As you arrive:

1. Start up your computer and plug it in.
2. ***Log into Angel*** and go to CSSE 120.
   Do the ***Attendance Widget*** –
   the PIN is on the board.
3. Go to the ***Course Schedule*** web page.
   Open the ***Slides*** for today if you wish.
4. Checkout today's project: `Session16_tkinter_ttk`

Session 16

GUIs, Event-driven programming, tkinter and ttk

Graphical User Interfaces (GUIs)

Event-driven programming

tkinter and ttk

Checkout today's project:

`Session16_tkinter_ttk`

Are you in the Pydev perspective? If not:

`Window ~ Open Perspective ~ Other` then `Pydev`

Messed up views? If so:

`Window ~ Reset Perspective`

Troubles getting today's project? If so:

No SVN repositories view (tab)? If it is not there:

`Window ~ Show View ~ Other`
then `SVN ~ SVN Repositories`

1. ***In your SVN repositories view (tab), expand your repository*** *(the top-level item) if not already expanded.*
   - If no repository, perhaps you are in the wrong Workspace. Get help.
2. ***Right-click on today's project,*** *then select* ***`Checkout`****.*
   Press ***OK*** *as needed.* The project shows up in the
   `Pydev Package Explorer`
   to the left. Expand and browse the modules under `src` as desired.

Data Collections

- Frequently several individual pieces of data are related
- We can collect them together in one object
- Examples:
  - A list or tuple contains an ordered sequence of items
  - A string contains an ordered sequence of characters
  - A custom object. Example from zellegraphics: A Line object contains two endpoints, a color, and the window in which it is drawn
  - A dictionary (defined soon) contains key-value pairs

Dictionaries – similar to Lists

- ***List*** – a collection of items. Access by ***position*** in the list.

```
animals = ['dog', 'cat', 'cow']
animals[1]
        'cat'
```

Maps from a ***number*** (index, ***position*** in the list) to a *value*

Values usually are homogeneous but are not required to be so (in Python).

- ***Dictionary*** – a collection of items. Access by ***key***.

```
pet = {'type' : 'cat',   'age' : 7,
       'name' : 'ginny', 'weight' : 12.6}
pet['name']
        'ginny'
pet['weight']
```

Maps from a ***key*** (anything immutable) to a *value*

Exercise: m0_dictionaries has an example. You might also, in main, define a ***dictionary*** that holds your name and GPA. Then a ***list of dictionaries*** with yours and two neighbors.

- No two items can have the same key.
- Note {…} instead of […]
- Often have lists of dictionaries

Graphical User Interfaces (GUIs) and Events

- A GUI receives many events and responds to them promptly.
  - What are examples of events that a GUI should handle?
  - Answer:
    - Mouse clicks
    - Mouse motion
    - Keyboard presses
    - Windows being covered or uncovered
    - Timers going off
    - And much more

Event-driven programming

Via *Polling*

Via an *Event Queue*

Associating events with functions is called ***binding***. The function to call is a ***callback*** that is called an ***event handler***.

Here events are repeatedly added to the ***Event Queue***.

Associate each event you are interested in with a function to call when that event occurs.

Repeatedly:

Poll devices to find what events have occurred.

Call the event handlers for those events.

Associate each event you are interested in with a function to call when that event occurs.

Repeatedly:

Take an item off the Event Queue.

Call the event handler for that event.

The thing that does this is called the ***event dispatcher***.

Event-driven programming – an example

Run this example in ***m0_simple_event_loop***.
Note the effect of the way-too-long *sleep*.

- Here is a simple example using zellegraphics.

```
while True:
        # Poll the devices to learn the events
        mouse = window.checkMouse()
        key = window.checkKey()

        # Respond to the events
        if mouse != None:
            mouse_handler(mouse, window)

        if key != None:
            key_handler(key, window)

        # Sleep to allow time for other threads to run
        time.sleep(1.0)    # Purposely set way too big here
```

tkinter and ttk

- We could use Zellegraphics and make our own buttons (Rectangles, see if mouse click is inside one) and so forth.
  - Why is this a bad idea?
- Instead, we will use a package the provides tools for making a pretty GUI easily. We'll use ***tkinter*** and ***ttk***.
- ***tkinter*** is the oldest GUI package that comes with the standard distribution of Pyton. It is based on a scripting language called ***Tcl***.
- ***ttk*** is an extension of *tkinter* that brings *tkinter's* capabilities and the look and feel of its widgets up to date.
- There are lots of other GUI packages for Python, but *ttk* is plenty powerful, pretty easy to use, and comes with Python.

Your first ttk program

Study and run this example in ***m1_ttk***. Questions?

Buttons, Checkboxes, Menus, etc. are called ***Widgets***.

- This program makes a button that says *Connect*. Pressing the button gives visual feedback (try it) but does nothing interesting.

```
from tkinter import *
from tkinter import ttk

main():
    root = Tk()

    button = ttk.Button(root, text='Connect')
    button.grid()

    root.mainloop()
```

ttk overwrites parts of *tkinter* and adds other stuff. So the order here is important. I'll just say "*ttk*" from now on, but realize that it really is a combination of the two.

Constructs a Tk object, called the ***root*** window.

Constructs a ttk ***Button***, attached to the root.

Draws the Button, using the ***grid*** layout manager.

Enters the ***mainloop***, which does the *event-handling* for you. **ttk is in control now. All you can do is respond to events.** The program stays in that *mainloop* until the root window is closed.

The rest of the examples

- The next examples in today's project show:
  - **m2_ttk_frames_and_grid**
    - Grouping the widgets into a Frame
    - Simple use of the grid layout manager
  - **m3_ttk_events_part1**
    - Writing event-handlers
    - Binding events to their event-handlers
    - From an Event, getting its widget.
    - Reconfiguring a widget.
  - **m4_ttk_events_part2**
    - How event handlers can share data, by having their widgets share data
    - How widgets can share data by using a shared dictionary
  - **m5_ttk_adding_your_event_loop**
    - How your robot can get into the action

Rest of Session

- ***Work on your project. Also recall that homework is due yesterday and today.***
  - Ask questions as needed!
- **Sources of help after class:**
  - **Assistants in the CSSE lab**
    - And other times as well (see link on the course home page)
  - **Email**
    - You get faster response from the above than from just your instructor

```
CSSE lab: Moench F-217
7 to 9 p.m.
Sundays thru Thursdays
```

```
csse120-staff@rose-hulman.edu
```