Overview

An `entry` widget provides users with a single-line text field where they can type in a string value. These can be just about anything: a name, a city, a password, social security number, etc.

Important Information

Creating an Entry

Entries are created using the `ttk.Entry` class:

```
# Create a variable to store the value of the entry
username = StringVar()
# Create the entry
# parent represents whatever container you want to put it in. This can be the root, or a frame.
entry = ttk.Entry(parent, textvariable=username)
```

Note: Notice that the `textvariable` option is what stores the value of what is typed into the entry rather than the `value` option.

A `width` configuration option may be specified to provide the number of characters wide the entry should be. This allows you, for example, to display a shorter entry for a zip or postal code.

```
# Set the width of the entry in character units
entry['width'] = 25
```

Entry Contents

We've seen how checkbutton and radiobutton widgets have a value associated with them. Entries do as well. That value is usually accessed through a linked variable specified by the `textvariable` configuration option.

Unlike the various buttons, entries don't have a text or image beside them to identify them. Use a separate label widget for that.

You can also retrieve or change the value of the entry widget without going through the linked variable. The `get()` method returns the current value.

```
# Get the current text of the entry without needing to reference the textvariable
current_value = entry.get()
```

However, you cannot change the value by using the `set()` method directly on the entry. You still need to use it on the `textvariable`.

The `delete` method allows you to remove part of all of the text in the widget, specified by a character position (`0` for the first character, `1` for the second, etc.); the keyword `'end'` refers to the position just past the last character.

For example:

```
# Delete the text between 2 indices
# In this case, delete from the beginning (0) to the end ('end') of the string
entry.delete(0, 'end')
```

But you could also do it like this:

```
# Delete the text between 2 indices
# In this case, delete from the beginning (0) to the fourth position (3) of the string
entry.delete(0, 3)
```

The insert method allows you to add new text at a given position:

```
# Insert some text into the entry
# In this case, text is inserted at the beginning (0)
# The text being inserted is 'Hello World!'
entry.insert(0, 'Hello World!")
```

Watching for Changes

Entry widgets don't have a `command` option to invoke a callback whenever the entry is changed. To watch for changes, monitor the linked variable via the `trace_add()` method.

```
# A function that will run when the value of the entry changes
def run_when_changed(*args):
    print("Value Changed!")

# Create a string variable for the entry
my_variable = StringVar()
# Link the string variable to the entry
entry['textvariable'] = my_variable
# Use trace_add() method to make it so whenever text is written to the entry, the run_when_changed function is called
my_variable.trace_add("write", run_when_changed)
```

You'll be fine if you stick with simple uses of `trace_add()` like that shown above. You might want to know that this is a small part of a much more complex system for observing variables and invoking callbacks when they are read, written, or deleted.

There are other methods that can be used with details about them below.

Method	Description
`trace_add(mode, callback)`	Adds an observer callback to a variable. The `mode` parameter specifies when the callback should be triggered, and `callback` is the function to run.
`trace_remove(mode, callback_name)`	Removes a previously added callback. Basically this detaches previously added functions from entries.
`trace_info()`	Returns a list of all currently registered traces for the variable.

There are also modes other than `'write'` that can be used to check how an entry is being interacted with.

Mode	Description
`"read"`	Triggers when the variable's value is read.
`"write"`	Triggers when the variable's value is written (changed). This is the most commonly used mode.
`"unset"`	Triggers when the variable is "unset" or deleted.
`"array"`	Used for tracing elements within a Tcl array variable. This mode is more advanced an it is unlikely you will use it.

Passwords

Entries can be used for passwords, where the actual contents are displayed as a bullet or other symbol. To do this, set the `show` configuration option to the character you'd like to display.

```
# Create a variable to track the value of the entry
password = StringVar()
# Create an entry where it displays '*' instead of the actual characters being typed
password_entry = ttk.Entry(parent, textvariable=password, show="*")
```

Entry States

Like the various buttons, entries can also be put into a disabled state via the `state` method (and queried with `instate`).

```
# Disable the entry
entry.state(['disabled'])
```

Entries can also use the state flag `readonly`; if set, users cannot change the entry, though they can still select the text in it (and copy it to the clipboard).

```
# Set the entry to read only
entry.state(['readonly'])
```

Copy, Change, Challenge

Copy

```
from tkinter import *
from tkinter import ttk

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

# Create a variable to store the entry value
username = StringVar()

# Create a label for the entry
label = ttk.Label(root, text="Username:")
label.grid()

# Create the entry
entry = ttk.Entry(root, textvariable=username, width=25)
entry.grid()

root.mainloop()
```

Change

Modify the program so that:

- The label text is changed to **"Email Address:"**
- The entry width is reduced to **15 characters**
- When the program starts, the entry already contains some placeholder text (use `insert`)

Challenge

Create a small **login-style form** using entry widgets:

- Add **two labels**: one for a username and one for a password
- Add **two entries**:
    - The username entry should display plain text
    - The password entry should hide input using the `show` option
- Add a **button** that, when clicked:
    - Prints the username and password values to the terminal
    - Clears both entry fields using `delete`