

# Drawing and Interaction with graphics.py

---

19 Oct 2010

CMPT140

Dr. Sean Ho

Trinity Western University

# Quiz 3 (5min, 10pts)

- What is **output** by this Python code? [2]

```
def add_ya(x):  
    x += "YA!"  
  
y = "hi"  
add_ya(y)  
print(y)
```

- In OO, a **class** is a user-defined type containing what **two** kinds of things? [3]
- In OO, what does the **constructor** do? [3]
- In Python, the constructor is always **named** \_\_\_\_\_

# Quiz 3: answers #1

- What is **output** by this Python code?

[2]

```
def add_ya(x):  
    x += "YA!"
```

```
y = "hi"
```

```
add_ya(y)
```

```
print(y)
```

- "hi"
- The **actual parameter** y is **not** changed, since its type is **str** (immutable), so it is passed by **value**

# Quiz 3: answers #2-4

- In OO, a **class** is a user-defined type containing what **two** kinds of things? **[3]**
  - **Attributes** (vars) and **methods** (functions)
  - *(2/3 pts for right idea but wrong terms)*
- In OO, what does the **constructor** do? **[3]**
  - Creates a new **object** and
  - **Initializes** its attributes
- In Python, the constructor is always **named**: **[2]**
  - **`__init__`** (*2 underscores*)

# Computing & Society Paper

- Computing scientist as **Godly Christian Leader**:
  - Not just **knowledge** about tools, but
  - **Wisdom** of how to use tools
  - ◆ To **serve** others and
  - ◆ To give glory to **God**
- Write a short **essay** on a topic of your choosing about **computers** and **society**:
  - ◆ ~ **5 pages** typed double-spaced 12pt 1in margins
  - ◆ Submit half-page **topic** by **Tues 2 Nov**
  - ◆ Paper **due** last day of our class (**Tues 7 Dec**)
    - Electronic submission (email, myCourses)

# Sample paper topics

- **Censorship** and free speech
  - Pornography, gambling, hate groups, etc.
- **Blogs**: effect on politics, social interaction, etc.
- **Artificial intelligence**: the nature of sentience
- **Violence** in video games (Columbine etc.)
- **Privacy**: online banking, ID theft, etc.
- **File sharing**: BitTorrent, etc.
- **Online dating** (e.g. eHarmony): pros/cons
- **Equity of access** / rural digital divide

# Essay / Position Paper

- Your essay should be a **position paper**:
  - Topic should have at least two **sides** (e.g. pro/con)
  - You should state (in the introductory paragraph) what your **position** is (**thesis**)
  - You should have at least 2-3 points, each, both **for** and **against** your position
- ◆ It is not necessary to **rebut** every point that contradicts your position:
- ◆ Be honest about **faults**/limitations of your thesis
  - Summary **intro/conclusion** paragraphs
  - Proper **English** (spelling, grammar) is important!

# Interaction in graphics.py

- Import library: `from graphics import *`
- Setup window: `win = GraphWin("win", 400,300)`
- Remember that **Python** code drives the graphics
  - Objects are **drawn** only when told to by code
- Get a **click** from the user:
  - `pt = win.getMouse()`
  - **Control** passes to the graphical window
  - Python **blocks** (waits) until user clicks
  - Returns a **Point** with **(x,y)** coordinates

# Private attributes

- How to **get** the (x,y) coordinates from a **Point**?
- We can directly access the **attributes**:  
`xcoord = pt.x`
- This also means we can **change** their values:  
`pt.x = 45`
- Attributes may be made **private** so that outside code cannot directly access/modify them:
  - In C++/Java, **declare** attributes **private**
  - In Python, give it a **name** starting with `__`:  
`self.__x = 0`

# Accessors/mutators (set/get)

- Access is provided to private attributes via **accessor** (get) and **mutator** (set) methods:
  - `xcoord = pt.getX()`
  - **Point** only has a **get** method, no set
- This allows **other** code which uses the class to read/write the attribute, but **only** through these **controlled** channels
  - E.g., **error-checking** to make sure the value is being set to a valid value
  - e.g., setting a **negative** x-coordinate, or even changing the type to **str**!

# Making clickable buttons

- Here's how to make **buttons** the user can click:
- Draw **Rectangles** for the **outlines** of the buttons
- Draw **Text labels** inside the rectangles
- Call **getMouse()** and get a **click** from the user:
  - For each **button**, check if the **(x,y)** coords of the user click lie **within** the button
- Consider storing button geometry in **lists**
- Real GUI **toolkits** have button classes



# Text entry box: Entry

- An **Entry** is a text box whose value can be **changed** by the user and **read** by the program:

```
amtIn = Entry(Point(200, 100), 5)
```

- Specify **centre** and **width** (in characters)

- **Set/get** methods for the text in the box:

```
amt = float( amtIn.getText() )
```

```
amtIn.setText("43.50")
```

- This works for regular **Text** boxes, too
- Your **program** doesn't know when the value **changes**, so use a **.getMouse()** and have the user **click** when done editing the **Entry** box

# Changing coordinate systems

- The default **coordinate system** puts **(0,0)** in the **top-left** corner and counts in units of **pixels**
- You can **change** the coordinate system of a window by specifying the **coordinates** of its four edges: (**left, bottom, right, top**):
  - # **bottom-left is (0,0), top-right is (6,4)**
  - win.setCoords(0., 0., 6., 4.)**
  - # **plot a point in the centre of the window**
  - Point(3, 2).draw(win)**
- Each **window** has its own coordinate system