Overview

A `label` is a widget that displays text or images, typically that users will just view but not otherwise interact with. Labels are used to identify controls or other parts of the user interface, provide textual feedback or results, etc.

Important Information

Creating a Label

Labels are created using the `ttk.Label` class. Often, the text or image the label will display are specified via configuration options at the same time:

```
# Create a label and text
# parent represents whatever container you want to put it in. This can be the root, or a frame.
label = ttk.Label(parent, text='Full name:')
```

Like frames, labels can take several different configuration options, which can alter how they are displayed.

Displaying Text

The `text` configuration option (shown above when creating the label) is the most commonly used, particularly when the label is purely decorative or explanatory. You can change what text is displayed by modifying this configuration option. This can be done at any time, not only when first creating the label.

```
# Set the label's text
label['text'] = "Hello World!"
```

Setting Text to a Variable that Changes

You can also have the widget monitor a variable in your script. Whenever the variable changes, the label will automatically update to display the new value of the variable. This is done with the `textvariable` configuration option:

```
# Create a variable that is an object of the StringVar class
resultsContents = StringVar()
# Attach the variable to the label, it will automatically update whenever the value is changed
label['textvariable'] = resultsContents
# Set a string value to the variable that the textvariable updates to
resultsContents.set('Initial value to display')
```

Tkinter only allows you to attach widgets to an instance of the `StringVar` class but not arbitrary Python variables. This class contains all the logic to watch for changes and communicate them back and forth between the variable and Tk. Use the get and set methods to read or write the current value of the variable.

```
# Get the value of the string variable
current = resultsContents.get()
# Set the value of the string variable
resultsContents.set('New value to display')
```

Displaying Images

Labels can also display an image instead of text. If you just want a static image displayed in your user interface, this is normally the way to do it. We'll go into images in more detail in a later chapter, but for now, let's assume you want to display a PNG stored in a file on disk. This is a two-step process. First, create an image "object." Then, tell the label to display that object via its image configuration option:

```
# Create a variable to store the image
image = PhotoImage(file='myimage.png')
# Set the label's image to the variable
label['image'] = image
```

Text and Images

Labels can also display both an image and text at the same time. You'll often see this in toolbar buttons. To do so, use the `compound` configuration option, which accepts the following values:

Configuration Option	Result
`none` *(default)*	display only the image if present; if there is no image, display the text specified by the `text` or `textvariable` options
`text`	text only (ignore any image provided)
`image`	image only (ignore any text provided)
`center`	text in the center of the image
`top`	image above text
`bottom`	image below text
`left`	image to left of text
`right`	image to right of text

For example:

```
# Create the label
label = Label(root)
# Create a variable to store the image
image = PhotoImage(file='smile.png')
# Add the image to the label
label['image'] = image
# Add text to the label (this can also be a textvariable instead of just text)
label['text'] = 'Smile'
# Set the compound configuration to choose how to display the image and text in relation to each other
label['compound'] = 'center'
# Add the label to the grid
label.grid()
```

Images and Resizing

An important part to understand about the `tkinter` library is that images **cannot** be resized. The image file you access and display will always display as its full resolution. If you want the image to display as a different size, you will need to do some of your own research into libraries that will allow you to do this.

A good place to start your research is by looking into the `Pillow` library.

Fonts

As with frames, you normally don't want to change things like fonts and colors directly. If you need to change them (e.g., to create a special type of label), the preferred method would be to create a new style, which is then assigned to the widget with its `style` option.

Unlike most themed widgets, the label widget also provides explicit widget-specific configuration options for fonts and colors as an alternative. Again, you should use these only in special one-off cases when using a style doesn't necessarily make sense.

You can specify the font used to display the label's text using the font configuration option. Here are the names of some predefined fonts you can use:

Font	Description
`TkDefaultFont`	Default for all GUI items not otherwise specified.
`TkTextFont`	Used for entry widgets, listboxes, etc.
`TkFixedFont`	A standard fixed-width font.
`TkMenuFont`	The font used for menu items.
`TkHeadingFont`	A font for column headings in lists and tables.
`TkCaptionFont`	A font for window and dialog caption bars.
`TkSmallCaptionFont`	Smaller captions for subwindows or tool dialogs.
`TkIconFont`	A font for icon captions.
`TkTooltipFont`	A font for tooltips.

For example:

```
# Set the font of the label
label['font'] = "TkDefaultFont"
```

Colors

The foreground (text) and background color of the label can also be changed via the `foreground` and `background` configuration options. You can specify colors as either color names (e.g., `red`) or hex RGB codes (e.g., `#ff34b3`).

For example:

```
# Set the text color of the label
label['foreground'] = "red"
# Set the background color of the label
label['background'] = "#00346a"
```

Layout

The geometry manager determines the overall layout of the label (i.e., where it is positioned within the user interface and how large it is). Yet, several options can help you control how the label is displayed within the rectangle the geometry manager gives it.

If the box given to the label is larger than the label requires for its contents, you can use the `anchor` option to specify what edge or corner the label should be attached to, which would leave any empty space in the opposite edge or corner. Possible values are specified as compass directions: `'n'` (north, or top edge), `'ne'`, (north-east, or top right corner), `'e'`, `'se'`, `'s'`, `'sw'`, `'w'`, or `'center'`.

For example:

```
# Set the label's anchor the the top left corner of the rectangle it is in
label['anchor'] = "nw"
```

Multi-line Labels

Labels can display more than one line of text. To do so, embed carriage returns (`\n`) in the `text` (or `textvariable`) string.

```
# Add text to the label (this can also be a textvariable instead of just text)
label['text'] = 'This will be on the first line!\nThis will be on the second line!'
```

Labels can also automatically wrap your text into multiple lines via the `wraplength` option, which specifies the maximum length of a line (in pixels, centimeters, etc.).

```
# Wrap the text if it goes over 100 pixels
label['wraplength'] = '100'
```

You can also control how multi-line text is justified via the `justify` option. It can have the values `left`, `center`, or `right`. If you have only a single line of text, you probably want the `anchor` option instead.

```
# Justify the text of a multiline label to the center
label['justify'] = 'center'
```

Copy, Change, Challenge

Copy

```
from tkinter import *
from tkinter import ttk

root = Tk()
root.title("Label Copy Example")

# Create a StringVar to control the label text
message = StringVar()
message.set("Hello, world!")

# Create the label
label = ttk.Label(root, textvariable=message)

# Add the label to the window
label.grid()

root.mainloop()
```

Change

Modify the program so that:

- The label text is set using `text` instead of `textvariable`
- The font is changed to `TkHeadingFont`
- The text color is changed to **blue**
- The label is anchored to the **top-left** of the space given to it

Challenge

Create a program that includes **two labels**:

- The first label should:
  - Display **multiple lines of text**
  - Use `wraplength` to limit the width of the text
  - Center the text using `justify`
- The second label should:
  - Use a `StringVar`
  - Start with one message
  - Change to a different message after the program starts (hint: update the `StringVar`)