Special Topics in Computer Science- CSC 4992

User Interfaces

Introduction to GUI programming

Terminal-Based User Interface (TUI)

Supports input via the keyboard and output via the monitor

In Python, the I/O functions are **`input`** and **`print`**

```
import math

radius = float(input('Enter the radius: '))
area = math.pi * radius ** 2
print('The area is', area)
```

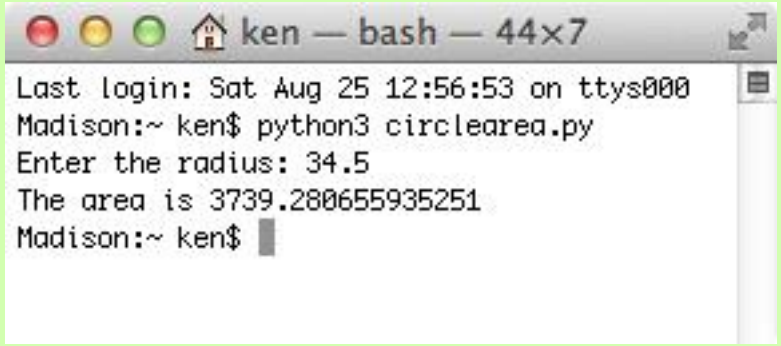
Terminal-Based User Interface (TUI)

Supports input via the keyboard and output via the monitor

In Python, the I/O functions are **`input`** and **`print`**

```
import math

radius = float(input('Enter the radius: '))
area = math.pi * radius ** 2
print('The area is', area)
```



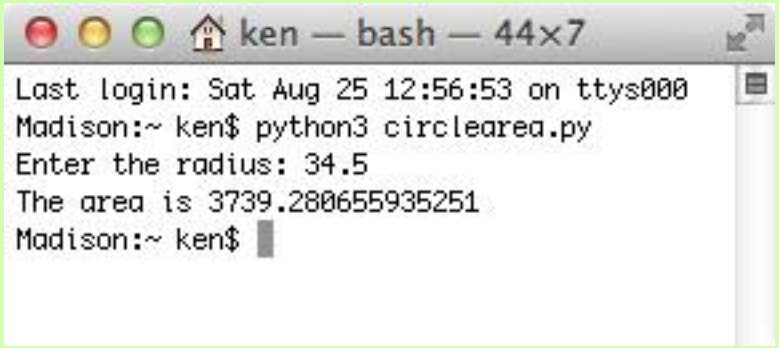
Problems with a TUI

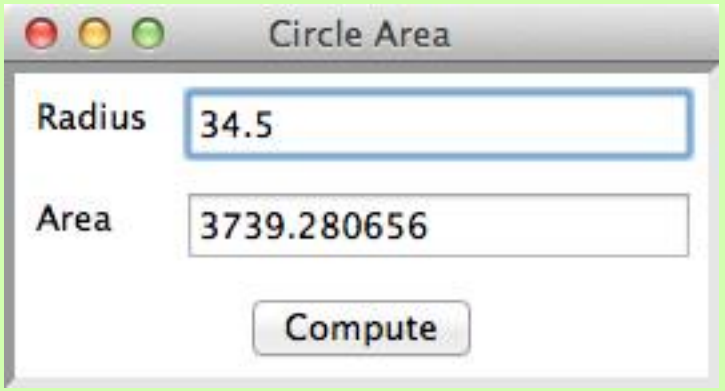
- Must enter inputs in a certain order
- Cannot back up to correct input mistakes or change one's mind
- Must re-enter all inputs to change just one
- I/O restricted to text

Graphical User Interface (GUI)

- Supports input via the keyboard
- Can output text and also graphical shapes representing desktop elements, such as windows, command buttons, data fields, and drop-down menus (also called “widgets”)
- Supports direct manipulation of desktop elements via the mouse or touchscreen

TUI vs GUI





Non-programmers (the 99%) do not use a TUI, they use a GUI

Only programmers (the 1%) use a TUI (and also a GUI)

Most beginning programmers program to a TUI, not a GUI

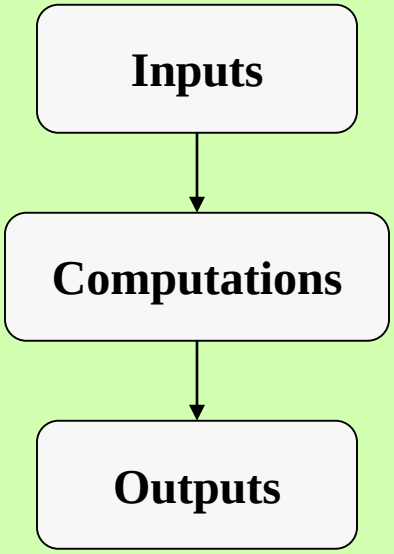
Programming a GUI

- Most modern programming languages (like Python and Java) include packages or modules for programming a GUI
- In Python, this module is called **`tkinter`**
- The model of computation for a GUI program is more complex than that of a TUI program

Models of Computation

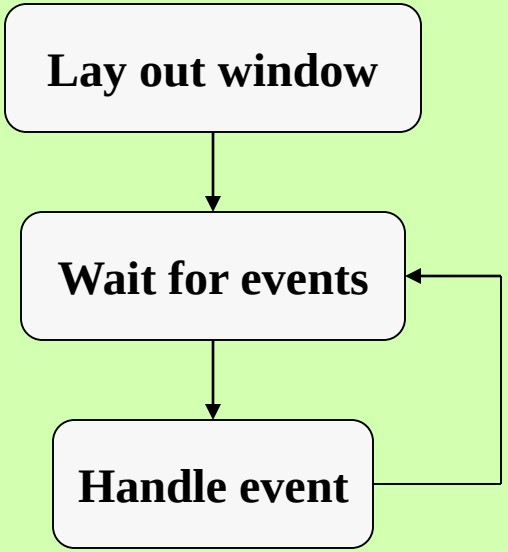
TUI

1. Obtain user inputs
2. Perform computations
3. Print results



GUI

1. Layout and pop up the window
2. Wait for user events
3. Handle a user event
4. Goto step 2



GUI Resources in Python

`tkinter`

http://docs.python.org/py3k/library/tkinter.html#module-tkinter

`breezypythongui`

http://home.wlu.edu/~lambertk/breezypythongui/index.html

What Is **breezypythongui**?

- A module of classes and functions that makes GUI programming with **tkinter** easy for beginners
- The module is free and open-source
- A tutorial and sample programs are available at the **breezypythongui** Web site

A First GUI Program: Hello World

```
from breezypythongui import EasyFrame
```

Import the abstract window class

A First GUI Program: Hello World

```
from breezypythongui import EasyFrame

class HelloWorld(EasyFrame):
    """Displays a greeting in a window."""
```

Define a subclass to specialize it for this application

A First GUI Program: Hello World

```
from breezypythongui import EasyFrame

class HelloWorld(EasyFrame):
    """Displays a greeting in a window."""

    def __init__(self):
        """Sets up the window and the label."""
        EasyFrame.__init__(self)
        self.addLabel(text = "Hello world!",
                      row = 0, column = 0)
```

Lay out the window and its widgets

A First GUI Program: Hello World

```
from breezypythongui import EasyFrame

class HelloWorld(EasyFrame):
    """Displays a greeting in a window."""

    def __init__(self):
        """Sets up the window and the label."""
        EasyFrame.__init__(self)
        self.addLabel(text = "Hello world!",
                      row = 0, column = 0)

# Instantiates and pops up the window.
if __name__ == "__main__":
    HelloWorld().mainloop()
```

Create and launch the application window

The Structure of Any GUI Program

```
from breezypythongui import EasyFrame

class <class name>(EasyFrame):

    def __init__(self):
        EasyFrame.__init__(self <optional args>)
        <code to set up widgets>

    <code to define event-handling methods>

# Instantiates and pops up the window.
if __name__ == "__main__":
    <class name>().mainloop()
```

Lay Out Widgets

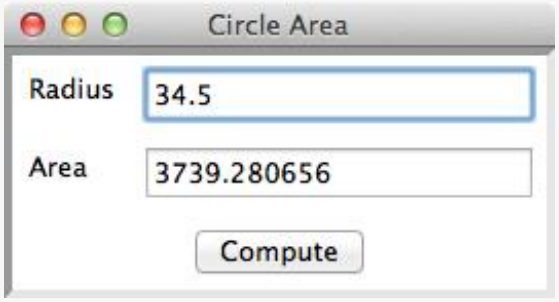
```
class CircleWithGUI(EasyFrame):
    """Computes and displays the area of a circle."""

    def __init__(self):
        """Sets up the window and widgets."""
        EasyFrame.__init__(self, title = "Circle Area")

        # Label and field for the input
        self.addLabel(text = "Radius",
                      row = 0, column = 0)
        self.radiusField = self.addFloatField(value = 0.0,
                                              row = 0,
                                              column = 1)

        # Label and field for the output
        self.addLabel(text = "Area",
                      row = 1, column = 0)
        self.areaField = self.addFloatField(value = 0.0,
                                            row = 1,
                                            column = 1)

        # The command button
        self.addButton(text = "Compute", row = 2, column = 0,
                       columnspan = 2, command = self.computeArea)
```



Define the Event Handler

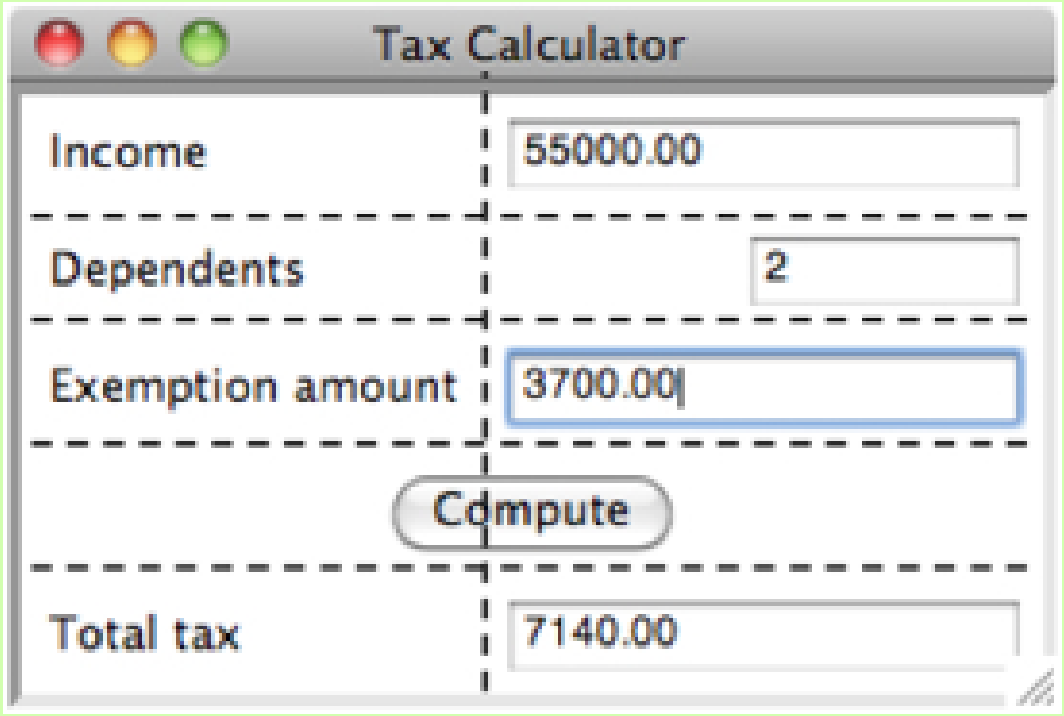
```
class CircleWithGUI(EasyFrame):
    """Computes and displays the area of a circle."""

    . . .


    # The event handling method for the button
    def computeArea(self):
        """Inputs the radius, computes the area,
        and outputs the result."""
        radius = self.radiusField.getNumber()
        area = math.pi * radius ** 2
        self.areaField.setNumber(area)

#Instantiate and pop up the window.
if __name__ == "__main__":
    CircleWithGUI().mainloop()
```

Grid Layout


Tax Calculator
Income
55000.00
Dependents
2
Exemption amount
3700.00
Compute
Total tax
7140.00

Grid Layout

```
# Label and field for the income
self.addLabel(text = "Income",
              row = 0, column = 0)
self.incomeField = self.addFloatField(value = 0.0,
                                       row = 0,
                                       column = 1)


# The command button
self.addButton(text = "Compute", row = 3, column = 0,
               columnspan = 2, command = self.computeTax)
```