Python & Pygame
RU4CS - August 19, 2014
Lars Sorensen - Laboratory for Computer Science Research
Rutgers University, the State University of New Jersey
Who Am I?

• Student Computing at the Laboratory for Computer Science Research, an affiliated center with the Department of Computer Science at Rutgers University

• Computer Science teacher at the HEROES Academy in New Brunswick.

• Working on my PhD in Education at the Graduate School of Education where I study Game Based Learning.
Who Am I?
HOW ARE WE GOING TO DO THIS?

- Python Basics
- What is Pygame
- Pygame Basics
- Next Level
- Lets look at a Tetris game
About Python

Python is a high-level general purpose programming language that emphasizes the readability of computer code.

It is dynamically typed and supports multiple paradigms including imperative, object oriented and functional styles.

It is well supported, a favorite of scientists and since it’s beginnings in 1989 has risen to be one of the most popular programming languages in use today. It is now used in the introductory sequence at 8 of the 10 highest ranking CS universities.

http://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext
About Python

Python is a good language to use for education for a few reasons:

It takes syntax out of the way. Many people find that pseudo code written out before coding is strikingly similar to the final Python code.

Not a toy language. Python, as it is learned, is useful in the business and scientific world and is well supported with many third party libraries and modules.

It’s fun and intuitive. Once you get started with Python you’ll see what I mean.
Basics

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough.
Errors should never pass silently.
In the face of ambiguity, refuse the temptation to import.
There should be one-- and preferably only one--
Now is better than never.
Although never is often better than ever.
Namespaces are one honking great idea.

# This is my first python program
# Author: Lars Sorensen

###

and then all of the lines are ignored

###

```python
x=5
y=7
z=x+y
print("The answer is:", z)
```

```bash
hello.py - C:\SimplePython\PythonPrograms\hello.py
```

```python
# This is my first python program
# Author: Lars Sorensen

###

and then all of the lines are ignored

###

```python
x=5
y=7
z=x+y
print("The answer is:", z)
```

```bash
hello.py - C:\SimplePython\PythonPrograms\hello.py
```
Sequences

```python
>>> mystr = 'Lars Sorensen'

>>> mystr[5]
'S'

>>> mylist = [23, "Lars", 45, "Sorensen"]

>>> mylist[3]
'Sorensen'

>>> mylist[2]
45
```
Sequences

```python
>>> str1
'hello'
>>> str1[0:3]
'hel'
>>> str1[3:5]
'lo'
>>> str1[:2]
'he'
```
```python
>>> str2
'spam'
>>> str3
'This is a really really really long string'
>>> len(str2)
4
>>> len(str3)
42
```
Loops

```python
>>> for i in range(6):
    print(i)

0
1
2
3
4
5
```

```python
>>> range(5, 10)
5 through 9

>>> range(0, 10, 3)
0, 3, 6, 9

>>> range(-10, -100, -30)
-10, -40, -70
```

```python
# This is a comment
#
# Author: Lars Sorensen
#
# Program to show how for loops work...
#
print("\nThis is my for loop program")

for i in range(10):
    print(" i = ", i)

print("\nThe End")
```

>>> # This is my for loop program
i = 0
i = 1
i = 2
i = 3
i = 4
i = 5
i = 6
i = 7
i = 8
i = 9
```
```
Loops

```python
# This is a comment.
#
# Author: Lars Sorensen
#
# Program to show how to iterate over a list.
#
print("\nThis is my for loop program")
MyList = [2, 4, 6, 8, 10, 12]

for i in MyList:
    print("i = ", i)

print("\nThe End")
```

```
This is my for loop program
i = 2
i = 4
i = 6
i = 8
i = 10
i = 12

The End
```
Loops

```python
# this is a comment
#
# Author: Lars Sorensen
#
# a simple program to play with a while loop
#
print("\nThis is the while loop program")

c = 0

while (c<10):
    print(c, "is less than 10!")
    c = c + 1

print("\nThe End")
```

This is the while loop program
0 is less than 10!
1 is less than 10!
2 is less than 10!
3 is less than 10!
4 is less than 10!
5 is less than 10!
6 is less than 10!
7 is less than 10!
8 is less than 10!
9 is less than 10!
The End
```
Loops

```python
# this is a comment
#
# Author : Lars Sorensen
#
# a simple program to play with a while loop
#
print("\nThis is the indefinite while loop program")

c = 'c'

while (c != 'a') : 
    print("The char is ":, c)
    c = input("Enter the next char: ")

print("\nThe End")
```

```python
>>> This is the indefinite while loop program
The char is : c
Enter the next char: f
The char is : f
Enter the next char: g
The char is : g
Enter the next char: t
The char is : t
Enter the next char: y
The char is : y
Enter the next char: u
The char is : u
Enter the next char: j
The char is : j
Enter the next char: a

The End
```
Functions

```python
# this is a comment
#
# Author: Lars Sorensen
#
# This program uses a small multiplication function
#

def times(a, b):
    print(a*b)

print("This is the multiplication function program\n")
times(2,5)
times(100,4)
```

```bash
>>> >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> RESTART >>>>
>>> This is the multiplication function program

10
400
```
Functions

```python
# this is a comment
#
# Author: Lars Sorensen
#
# This program uses a small multiplication function
#

def times(a, b):
    print(a*b)

print("This is the multiplication function program\n")
times(2,5)
times(100,4)

>>> -------------------------------  RESTART -------------------------------
>>> 
This is the multiplication function program

10
400
Traceback (most recent call last):
  File "C:\SimplePython\PythonPrograms\times.py", line 14, in <module>
    print(a)
NameError: name 'a' is not defined
```
Functions

```python
# this is a comment
#
# Author: Lars Sorensen
#
# This program uses a small multiplication function
#

def times(a, b):
    return a*b

print("This is the multiplication function program")
a = times(2,5)  # remember, times is a fruitful function
print(a)

>>> ================================ RESTART ================================
>>> 
This is the multiplication function program

10
```
OOP

```python
# this is a comment
#
# Author: Lars Sorensen
#
# A program that creates and uses a simple class
#
from random import randrange

class MyDie:
    def __init__(self, sides):
        self.sides = sides
        self.value = 1

    def roll(self):
        self.value = randrange(1, self.sides+1)

    def getValue(self):
        return self.value

    def setValue(self, value):
        self.value = value

# The main program

d1 = MyDie(6)
print(d1.getValue())

d1.roll()
print(d1.getValue())

d2 = MyDie(20)
for i in range(20):
    d2.roll()
    print(d2.getValue())

>>> =============
```

PYTON BASICS
6. Modules

If you quit from the Python interpreter and enter it again, the definitions you have made (functions and variables) are lost. Therefore, if you want to write a somewhat longer program, you are better off using a text editor to prepare the input for the interpreter and running it with that file as input instead. This is known as creating a script. As your program gets longer, you may want to split it into several files for easier maintenance. You may also want to use a handy function that you’ve written in several programs without copying its definition into each program.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module (the collection of variables that you have access to in a script executed at the top level and in calculator mode).

A module is a file containing Python definitions and statements. The file name is the module name with the suffix .py appended. Within a module, the module’s name (as a string) is available as the value of the global variable __name__. For instance, use your favorite text editor to create a file called fibo.py in the current directory with the following contents:

```python
# Fibonacci numbers module
def fib(n):  # write Fibonacci series up to n
    a, b = 0, 1
    while b < n:
        print(b)
        a, b = b, a + b
```
Python Basics

Let's do a Hello World
Tons of Resources for Learning Python
## Tons of Resources for Learning Python

<table>
<thead>
<tr>
<th>ID</th>
<th>Description / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Multiples of 3 and 5</td>
</tr>
<tr>
<td>2</td>
<td>Even Fibonacci numbers</td>
</tr>
<tr>
<td>3</td>
<td>Largest prime factor</td>
</tr>
<tr>
<td>4</td>
<td>Largest palindrome product</td>
</tr>
<tr>
<td>5</td>
<td>Smallest multiple</td>
</tr>
<tr>
<td>6</td>
<td>Sum square difference</td>
</tr>
<tr>
<td>7</td>
<td>100th prime</td>
</tr>
<tr>
<td>8</td>
<td>Largest product in a series</td>
</tr>
<tr>
<td>9</td>
<td>Special Pythagorean triplet</td>
</tr>
<tr>
<td>10</td>
<td>Summation of primes</td>
</tr>
<tr>
<td>11</td>
<td>Largest product in a grid</td>
</tr>
<tr>
<td>12</td>
<td>Highly divisible triangular number</td>
</tr>
<tr>
<td>13</td>
<td>Large sum</td>
</tr>
<tr>
<td>14</td>
<td>Longest Collatz sequence</td>
</tr>
</tbody>
</table>
WHAT IS PYGAME?
WHAT IS PYGAME?

PyGame is a Python library for game development. It is designed to be easy to learn and use, and it provides a wide range of features for creating games.

PyGame supports a variety of programming constructs, including events, input, graphics, and sound. It also includes a built-in debugger, which makes it easy to debug your code.

PyGame is written in Python, which makes it easy to integrate with other Python programs. It is also compatible with other programming languages, such as C, C++, and Java.

PyGame is a powerful and flexible library that is perfect for anyone looking to create games in Python.
About

Pygame is simply a set of modules designed for writing video games with the Python programming language.

The project began in the summer of 2000 as a Python wrapper for the SDL (Simple DirectMedia Library) that was being used with C.

It has been an open community project since about 2005 and now has been downloaded over a million times.
What is Pygame?

About

Pygame does not require OpenGL or any other libraries to function.

It’s truly portable and runs on any system that Python does.

It’s simple and fun, allowing the programmer to get up to speed quickly with simple 2D game development.

The project homepage can be found at: http://pygame.org/news.html
WHAT IS PYGAME?

[Images of various games and game interfaces]
PYGAME BASICS

- Hello World
- Game Loops
- Graphics Coordinates
- Draw Shapes
- Animation
- Sound
Hello World for Pygame

```python
# Hello World for Pygame
#
# Lars Sorensen for RU4CS in August 2014
#

import pygame, sys
from pygame.locals import *

pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Hello World!')

# Game Loop
while True:  # main game loop
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    pygame.display.update()
```
import pygame, sys
from pygame.locals import *

pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Hello World!')
Pygame Basics

Game Loop

- Handle Events
- Update Game State
- Draw Screen
Game Loops and Game States

```python
# Game Loop
while True:  # main game loop
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    pygame.display.update()
```
Hello World

# Hello World for Pygame
#
# Lars Sorensen for RU4CS in August 2014
#

import pygame, sys
from pygame.locals import *

pygame.init()
DISPLAYSURF = pygame.display.set_mode((400, 300))
pygame.display.set_caption('Hello World!')

# Game Loop
while True:  # main game loop
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
        pygame.display.update()
Hello World
Graphics - How the Screen is Handled

The 2D Screen Coordinates: The origin is located at the top-left corner, with x-axis pointing left and y-axis pointing down.
Before we begin to draw we need to understand colors

<table>
<thead>
<tr>
<th>Color</th>
<th>RGB Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aqua</td>
<td>(0, 255, 255)</td>
</tr>
<tr>
<td>Black</td>
<td>(0, 0, 0)</td>
</tr>
<tr>
<td>Blue</td>
<td>(0, 0, 255)</td>
</tr>
<tr>
<td>Fuchsia</td>
<td>(255, 0, 255)</td>
</tr>
<tr>
<td>Gray</td>
<td>(128, 128, 128)</td>
</tr>
<tr>
<td>Green</td>
<td>(0, 128, 0)</td>
</tr>
<tr>
<td>Lime</td>
<td>(0, 255, 0)</td>
</tr>
<tr>
<td>Maroon</td>
<td>(128, 0, 0)</td>
</tr>
<tr>
<td>Navy Blue</td>
<td>(0, 0, 128)</td>
</tr>
<tr>
<td>Olive</td>
<td>(128, 128, 0)</td>
</tr>
<tr>
<td>Purple</td>
<td>(128, 0, 128)</td>
</tr>
<tr>
<td>Red</td>
<td>(255, 0, 0)</td>
</tr>
<tr>
<td>Silver</td>
<td>(192, 192, 192)</td>
</tr>
<tr>
<td>Teal</td>
<td>(0, 128, 128)</td>
</tr>
<tr>
<td>White</td>
<td>(255, 255, 255)</td>
</tr>
<tr>
<td>Yellow</td>
<td>(255, 255, 0)</td>
</tr>
</tbody>
</table>
Color Objects

16,777,216 different colors

```python
# set up the colors
BLACK = ( 0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = ( 0, 255, 0)
BLUE = ( 0, 0, 255)
```

You can create colors with varying opacity by adding a fourth “alpha” variable with 255 being completely opaque and 0 being completely transparent. You need to setup a special drawing Surface to do this.
Drawing Simple Shapes

We will start by drawing a simple rectangle

First some setup

```python
import pygame, sys
from pygame.locals import *

pygame.init()

# set up the window
DISPLAYSURF = pygame.display.set_mode((500, 400), 0, 32)
pygame.display.set_caption('Drawing')

# set up the colors
BLACK = (  0,   0,   0)
WHITE = (255, 255, 255)
RED   = (255,   0,   0)
GREEN = (  0, 255,   0)
BLUE  = (  0,   0, 255)

# draw on the surface object
DISPLAYSURF.fill(WHITE)
```
Drawing Simple Shapes

We will start by drawing a simple rectangle

Now we create a rectangle object

```python
# Make a rectangle object and draw it
myRect = pygame.Rect(10, 10, 100, 100)
```
Drawing Simple Shapes

We will start by drawing a simple rectangle

Then we draw it...

```python
pygame.draw.rect(DISPLAYSURF, BLUE, myRect, 4)
```
# Make a rectangle object and draw it
myRect = pygame.Rect(10,10,100,100)
pygame.draw.rect(DISPLAYSURF, BLUE, myRect, 4)

print(myRect.top, myRect.left)
print(myRect.bottom, myRect.right)

These results are displayed in the IDLE output →

$$\begin{array}{c}
>>> 10 10 \\
>>> 110 110
\end{array}$$
Here's the whole shebang...

```python
import pygame, sys
from pygame.locals import *

pygame.init()

# set up the window
DISPLAYSURF = pygame.display.set_mode((500, 400), 0, 32)
pygame.display.set_caption('Drawing')

# set up the colors
BLACK = ( 0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

# draw on the surface object
DISPLAYSURF.fill(WHITE)

# Make a rectangle object and draw it
myRect = pygame.Rect(10, 10, 100, 100)
pygame.draw.rect(DISPLAYSURF, BLUE, myRect, 4)

print(myRect.top, myRect.left)
print(myRect.bottom, myRect.right)

# run the game loop
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
            pygame.display.update()
```
Drawing Simple Shapes

Here’s the results...
Drawing Simple Shapes

Because we used objects we can just change the attributes

# Make a rectangle object and draw it
myRect = pygame.Rect(10,10,200,300)
pygame.draw.rect(DISPLAYSURF, GREEN, myRect, 4)
Drawing Simple Shapes

Now, what if we don’t want to create an object for every single line and shape on the screen? If we know we will not be using the object over and over again we have to option to just draw to the screen.

```python
# draw a rect on the fly
pygame.draw.rect(DISPLAYSURF, RED, (200, 150, 100, 50))
```
Drawing Simple Shapes

Now lets try some other shapes...

```python
# draw a rect on the fly
pygame.draw.rect(DISPLAYSURF, RED, (200, 150, 100, 50))

pygame.draw.polygon(DISPLAYSURF, GREEN, [(146, 0), (291, 106), (236, 277), (56, 277), (0, 106)])

pygame.draw.line(DISPLAYSURF, BLUE, (60, 60), (120, 60), 4)
pygame.draw.line(DISPLAYSURF, BLUE, (120, 60), (60, 120))
pygame.draw.line(DISPLAYSURF, BLUE, (60, 120), (120, 120), 4)

pygame.draw.circle(DISPLAYSURF, BLUE, (300, 50), 20, 0)

pygame.draw.ellipse(DISPLAYSURF, RED, (300, 250, 40, 80), 1)
```

Notice the ellipse. Here we give it the coordinates for a rectangle
And pygame draws an ellipse inside it...
Drawing Simple Shapes

```python
import pygame
sys
from pygame.locals import *

pygame.init()

# set up the window
DISPLAYSURF = pygame.display.set_mode((500, 400), 0, 32)
pygame.display.set_caption('Drawing')

# set up the colors
BLACK = ( 0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)

# draw on the surface object
DISPLAYSURF.fill(WHITE)

# draw a rect on the fly
pygame.draw.rect(DISPLAYSURF, RED, (200, 150, 100, 50))

pygame.draw.polygon(DISPLAYSURF, GREEN, ((146, 0), (291, 106), (236, 277), (56, 277), (0, 106)))

pygame.draw.line(DISPLAYSURF, BLUE, (60, 60), (120, 60), 4)
pygame.draw.line(DISPLAYSURF, BLUE, (120, 60), (60, 120))
pygame.draw.line(DISPLAYSURF, BLUE, (60, 120), (120, 120), 4)

pygame.draw.circle(DISPLAYSURF, BLUE, (300, 50), 20, 0)

pygame.draw.ellipse(DISPLAYSURF, RED, (300, 250, 40, 80), 1)

# run the game loop
while True:
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    pygame.display.update()
```
Drawing Simple Shapes
Animation
Animation

Just think about flip books
Animation

```python
import pygame, sys
from pygame.locals import *

pygame.init()

import time
FPS = 18 # frames per second setting
fpsClock = pygame.time.Clock()
```

Important to create a clock object and control the time the image is on the screen. Otherwise the computer will run as fast as it can...
Animation

Setup the initial screen

```python
# set up the window
DISPLAYSURF = pygame.display.set_mode((400, 300), 0, 32)
pygame.display.set_caption('cat animation')

WHITE = (255, 255, 255)
catImg = pygame.image.load('cat.png')
catx = 280
caty = 220
direction = 'left'
```
Animation

The game loop begins -

We see what direction the Cat is going in and move it 5 pixels in that direction

Until we hit the limit of the screen, then we change direction

At the end of each state change we update the screen.

```python
while True:  # the main game loop
    DISPLAYSURF.fill(WHITE)

    if direction == 'right':
        catx += 5
        if catx == 280:
            direction = 'down'
    elif direction == 'down':
        caty += 5
        if caty == 220:
            direction = 'left'
    elif direction == 'left':
        catx -= 5
        if catx == 10:
            direction = 'up'
    elif direction == 'up':
        caty -= 5
        if caty == 10:
            direction = 'right'

    DISPLAYSURF.blit(catImg, (catx, caty))
```
Animation

Now we jump into the event detection loop

```python
for event in pygame.event.get():
    if event.type == QUIT:
        pygame.quit()
        sys.exit()

    pygame.display.update()
    fpsClock.tick(FPS)
```

See how we tick our clock above
import pygame, sys
from pygame.locals import *

pygame.init()

import time
FPS = 18  # frames per second setting
fpsClock = pygame.time.Clock()

# set up the window
DISPLAYSURF = pygame.display.set_mode((400, 300), 0, 32)
pygame.display.set_caption('cat animation')

WHITE = (255, 255, 255)
catImg = pygame.image.load('cat.png')
catx = 280
caty = 220
direction = 'left'

while True:  # the main game loop
    DISPLAYSURF.fill(WHITE)
    if direction == 'right':
        catx += 5
    if catx == 200:
        direction = 'down'
    elif direction == 'down':
        caty += 5
    if caty == 220:
        direction = 'left'
    elif direction == 'left':
        catx -= 5
    if catx == 10:
        direction = 'up'
    elif direction == 'up':
        caty -= 5
    if caty == 10:
        direction = 'right'

    DISPLAYSURF.blit(catImg, (catx, caty))

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
        pygame.display.update()
    fpsClock.tick(FPS)
PYGAME BASICS

Animation
Animation

Let's play with the speed

How could we change direction?
Playing Sounds
Playing Sounds

I am not sure there is working sound on these machines so you might be just following along if you are using a Rutgers machine.

That’s OK.

Now. Like a lot of simple 2D video games we will have a sound Play during the game play and then an ending sequence when the Game is over.
Playing Sounds

```python
soundObj2 = pygame.mixer.Sound('gameover.wav')
soundObj = pygame.mixer.Sound('beeps.wav')

while True:  # the main game loop
    DISPLAYSURF.fill(WHITE)
    soundObj.play()
```
Playing Sounds

```python
for event in pygame.event.get():
    if event.type == QUIT:
        soundObj.stop()
        soundObj2.play()
        time.sleep(3)
        pygame.quit()
        sys.exit()
```
Playing Sounds

We can also setup background music...

```python
pygame.mixer.music.load('plush.mp3')
pygame.mixer.music.play(-1, 0.0)
```
Playing Sounds

**Pygame Basics**

**pygame.mixer**

*pygame module for loading and playing sounds*

- `pygame.mixer.init` — initialize the mixer module
- `pygame.mixer.pre_init` — preset the mixer init arguments
- `pygame.mixer.quit` — uninitialized the mixer
- `pygame.mixer.get_init` — test if the mixer is initialized
- `pygame.mixer.quit` — stop playback of all sound channels
- `pygame.mixer.pause` — temporarily stop playback of all sound channels
- `pygame.mixer.unpause` — resume paused playback of sound channels
- `pygame.mixer.fadeout` — fade out the volume on all sounds before stopping
- `pygame.mixer.set_num_channels(num_channels)` — set the total number of playback channels
- `pygame.mixer.get_num_channels()` — get the total number of playback channels
- `pygame.mixer.set_reserved(num_channels)` — reserve channels from being automatically used
- `pygame.mixer.find_channel()` — find an unused channel
- `pygame.mixer.get_busy()` — test if any sound is being mixed
- `pygame.mixer.Sound(filename)` — Create a new Sound object from a file or buffer object
- `pygame.mixer.Channel(channel)` — Create a Channel object for controlling playback

This module contains classes for loading Sound objects and controlling playback. The mixer module is optional and depends on SDL_mixer. Your program must call `pygame.mixer.init()` to initialize the mixer module. The mixer module has a limited number of channels for playback of sounds. Usually programs tell pygame to start playing audio and it selects an available channel over the number of channels and their use.

All sound playback is mixed in background threads. When you begin to play a Sound object, it will return immediately while the sound continues to play. The mixer also has a special streaming channel. This is for music playback and is accessed through the `pygame.mixer.music` module.

The mixer module must be initialized like other Pygame modules, but it has some extra conditions. The `pygame.mixer.init()` function takes several options. If you cannot perform Sound resampling, so the mixer should be initialized to match the values of your audio resource.

**Note:** Not to get less laggy sound, use a smaller buffer size. The default is set to reduce the chance of scratchy sounds on older computers. You can change it by example: `pygame.mixer.pre_init(44100, -16, channels=2, buffer=4096)`. The default size was changed from 1024 to 3072 in Pygame 1.8.

**pygame.mixer.init()**

*Initialize the mixer module*

```
init(frequency=44100, size=-16, channels=2, buffer=4096) -> None
```

Initialize the mixer module for Sound loading and playback. The default arguments can be overridden to provide specific audio mixing. Keyword arguments for the Sound module are:

- `frequency`: Specifies the frequency of the sound in Hertz (Hz). The default is 44100 Hz.
- `size`: Specifies the bits per sample, which determines the audio depth. The default is 16 bits per sample.
- `channels`: Specifies the number of channels. The default is 2 channels.
- `buffer`: Specifies the buffer size in bytes. The default is 4096 bytes.
NEXT LEVEL STUFF
• Collision Detection
• Handling Keyboard Input
Collision Detection
Collision Detection

I am going to give you the code for “The Reaper”

We will get it running and then trace the code to see what is Going on with it...
Collision Detection

Reaper code is found at http://www.cs.rutgers.edu/~biglars/RU4CS/reaper.html

Create a new window in IDLE

Paste in the code, save and run.

Reaper has no special resources so it will run with only the code, no pictures or music to worry about.
Collision Detection

Now let’s go back to the code and see what it’s doing

```python
def doRectsOverlap(rect1, rect2):
    for a, b in [(rect1, rect2), (rect2, rect1)]:
        # Check if a's corners are inside b
        if ((isPointInsideRect(a.left, a.top, b)) or
            (isPointInsideRect(a.left, a.bottom, b)) or
            (isPointInsideRect(a.right, a.top, b)) or
            (isPointInsideRect(a.right, a.bottom, b))):
            return True

    return False

def isPointInsideRect(x, y, rect):
    if (x > rect.left) and (x < rect.right) and (y > rect.top) and (y < rect.bottom):
        return True
    else:
        return False
```
Collision Detection

Some Setup

```python
# set up direction variables
DOWNLEFT = 1
DOWNRIGHT = 3
UPLEFT = 7
UPRIGHT = 9
MOVESPEED = 4
```
Collision Detection

# set up the colors
BLACK = (0, 0, 0)
GREEN = (0, 255, 0)
WHITE = (255, 255, 255)

# set up the bouncer and food data structures
foodCounter = 0
NEWFOOD = 40
FOODSIZE = 20
bouncer = {'rect':pygame.Rect(300, 100, 50, 50), 'dir':UPLEFT}
foods = []
for i in range(20):
    foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - FOODSIZE),
                           random.randint(0, WINDOWHEIGHT - FOODSIZE),
                           FOODSIZE, FOODSIZE))
Collision Detection

```python
# run the game loop
while True:
    # check for the QUIT event
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    foodCounter += 1
    if foodCounter >= NEWFOOD:
        # add new food
        foodCounter = 0
        foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - FOODSIZE),
                                  random.randint(0, WINDOWHEIGHT - FOODSIZE),
                                  FOODSIZE, FOODSIZE))

    # draw the black background onto the surface
    windowSurface.fill(BLACK)
```
Collision Detection

```python
# draw the black background onto the surface
windowsSurface.fill(BLACK)

# move the bouncer data structure
if bouncer['dir'] == DOWNLEFT:
    bouncer['rect'].left -= MOVESPEED
    bouncer['rect'].top += MOVESPEED
if bouncer['dir'] == DOWNRIGHT:
    bouncer['rect'].left += MOVESPEED
    bouncer['rect'].top += MOVESPEED
if bouncer['dir'] == UPLEFT:
    bouncer['rect'].left -= MOVESPEED
    bouncer['rect'].top -= MOVESPEED
if bouncer['dir'] == UPRIGHT:
    bouncer['rect'].left += MOVESPEED
    bouncer['rect'].top -= MOVESPEED
```
Collision Detection

```python
# check if the bouncer has move out of the window
if bouncer['rect'].top < 0:
    # bouncer has moved past the top
    if bouncer['dir'] == UPLEFT:
        bouncer['dir'] = DOWNLEFT
    if bouncer['dir'] == UPRIGHT:
        bouncer['dir'] = DOWNRIGHT
if bouncer['rect'].bottom > WINDOWHEIGHT:
    # bouncer has moved past the bottom
    if bouncer['dir'] == DOWNLEFT:
        bouncer['dir'] = UPLEFT
    if bouncer['dir'] == DOWNRIGHT:
        bouncer['dir'] = UPRIGHT
if bouncer['rect'].left < 0:
    # bouncer has moved past the left side
    if bouncer['dir'] == DOWNLEFT:
        bouncer['dir'] = DOWNRIGHT
    if bouncer['dir'] == UPLEFT:
        bouncer['dir'] = UPRIGHT
if bouncer['rect'].right > WINDOWWIDTH:
    # bouncer has moved past the right side
    if bouncer['dir'] == DOWNRIGHT:
        bouncer['dir'] = DOWNLEFT
    if bouncer['dir'] == UPRIGHT:
        bouncer['dir'] = UPLEFT
```
Collision Detection

```python
# draw the bouncer onto the surface
pygame.draw.rect(windowSurface, WHITE, bouncer['rect'])

# check if the bouncer has intersected with any food squares.
for food in foods[:]:
    if doRectsOverlap(bouncer['rect'], food):
        foods.remove(food)

# draw the food
for i in range(len(foods)):
    pygame.draw.rect(windowSurface, GREEN, foods[i])

# draw the window onto the screen
pygame.display.update()
mainClock.tick(40)
```
Collision Detection

5 times through the song
Collision Detection

7 Hours
Collision Detection

Almost a Day
Collision Detection
Collision Detection

Collision Detection final thoughts
Keyboard Input

Now we are going to control the movement of the reaper
Next Level Stuff

Keyboard Input

Go grab the kinput.py code here

http://www.cs.rutgers.edu/~biglars/RU4CS/kinput.html

Save this code to a file called kinput.py and get it running
Keyboard Input

```python
# set up movement variables
moveLeft = False
moveRight = False
moveUp = False
moveDown = False

# run the game loop
while True:
    # check for events
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
        if event.type == KEYDOWN:
            # change the keyboard variables
            if event.key == K_LEFT or event.key == ord('a'):
                moveRight = False
                moveLeft = True
            if event.key == K_RIGHT or event.key == ord('d'):
                moveLeft = False
                moveRight = True
            if event.key == K_UP or event.key == ord('w'):
                moveDown = False
                moveUp = True
            if event.key == K_DOWN or event.key == ord('s'):
                moveUp = False
                moveDown = True
```
`Keyboard Input`:

```python
if event.type == KEYUP:
    if event.key == K_ESCAPE:
        pygame.quit()
        sys.exit()
    if event.key == K_LEFT or event.key == ord('a'):
        moveLeft = False
    if event.key == K_RIGHT or event.key == ord('d'):
        moveRight = False
    if event.key == K_UP or event.key == ord('w'):
        moveUp = False
    if event.key == K_DOWN or event.key == ord('s'):
        moveDown = False
    if event.key == ord('x'):
        player.top = random.randint(0, WINDOWHEIGHT - player.height)
        player.left = random.randint(0, WINDOWWIDTH - player.width)

if event.type == MOUSEBUTTONDOWN:
    foods.append(pygame.Rect(event.pos[0], event.pos[1],
                              FOODSIZE, FOODSIZE))
```
foodCounter += 1
if foodCounter >= NEWFOOD:
    # add new food
    foodCounter = 0
    foods.append(pygame.Rect(random.randint(0, WINDOWWIDTH - FOODSIZE),
                              random.randint(0, WINDOWHEIGHT - FOODSIZE),
                              FOODSIZE, FOODSIZE))

    # draw the black background onto the surface
    windowSurface.fill(BLACK)

    # move the player
    if moveDown and player.bottom < WINDOWHEIGHT:
        player.top += MOVESPEED
    if moveUp and player.top > 0:
        player.top -= MOVESPEED
    if moveLeft and player.left > 0:
        player.left -= MOVESPEED
    if moveRight and player.right < WINDOWWIDTH:
        player.right += MOVESPEED

    # draw the player onto the surface
    pygame.draw.rect(windowSurface, WHITE, player)
Keyboard Input

```python
# check if the player has intersected with any food squares.
for food in foods[:]:
    if player.colliderect(food):
        foods.remove(food)
```

Important: Notice here that we let pygame do the collision detection for us and we don’t code our own
Keyboard Input

```python
#Keyboard Input

# run the game loop
while True:
    # check for events
    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
        if event.type == KEYDOWN:
            # change the keyboard variables
            if event.key == K_ESCAPE:
                pygame.quit()
                sys.exit()
            if event.key == K_LEFT or event.key == ord('a'):
                moveleft = True
                moveright = False
            if event.key == K_UP or event.key == ord('w'):
                moveup = True
                movedown = False
            if event.key == K_DOWN or event.key == ord('s'):
                movedown = True
                moveup = False
            if event.key == K_RIGHT or event.key == ord('d'):
                moveright = True
                moveleft = False
    # handle key releasing
    if event.type == KEYUP:
        if event.key == K_ESCAPE:
            pygame.quit()
            sys.exit()
        if event.key == K_LEFT or event.key == ord('a'):
            moveleft = False
            moveright = True
        if event.key == K_UP or event.key == ord('w'):
            moveup = False
            movedown = True
        if event.key == K_DOWN or event.key == ord('s'):
            movedown = False
            moveup = True
        if event.key == K_RIGHT or event.key == ord('d'):
            moveright = False
            moveleft = True

# set up game variables
moveleft = False
moveright = False
moveup = False
movedown = False
foodcounter = 1

# set up the screen
screensize = 640, 480
screensurface = pygame.display.set_mode(screensize)
pygame.display.set_caption('Input')

# set up the window
windowwidth = 400
windowheight = 200
windowSurface = pygame.display.set_mode((windowwidth, windowheight))

# set up the colors
BLACK = (0, 0, 0)
GREEN = (0, 255, 0)
WHITE = (255, 255, 255)
FOODSIZ = 16
FOODSIZE = 16
FOODSIZE = (FOODSIZ, FOODSIZ)

# set up the food and food data structure
foodcounter = 0
FOODS = 20
food = pygame.Rect(300, 100, 50, 50)
FOODS = [()]
for i in range(20):
    food = pygame.Rect(random.randint(0, windowwidth - FOODSIZE[0]),
                       random.randint(0, windowheight - FOODSIZE[1]),
                       FOODSIZE[0], FOODSIZE[1])
    food = food
    FOODS.append(food)

# set up movement variables
moveleft = False
moveright = False
moveup = False
movedown = False

# run the program
foodcounter = 1
if foodcounter > 1:
    # add new food
    foodcounter = 0
    FOODS.append(pygame.Rect(random.randint(0, windowwidth - FOODSIZE[0]),
                               random.randint(0, windowheight - FOODSIZE[1]),
                               FOODSIZE[0], FOODSIZE[1]))

    # draw the black background onto the surface
    windowSurface.fill(BLACK)

    # move the player
    if moveright and player.top < windowheight - player.height:
        if moveright and player.top < 0:
            moveright = False
    if moveleft and player.left > 0:
        if moveleft and player.left > windowwidth - player.width:
            moveleft = False
    if moveup and player.top > 0:
        if moveup and player.top > windowheight - player.height:
            moveup = False
    if movedown and player.bottom < windowheight:
        if movedown and player.bottom < windowheight - player.height:
            movedown = False

    # check the player has intersected with any food squares.
    for food in FOODS:
        if player.colliderect(food): food
    remove_food()

    # draw the food
    for food in FOODS:
        pygame.draw.rect(windowSurface, GREEN, food)

    # draw the window onto the screen
    pygame.display.update()
    mainClock.tick(60)
```
NEXT LEVEL STUFF

Keyboard Input

?
NEXT LEVEL STUFF

Keyboard Input
A LOOK AT TETRIS
A full game
The Tetris code is here:

http://www.cs.rutgers.edu/~biglars/RU4CS/tetris.html

Also, do not forget the Two music files! They Are MIDIs and are called

tetrisb.mid and tetrisc.mid

They are in the same directory as the tetris code. Place them in the same directory you keep your Python code saved and you’ll be OK
After today’s workshop there is nothing in the Tetris code that you haven’t seen before
LET’S TALK
What’s your initial take? Is Pygame something you would use in the classroom?
RESOURCES
Pygame Resources

http://www.pygame.org/news.html
Pygame Resources

http://inventwithpython.com/pygame/
Pygame Resources

Invent with Python
Learn to program by making computer games.

Invent Your Own Computer Games with Python
2nd Edition

Learn how to program!
"Invent Your Own Computer Games with Python" is a free book (as in, open source) and a free eBook (as in, no cost to download) that teaches you how to program in the Python programming language. Each chapter gives you the complete source code for a new game, and then teaches the programming concepts from the example.

"Invent with Python" was written to be understandable by kids as young as 10 to 12 years old, although it is great for anyone of any age who has never programmed before.

This second edition has revised and expanded content, including a Pygame tutorial to make games with graphics, animation, and sound.

The book and the programs are open source, and are available under a Creative Commons Attribution-Noncommercial-Share Alike License, meaning they are free to copy and distribute. See the license for full details. A print copy can be bought on Amazon.com for $25.

Buy the book on Amazon.com!
Read the book online for free!

Download the .pdf version for free. (Alternate link)

http://inventwithpython.com/index.html
Python Resources

**Biglars Python Resources**

**Learning Python**
- Python.org - The main Python page. A great portal for all things Python and the place to get software.
- PYCON.org - Python in Education - Special Interest group for Education.
- ...
- Learn Python: The Hard Way - Great site to begin learning Python with and it's free online.
- Invent Your Own Computer Games with Python - Great way to learn Python with a purpose, nice pacing.
- Computer Science with Python and Pygame: PCC gets it right, big fan of the site.
- Online Python Tutor - An online program visualization tool. Type in Python code and execute it step by step.

**Books**
- *Python Programming: An Introduction to Computer Science* by John Zelle's book is the only book I would ever have a student buy.
- *The Whole of OSS by goodness* - Great books to have around and fiddle with. Grab them if you see them lying around.

---

Python is a programming language that lets you work quickly and integrate systems more effectively. **Learn More**

**Get Started**
- Whether you're new to programming
- Download Python source code and installers
- Docs Documentation for Python
- Jobs Looking for work or have a Python job?
THANK YOU!