ECS 10

3/8

More on modules

- You know how to write programs sophisticated enough to do real work.
- Lots of real work requires you to use modules.
- We know nodules contain new functions.

```
from inputCheck import canBeInt
```

- Import new functions at top of program
- Use them like any other function or method
- How would you import function drawPlot ?

```
from tkinter import *
```

- Imports all of the stuff in the module

Importing a module

- If the module has a main() function, it gets run automatically when the module is imported.
- Sometimes this is a good idea.
- In this case, the main() function just demonstrates the function on some bogus data; we don't want to do that every time the module is imported.
- These lines detect that it is being called when it is imported, and returns.

```
if __name__ != "__main__":
        return
```

Classes

- Modules also usually contain new data types!
- These are called classes.
- It's important to know a little bit about classes to use modules.

Example: the tkinter module

- Used to define Graphical User Interfaces (one of many).
- GUI modules unfortunately always seem complicated.
- Let's make a window.

Classes

- A **class** is like a new data type.
- Classes are most often added by modules. The tkinter module adds the Tk class (among others)
- An **object** in Python is any data item.

```
root  = tk.Tk()
```

- Creates an object which is an instance of class Tk, and puts it into variable root.
- Kind of like an "object of type Tk" – but with different jargon.

Data as objects

```
name = 'Nancy Drew'
root = tk.Tk()
```

- *name* is a variable, containing a string.
- *root* is a variable, containing a Tk, whatever that is.
- Strings and Tks are two kinds of objects; the class string is built-in to Python, but the class Tk was added by a module.

Instance

- There can be lots of objects of type string in a program, and similarly lots of objects of type Tk.
- Each object is an **instance** of its class.

Factory Functions

- Create instances of the class.
- Often these are the only functions (as opposed to methods) in the module.

```
root = tk.Tk()
```

Methods

- Any function that works only with an object of a particular class is given as a method.
- This is the difference between a function and a method – methods are functions that belong to specific kinds of objects.

```
root.title("cute little window")
root.geometry("200x100")
```

The tkinter main loop

```
root.mainloop()
```

- Puts up the window
- Stays in this function until something happens to the window.
- So far, this is just killing the window. But soon we will have buttons!

The canvas object

- Put a "sketchpad" in our window, so we can draw some graphics.

```
c = Canvas(root, width=600,height=400,bg ='white')
```

- A canvas object – one of the possible GUI features.
- Attached to the root window.

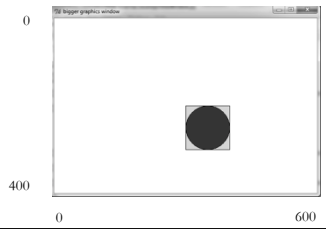
```
canvas.grid(column=0, row=0)
```

- Where exactly it goes in root window (fills it up).

Some graphics

- Draw things like lines, rectangles, ovals.

```
canvas.create_rectangle(300,200,400,300)
canvas.create_oval(300,200,400,300,fill="darkRed")
```



Lines

```
canvas.create_line(100,y,105,y, width=2)
```

- From (x,y) point to another (x,y) point

```
dataList = [ (20,40), (30, 50), (50, 100) ]
canvas.create_line(dataList, fill='black')
```

- Connects all of the point tuples in the input list

The drawPlot function

- Uses the canvas
- Draws the axes as lines

Object Oriented Programming

- A philosophy about how to organize programs.
- Keep data organized into objects, along with methods to use or modify that data.
- Many languages, including Python, have a lot of syntactic sugar to make object oriented programming easier.
- Lots of modules are organized as collections of classes.

The future

- If you go on to use Python, you'll use a lot of modules that have classes in them.