ECS 10

3/11

Plans

- Final is Wds Mar 20, 1-3pm, in this room.
- Will be very like midterm 2, but with more review of midterm 1 material. Two small programs.
- Makeup program due tomorrow 10pm
  - Lab hours today 11-1
  - Lab hours tomorrow 12-2
- Prog 6 due Sunday Mar 17
- If your previous HW grades were good do not hand in both. We will replace your lowest score with the makeup, even if it lowers your grade.

Recap MT2 program

- Input:

```
Afghanistan: Islamic Republic of Afghanistan, Jamhuri-ye Islami-ye Afghanistan
Akrotiri:
Albania: Republic of Albania, Republika e Shqiperise, Shqiperia
Algeria: People's Democratic Republic of Algeria, Al Jaza'ir
American Samoa: Territory of American Samoa, AS
```

- Output:

```
Country name: AS
The standard name of AS is American Samoa
Country name: American Samoa
The standard name of American Samoa is American Samoa
```

Look at the output

- If user inputs alternative names, and it reports standard names, then the keys are alternative names and the values are standard names.
- You could use a tuple (sadly not a list…) of alternative names as keys. What is wrong with this idea?
- You could use standard names as keys and a list of alternative names as values and then read the whole dictionary to find every answer. What is wrong with this idea?

What if…

- The output is a file containing cities and their populations, largest to smallest, input is a file with cities by country, including populations.

```
Tokyo–Yokohama 37,126,000
Jakarta 26,063,000
Seoul–Incheon 22,547,000
Delhi 22,242,000
Shanghai 20,860,000
Manila 20,767,000
Karachi 20,711,000
New York 20,464,000
```

Data structure: list of tuples

```
[ (22242000, Delhi), (66000, Davis)…)]
```

- Each tuple is (integer, string)
- Put population first so you can sort by population.
- Write population second to file.

Last time: Using Objects

- All data in Python is some kind of object.
- Built-in data types are integer, string, dictionary…
- Classes are “new” data types defined by programs.
- Lots of modules are organized entirely as collections of classes.
- Factory functions create instances of the new classes, that is, objects belonging to the new classes.
- Methods for the new classes are defined in the module, do most of the work.

More tkinter: Label class

- Put text into your window.

```
label = Label(root, text="Yowza!")
```

- Objects store data as well as methods.
- tkinter lets the user have access to some (but not all) of the data stored in an object.
- These are the attributes.

```
print(label.cget("text"))
```

Keyword parameters

- A function or method can have a lot of parameters.
- A label can have text, but also lots of other parameters (colors, behavior when window is active, images, border styles…)
- Rather than have to specify all parameters every time you make a label, specify only the ones you want and use default values for the rest.
- Specify parameters by variable name rather than position – these are called keyword arguments.
- Tkinter uses keyword arguments for most attributes.

Example: changing the font

- Need to include the font part of tkinter

```
from tkinter.font include Font
```

- Then define the font you want

```
bigFont = Font(family='Times', size=20)
label = Label(root, text="Yowza!", font=bigFont)
```

Images

- Can also put pictures into labels.

```
flower = PhotoImage(file="Rhododendrum.gif")
pic = Label(root, image=flower)
```

- Only guaranteed to handle certain image formats (gif, ppm, pnm), sadly.

Making a button

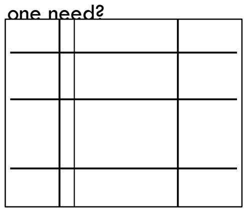
- Buttons are like labels, but they ought to do something.

```
like = Button(root, text="Like!", font=big, \
        command=whatButtonDoes)
```

- The command should be the name of a function in your program.
- That function gets run when the button is pushed.

Grid layout

- Basic idea: arrange things in rows and columns.
- This is the visual design; start with a plan. How big are your components, how many rows or columns does each one need?



Frame

- Gives you more options than root window

```
frame = Frame(root)
frame.grid()
```

- Frame goes into root window
- Put the buttons, pictures, labels inside the frame

Spanning columns

- Add a background picture spanning the whole frame.
- Add it first so buttons end up on top.

```
pic.grid(column=0, row=0, columnspan=3, rowspan=3)
```

Place widgets

```
label.grid(column=0,row=0,columnspan=3)
…
like.grid(column=0,row=4)
```

- Places the label in middle of whole top row.
- Places like button on lower left