELF Header (64 bytes, `-h` switch)
- Program Header (`-l` switch)
- Section Header (`-S` switch)

#### ELF File Types

- Relocatable
- Executable
- Shared Object Files

## Compilations (cont.)

ELF Header:

```
Magic: 7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00  
Class: ELF64  
Data: 2s complement, little endian  
Version: 1 (current)  
OS/ABI: UNIX - System V  
ABI Version: 0  
Type: REL (Relocatable file)  
Machine: Advanced Micro Devices X86-64  
Version: 0x1  
Entry point address: 0x0  
Start of program headers: 0 (bytes into file)  
Start of section headers: 384 (bytes into file)  
Flags:  
Size of this header: 64 (bytes)  
Size of program headers: 0 (bytes)  
Number of program headers: 0  
Size of section headers: 64 (bytes)  
Number of section headers: 9  
Section header string table index: 1
```

Figure 9: ELF File Header `readelf -h obj.o`

# Linking Process

## Static Linking

- Code and variables resolved at compile time by interpreter ld, and copied into target application as a stand-alone executable
- .a filename convention
- Benefits: No dependency problems (single executable file)
- Cost: Large file size, Library code possibly loaded multiple times in memory

```
// file get15.c  
// gcc -c get15.c
```

```
int get15() {  
    return 15;  
}
```

```
// ar -cvr libget15.a get15.o
```

```
// file main.c  
// gcc main.c get15.o  
// gcc main.c libget15.a  
// gcc main.c -lget15
```

```
int main() {  
    return get15();  
}
```

## Linking Process (cont.)

### Dynamic Linking

- Code and variables resolved at load time by runtime interpreter ld-linux.so.2, and copied into target executable as symbols
- .so file convention
- Benefit: Small file size, library code loaded once in shared memory
- Cost: Dependency management

```
// file get15.c  
// gcc -shared get15.c -o libget15.so
```

```
// file main.c  
// gcc main.c -lget15
```

## Linking Process (cont.)

```
# Output: readelf -S a.out
[21] .dynamic      DYNAMIC      0000000000003de8  00002de8
     00000000000000001f0  000000000000000010  WA      7      0      8

#Output: readelf -d a.out
Dynamic section at offset 0x2de8 contains 27 entries:
  Tag      Type           Name/Value
 0x0000000000000001 (NEEDED)   Shared library: [libget15.so]
 0x0000000000000001 (NEEDED)   Shared library: [libc.so.6]
 0x000000000000000c (INIT)    0x1000
# ... continues
```

- Use symbolic links when dealing with multiple versions

```
gcc -shared -Wl,-soname,libget15.so.1 -o libget15.so.1.0 get15.c
ln -sf libget15.so.1.0 libget15.so.1
ln -sf libget15.so.1.0 libget15.so
```

## Automated Builds



- Single-file programs do not work well when code gets large
- Larger programs are split into multiple files

## Automated Builds (cont.)

- Retyping commands is wasteful (Use ↑ or CTRL+R as shortcut)

### GNU Make

- A utility for automatically compiling ("building") executables and libraries from source code.
- Often used for C programs, but not language-specific
- Follows Makefile format

```
myprogram : file1.c file2.c file3.c  
          gcc -o myprogram file1.c file2.c file3.c
```

- Launch as `make` for first target, or `make myprogram` with direct target name
- Runs commands only if needed (based on timestamp)

## Automated Builds (cont.)

```
all: aprogram

aprogram : foo.o bar.o
    gcc -o aprogram foo.o bar.o

foo.o: foo.c
    gcc -c foo.c

bar.o: bar.c
    gcc -c bar.c
```

- Standard Makefile targets: all, install, clean, distclean, ...
- Standard Variables: CC, CFLAGS, CXX, CXXFLAGS, LDFLAGS, ...

# Debugging Using GDB / DDD

## Code Debugging

- Step through a program line by line
- Inspect variables and objects as it steps through
- Inspect disassembled code as it steps through
- Inspect call stack as it steps through

# Debugging Using GDB / DDD (cont.)

## Code Debugging

### GNU Debugger GDB

- Compile Time: `gcc -g myCode.c`
- Run Time: `gdb ./a.out`, followed by `run arg1 arg2`
- NCurses based frontend using `gdb ./a.out -tui`, or launching as normal, and issuing `layout src` after inserting any breakpoint.
- Commands:
  - Breakpoints `break file.c:10`, OR `break 10`, OR `break myFunc`
  - Delete breakpoints using `delete` or specifically by name
  - To view code: `list`
  - To view disassembled code: `mi|disassemble myFunc`
  - Iterate through code: `continue`, `step`, `next`
  - Inspect variables using: `print variableName`

# Debugging Using GDB / DDD (cont.)

## Code Debugging

[GNU Project - Software](#)



- [About DDD](#)
- [DDD News](#)
- [Getting DDD](#)
- [Building DDD](#)
- [Documentation](#)
- [Alpha Releases](#)
- [Reporting Bugs](#)
- [Where can I learn more about the debuggers DDD uses?](#)
- [Help and Assistance](#)
- [References](#)

## What is DDD?

GNU DDD is a graphical front-end for command-line debuggers such as [GDB](#), [DBX](#), WDB, [Ladebug](#), JDB, XDB, [the Perl debugger](#), the bash debugger [bashdb](#), the GNU Make debugger [remake](#), or the Python debugger [pydb](#). Besides ``usual'' front-end features such as viewing source texts, DDD has become famous through its interactive graphical data display, where data structures are displayed as graphs.



# Debugging for Memory Problems using Valgrind

## Memory Profilers

### Typical Memory Problems

- Uninitialized Variables
- Read/Write to un-allocated Memory (maybe segfaults generated)
- Deleting or Freeing dynamically created memory twice
- Memory Leaks

```
void f() {  
    int *x = malloc(10 * sizeof(int));  
    x[10] = 0;  
}  
  
int main(int argc, char *argv[]) {  
    int n, i;  
    f();  
    for (i = 0; i < n; i++);  
    return 0;  
}
```

```
valgrind --leak-check=full ./a.out
```

# Debugging for Memory Problems using Valgrind (cont.)

## Memory Profilers

```
==23294== Invalid write of size 4
==23294==   at 0x108728: f (in /home/omar/work/codes/c/valgrind/a.out)
==23294==   by 0x108749: main (in /home/omar/work/codes/c/valgrind/a.out)
==23294== Address 0x5204068 is 0 bytes after a block of size 40 allocated
==23294==   at 0x4C2EF1F: malloc (vg_replace_malloc.c:299)
==23294==   by 0x10871B: f (in /home/omar/work/codes/c/valgrind/a.out)
==23294==   by 0x108749: main (in /home/omar/work/codes/c/valgrind/a.out)
==23294==
==23294== Conditional jump or move depends on uninitialised value(s)
==23294==   at 0x10875D: main (in /home/omar/work/codes/c/valgrind/a.out)
==23294==
==23294== More than 10000000 total errors detected. I am not reporting any more.
==23294==
==23294== HEAP SUMMARY:
==23294==   in use at exit: 40 bytes in 1 blocks
==23294==   total heap usage: 1 allocs, 0 frees, 40 bytes allocated
==23294==
==23294== 40 bytes in 1 blocks are definitely lost in loss record 1 of 1
==23294==   at 0x4C2EF1F: malloc (vg_replace_malloc.c:299)
==23294==   by 0x10871B: f (in /home/omar/work/codes/c/valgrind/a.out)
==23294==   by 0x108749: main (in /home/omar/work/codes/c/valgrind/a.out)
==23294==
```

# Debugging for Memory Problems using Valgrind (cont.)

## Memory Profilers

```
==23294== LEAK SUMMARY:
==23294==    definitely lost: 40 bytes in 1 blocks
==23294==    indirectly lost: 0 bytes in 0 blocks
==23294==    possibly lost: 0 bytes in 0 blocks
==23294==    still reachable: 0 bytes in 0 blocks
==23294==    suppressed: 0 bytes in 0 blocks
```

# Observing Memory Usage Behavior using Massif

## Memory Profilers

- Memory usage as a function of allocation/deallocation event on heap (including stack)
- Usage: `valgrind --tool=massif --time-unit=B ./a.out`
- Visualization usage: `massif-visualizer massif.out.pid`

# Observing Memory Usage Behavior using Massif (cont.)

## Memory Profilers



# Performance Measurement

## Code Profilers

### Profiler

An analysis that may measure usage of instructions in terms of their frequency or duration of execution. The goal of profiling is to aid program optimization.

- Common and Powerful Tools for Profiling Code: time, Fine-grained profiling, callgrind, gprof, oprofile, ...

```
time ./a.out
time ./a.out
# output of my program, if any.
real 0m0.003s # Wall clock time
user 0m0.001s # Userspace (accumulated CPU for both library + user space)
sys  0m0.001s # Kernelspace (I/O and/or syscalls)
```

- General Tip: Run for sufficiently long period + allow for relaxation period.

# Performance Measurement (cont.)

## Code Profilers

### Fine Grained profiling

- Multiple Solutions !!!

```
struct timespec start, finish;           // Defined in time.h
// struct timespec contains tv_sec, and tv_nsec
// struct timeval contains tv_sec, and tv_usec

/* vs CLOCK_REALTIME (NTP dependent) */
clock_gettime(CLOCK_MONOTONIC, &start);
    /* Code Here */
clock_gettime(CLOCK_MONOTONIC, &finish);

double elapsed = (finish.tv_sec - start.tv_sec);
elapsed += (finish.tv_nsec - start.tv_nsec) / 1e9;
```

# Performance Measurement (cont.)

## Code Profilers

### callgrind

- Part of valgrind,
- Compiler Flags: `gcc -g myCode.c`
- Call using valgrind: `valgrind --tool=callgrind ./a.out`
- View output on Terminal: `callgrind_annotate --auto=yes callgrind.out.NNNN`
- View output in GUI: `kcacheGrind callgrind.out.NNNN`

# Performance Measurement (cont.)

## Code Profilers

### gprof

- Compile as: `gcc myCode.c -pg`
- Run as: `./a.out` (will be slower than without profiling), output in `gmon.out`
- View gprof using: `gprof ./a.out`

%	cumulative	self		self	total	
time	seconds	seconds	calls	s/call	s/call	name
39.47	12.01	12.01	1	12.01	21.38	func1
30.79	21.38	9.37	1	9.37	9.37	func2
30.79	30.75	9.37	1	9.37	9.37	new_func1
0.13	30.79	0.04				main

# Performance Measurement (cont.)

## Code Profilers

```
#include <stdio.h>

void func1(void)           void func2(void)           void new_func1(void)
{   printf("\n Inside func1 \n");   {   printf("\n Inside func2 \n");   {   printf("\n Inside new_func1()\n");
    for(int i = 0; i<0xffffffff; i++);   for(int i = 0; i<0xfffffffcaa; i++);   for(int i = 0; i <0xfffffffbee; i++);
    new_func1();                         return;                     return;
    return;                           }                         }
}                                     }

int main(void)
{
    printf("\n Inside main()\n");
    for(int i = 0; i<0xfffffff; i++);
    func1();
    func2();

    return 0;
}
```

# Performance Measurement (cont.)

## Code Profilers

### Call Graph Output

	index	%	time	self	children	called	name
[1]		100.0		0.04	30.75		main [1]
				12.01	9.37	1/1	func1 [2]
				9.37	0.00	1/1	func2 [3]
							-----
[2]		69.4		12.01	9.37	1/1	main [1]
				12.01	9.37	1	func1 [2]
				9.37	0.00	1/1	new_func1 [4]
							-----
[3]		30.4		9.37	0.00	1/1	main [1]
				9.37	0.00	1	func2 [3]
							-----
[4]		30.4		9.37	0.00	1/1	func1 [2]
				9.37	0.00	1	new_func1 [4]
							-----

# Performance Measurement (cont.)

## Code Profilers

### OProfile

- Install it
- Run any code as: `sudo operf ls` (previously it was a kernel module as `CONFIG_OPROFILE`)
- output written to: `oprofile_data` in current directory
- View output as: `opreport`
- Cleanup by removing directory

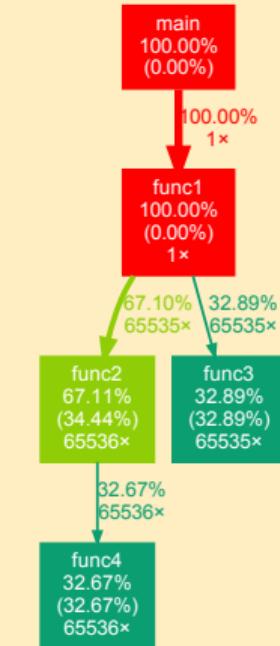
# Performance Measurement (cont.)

## Code Profilers

### gprof2dot

- Can be called with any profiler tools
- Generates interesting graphics for documentation / publications
- Usage:

```
gprof ./a.out | gprof2dot | dot -Teps -o output.eps
```



## Code Coverage using Gcov

- Find out different un-used portions of code during various tests.

- Compile using:

```
gcc -Wall -fprofile-arcs -ftest-coverage myCode.c
```

or

```
gcc myCode.c --coverage
```

- Run your program

- Call gcov myCode.c

```
for (i = 1; i < 10; i++)
{
    if (i % 3 == 0)
        printf ("%d is divisible by 3\n", i);
    if (i % 11 == 0)
        printf ("%d is divisible by 11\n", i);
}
```

## Code Coverage using Gcov (cont.)

### Coverage output

```
10:  8:  for (i = 1; i < 10; i++)
-:  9:    {
9: 10:      if (i % 3 == 0)
3: 11:        printf ("%d is divisible by 3\n", i);
9: 12:      if (i % 11 == 0)
#####: 13:        printf ("%d is divisible by 11\n", i);
-: 14: }
```

# Cross Compilation

- Compiler support required to merge object files of different languages
- Why? Performance and Native Calls

## Wrapper Libraries

- C extension modules (for Python, Cython)
- Java extension modules (for Python, Jython)
- Mex files (for Matlab)
- Swift - Objective C extensions
- ...

## Cross Compilation (cont.)

### Example 1 (C Code)

```
from ctypes import *
so_file = "./fputs.so"
myCFunctions = CDLL(so_file)

myCFunctions.myfputs(b"Hello", b"write.txt")

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int myfputs(char *s, char *f) {
    FILE *fp = fopen(f, "w");
    int ret = fputs(s, fp);
    fclose(fp);
    return ret;
}
```