

# Let the PyGames Begin!

Professor Bob Brown

College of Computing and Software Engineering  
Kennesaw State University

[Bob.Brown@Kennesaw.edu](mailto:Bob.Brown@Kennesaw.edu)

# Please Help Professor Brown

- This class is new to Professor Brown.
- He needs to know how he's doing.
- Please answer a 10-question survey.
- This is *not for a grade*.
- Put your computer number, not your name on it.
- Put your grade on it.
- If you're not sure, pick the answer that seems best.

# The Plan !

In the next few weeks we will

- Work together on Python projects at first.
- Work in teams on independent projects.
- Learn as we go along!

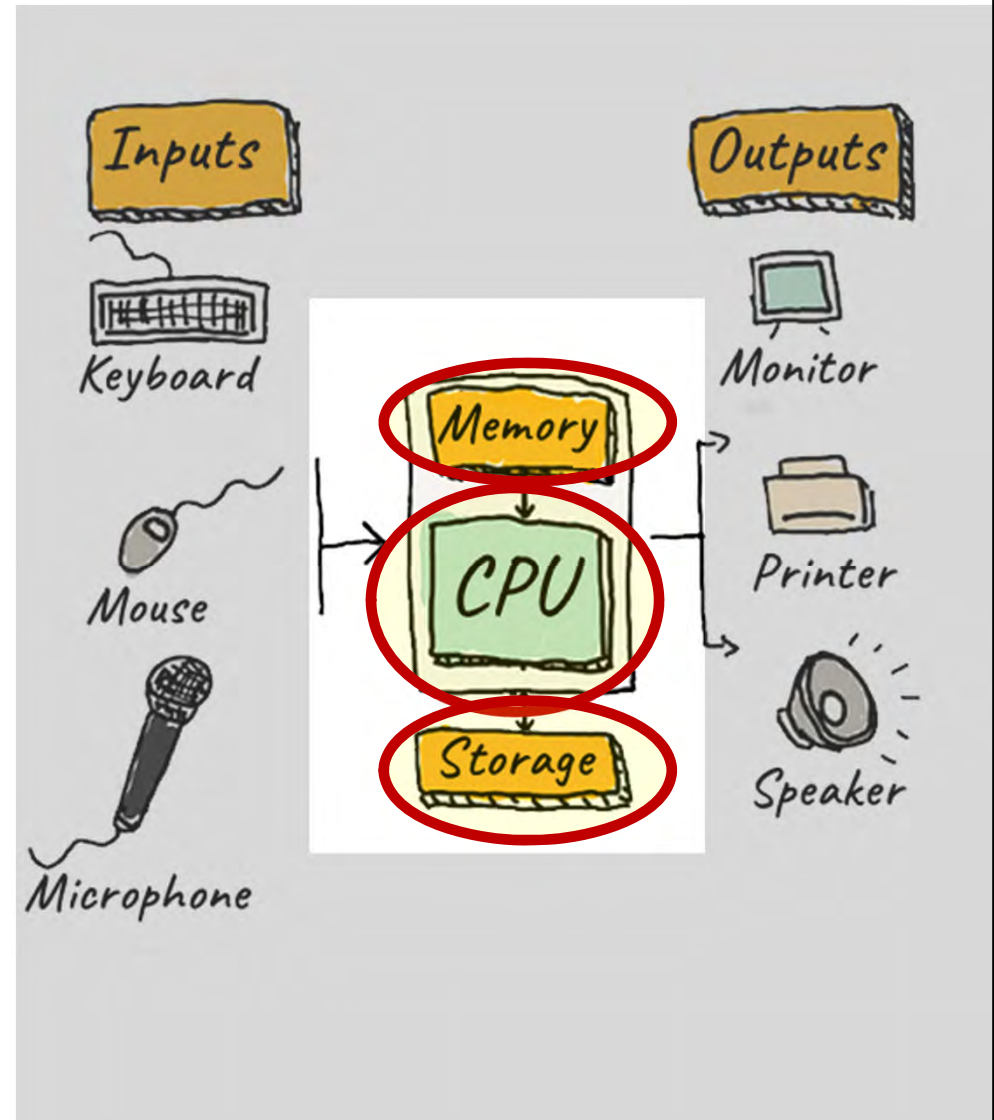


# What is a Computer?

- A computer is a machine.
- It takes simple actions in response to commands.
- The commands can be stored, to be used over and over.
- The stored commands are called a *program*.
- Programs are written in computer *code*.

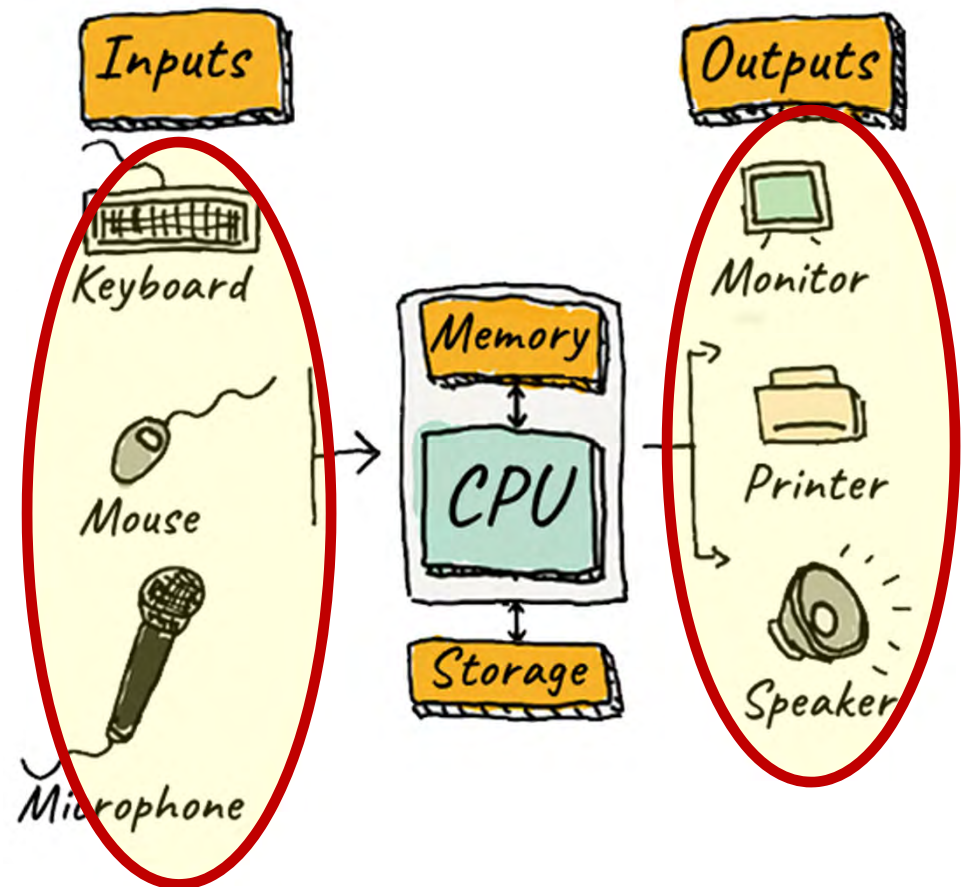
# The CPU and Memory

- The ***CPU*** executes commands, one at a time.
- The ***memory*** holds the program and its data. Called ***RAM***.
- ***Storage*** holds programs and data. ***Hard disk, SSD, or flash*** memory.



# Input and Output

- ***Inputs*** are processed by the CPU as commanded by the program.
- ***Outputs*** are produced by the CPU as commanded by the program.
- There are different kinds of ***devices*** for input and output.



# Another Way to Think

- Computers receive *inputs*.
- Inputs are *processed* by a *program*.
- Which produces the *outputs*.



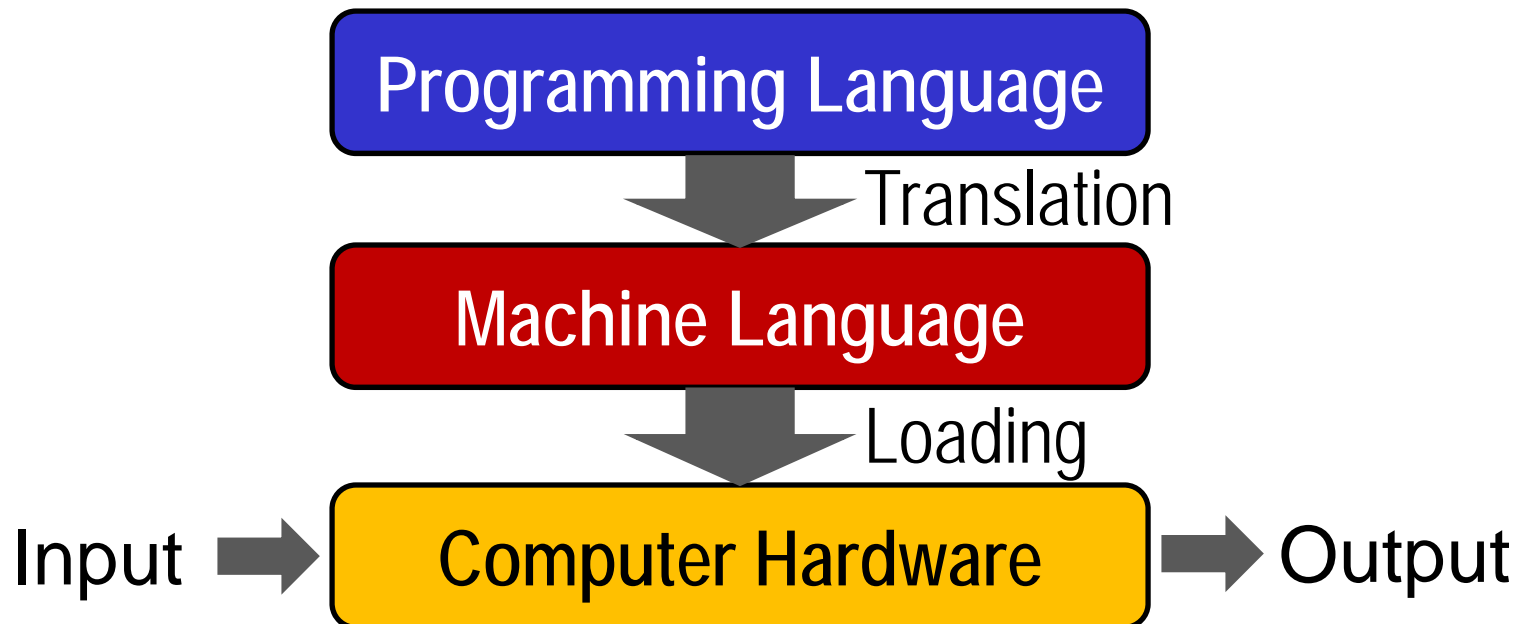
# Computers, Humans, and Languages

- Humans speak English (Or Spanish, or French, or Japanese...)
- Computers understand only strings of ones and zeros, called *machine language*.
- Humans write for computers in highly-structured languages called *programming languages*.



# Translation to Machine Language

Programming languages must be *translated* to ones and zeros for the computer.




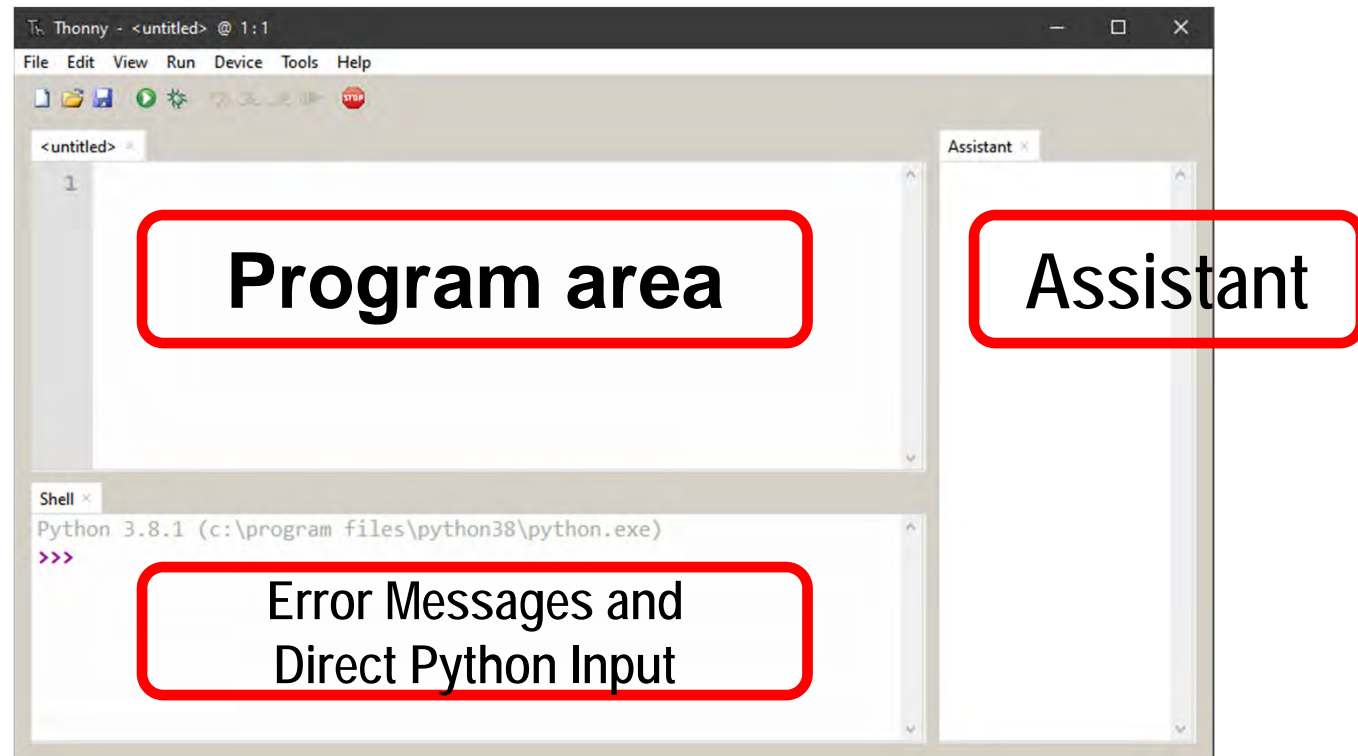
# The Python Language

- Capitalization matters
- Indentation matters
- It's not named for a snake

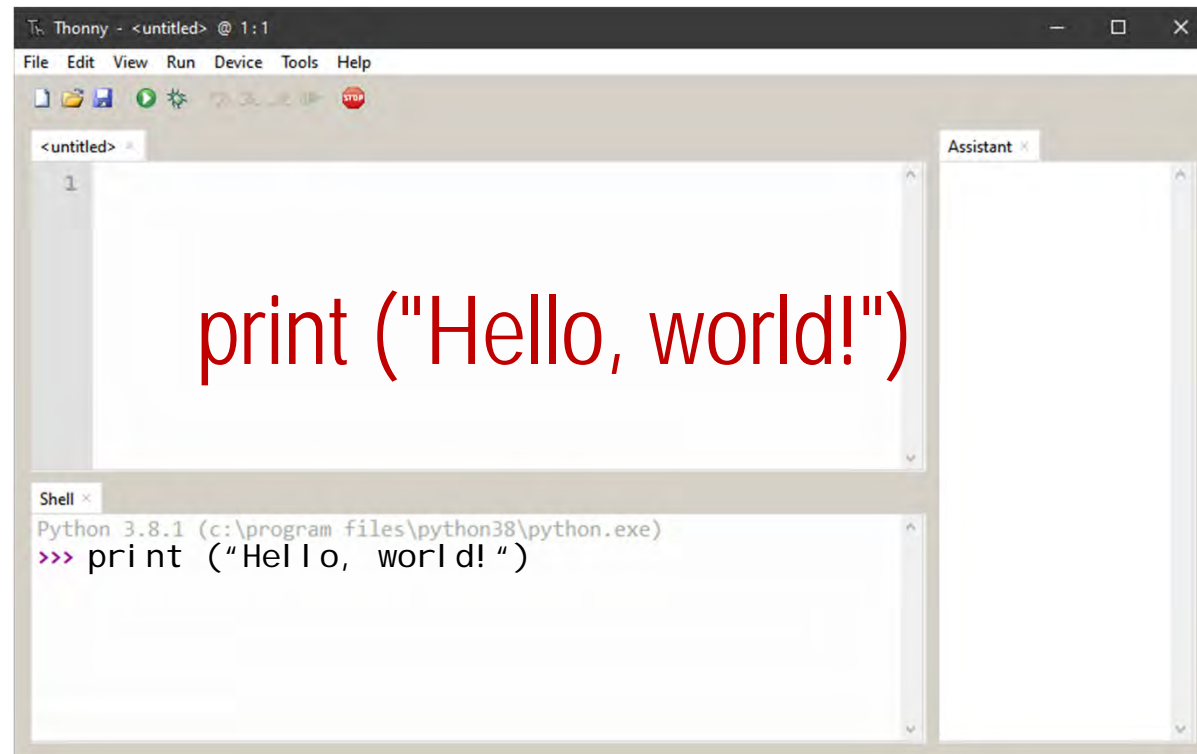


# The “Thonny” Python Environment

To bring up the app, click  and type “thonny”



# Something to Try

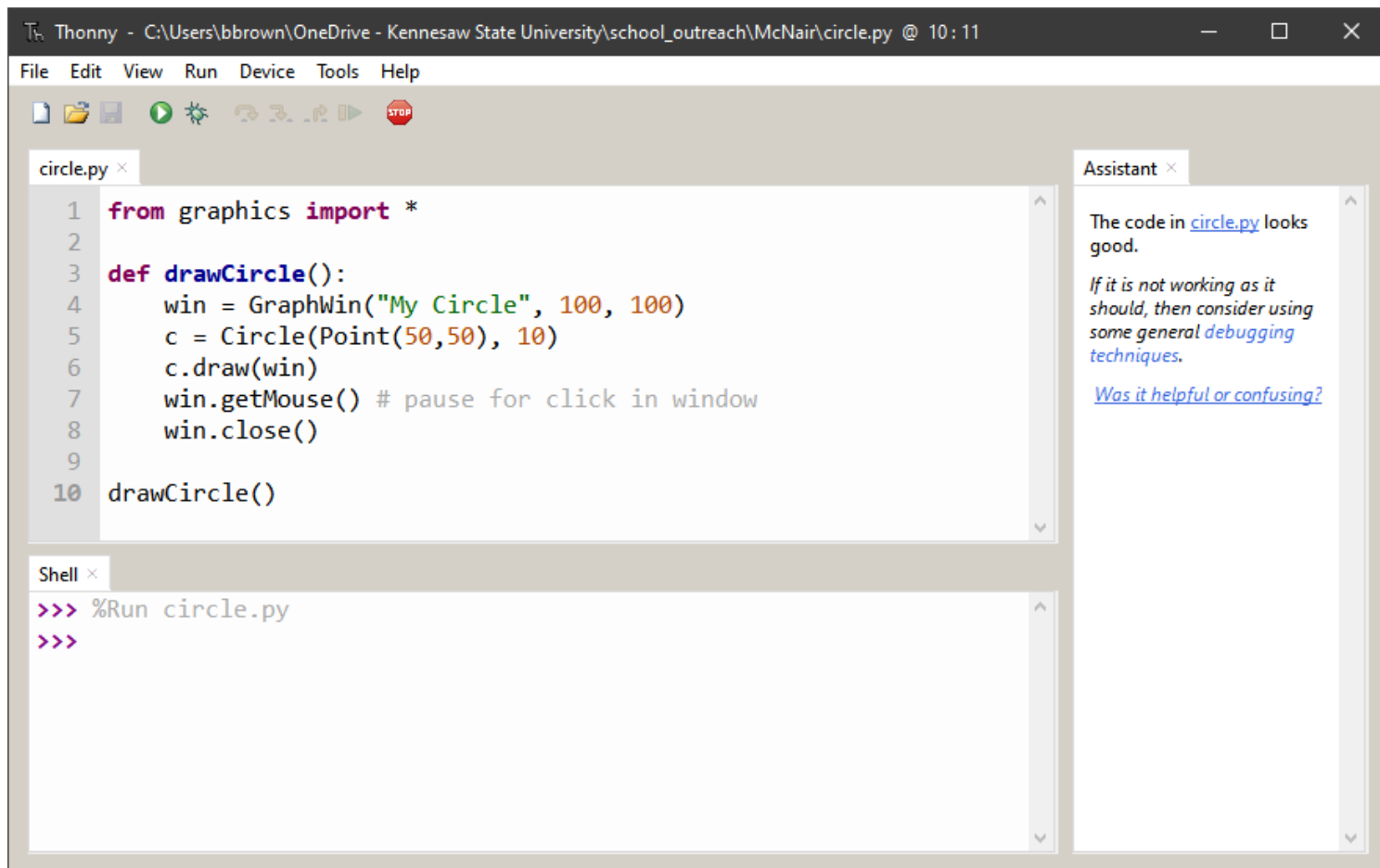


# A Program of Your Own

## Algorithm:

- Use “import” to get the graphics library.
- Create a function that does this:
  - Define a 100 x 100 window
  - Define a circle within the window
  - Draw the circle
  - Wait for a mouse click
  - Close the graphics window
- Run the function

# Here it is in Thonny



The screenshot shows the Thonny Python IDE interface. The title bar indicates the file path: C:\Users\bbrown\OneDrive - Kennesaw State University\school\_outreach\McNair\circle.py @ 10:11. The menu bar includes File, Edit, View, Run, Device, Tools, and Help. The toolbar contains icons for file operations and execution. The main editor window displays the code in circle.py:

```
1 from graphics import *
2
3 def drawCircle():
4     win = GraphWin("My Circle", 100, 100)
5     c = Circle(Point(50,50), 10)
6     c.draw(win)
7     win.getMouse() # pause for click in window
8     win.close()
9
10 drawCircle()
```

Below the editor is the Shell window, which shows the command to run the script:

```
>>> %Run circle.py
>>>
```

On the right side, the Assistant panel provides feedback:

The code in [circle.py](#) looks good.

If it is not working as it should, then consider using some general [debugging techniques](#).

[Was it helpful or confusing?](#)



# Big Enough to Read, and Run!

```
from graphics import *  
  
def drawCircle():  
    win = GraphWin("My Circle", 100, 100)  
    c = Circle(Point(50,50), 10)  
    c.draw(win)  
    win.getMouse() # pause for click  
    win.close()
```

5 lines above indented; handout is in error.

```
drawCircle()
```

# Let's Look, Line by Line

```
from graphics import *
```

```
def drawCircle():  
    win = GraphWin("My Circle", 100, 100)  
    c = Circle(Point(50,50), 10)  
    c.draw(win)  
    win.getMouse() # pause for click  
    win.close()
```

```
drawCircle()
```



# Let's Look, Line by Line

```
from graphics import *
```

```
def drawCircle():
```

```
    win = GraphWin("My Circle", 100, 100)
```

```
    c = Circle(Point(50,50), 10)
```

```
    c.draw(win)
```

```
    win.getMouse() # pause for click
```

```
    win.close()
```

```
drawCircle()
```

# Let's Look, Line by Line

```
from graphics import *
```

```
def drawCircle():  
    win = GraphWin("My Circle", 100, 100)  
    c = Circle(Point(50,50), 10)  
    c.draw(win)  
    win.getMouse() # pause for click  
    win.close()
```

```
drawCircle()
```

# Let's Look, Line by Line

```
from graphics import *  
  
def drawCircle():  
    win = GraphWin("My Circle", 100, 100)  
    c = Circle(Point(50,50), 10)  
    c.draw(win)  
    win.getMouse() # pause for click  
    win.close()  
  
drawCircle()
```

# Let's Look, Line by Line

```
from graphics import *  
  
def drawCircle():  
    win = GraphWin("My Circle", 100, 100)  
    c = Circle(Point(50,50), 10)  
    c.draw(win)  
    win.getMouse() # pause for click  
    win.close()  
  
drawCircle()
```

# Let's Look, Line by Line

```
from graphics import *  
  
def drawCircle():  
    win = GraphWin("My Circle", 100, 100)  
    c = Circle(Point(50,50), 10)  
    c.draw(win)  
    win.getMouse() # pause for click  
    win.close()  
  
drawCircle()
```

# Let's Look, Line by Line

```
from graphics import *  
  
def drawCircle():  
    win = GraphWin("My Circle", 100, 100)  
    c = Circle(Point(50,50), 10)  
    c.draw(win)  
    win.getMouse() # pause for click  
    win.close()
```

```
drawCircle()
```

# Let's Look, Line by Line

```
from graphics import *  
  
def drawCircle():  
    win = GraphWin("My Circle", 100, 100)  
    c = Circle(Point(50,50), 10)  
    c.draw(win)  
    win.getMouse() # pause for click  
    win.close()
```

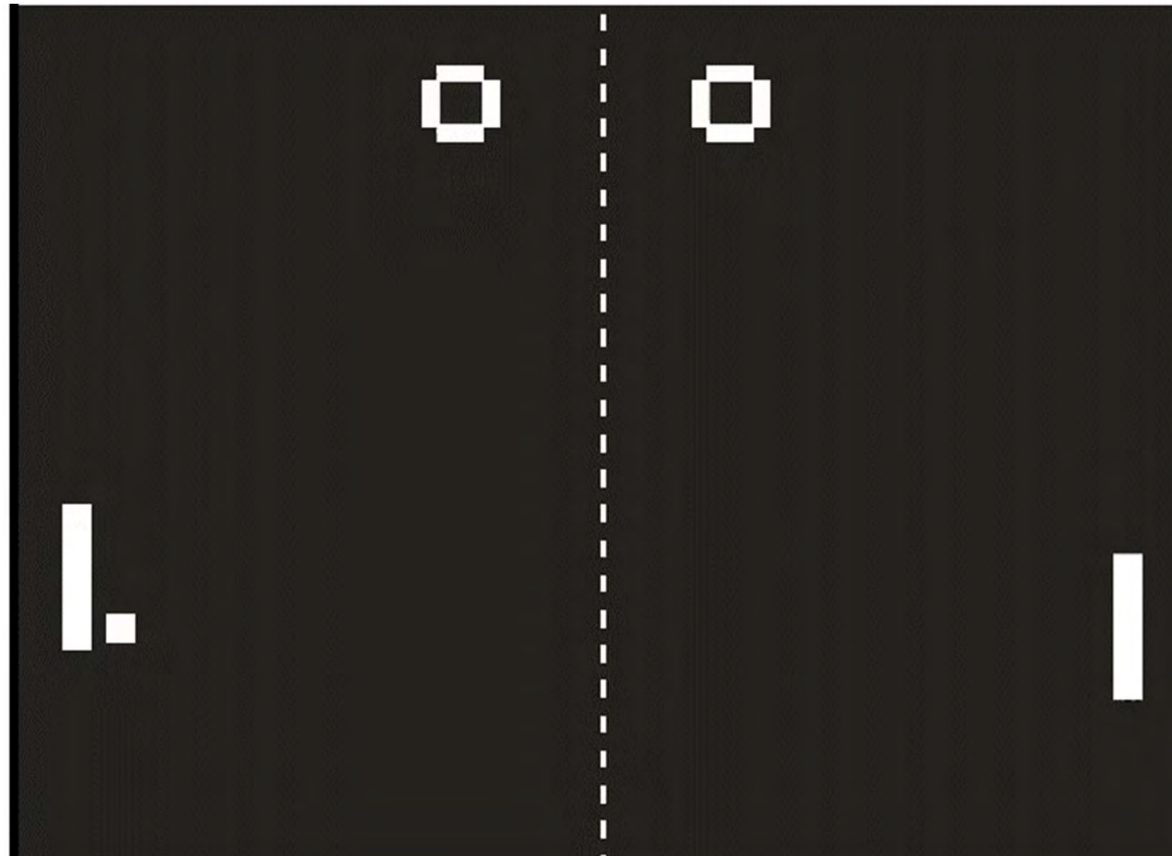
**drawCircle()**

# Programming Challenges

- Can you change the size of the window?
- Can you change the size of the circle?
- Can you change where the circle is drawn?



# An Early Computer Game



# Getting Started with Pong

```
# Import the pygame library and initialise  
the game engine  
import pygame  
pygame.init()  
# Define some colors  
BLACK = (0,0,0)  
WHITE = (255,255,255)  
# Open a new window  
size = (700, 500)  
screen = pygame.display.set_mode(size)  
pygame.display.set_caption("Pong")
```

# Getting Started with Pong

```
# The loop will carry on until the user exit  
# the game (e.g. clicks the close button).  
carryOn = True
```

```
# The clock will be used to control how fast  
# the screen updates  
clock = pygame.time.Clock()
```

# Getting Started with Pong

```
# ----- Main Program Loop -----  
while carryOn:  
    # --- Main event loop  
    for event in pygame.event.get():  
        # User did something  
        if event.type == pygame.QUIT:  
            # If user clicked close  
            carryOn = False  
            # Flag that we are done  
            # so we exit this loop  
  
    # --- Game logic will go here
```

# Getting Started with Pong

```
# --- Drawing code will go here
# First, clear the screen to black.
screen.fill(BLACK)
#Draw the net
pygame.draw.line(screen,
    WHITE, [349, 0], [349, 500], 5)
# --- Update the screen
pygame.display.flip()
```

```
# Once we have exited the main program loop
# we can stop the game engine:
pygame.quit()
```

# You Can Run the Code

- You can run the code as it is now.
- It may take 30 seconds or more to start.
- Next time, we'll add to the game.

# Let the PyGames Begin

Professor Bob Brown

College of Computing and Software Engineering  
Kennesaw State University

[Bob.Brown@Kennesaw.edu](mailto:Bob.Brown@Kennesaw.edu)