First Pygame Game

Making Game with Python (2)

Last Time

- Software Installation:
  - Install python 3.6.1 or greater
  - Install `pygame`
- Python primitives:
  - Scalar objects: int, float, bool, NoneType,
  - Type conversion
  - Operations: +, -, /, *, **
- Syntax:
  - variable
  - print
  - assignment

Today

- String object
- Input / Output
- Comparison operators
- Branching and conditionals
- Indentation
- Iteration and loops
- Pygame module
- Hello world pygame

String

- Letters, special characters, spaces, digits
- Enclose in quotation marks or single quotes
  - hi = ‘Hello there’
  - hi = “Hello there”
- Concatenate strings
  - name = ‘Andrew’
  - greeting = ‘Hello ’ + name ⇒ ‘Hello Andrew’
- Copy string
  - echo = ‘echo ‘
  - multi_echo = echo * 3 ⇒ ‘echoechoecho’

Input / Output

- Binds key input to a variable
  - text = input(‘Input a integer number’) ⇒ keyboard input: ‘Hi there’
  - print(text) ⇒ output: ‘Hi there’
- Input gives you a string so it must be converted to int or float if working with number
  - num = int(input(‘Type a integer number’)) ⇒ keyboard input: ‘5’
  - print(num) ⇒ num is integer 5

Comparison operators on int, float, string

- Comparisons below evaluate to a Boolean (True or False): x and y can be int, float, or string
- Example: x = 3; y=5
  - x == y ⇒ False
  - x != y ⇒ True
  - x > y ⇒ False
  - x >= y ⇒ False
  - x < y ⇒ True
  - x <= y ⇒ True

Logic Operators

- a and b are Boolean variables
  - not a → True if a is False; False if a is True
  - a and b → True if both are True
  - a or b → True if either or both are True

a	b	a and b	a or b
True	True	True	True
True	False	False	True
False	True	False	True
False	False	False	False

Control Flow - Branching

```
if <condition>:
    <expression>
    …
<expression>
```

```
if <condition>:
    <expression>
    ...
else:
    <expression>
    …
<expression>
```

```
if <condition>:
    <expression>
    <expression>
    …
elif <condition>::
    <expression>
    <expression>
    …
else:
    <expression>
    <expression>
    …
<expression>
```

- <condition> has a value True or False
- Evaluate expressions in that block if <condition> is True

Indentation

- Matters in python

```
x = float(input(‘Enter a number for x: ‘))
y = float(input(‘Enter a number for y: ‘))

if x == y:
        print(‘x and y are equal’)
elif x < y:
        print(‘x is smaller‘)
else:
        print (‘y is smaller’)
print(‘Thanks’)
```

Control Flow: while loops

```
while <condition>:
    <expression>
    <expression>
    …
<expression>
```

- <condition> evaluates to a Boolean
- If <condition> is True, do all steps inside while code block
- Check <condition> again
- Repeat until <condition> is False

while loop example

```
x = print(‘You’re in the Lost Forest. You need to choose: Go left or right‘)

while x == ‘right’:
        x = input(‘type: left or right?‘)

print(‘You got out of the Lost Forest’)
```

Control Flow: for loops

```
for n in range(5):
        print(n)
```

- range(5) generate a sequence of numbers
- Each time through the loop, <variable> takes a value
- First time, <variable> starts at the first value (0)
- Next time, <variable> get the second value (1)
- Continue until last element is read

range(start, stop, step)

- Default values are start = 0, step = 1
- Loop until value is stop - 1

```
For i in range(10):                    # equal to range(0, 10, 1)
    print(i)                           ⇒ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

for i in range(7, 10):
    print(i)                           ⇒ 7, 8, 9


for i in range(7, 10, 2):
    print(i)                           ⇒ 7, 9
```

break statement

- Immediately exits whatever loop it is in
- exits only innermost loop

```
while <condition>:
        while <condition>:
                <expression 1>
                break
                <expression 2>
        <expression 3>
<expression 4>
```

for vs while loops

for loops

- Know number of iterations
- Can end early via break
- Uses a counter
- Can rewrite a for loop using a while loop

```
# shortcut with for loop
for n in range(5):
        print(n)
```

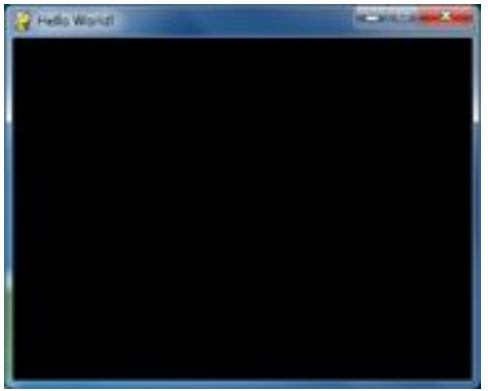
while loops

- Unbounded number of iteration
- Can end early via break
- Can use a counter but must initialize before loop and increment it inside loop
- May not be able to rewrite a while loop using a for loop

```
n = 0
while n < 5:
        print(n)
        n = n+1
```

Pygame Surface Object

- Surface objects represent rectangle 2D images
  - Display surface:
    pygame.display.set_mode((width, height))
  - pygame.display.set_caption(‘Hello World’)



Hello World Game

```
import pygame, sys
from pygame.locals import QUIT

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

while True:
    for event in pygame.event.get():  # event from keyboard or mouse
        if event.type == QUIT:
            pygame.quit()
            sys.exit()
    pygame.display.update()
```

import module vs from module import *

```
import pygame.locals

If event == pygame.locals.QUIT:
```

```
from pygame.locals import QUIT

If event == QUIT:
```

Game Loop

- Handles event:
  - Events from keyboard
  - Events from mouse
- Updates the game state
- Draws the game state to the screen

Pygame.event.Event objects

- Events:
  - keyboard
  - Mouse
- pygame.event.get():
  - List of event objects which happened since the last time the pygame.event.get() was called
  - Or list of the events which have happened since the start of the program if pygame.event.get() has never been called

Game termination

```
If event == QUIT:

        pygame.quit()

        sys.exit()
```