Python Programming

Stensat Group LLC

Introduction to Python

- Python is an interpreted language. This means that a compiler is not used to generate a executable file. Instead, the python program interprets the written program directly.
- This lesson assumes basic knowledge of python. You can learn the basics of python at www.learnpython.org
- There are a few ways to write python programs. The method shown here is very simple and straight forward.
- This lesson will introduce you to the pygame library and the socket library that will be used to interact with the SYST101 and SYST395 kits.
- Both Windows and Mac OS X will be covered.

Python Installation for Windows

- For Windows, it is assumed Windows 10 or Windows 7 is being used. Python version 3.7.2 will be used.
- Go to www.python.org/downloads/release/python-372/
- Scroll down to Files and select **Windows x86-64 executable installer**.
- Once downloaded, start the installer program. At the start windows, select **Add Python 3.7 to Path** then click on **Install Now.**
- Python will now install and show up in the Start menu. Close the window when the installation has completed.

PyGame Installation for Windows

- In the start up menu, select **Windows Powershell** and select **Windows Powershell**. Do not select the ISE version.
- In the Powershell, type **pip install pygame** and press **Enter**. This will install the pygame library for Python.
- Once completed, exit **Windows Powershell**. This completes the installation.
- To start Python, select **Python IDLE** in the **Start** menu.

Python installation for Mac

- It is assumed the latest Mac OS X release is being used.
- Go to www.python.org/downloads/mac-osx
- Locate **python 3.7.2 – Dec 24, 2018** and select **Mac OS 64-bit installer**.
- The installer will be downloaded. Once downloaded, double click on the program and it will go through the installation process.
- Once installed, the Python 3.7 will be located in the Application folder. Open the Python 3.7 folder and you can double click on IDLE.app to start Python.

PyGame for Mac

- Open the terminal application.
- In the terminal application enter the following:
  - **pip3 install pygame** and press **Enter.**
- The library has been installed.

Python IDLE

- The Python IDLE is a python shell. You can enter python commands and and they will execute immediately.
- Start Python IDLE. At the **>>>** prompt, enter the command:

```
>>>print(“Hello world”)
Hello world
```

- Equations can be entered and Python will generate the answer:

```
>>>5+3
8
```

- Python uses indentation to define a block of code. This means all code indented after an if statement, while statement or other conditional statement will be executed.

```
if 10 > 5:
    print(“10 is larger”)
    print(“Second line”)
```

-

Python IDLE

- Here is a while loop:

```
while 1:
    print(“this is a loop”)
```

- You can stop the while loop by pressing <Ctrl> and C keys.

Python IDLE

- The IDLE will also let you create new programs using an editor.
- Select menu File and New.
- An editor will open and you can enter a python program.
- Try the program on the right.
- Save the file. You will be prompted for a name. By convention, use .py for the file name extension.

```
while 1:
    print(“this is a loop”)
```

- Once saved, select menu **Run** and **Run Module**.
- Press CTRL and C to stop the program.
- Now you should have a basic understanding on using the IDLE.

More Basic Python

- Variables are used to store values or strings. You do not need to declare what type of variable it is. It is automatically determined when a value is assigned.
- Variables can change types by assignment.
- Variables must start with a letter. It cannot start with a number.
- Variables can only be made up of letters, numbers and _.

```
a = 5
b = “Fred”
print(a)
print(b)
print(a,b)
```

```
a = 5
print(a)
a = “Fred”
print(a)
```

Stensat Group LLC

Python Math

- Math operations is simple in Python. Just use equations.
- Notice the # sign and the text to the right. This is a comment. Comments are useful to explain what the code is supposed to do.

```
a = 5 + 6              # addition
b = 12.43              # floating point
c = b – 3.14           # subtraction
dog = 45.834           # assignment
cat = dog * b          # multiplication
dd = cat / a           # division
```

Python Comparison

- Variable and values can be compared with each other. Python provides multiple ways to do comparisons

  == is equal

  != is not equal

  > greater than

  < less than

  >= greater than or equal

  <= less than or equal

- Enter the code into a program and run it. You can change the values of the variables and see how execution changes.

```
a = 45
b = 32
if a > b:
    print(“a is greater than b”)
if a < b:
    print(“a is less than b”)
if a == b:
    print(“a is equal to b”)
if a != b:
    print(“a is not equal to b”)

if a > b:
    print(“a is greater”)
elif:
    print(“b is greater”)
```

Python Input

- A python program can prompt for an input to by typed.
- The input function uses a string argument to display a prompt.
- Once a user types in the information and presses the ENTER key, the program will put the information in a string variable.
- The print statement displays the result.

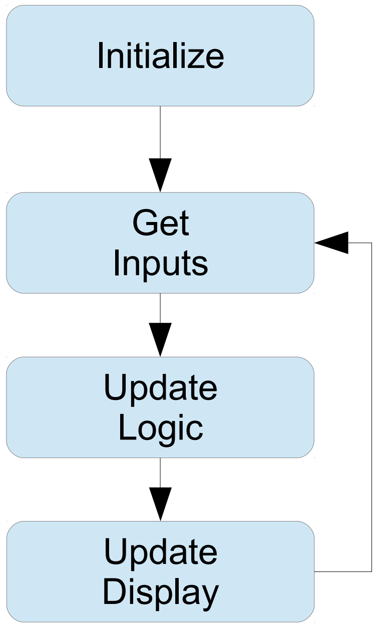
```
while 1:
     v = input("Enter your name: ")
     print("Your name is",v)
```

Python Rules of Coding

- When indenting, it is very critical to use the same number of spaces otherwise the code will not be interpreted correctly.
- Indented code is used to identify code associated with while, for, if and other conditional statements.
- Tabs and spaces are not interchangeable and will cause program errors or incorrect code execution.

PyGame

- Pygame is a library that provides the user access to graphics and user interfaces for developing games or other interactive programs. Pygame is a relatively simple library that allows simple programming.
- A program using Pygame can be divided into a few sections.
  - The first section is initialization which is done once
  - The rest of the sections are part of a large continuous loop.
    - The second section deals with user inputs. It can be used to detect keyboard events and mouse events and even joystick events.
    - The third section is the logic section that interprets the input events and other activities and updates variables and objects.
    - The fourth section updates the display. This is where the graphics is updated based on user input and logic updates.



First Program

- Try the program to the right.
- The bold area is the initialization section. The pygame library is initialized and a window is created. Variable done is set to false. This is used to exit the program.

```
import pygame

pygame.init()
scn = pygame.display.set_mode((800,400))
done = False

while not done:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      done = True
      pygame.quit()
      quit()
  pygame.draw.circle(scn,(255,255,255),(400,200),50)
  pygame.display.flip()
```

First Program

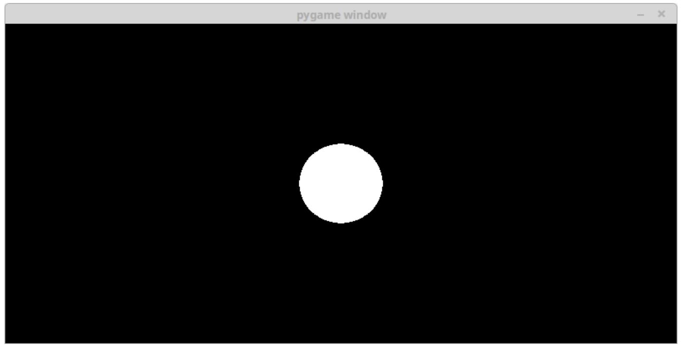
- The next section processes any inputs.
- The for loop goes through any events detected.
- The only event to be detected is the click of the close button on the program window.
- **pygame.quit()** will only close the program window. **quit()** is needed to exit the python program.

```
import pygame

pygame.init()
scn = pygame.display.set_mode((800,400))
done = False

while not done:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      done = True
      pygame.quit()
      quit()
  pygame.draw.circle(scn,(255,255,255),(400,200),50)
  pygame.display.flip()
```

First Program

- The last section is the display update. A circle is drawn in the center of the window and the **pygame.display.flip()** function updates the window.

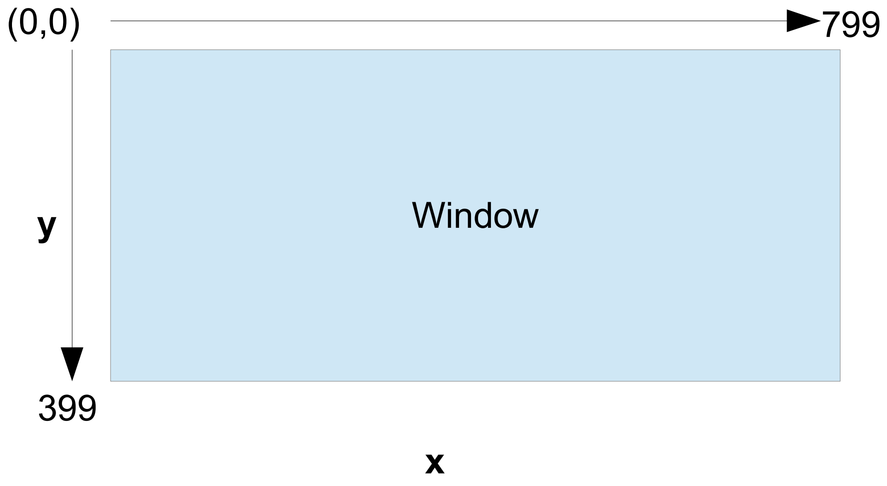
```
import pygame

pygame.init()
scn = pygame.display.set_mode((800,400))
done = False

while not done:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      done = True
      pygame.quit()
      quit()
  pygame.draw.circle(scn,(255,255,255),(400,200),50)
  pygame.display.flip()
```



Result of Program

First Program

- Pygame uses a coordinate system shown below. 0,0 is at the top left corner.
- pygame.display.set_mode((800,400)) uses a tuple for an argument. The tuple is the x,y size of the screen.
- You will notice that pygame library uses tuples for different arguments such as position and color.



First Program

- The for loop will sequence through all events captured and queued.
- Once there are no more events, the for loop will exit.
- This section is where all events are processed.
- pygame.QUIT event is triggered when the Close button on the window is clicked.

```
for event in pygame.event.get():
  if event.type == pygame.QUIT:
    done = True
    pygame.quit()
    quit()
```

First Program

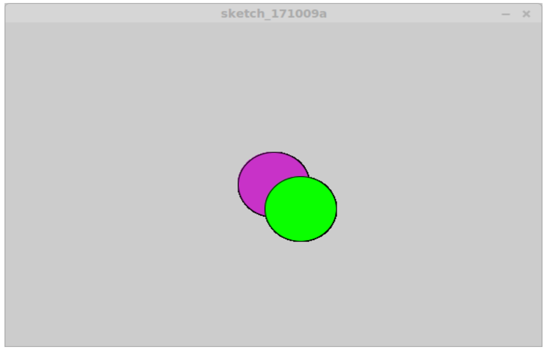
- The pygame.draw.circle() function renders a solid circle.
  - The first argument identifies the window to use.
  - The second argument is a tuple for the color of the circle. The value ranges from 0 to 255 and sets the intensity of red, green and blue.
  - The third argument is a tuple specifying the position of the circle. The coordinate is the center of the circle.
  - The last argument is the radius.

```
pygame.draw.circle(scn,(255,255,255),(400,200),50)
```

Adding to the First Program

- Add a second circle to the program after the first circle. Change the color and position it 50 pixels to the left and down.
- Run the program again.
- The second circle should be covering the first one partially. This shows that the order of rendering objects is important and newer objects will be rendered on top of older objects.

```
pygame.draw.circle(scn,(0,255,0),(450,250),50)
```



Ellipse drawn over ellipse

Interacting with the Program

- The mouse can be used to interact with the program. There is a function to detect when a mouse button is pressed.
- **pygame.mouse.get_pressed()[0]** is checked to determine if the left mouse button is pressed. Change 0 to 1 and the right mouse button is checked.
- The result is true if the mouse button is pressed.
- The coordinates for the mouse are taken from the **pygame.mouse.get_pos()** function.

```
import pygame

pygame.init()
scn = pygame.display.set_mode((800,400))
done = False
mX = 0
mY = 0
while not done:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      done = True
      pygame.quit()
      quit()
  if pygame.mouse.get_pressed()[0] == True:
    mX,mY = pygame.mouse.get_pos()
```

Interacting with the Program

- A new function added is **scn.fill()**. This function erases the window with the specified color. The window is erased to black.
- A circle is then drawn with the position at the mouse coordinates.
- try out the program.

```
import pygame

pygame.init()
scn = pygame.display.set_mode((800,400))
done = False
mX = 0
mY = 0
while not done:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      done = True
      pygame.quit()
      quit()
  if pygame.mouse.get_pressed()[0] == True:
    mX,mY = pygame.mouse.get_pos()
  scn.fill((0,0,0))
  pygame.draw.circle(scn,(255,0,255),(mX,mY),0)
  pygame.display.update()
```

Keyboard Interaction

- Pygame can detect when a key is pressed.
- The event is called KEYDOWN. This event will occur as long as a key is pressed.
- The program changes the color of circle based on which key is pressed. The up and down arrow keys are used.
- Modify the program and make all the changes in bold.

```
import pygame

pygame.init()
scn = pygame.display.set_mode((800,400))
done = False
mX = 0
mY = 0
color = (255,255,255)
while not done:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      done = True
      pygame.quit()
      quit()
    if event.type == pygame.KEYDOWN:
      if event.key == pygame.K_UP:
        color = (0,0,255)
      if event.key == pygame.K_DOWN:
        color = (255,0,0)
  if pygame.mouse.get_pressed()[0] == True:
    mX,mY = pygame.mouse.get_pos()
  scn.fill((0,0,0))
  pygame.draw.circle(scn,color,(mX,mY),0)
  pygame.display.update()
```

Keyboard Interaction

- Any key on the keyboard can be used. Pygame identifies them with names.
- The names always start with **pygame.K_**
- For number keys, add the number to the end of the name. For letters, add the letter. The letters are case sensitive so upper and lower case keys can be identified.
  - pygame.K_a
  - pygame.K_A

```
import pygame

pygame.init()
scn = pygame.display.set_mode((800,400))
done = False
mX = 0
mY = 0
color = (255,255,255)
while not done:
  for event in pygame.event.get():
    if event.type == pygame.QUIT:
      done = True
      pygame.quit()
      quit()
    if event.type == pygame.KEYDOWN:
      if event.key == pygame.K_UP:
        color = (0,0,255)
      if event.key == pygame.K_DOWN:
        color = (255,0,0)
  if pygame.mouse.get_pressed()[0] == True:
    mX,mY = pygame.mouse.get_pos()
  scn.fill((0,0,0))
  pygame.draw.circle(scn,color,(mX,mY),0)
  pygame.display.update()
```

Other Graphics

- Pygame has other shapes that can be rendered.
  - Rectangle

    pygame.draw.rect(scn,(color),(x,y,width,height),width)
  - Ellipse

    pygame.draw.ellipse(scn,(color),x,y,width,height),width)
  - When width is zero, the shape is filled with the color otherwise the shape is drawn as an outline with width being the number of pixels wide of the outline.
- Other shapes can be found at https://www.pygame.org/docs/ref/draw.html

End

- More details about pygame can be found here:
  - https://www.pygame.org/docs/