



Curtin University

# Graphical User Interfaces

ISYS5002, School of Marketing and Management

## ***ELECTRONIC WARNING NOTICE FOR COPYRIGHT STATUTORY LICENCES***

### **WARNING**

This material has been reproduced and communicated to you by or on behalf of **Curtin University** in accordance with section 113P of the *Copyright Act 1968 (the Act)*

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.



I acknowledge the traditional custodians of  
the land on which I work and live, and  
recognise their continuing connection to land,  
water and community. I pay respect to elders  
past, present and emerging.



# Today

- Discuss User Interfaces
- Explain different interaction styles
- User support which should be built-in to user interfaces
- See how to create an 'app' using Python notebooks



# The User Interface

- System users often judge a system by its interface rather than its functionality
- A poorly designed interface can cause a user to make catastrophic errors
- Poor user interface design is the reason why so many software systems are never used



# Graphical User Interface

<b>Characteristic</b>	<b>Description</b>
Windows	Multiple windows allow different information to be displayed simultaneously on the user's screen.
Icons	Icons different types of information. On some systems, icons represent files; on others, icons represent processes.
Menus	Commands are selected from a menu rather than typed in a command language.
Pointing	A pointing device such as a mouse is used for selecting choices from a menu or indicating items of interest in a window.
Graphics	Graphical elements can be mixed with text on the same display.



# Advantages

- They are easy to learn and use. Users without experience can learn to use the system quickly.
- The user may switch quickly from one task to another and can interact with several different applications.
- Information remains visible in its own window when attention is switched.
- Fast, full-screen interaction is possible with immediate access to anywhere on the screen



# UI Design Principles

- User familiarity
- Consistency
- Minimal surprise
- Recoverability
- User guidance
- User diversity (Accessibility – ISYS3004)



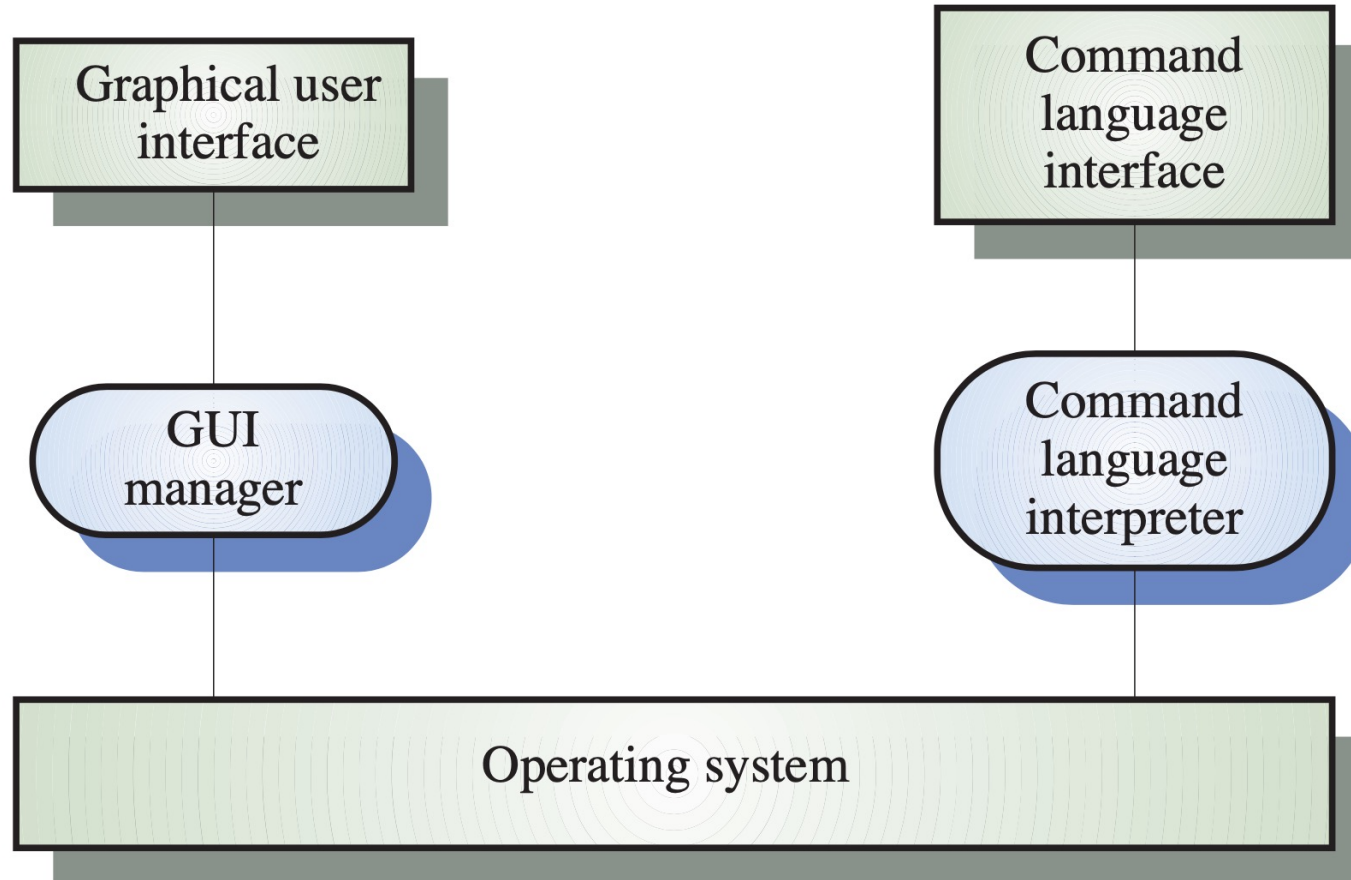


# Interaction Styles

- Direct manipulation (video games)
- Menu selection (general software)
- Form fill-in (loan application, stock control)
- Command language (operating systems)
- Natural language (Siri, Hey Google)



# Modern Systems - Multiple Interfaces



# User Support

- Guidance (Tool Tips)
- Help System
  - I want information
  - I need help!
- Error Messages
- User Documentation



# GUI - History

- Punch Cards
- Text-Based (Teletype)
- Text-Based (Monitor)
- Sketch Pad (Ivan Sutherland's light pen)
- Stanford Online System (hyperlinks)
- Xerox PARC (first mouse)
- . . . .



# Common Widgets

- Buttons
- Radio buttons
- Check Boxes
- Text Boxes
- Sliders
- Date Pickers
- Colour Pickers
- Tabs
- Canvas
- Plots/Charts



# App (in a notebook)

- **Change variables**
- **Use input()**
- **Colab forms**
- **Widgets**
  - Forms**
  - Dashboard**
- **Share**
  - Module/Package (Nbdev -> PyPi)
  - Interactive (Binder, NBViewer)
  - Report (Slides, HTML, PDF)



# App in a notebook

jupyter example\_app Last Checkpoint: 2 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 2

Appmode

```
In [1]: from future import division
import ipywidgets as ipw

output = ipw.Text(placeholder="0", layout=ipw.Layout(width="212px"), disabled=True)

def on_click(btn):
    if btn.description == "=":
        try:
            output.value = str(eval(output.value))
        except:
            output.value = "ERROR"
    elif btn.description == "AC":
        output.value = ""
    elif btn.description == "del":
        output.value = output.value[:-1]
    else:
        output.value = output.value + btn.description

def mk_btn(description):
    btn = ipw.Button(description=description, layout=ipw.Layout(width="50px"))
    btn.on_click(on_click)
    return btn

row0 = ipw.HBox([mk_btn(d) for d in ("(", ")", "del", "AC")])
row1 = ipw.HBox([mk_btn(d) for d in ("7", "8", "9", "/" )])
row2 = ipw.HBox([mk_btn(d) for d in ("4", "5", "6", "*" )])
row3 = ipw.HBox([mk_btn(d) for d in ("1", "2", "3", "-" )])
row4 = ipw.HBox([mk_btn(d) for d in ("0", ".", "=", "+" )])
ipw.VBox((output, row0, row1, row2, row3, row4))
```

0

(	)	del	AC
7	8	9	/
4	5	6	*
1	2	3	-
0	.	=	+

In [ ]:



jupyter Edit App Logout

6 \* 7

(	)	del	AC
7	8	9	/
4	5	6	*
1	2	3	-
0	.	=	+




# App (from a notebook)

- Native GUI (**tkinter**, pyQT, wxPython, PySide )  
Cross platform can be difficult
- Web Page (**cherrypy**, bottle, flask, Django, pyscript)  
Need to know some HTML and CSS
- Web App (**Anvil**, Streamlit, Volia, Dash, Panel)
- Native Cross Platform App (Kivy, Beeware)

Curtin University

# App from a Notebook (anvil)

 anvil

Built with Anvil

## Pay Calculator

Name

Your Name

Date

Salary

4000

Sales

1000

CALCULATE

# Can you

- Suggest some general design principles for user interface design
- Explain different interaction styles
- Describe the user support which should be built-in to user interfaces
- Outline some of the ways create an 'app' using notebooks

