



# Graphical User Interfaces

Using Python `tkinter`

Dr Ian Cornelius



**Hello**

# Hello (1)

## Learning Outcomes

1. Understand the concept of GUI and their purpose in Python
2. Demonstrate knowledge on how to use GUI widgets in a body of work



# Graphical User Interfaces

# Graphical User Interfaces (1)

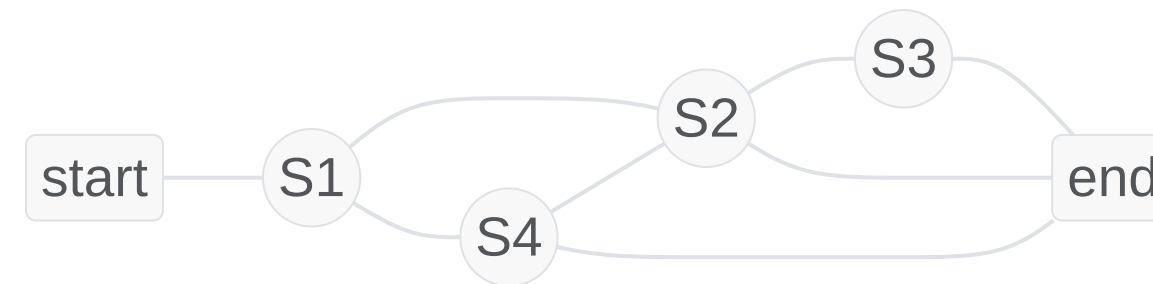
- Graphical User Interfaces is often abbreviated to GUI
- Helps a user to interact with the software through the use of *visual indicators*
- To-date you have created applications that are procedural in pattern
  - i.e. it follows a series of steps in a pre-determined pattern
- GUIs are different to Command-Line Interfaces (CLI)
  - a CLI will perform a series of tasks in a pre-determined order
  - a GUI will wait for a series of inputs from the user
    - any tasks performed are in control by the user



# Graphical User Interfaces (2)

## Event Driven Programming

- Event driven programming is where a program reacts to events
- An *event* has some sort of action that is associated to it
  - the order and frequency of an event is unpredictable
- Event driven programming does not have a predefined sequence of actions to be performed
  - nor does it have a pre-defined end



# Graphical User Interfaces (3)

## GUI Programming

- GUIs consist of the following structure:
  - a selection of icons and widgets displayed to the user, organised inside a window
  - functions that will process user and application events
  - association of user events with specific functions
  - an infinite conditional statement to process user events
- Python does not support GUI or event driven programming natively
- GUIs are implemented using an additional library
  - i.e. `tkinter`, `PySide`, and `wxPython`

[Full List of Additional GUI Libraries](#)



# tkinter Module



# tkinter Module (1)

- For this module, you will be introduced to `tkinter`
  - a GUI module that is pre-installed with Python
  - an abbreviation for **TK Interface**
- Requires loading the library/module `tkinter` to be used

```
import tkinter
```

- `tkinter` is a platform independent package that consists of a variety of GUI elements
  - e.g. button, label, menu, frame etc.
  - the GUI elements are often referred to as **widgets**
- Provides an object-oriented interface to the GUI toolkit

# tkinter Module (2)

## tkinter Widgets i

- The module consists of 15 widgets to aid in building a GUI

Widget	Description
Button	Displays a button in your application.
Canvas	Enables you to draw shapes, such as lines, circles and rectangles.
Check Button	Can be used to display a number of options as a checkboxes. The user is able to select multiple options at a time.
Entry	Displays an input field for a single line of text. Allows a user to enter data.
Frame	This is a container, and is where all other widgets will be organised.
List Box	Provides a list of options to a user.

# tkinter Module (3)

## tkinter Widgets ii

Widget	Description
Menu	Provides a method to display various commands to the user. The commands themselves are stored in a Menu Button.
Menu Button	Used to create a menu in your application.
Message	Displays an input field to display multi-line text.
Radio Button	Displays a number of options to the user as a radio button. The user can only select <i>one</i> option at a time.

# tkinter Module (4)

## tkinter Widgets iii

Widget	Description
Scale	Used to display a slider to the user.
Scrollbar	Provides scrolling capability to various widgets, i.e. list-boxes and messages.
Text	Can be used for the user to insert multi-lines of text to the user.

# tkinter Module (5)

## tkinter Widgets iv

Widget	Description
Top Level	Used to provide a separate window container.
Spin Box	Displays a fixed number of values to the user for selection.
Paned Window	A container that may contain any number of panes that are arranged horizontally or vertically.
Label Frame	A simple container that acts as a spacer or container for complex layouts.
Message Box	Enables the developer to display messages to the user.

# tkinter Module (6)

## Widget Attributes i

- Each widget will consist of an *attribute*
  - an attribute is an underlying method/function of a widgets instance
- There are a standard list of attributes available for each widget:
  1. Dimensions
  2. Colours
  3. Fonts
  4. Anchors
  5. Relief Styles
  6. Bitmaps

# tkinter Module (7)

## Widget Attributes ii

### Dimensions

- A variety of dimensions for a widget can be described using different units
  - a dimension could be: length, width, height etc.
- If a dimension of a widget is set as an integer, it is assumed to be in pixels
- A particular measurement type can be set by using a string literal

Measurement Type	Example
Centimeter	"1c"
Inch	"1i"
Millimeter	"10mm"

# tkinter Module (8)

## Widget Attributes iii

### Colours

- Specifying colours can be achieved in one of two ways:
  1. Use a string to specify the red, green and blue (RGB) values in hexadecimal
  2. Use a locally defined color name

### Method 1: Hexadecimal Values

- Strings are used to define the red, green and blue (RGB) values in a hexadecimal format
  - `#rgb` four bits per colour, i.e. `#fff` is white
  - `#rrggbb` eight bits per colour, i.e. `#000000` is black
  - `#rrrrggggbbb` twelve bits per colour, i.e. `#000fff000` is green

### Method 2: Defined Colour Name

- Locally defined names for a colour can be used
  - i.e. `'white'`, `'black'`, `'red'`, `'green'`



# tkinter Module (9)

## Widget Attributes iv

### Fonts

- Specifying a font can be done in one of two ways

### Method 1: Tuple Format

- Tuples can be used to store the font family, size and formatting style

```
# 0           # 1           # 2  
( 'Helvetica', '16', 'bold underline' )
```

Index	Definition
0	Defines to the font family
1	Defines the font size
2	Defines to the formatting of the text

# tkinter Module (10)

## Widget Attributes v

### Fonts continued

#### Method 2: Font Objects

- Font objects can be created using the `tkFont` module
- Use the constructor of the `Font` class and a variety of options to change the formatting

Option	Definition
<code>family</code>	Defines the font family to be used as a string
<code>size</code>	Defines the font size
<code>weight</code>	<code>bold</code> for bold formatting, <code>normal</code> for standard formatting
<code>slant</code>	<code>italic</code> for italic formatting, <code>roman</code> for standard formatting
<code>underline</code>	<code>1</code> for underline formatting, <code>0</code> for no underline
<code>overstrike</code>	<code>1</code> for overline formatting, <code>0</code> for no over-strike

- An example:

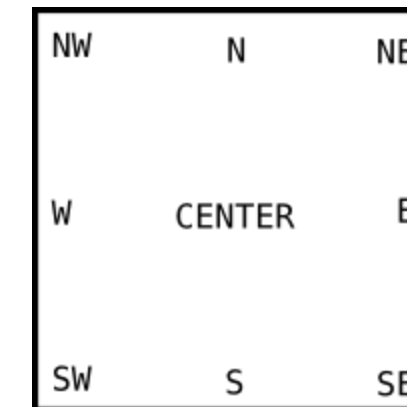
```
import tkFont
font1 = tkFont.Font(family='Helvetica', size=16, weight='bold', underline=1)
```

# tkinter Module (11)

## Widget Attributes vi

### Anchors

- Anchors are constants to control where items are positioned relative to their context
  - i.e. an anchor can specify where a widget is located inside a frame
- The constants are given as a compass point
  - i.e. north is top and west is left

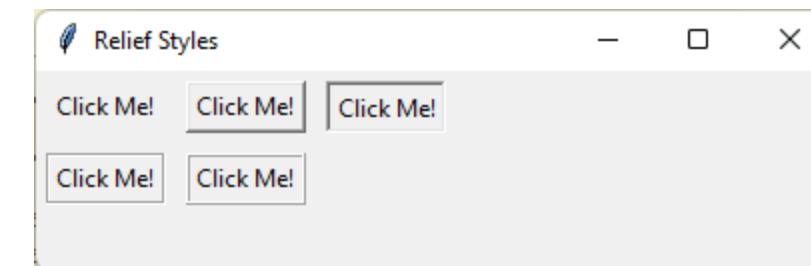


# tkinter Module (12)

## Widget Attributes vii

### Relief Styles

- Provides a simulated three-dimensional effect around the outside of a widget
- The width of the borders are dependant upon the `borderwidth` option













# tkinter Module (13)

## Widget Attributes viii

### Bitmaps

- Bitmaps can be used to add icons to buttons etc.

Bitmap Value	Icon	Bitmap Value	Icon
error		gray75	
gray50		gray25	
gray12		hourglass	
info		questhead	
question		warning	

- Your own bitmap icons can also be used, using the file extension `.xbm`
- Instead of using a string from the table above, provide a string with the following:
  - `@` symbol
  - path to the bitmap file

```
@directory/file_name.xbm
```



**Goodbye**

# Goodbye (1)

## Questions and Support

- Questions? Post them on the **Community Page** on Aula
- Additional Support? Visit the [Module Support Page](#)
- Contact Details:
  - Dr Ian Cornelius, [ab6459@coventry.ac.uk](mailto:ab6459@coventry.ac.uk)