Lecture Notes

These lecture notes are brief and provide an outline for what was taught. This material was taught in three days.

Part 1: Printing to the screen

To print something in python 3.2, print (“”)

```
print(“Hello Class”)
print(“You guys are cool”)
print(“This will print to the screen”)
```

-What comments are

```
#
‘’’
’’’
```

Questions:

Call on someone to go to the board. and comment out some code.

Part 2: Variables

-Cup Model. Reassigning Variables

```
intVar = 1
intVar = 2
```

-Strings, Integer, Boolean, Floats, & lists.

```
variable name = initialvalue
name = “Omer”
intVar = 1
floatVar = 1.2
booleanVar = True
listVar = [123, “second thing”, True]
```

-Multiple Assignments

```
a = b = c = 1
```

-Printing Out Variables

```
age = 50
print(“what is inside of age is ” ,age )
```

print(“furst thing in listVar is ”, ListVar[0])

Questions

What will print(listVar[2]) do?
What is a boolean?
What is diff between int and float.
Show it on a number line

Part 3: Math

+ - / < > <= >=

-how it prints out the result of math mathematical operations

print ("I will now count my chickens:")
print ("Hens"**,25 + 30 / 6)**

explain the importance of ()

print ("Roosters"**, 100 - 25 * 3 / 4)**

make them ()
print ("Is it true that 3 + 2 < 5 - 7?")

print (3 + 2 < 5 - 7)
print ("Is it greater?"**, 5 > -2**)

firstname = poof

lastname = ball

print(“full name”, firstname + lastname)

Questions:

What is the order of operations?
Why does / round down?

Part 4: Reading input from the user

```
name = raw_input()
age = int(raw_input())
decimal = float(raw_input())
```

Questions:
input and raw input

Part 5 Student Exercises

Beginner: You will create python variables for name, age. prints out true if you're older than 12.

Intermediate : Create a program that gets meal cost with tip and tax

Work in groups

Part 6 They Do

Go to the variables exploration module

Part 7 Variables & Manipulation

Variables:

variable name = initialvalue
name = “Omer”
intVar = 1
floatVar = 1.2
booleanVar = True
listVar = [123, “second thing”, True]

Math:

On Ints and Floats
+=
-=

*=
/=
On Strings
+
On Booleans
and
or
not

Part 8 Boolean Logic

-Venn Diagrams: Not, And, Or

-Activity Result Of These. Make each line false (substitution if they get stuck)

True and True
False and True
True or False
(8>7) or (8<7)
not (8==7+1)

17 < 253, 100 == (2 * 50), 19 <=19, -22 >= -18, 99 != (98 +1)
booleanVar = (not (6>9)) or (7==3+4) and (7!=9000-8342)

If Statements:

variable name = initialvalue
if check:
    indented block
elif check:

elif check:

else:

```
cars = 100
trucks = 30
people = 20
if cars > people:
   print "We should take the cars."
elif cars < people:
```

```
    print "We should not take the cars."
else:
    print "We can't decide."
```

If Statement Activity

```
if 1+4/2 == 3:
    touch your toes

if (1+4)/2 == 3:
    touch your toes

if not 3>4:
    say poof is cool

a = 11
b = 2
if(a/b == 5):
    jump
```

Part 9 Loops

While Loops

```
while checkIsTrue:
    # code to run when code is true
    # indented in
    if check:
        this must also be indented

code to be run when while loop is done

while lives > 0:
    print("i am alive")
    fight()
    if lives == 1:
        break

print("lives is not greater than 0")
```

-Vending machine example with break statement

```
while true:
    guess = int(raw_input())
    if guess == 14:
        break
```

for loops are how we go through lists

```
fruits = ['banana', 'apple', 'orange', 'tomato', 'pear', 'grape']

print 'You have...'
for f in fruits:
    if f == 'tomato':
        print 'A tomato is not a fruit!' # (It actually is.)
    print 'A', f
    else:
    print 'A fine selection of fruits!'
```

Loop that runs once

```
loop_condition = True

while loop_condition:
    print "I am a loop"
    loop_condition = False'
```

Incrementing Through A Loop

```
num = 1

while num<11:  # Fill in the condition
    print num*num
    num +=1
    # Print num squared
    # Increment num (make sure to
do this!)
```

Simple Error:
A common application of a `while` loop is to check user input to see if it is valid. For example, if you ask the user to enter `y` or `n` and they instead enter `7`, then you should re-prompt them for input.

```
choice = raw_input('Enjoying the course? (y/n)')
while choice != 'y' and choice != 'n':  # Fill in the condition (before the colon)
    choice = raw_input("Sorry, I didn't catch that. Enter again: ")
```

Infinite Loops

```
count = 0

while count < 10: # Add a colon
    print count
    count+=1
    # Increment count
```

for loops in numbers

```
print "Counting..."

for i in range(20):
    print i
```

for loops in string

```
phrase = "A bird in the hand..."

# Add your for loop
for char in phrase:
    if char == 'A' or char == 'a':
        print 'X',
    else:
        print char,
```

-create program that takes someone age and tells them what grade they are in. do it in groups

-advanced students will do it with a while loop

Part 10 Object Oriented Programming

Functions

Code that is like a box
Takes in stuff and returns something
Can be reused. create functions for things that will be called twice

```
def checkLives(life):
    while lives > 0:
        attack random.randrange(0, 100)
        lives= fight(life,attack,100 )

def fight(life, damage, health):
    if health - damage < 0:
        return (life-1)
```

Classes

Classes are like blueprints for creating an object
You will need a class for each similar thing
Enemy Class (multiple enemies), Tree Class (multiple trees), Floor Class (multiple floors)
init method (self)

```
import pygame

class Player(pygame.sprite.Sprite): # parent object. they share similar genes

    def __init__(self):
        super().__init__() # Extends the pygame object
        self.image = pygame.image.load("").convert_alpha() # pixel format
        self.rect = self.image.get_rect(center=(starting x ,starting y))
    def update(self):
        # do stuff like switching costume and directions and bounds and collisions
```

Part 11 Pygame

Main Py Game

1) Import everything you need. Initialize pygame

```
import pygame
import random
from pygame.locals import *
from player import Player
```

2) Background
a) set the screen object
b) create the background (2 for scrolling)
c) draw the background image on the screen object
everything in pygame is rectangles.
you must create the rectangle and draw an image on it

```
screen = pygame.display.set_mode((640, 640))
background = pygame.image.load("").convert_alpha();
screen.blit(background, (0, 0))          # draws background at origin
```

3) Font if you want
4) Create enemies and players. add them to the groups make the draw methods maybe

```
poof = Player()
```

5) While True
A) Check for events
type and key
movement/action

```
while True:
        for event in pygame.event.get():
                if event.type == QUIT:
                        pygame.quit()
                        sys.exit()
        keys = pygame.key.get_pressed()
        if keys[K_w]:
                poof.rect.y += 5
        screen.blit(background, (0, 0))
        poof.update
        pygame.display.update()
```