# Pygame

Knoxville Game Design

June 2020

Levi D. Smith

# Overview

- Pygame is similar to SDL in structure
- Good for SDL developers who don't want to deal with the complexities of C or C++
- No web builds.  Bad for game jams.
- Can take advantage of available Python libraries (pip)
- Not good for beginners (no built in GUI objects)

# Python differences from other languages

- Boolean expressions – “*or*”, “*and*”
  - No double pipe or ampersand;  Must be lowercase
- Boolean values – “True”, “False”
  - Case is important (first letter capitalized, other lowercase)
- Space is significant for methods, loops, and *if* statements
- No line termination (semicolon) or block characters (curly braces)
- *self* is implied first parameter
- No constants
- null is *None*
- *elif* instead of else if
- No *switch* statement (can simulate with dictionary)
- *toString* method is *__str__()*

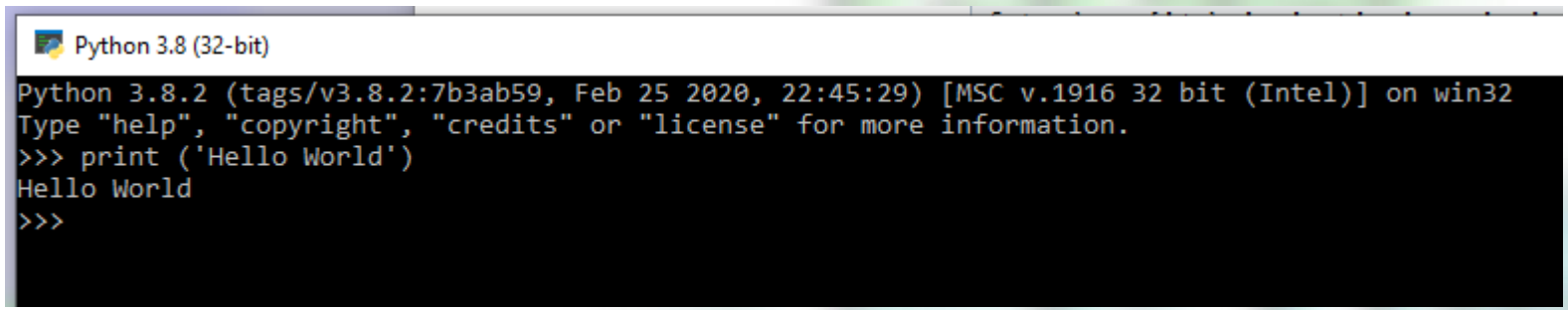
KNOX GAME DESIGN

# Installation

- Download Python from www.python.org/downloads
- Optional - add python install directory to path environment variable

# Hello World

- Start Python shell
- print ('hello world')



# Number guessing game

- Can put code in text file with .py extension
- Run by calling the python executable with the code file as a

numguess.py

```
1  from random import randrange
2
3  print('Number Guessing Game - 2020 Levi D. Smith')
4
5  iSecretNum = randrange(1,100)
6  iGuessNum = -1
7  iGuessCount = 0
8
9  while (iGuessNum != iSecretNum):
10     print('Guess the number from 1 to 100')
11     iGuessNum = int(input())
12     iGuessCount += 1
13
14     if (iGuessNum > iSecretNum):
15         print("Lower")
16     elif (iGuessNum < iSecretNum):
17         print("Higher")
18     else:
19         print("Correct!")
20
21 print('Secret number is ' + str(iSecretNum) + '.')
22 print('Total number of guesses ' + str(iGuessCount) + '.')
```

```
Number Guessing Game - 2020 Levi D. Smith
Guess the number from 1 to 100
50
Lower
Guess the number from 1 to 100
25
Higher
Guess the number from 1 to 100
37
Higher
Guess the number from 1 to 100
42
Lower
Guess the number from 1 to 100
39
Lower
Guess the number from 1 to 100
38
Correct!
Secret number is 38.
Total number of guesses 6.
```

KNOX GAME DESIGN

# Variables

- Types
  - int - 42
  - float - 3.14
  - string - "hello world" (single or double quotes)
- Casting
  - number to string - str(42)
  - string to number - int("42")
- Concatenate two strings
  - "foo " + str(42)
- Arrays
  - fib = [ 1, 1, 2, 3, 5, 8, 13]

# Scoping

- Globals
  - Declare using *global <variable>*
  - Can't declare and assign on same line
  - If assigning to global variable in method, then it becomes local
- Alternatively, pass variable as parameter to methods
- Recommendation - Put as much as possible in classes to avoid declaring global variables

```
1   global x
2   x = 42
3
4   def foo():
5       x = 3
6       print("foo x: " + str(x)) #x is local, 3
7
8       y = "42"
9       y = int(y) + 1
10      print("foo y: " + str(y))
11
12  def bar():
13      print("bar x: " + str(x)) #x is global, 42
14
15
16  foo()
17  bar()
18
19  print("x: " + str(x))
20
```

KNOX GAME DESIGN

# Dictionary

- Similar to a hash table/hash map in other languages
- No collisions as with hash tables
- Can loop over key/value pairs

```
#Check Yaku matches
strKey = 'kasu'
if (not strKey in self.scoreDict):
    if (iNormals >= 10):
        iPoints = 1 + (iNormals - 10)
        self.scoreDict[strKey] = iPoints
        self.hasNewScore = True
```

```
#Build total score and score text
self.iTotalPoints = 0
self.score_text = "Score\n"
for key in self.scoreDict:
    self.score_text += key + " " + str(self.scoreDict[key]) + "\n"
    self.iTotalPoints += self.scoreDict[key]

self.score_text += "Points " + str(self.iTotalPoints)
```

# Looping

- Loop over an array

```
vals = ["foo", "bar", "baz"]
for v in vals:
    print(v)
```

- Loop over an array with index

```
vals = ["foo", "bar", "baz"]
for idx, v in enumerate(vals):
    print(str(idx) + ": " + v)
```

- Loop over Dictionary

```
myDict = {'a' : 'alpha', 'b' : 'bravo', 'c' : 'charlie'}
for key in myDict:
        print('key: ' + key + ' value: ' + myDict[key])
```

# Installing Pygame

- cd to <python>\Scripts
- pip install pygame

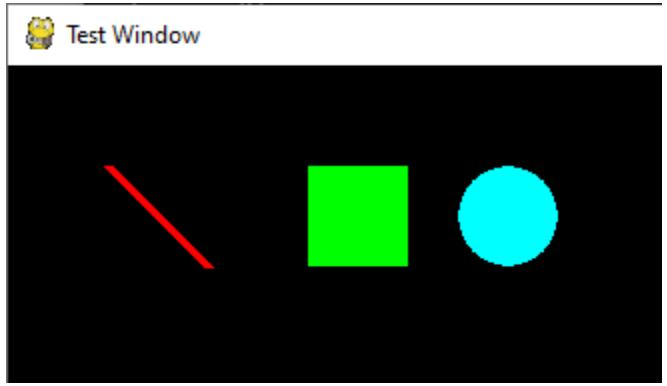
```
D:\Program Files\Python\Python38-32\Scripts>pip install pygame
Collecting pygame
  Downloading https://files.pythonhosted.org/packages/d2/ba/8e4f8fae51bd9d5766f1f20c9ce451e93929ee9
efdd2784b1a7b469ea76e/pygame-1.9.6-cp38-cp38-win32.whl (4.4MB)
     |                                | 4.4MB 939kB/s
Installing collected packages: pygame
Successfully installed pygame-1.9.6
```

KNOX GAME DESIGN

# Drawing to the Window

test_window.py

```
1  import pygame
2
3  pygame.init()
4  display = pygame.display.set_mode((1280, 720))
5  pygame.display.set_caption('Test Window')
6
7
8  clock = pygame.time.Clock()
9
10
11 keepLooping = True
12 while keepLooping:
13     for event in pygame.event.get():
14         if event.type == pygame.QUIT:
15             keepLooping = False
16
17     #Draw Line
18     c = (255, 0, 0)
19     pygame.draw.line(display, c, (50, 50), (100, 100), 5)
20
21     #Draw Rectangle
22     c = (0, 255, 0)
23     pygame.draw.rect(display, c, (150, 50, 50, 50))
24
25     #Draw Circle
26     c = (0, 255, 255)
27     pygame.draw.circle(display, c, (250, 75), 25)
28
29     pygame.display.update()
30     clock.tick(60)
31
32 pygame.quit()
33 quit()
```



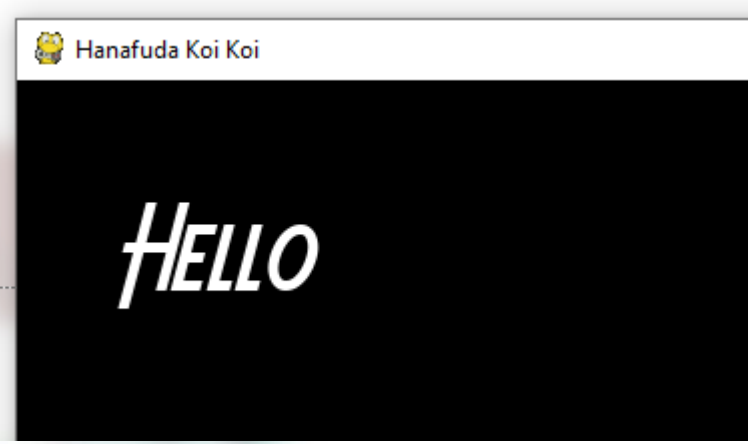

Note - transparency, must use a surface like in SDL

# Drawing Text

```
47    pygame.init()
48    display = pygame.display.set_mode(SCREEN_SIZE)
```

```
20
21     font = pygame.font.Font("Aerovias Brasil NF.ttf", 64)
22
```

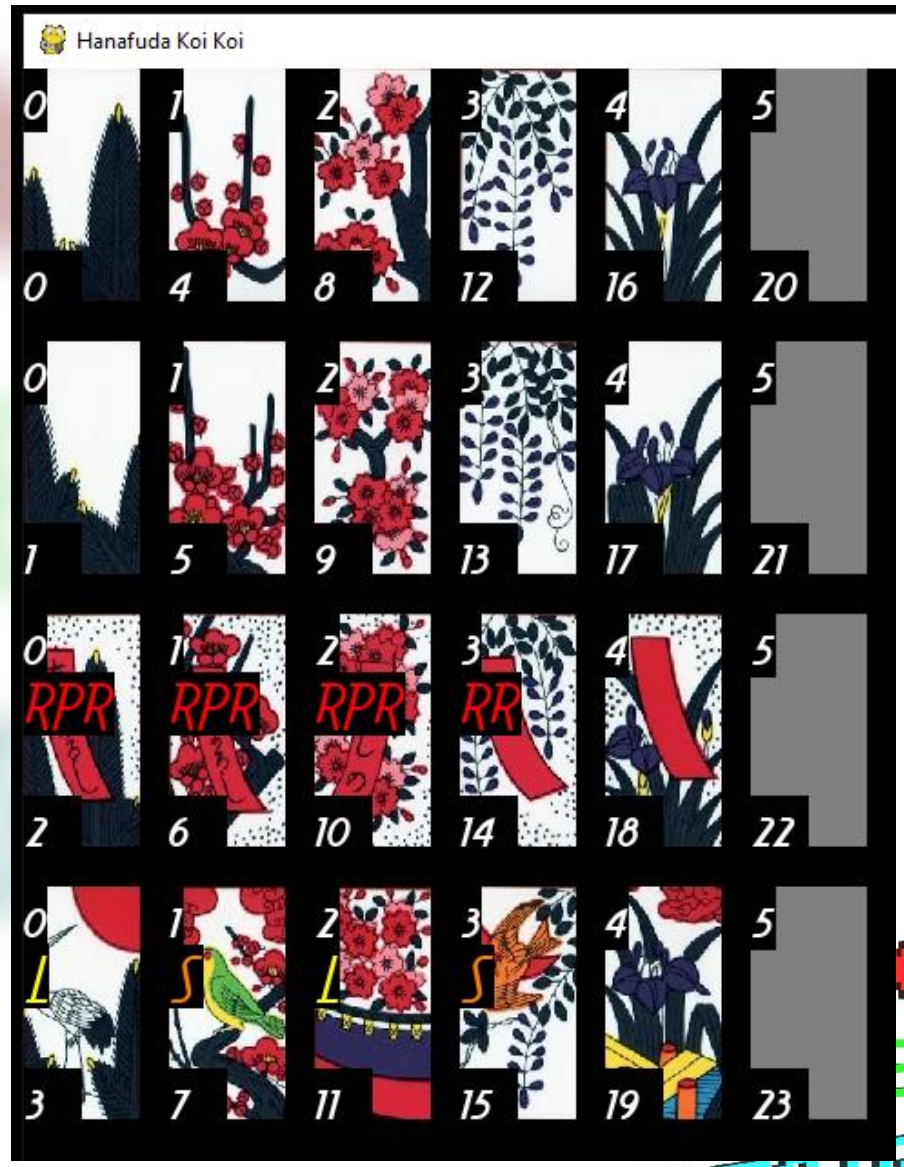
```
37
38    c = (255, 255, 255)
39    text = font.render('Hello', True, c)
40    display.blit(text, (50, 50))
41
```




KNOX GAME DESIGN

# Loading and Drawing Images

- Load
  - img = pygame.image.load('<image_file>.jpg')
- Display
  - display.blit(img, (x, y))
- Scale
  - img = pygame.transform.scale(img, (w, h))



# Game Loop

- Include pygame with *import pygame*
- Start pygame with *pygame.init()*
- Create loop that stops on *pygame.QUIT*
- On each loop, process all events using *pygame.event.get()*
- Recommend creating update and draw methods
- Call *pygame.display.update()* to update the display
- *clock.tick(60)* – set to run at 60 frames per second
- Use *pygame.quit()* on exit

```
import pygame


def main():
    pygame.init()
    display = pygame.display.set_mode((1280, 720))
    pygame.display.set_caption('Test Window')

    clock = pygame.time.Clock()

    keepLooping = True
    while keepLooping:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                keepLooping = False
            elif event.type == pygame.KEYDOWN:
                if (event.key == pygame.K_LEFT):
                    ship.vel_x = -5
                elif (event.key == pygame.K_RIGHT):
                    ship.vel_x = 5

        update()
        draw(display)
        pygame.display.update()
        clock.tick(60)

    pygame.quit()
    quit()

main()
```

# Handling Input

- Similar to SDL
- Check *event.type == pygame.KEYDOWN* for key press
- Use *event.key* to check which key was pressed
- Key constants are in the form *pygame.K_<key>*
  - Example *pygame.K_LEFT*, *pygame.K_a*
- Check *event.type == pygame.KEYUP* for key release

```
elif event.type == pygame.KEYDOWN:
    if (event.key == pygame.K_LEFT or event.key == pygame.K_a):
        ship.moveLeft()
    elif (event.key == pygame.K_RIGHT or event.key == pygame.K_d):
        ship.moveRight()
    elif (event.key == pygame.K_UP or event.key == pygame.K_w):
        ship.moveUp()
    elif (event.key == pygame.K_DOWN or event.key == pygame.K_s):
        ship.moveDown()
    elif (event.key == pygame.K_ESCAPE or event.key == pygame.K_q):
        keepLooping = False
elif event.type == pygame.KEYUP:
    if (event.key == pygame.K_LEFT or event.key == pygame.K_a):
        ship.stopMovingLeft()
    elif (event.key == pygame.K_RIGHT or event.key == pygame.K_d):
        ship.stopMovingRight()
    elif (event.key == pygame.K_UP or event.key == pygame.K_w):
        ship.stopMovingUp()
    elif (event.key == pygame.K_DOWN or event.key == pygame.K_s):
        ship.stopMovingDown()
```

KNOX GAME DESIGN

# Classes and Objects

- Use *class* followed by class name and colon
- Instantiate object with *<variable> = Ship()*
  - Don't use *new* keyword
- Define instance variable and methods inside of class
- Use *self.<variable>* to refer to instance variables
- Use *__init__(self, <param1>, <param2>,…)* to define "constructor"
- Delete objects with *del*
- Object variables and methods have public scope
- Use *self.* to create a list, otherwise you will get one list shared among all object instances

```
class Ship:
    x = 640
    y = 600
    w = 64
    h = 64

    vel_x = 0
    vel_y = 0

    def draw(self, display):
        #Draw Ship
        c = (0, 255, 255)
        pygame.draw.rect(display, c, (self.x, self.y, self.w, self.h))

    def update(self):
        self.x += self.vel_x
        self.y += self.vel_y

        if (self.x < 0):
            self.x = 0
        elif (self.x > 1280 - self.w):
            self.x = 1280 - self.w

        if (self.y < 0):
            self.y = 0
        elif (self.y > 720 - self.h):
            self.y = 720 - self.h

    def moveLeft(self):
        self.vel_x = -5

    def moveRight(self):
        self.vel_x = 5
```

KNOX GAME DESIGN

# Object Oriented

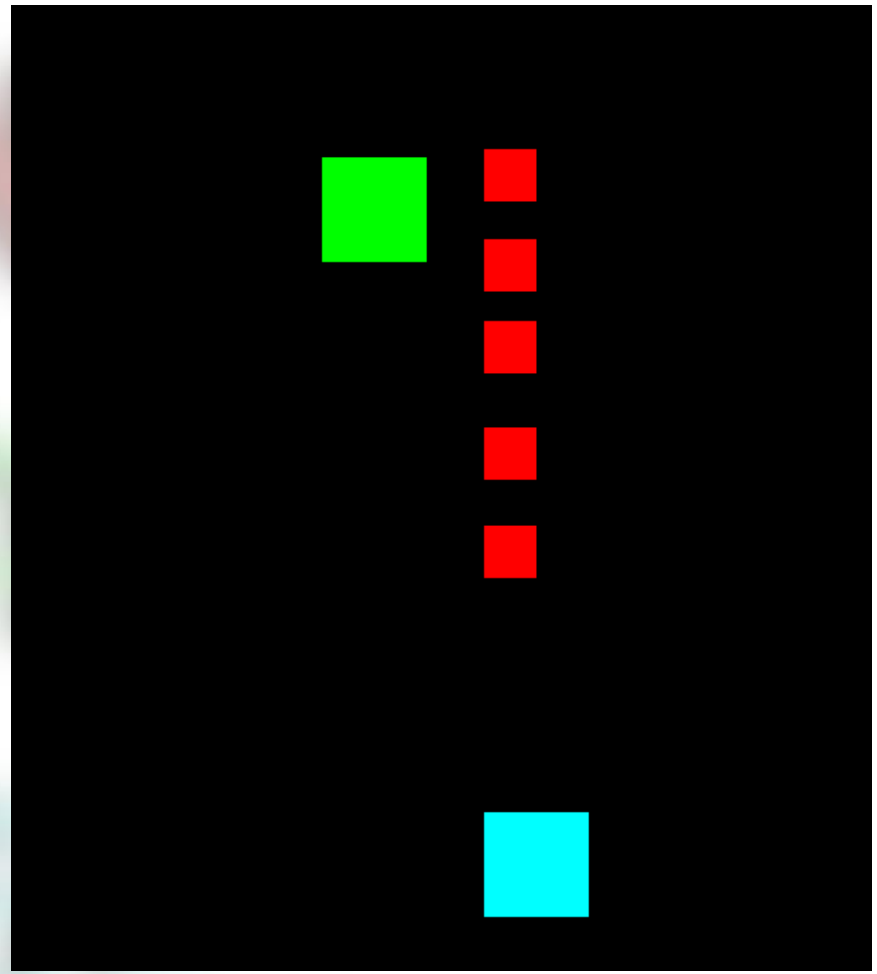
```
class <superclass>:
        def superclass_method(self):
                #code
class <subclass>(<superclass>):
        def subclass_method(self):
                self.superclass_method()
```

- Use *pass* to leave method implementation empty to be (optionally) implemented by subclass
- Superclass init isn't automatically called
  - super().__init__()

KNOX GAME DESIGN

# Lists/Arrays

- Create an empty list
  - *bullets = []*
- Append a new bullet to the list
  - *bullets.append(Bullet(64, 64))*
- Loop through list
  - *for bullet in bullets:*
    *<do something>*
- Remove array element
  - One - list.remove(object)
  - Call - list.clear()
- Check if item is in list
  - if  (item in list):



KNOX GAME DESIGN

# Music / Sound Effects

```
def load_audio(self):
    pygame.mixer.music.load('audio/PyKoiKoi_game.mp3')
    pygame.mixer.music.set_volume(0.5)
    self.sound_effects['card_drop'] = pygame.mixer.Sound('audio/card_drop.wav')
    self.sound_effects['next_player'] = pygame.mixer.Sound('audio/next_player.wav')
```

- Music
  - Use *play(-1)* for infinite loops
  - *SONG_END* event to do something when the music stops

```
pygame.mixer.music.play(-1)
```

- Sound Effects
  - Load into variable and call *play()*

```
self.gamemanager.sound_effects['card_drop'].play()

sound_effects['next_player'].play()
```

# A GUI Button Class

- PyGame does not include GUI components like Buttons, Input Fields, so you have to make your own
- Can assign a method to an instance variable to be executed on click
- Loop through all of the buttons from game manager, and check for mouse pressed (click) or mouse move (hover)
- Changing cursor
  - Best way is to set cursor to arrow before processing clickable items (buttons, cards)
  - Then  set cursor to diamond when an item is hovered
  - Otherwise, the cursor will be set back to arrow when the next clickable item is processed.  One cursor, many clickable items.
  - Better - keep track of the cursor state, and only set it if it changes state, to reduce cursor flicker

```
#2020 Levi D. Smith - levidsmith.com
import pygame

from drawhelper import DrawHelper

class Button:
    def __init__(self, str, x, y):
        self.strLabel = str
        self.x = x
        self.y = y
        self.w = 128
        self.h = 32
        self.action = None
        self.bkg_color_normal = (0, 0, 0)
        self.bkg_color_highlight = (0, 0, 255)
        self.bkg_color = self.bkg_color_normal
        self.isHidden = False


    def draw(self, display, font):
        if (not self.isHidden):
            pygame.draw.rect(display, self.bkg_color, [self.x, self
            .y, self.w, self.h])
            DrawHelper.drawTextShadow(self.strLabel, self.x, self.y
            , (255, 255, 255), display, font[1])

    def isClicked(self, x, y):
        if (not self.isHidden):
            if (x > self.x and x < self.x + self.w and y > self.y
            and y < self.y + self.h):
                return True
            else:
                return False

    def isHovered(self, x, y):
        if (not self.isHidden):
            if (x > self.x and x < self.x + self.w and y > self.y
            and y < self.y + self.h):
                self.bkg_color = self.bkg_color_highlight
                return True
            else:
                self.bkg_color = self.bkg_color_normal
                return False

    def hide(self):
        self.isHidden = True

    def show(self):
        self.isHidden = False
```

# Command Line Parameters

- import sys
- if ("-param" in sys.argv):
  - #do something

KNOX GAME DESIGN

# Networking

- import urllib.request
- urllib.request.urlopen(<url>).read()

```python
#2020 Levi D. Smith - levidsmith.com
import urllib.request
import hashlib

class LeaderboardManager:

    def __init__(self):
        self.iGameID = 6751
        f = open("leaderboard.key", "r")
        self.strKey = f.readline()
        f.close()

    def submitScore(self, strName, iScore):
        strToHash = strName + str(iScore) + str(self.iGameID) + self.strKey
        print("strToHash: " + strToHash)
        hash = hashlib.md5(strToHash.encode())
        print("MD5 sum of has string " + hash.hexdigest())
        urllib.request.urlopen("https://levidsmith.com/scores/AddScore.php?name=" + strName + "&game=" + str(self.iGameID) + "&score=" + str(iSc

    def getTopScores(self):
        print(urllib.request.urlopen("https://levidsmith.com/scores/TopScores.php?game=" + str(self.iGameID)).read())
```

# See also

- https://www.pygame.org/news.html
- https://www.w3schools.com/python/default.asp
- http://cs.roanoke.edu/Fall2013/CPSC120A/pygame-1.9.1-docs-html/ref/index.html
- https://nerdparadise.com/programming/pygame
- https://docs.python.org

KNOX GAME DESIGN