

Introduction to Python 3

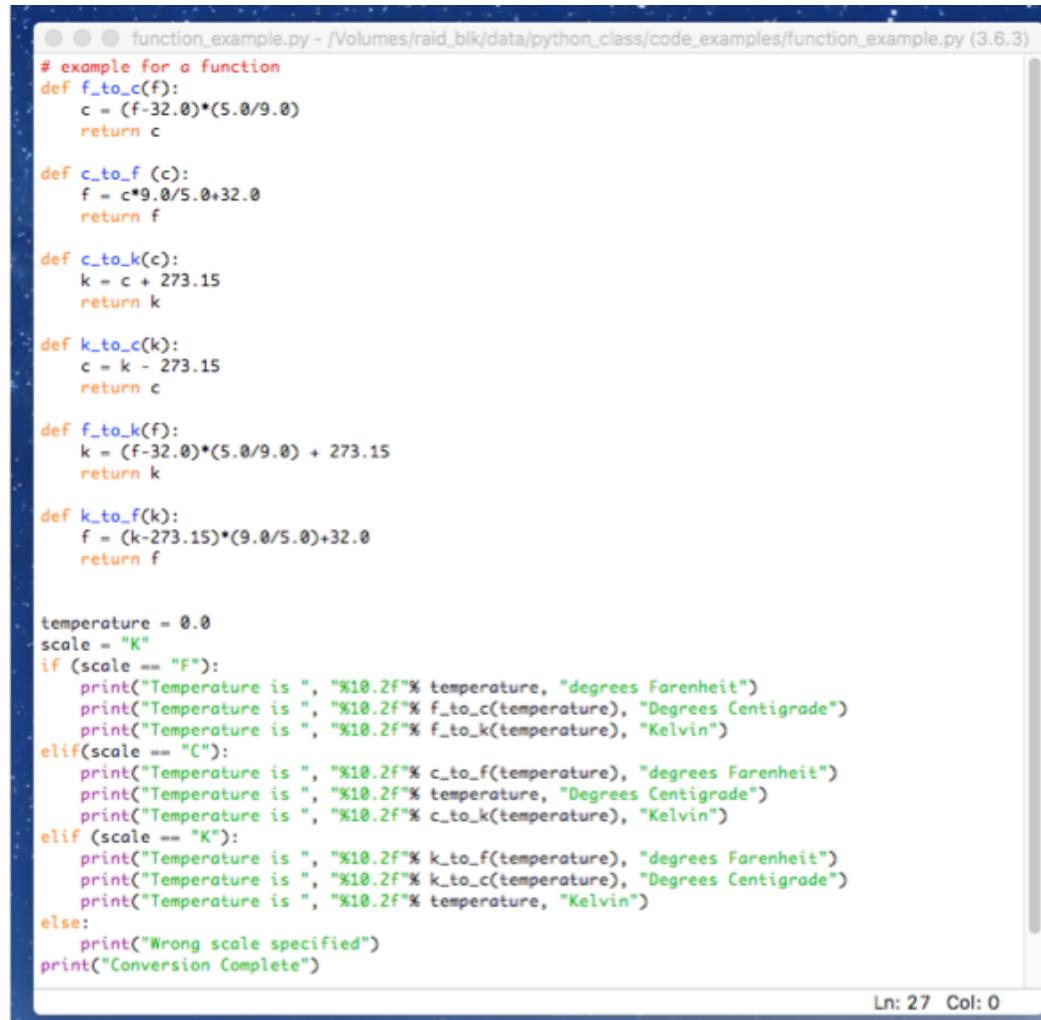
Week 2

Questions on last week?

Functions

- When writing programs, if we use the same code multiple times, it is more efficient to make it a function that has an arbitrary input and output
- For example, if we had a set of temperature reading every day for a year in degrees Fahrenheit and wanted to convert them to degrees Celsius, we could write a `f_to_c` function
 - This is clearly a trivial example

Example Function



```
# example for a function
def f_to_c(f):
    c = (f-32.0)*(5.0/9.0)
    return c

def c_to_f(c):
    f = c*9.0/5.0+32.0
    return f

def c_to_k(c):
    k = c + 273.15
    return k

def k_to_c(k):
    c = k - 273.15
    return c

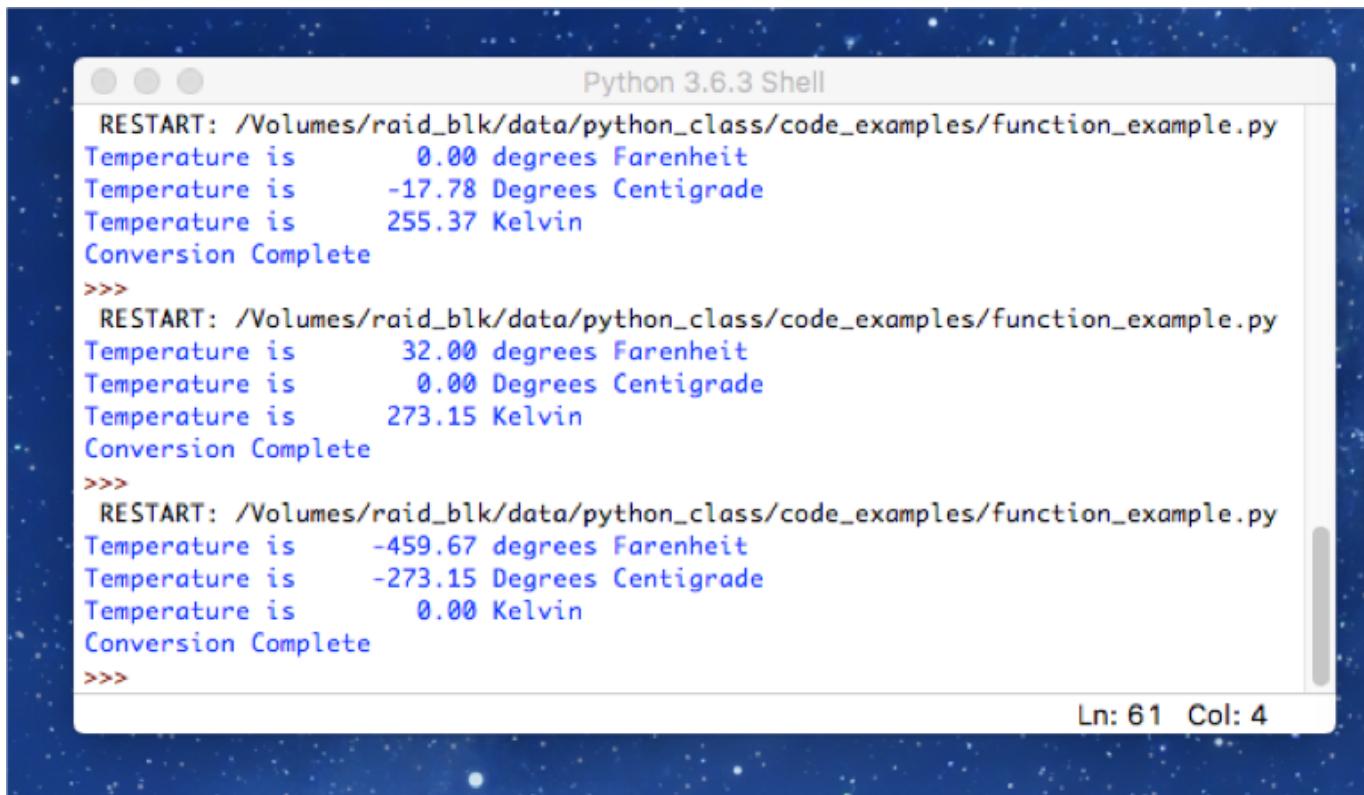
def f_to_k(f):
    k = (f-32.0)*(5.0/9.0) + 273.15
    return k

def k_to_f(k):
    f = (k-273.15)*(9.0/5.0)+32.0
    return f

temperature = 0.0
scale = "K"
if (scale == "F"):
    print("Temperature is ", "%10.2f"% temperature, "degrees Farenheit")
    print("Temperature is ", "%10.2f"% f_to_c(temperature), "Degrees Centigrade")
    print("Temperature is ", "%10.2f"% f_to_k(temperature), "Kelvin")
elif(scale == "C"):
    print("Temperature is ", "%10.2f"% c_to_f(temperature), "degrees Farenheit")
    print("Temperature is ", "%10.2f"% temperature, "Degrees Centigrade")
    print("Temperature is ", "%10.2f"% c_to_k(temperature), "Kelvin")
elif (scale == "K"):
    print("Temperature is ", "%10.2f"% k_to_f(temperature), "degrees Farenheit")
    print("Temperature is ", "%10.2f"% k_to_c(temperature), "Degrees Centigrade")
    print("Temperature is ", "%10.2f"% temperature, "Kelvin")
else:
    print("Wrong scale specified")
print("Conversion Complete")
```

Ln: 27 Col: 0

Output from example function



The screenshot shows a terminal window titled "Python 3.6.3 Shell". It displays three separate runs of a script named "function_example.py". Each run prints out the conversion of a temperature value between Fahrenheit, Centigrade, and Kelvin.

```
Python 3.6.3 Shell
RESTART: /Volumes/raid_blk/data/python_class/code_examples/function_example.py
Temperature is      0.00 degrees Farenheit
Temperature is     -17.78 Degrees Centigrade
Temperature is     255.37 Kelvin
Conversion Complete
>>>
RESTART: /Volumes/raid_blk/data/python_class/code_examples/function_example.py
Temperature is      32.00 degrees Farenheit
Temperature is      0.00 Degrees Centigrade
Temperature is     273.15 Kelvin
Conversion Complete
>>>
RESTART: /Volumes/raid_blk/data/python_class/code_examples/function_example.py
Temperature is     -459.67 degrees Farenheit
Temperature is     -273.15 Degrees Centigrade
Temperature is      0.00 Kelvin
Conversion Complete
>>>
```

Ln: 61 Col: 4

Variables are implicitly local unless declared as global

- Variable names are local to a program and a function
- The names of variables inside a function are not related to those outside the function
- When a function is called only the values are passed to the function
 - The names of the variables are not passed to the function
- Any variables used inside a function can not be referenced outside a function
 - The value of a variable (but not its name) can be passed in the return function
- Even though variables inside and outside a function are unrelated, it is recommended that variable names not be reused

Local Variable Example

The screenshot shows a Python development environment with two main panes. The top pane is a code editor titled "local_variable_example.py" with the file path "/Volumes/raid_blk/data/python_class/code_examples/local_variable_example.py (3.6.3)". It contains the following code:

```
# example for local variables
def x_sqrd(x):
    x = x*x
    print("x inside the function is", x)
    return x

x=25.0
print("x outside the function is", x)
y = x_sqrd(x)
print("y outside the function is", y)
print("x outside the function is", x)
```

The bottom pane is a "Python 3.6.3 Shell" window. It displays the Python interpreter's welcome message and a series of prompts and responses. The prompt "RESTART:" indicates the script was run again.

```
Python 3.6.3 (v3.6.3:2c5fed86e0, Oct  3 2017, 00:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: /Volumes/raid_blk/data/python_class/code_examples/local_variable_example.py
x outside the function is 25.0
x inside the function is 625.0
y outside the function is 625.0
x outside the function is 25.0
>>>
```

Global Variables

- A global variable is a variable known to all functions and can be used in any functions in the program
- It is not recommended to use global variables!
 - If a variable name is accidentally reused or changed inside a function it can make debugging code very hard
 - It is better to pass data using arguments in a function and arguments in a return
- There are times when global variables are needed
 - Could not find any examples in Python
 - Unrelated to Python--- DSP programming C; had to pass large amount of data to a function via a global to make the code run correctly
- We will discuss global variables in more detail in the last week

Global Example

The screenshot shows a Python development environment with two windows. The top window is a code editor for `global_variable_example.py`, and the bottom window is a Python shell.

Code Editor Content:

```
# example for local variables
global x

def x_sqrd():
    global x
    x = x*x
    print("x inside the function is", x)
    return x

x=25.0
print("x outside the function is", x)
y = x_sqrd()
print("y outside the function is", y)
print("x outside the function is", x)
```

Python Shell Output:

```
>>> RESTART: /Volumes/raid_blk/data/python_class/code_examples/global_variable_example.py
x outside the function is 25.0
x inside the function is 625.0
y outside the function is 625.0
x outside the function is 625.0
>>>
```

Libraries

- When there is a set of code that many people write and/or use, languages will be extended with functions that are then called libraries
- Libraries are called just like functions
- But they have to be included using the “import” statement before they are used.

Some common built-in libraries

- time
- math
- random
- os
- sys
- For more information on libraries see:
 - <https://docs.python.org/3/library/index.html>

Some Useful Libraries

- wxPython. A gui toolkit for python.
- NumPy. It provides some advance math functionalities to python.
- SciPy. It is a library of algorithms and mathematical tools for python
- matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.
- Pygame. This library will help you achieve your goal of 2d game development. It has other uses, also.
- Scapy. A packet sniffer and analyzer for python made in python.

Getting libraries from the internet

- If you know the name of the library you can often download it directly using the pip command in a terminal window
- For Python 3, the command is pip3
- It should have been installed with Python
- We will show it use in a few slides

Importing libraries

- We import libraries with the import command
- The import command should be at the start of the program
 - import library_name
- This will install all functions of the library
- If code size is important, only the functions that you want to use can be installed
 - from library_name import function_name1 function_name2 ...

Time Library

- The time library has many functions
- See <https://docs.python.org/3/library/time.html>
- Some of the useful functions are:
 - `time.sleep(time_in_seconds)`
 - `time.localtime()`
 - `time.time()`
 - On Windows and Unix Systems, return the time in seconds January 1, 1970, 00:00:00 (UTC) with leap seconds are not counted towards the time in seconds. This is commonly referred to as Unix time.
 - `time.timezone()`
 - `time.daylight()`

Example use of time library

The image shows a Mac OS X desktop environment. At the top, there is a menu bar with standard Apple menu items like 'File', 'Edit', etc. Below the menu bar, there are two windows:

- Code Editor Window:** The title bar says 'libraries_1.py - /Volumes/raid_blk/data/python_class/code_examples/libraries_1.py (3.6.3)'. The code inside the window is as follows:

```
import time          # import entire time library
from time import localtime      # import only localtime from time library
from time import sleep        # import only sleep from time library
t = time.localtime()        # local time is a structure[year, month,
                           # day, hour, min, sec, day of week, day of year, dst on off]
print(t[3],":",t[4],":",t[5])    # hour minutes, seconds
time.sleep(2)            # sleep for 2 seconds
t1=localtime()
print("%d"%t1[3],":",t1[4],":",t1[5])
sleep(5)
t1=localtime()
print("%d"%t1[3],":",t1[4],":",t1[5])
```

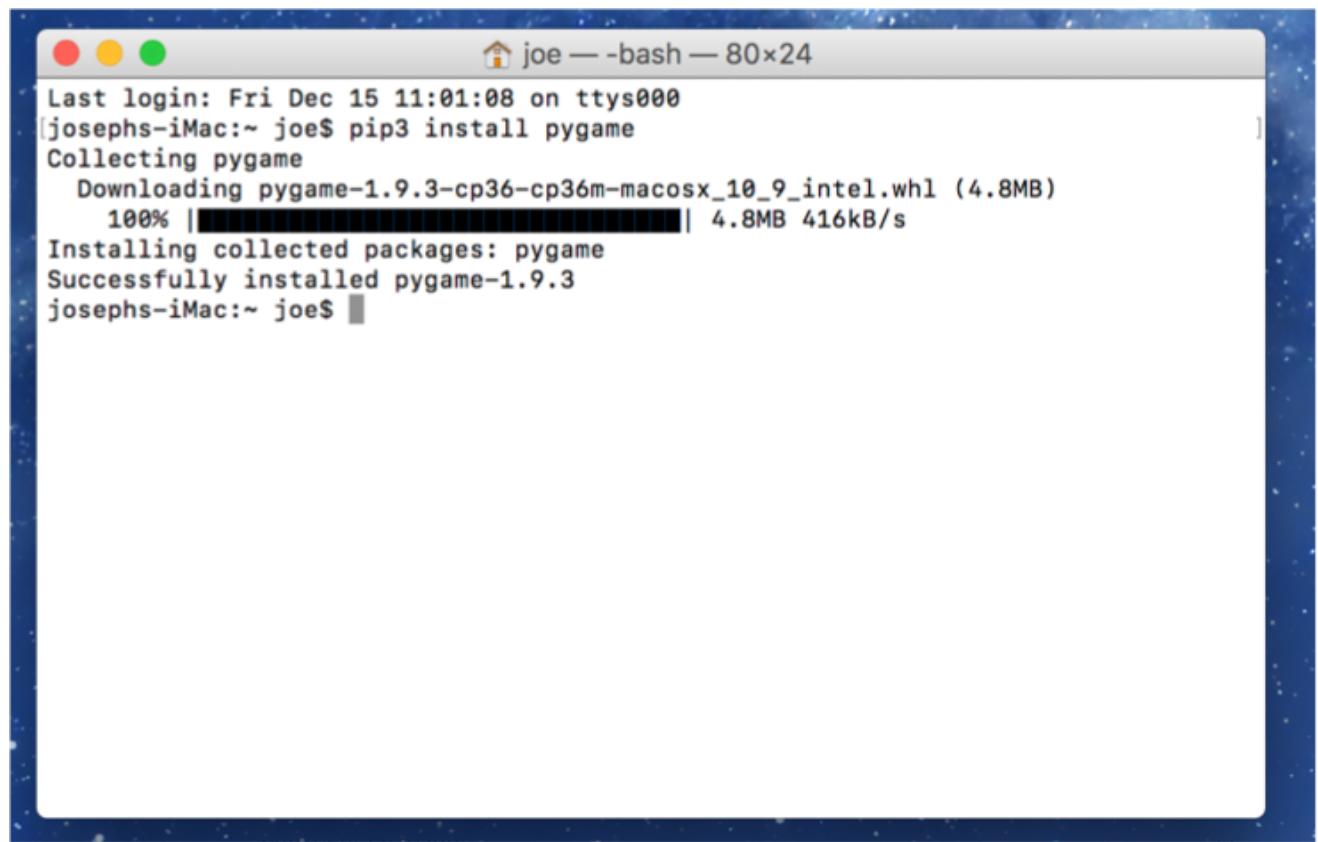
Ln: 7 Col: 53
- Terminal Window:** The title bar says 'Python 3.6.3 Shell'. The window displays the Python interpreter output:

```
Python 3.6.3 (v3.6.3:2c5fed86e0, Oct  3 2017, 00:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: /Volumes/raid_blk/data/python_class/code_examples/libraries_1.py =
16 : 50 : 12
16 : 50 : 14
16 : 50 : 19
>>>
```

Ln: 9 Col: 4

pygame library

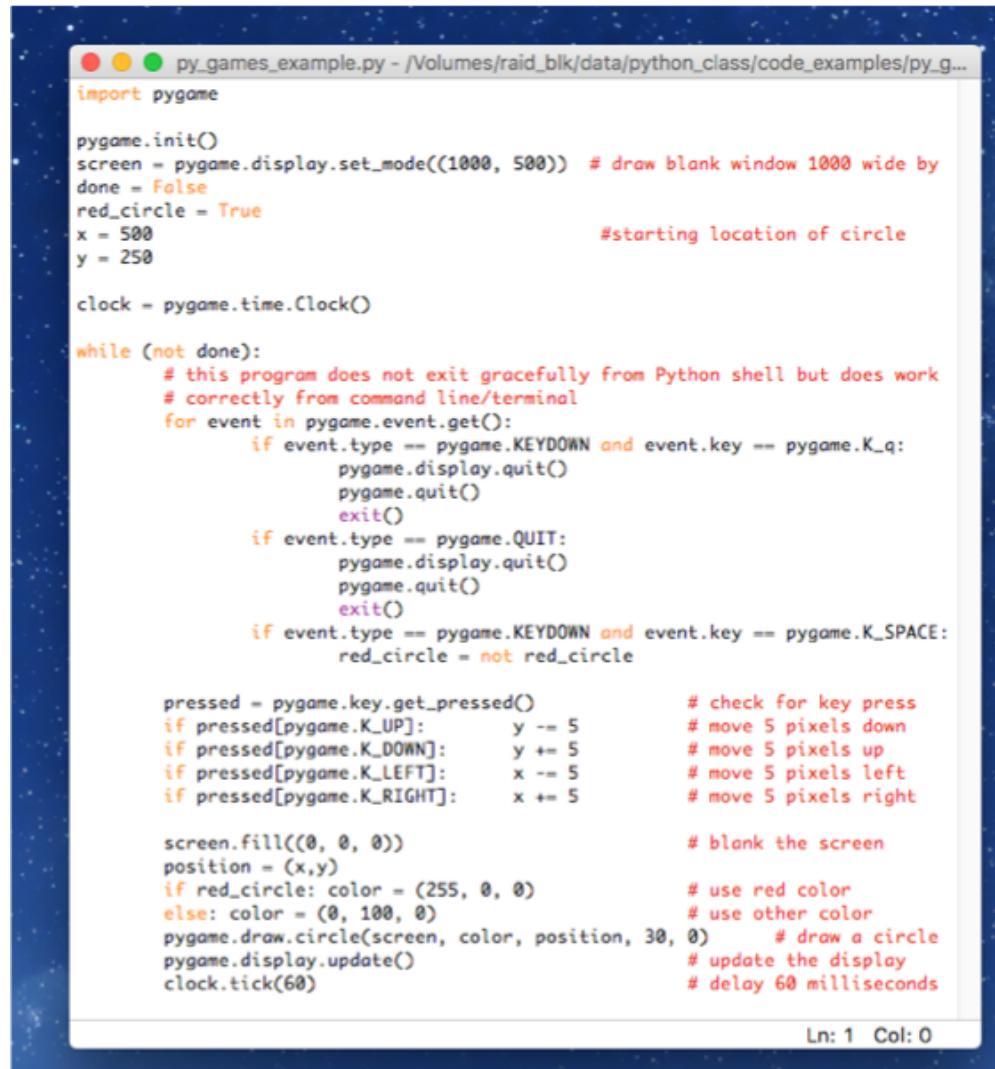
- Install pygame from terminal window



A screenshot of a Mac OS X terminal window titled "joe — -bash — 80x24". The window shows the command "pip3 install pygame" being run. The output indicates that pygame version 1.9.3 is being downloaded from a macOS intel wheel file (4.8MB) at 416kB/s, and it is successfully installed.

```
Last login: Fri Dec 15 11:01:08 on ttys000
[josephs-iMac:~ joe$ pip3 install pygame
Collecting pygame
  Downloading pygame-1.9.3-cp36-cp36m-macosx_10_9_intel.whl (4.8MB)
    100% |██████████| 4.8MB 416kB/s
Installing collected packages: pygame
Successfully installed pygame-1.9.3
josephs-iMac:~ joe$ ]
```

Example pygame Program



A screenshot of a terminal window titled "py_games_example.py - /Volumes/raid_blk/data/python_class/code_examples/py_g...". The window contains Python code for a simple pygame program. The code initializes pygame, sets up a window, and handles user input for moving a red circle. It includes comments explaining the logic for exiting the program and handling key presses.

```
import pygame

pygame.init()
screen = pygame.display.set_mode((1000, 500)) # draw blank window 1000 wide by
done = False
red_circle = True
x = 500                                     #starting location of circle
y = 250

clock = pygame.time.Clock()

while (not done):
    # this program does not exit gracefully from Python shell but does work
    # correctly from command line/terminal
    for event in pygame.event.get():
        if event.type == pygame.KEYDOWN and event.key == pygame.K_q:
            pygame.display.quit()
            pygame.quit()
            exit()
        if event.type == pygame.QUIT:
            pygame.display.quit()
            pygame.quit()
            exit()
        if event.type == pygame.KEYDOWN and event.key == pygame.K_SPACE:
            red_circle = not red_circle

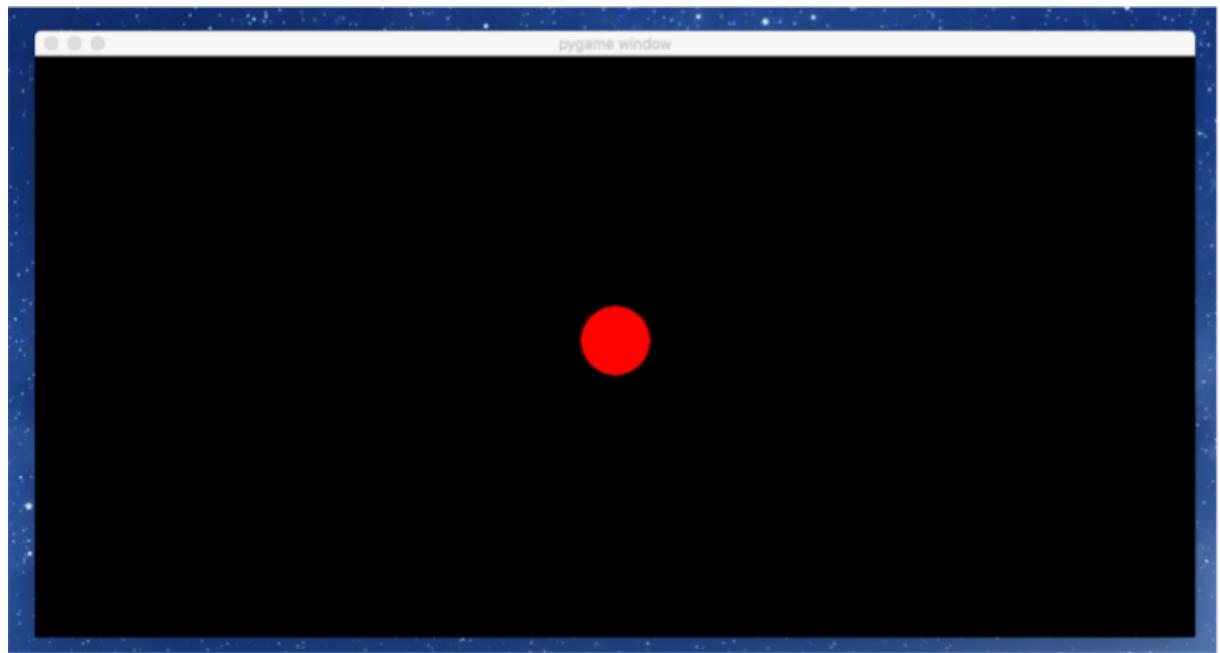
    pressed = pygame.key.get_pressed()           # check for key press
    if pressed[pygame.K_UP]:                     y -= 5      # move 5 pixels down
    if pressed[pygame.K_DOWN]:                   y += 5      # move 5 pixels up
    if pressed[pygame.K_LEFT]:                   x -= 5      # move 5 pixels left
    if pressed[pygame.K_RIGHT]:                  x += 5      # move 5 pixels right

    screen.fill((0, 0, 0))                      # blank the screen
    position = (x,y)
    if red_circle: color = (255, 0, 0)          # use red color
    else: color = (0, 100, 0)                    # use other color
    pygame.draw.circle(screen, color, position, 30, 0) # draw a circle
    pygame.display.update()                      # update the display
    clock.tick(60)                             # delay 60 milliseconds
```

Ln: 1 Col: 0

Program Running

- Circle can
be moved with cursor



Printing

- Printing for Python 3 is different from Python 2
- We will use the Python 3 formatting conventions
- Printing is necessary to get output/results from your program
- We saw examples of printing in previous examples
- These examples will be more detailed
- We can print text and variables on single and multiple lines

Formatting of Variables when printing

- %c character
- %s string conversion via str() prior to formatting
- %i signed decimal integer
- %d signed decimal integer
- %u unsigned decimal integer
- %o octal integer
- %x hexadecimal integer (lowercase letters)
- %X hexadecimal integer (UPPERcase letters)
- %e exponential notation (with lowercase 'e')
- %E exponential notation (with UPPERcase 'E')
- %f floating point real number
- %g the shorter of %f and %e
- %G the shorter of %f and %E

Printing examples

- `print ("This is a line of text")`
- `print ("This is a second line of text")`
- `print ("This is a line of text\nThis is a second line of text")`
- The `\n` tells the program to start a new line
- A `\` before a special command will print the command
- `print("\n")` will print `\n`
- `print("\n")` will move to the next line for the next print statement

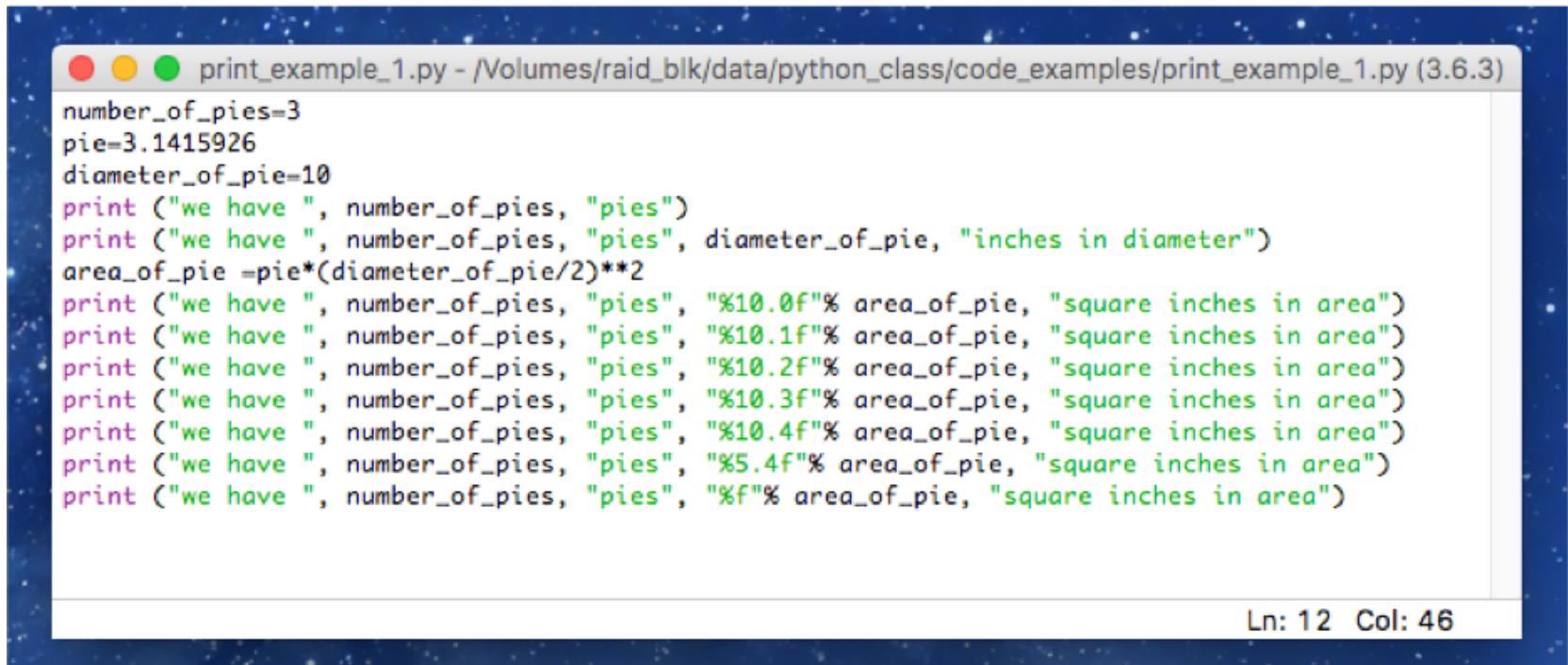
Some Common Formatting Expressions

- \n new line
- \' '
- \" "
- \f form feed
- \r carriage return (CR) move to beginning of same line
- \t tab

Formatting variables for Printing

- Print decimal variable with **Z** spaces and **y** digits after the decimal point
 - `%Z.yd%"%`
- Print floating variable with **Z** spaces and **y** digits after the decimal point
 - `%Z.yf%"%`
- Print octal variable with **Z** spaces and **y** digits after the octal point
 - `%Z.yo%"%`
- Print binary variable with **Z** spaces and **y** digits after the binary point
 - `%Z.yb%"%`
- Print hex variable with **Z** spaces and **y** digits after the hex point
 - `%Z.yX%"%`
- **Note: use of red Z, blue y, and bold type is for clarity and not a part of the formatting**

Print_example_1.py

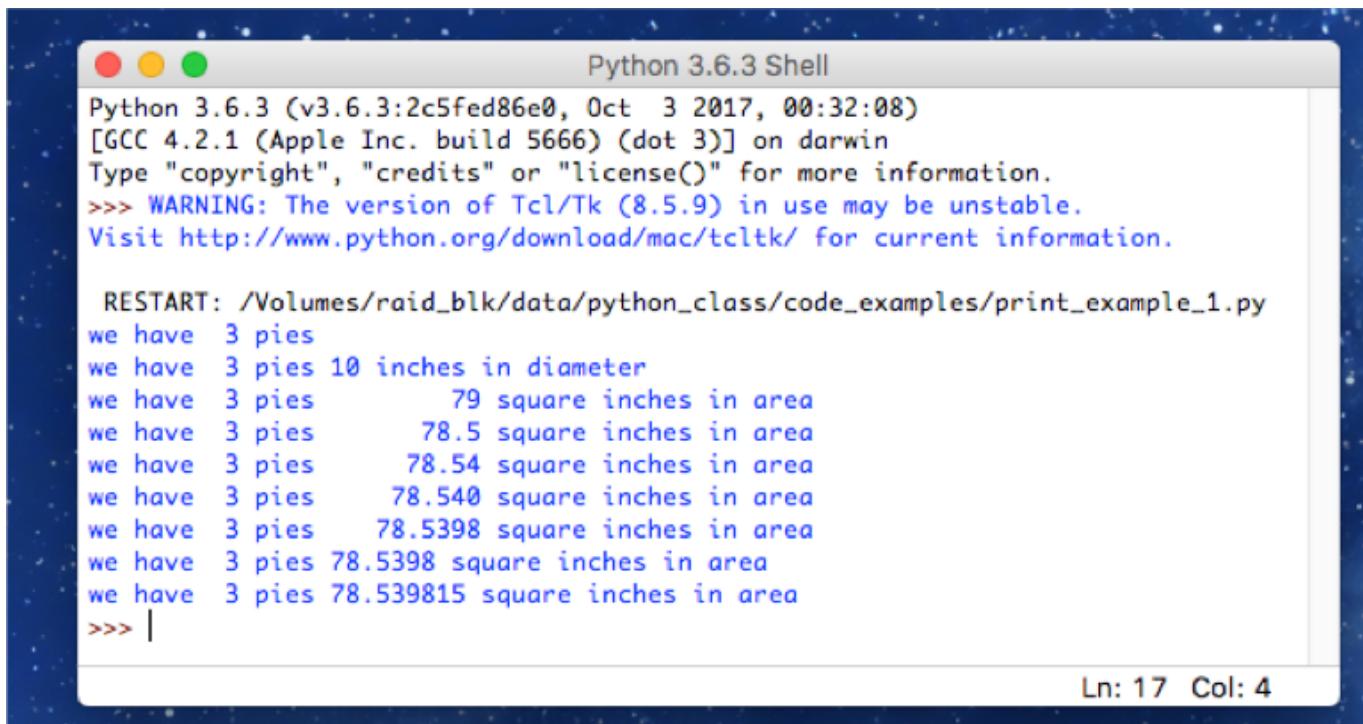


A screenshot of a terminal window titled "print_example_1.py - /Volumes/raid_blk/data/python_class/code_examples/print_example_1.py (3.6.3)". The window contains the following Python code:

```
number_of_pies=3
pie=3.1415926
diameter_of_pie=10
print ("we have ", number_of_pies, "pies")
print ("we have ", number_of_pies, "pies", diameter_of_pie, "inches in diameter")
area_of_pie =pie*(diameter_of_pie/2)**2
print ("we have ", number_of_pies, "pies", "%10.0F"% area_of_pie, "square inches in area")
print ("we have ", number_of_pies, "pies", "%10.1F"% area_of_pie, "square inches in area")
print ("we have ", number_of_pies, "pies", "%10.2F"% area_of_pie, "square inches in area")
print ("we have ", number_of_pies, "pies", "%10.3F"% area_of_pie, "square inches in area")
print ("we have ", number_of_pies, "pies", "%10.4F"% area_of_pie, "square inches in area")
print ("we have ", number_of_pies, "pies", "%5.4F"% area_of_pie, "square inches in area")
print ("we have ", number_of_pies, "pies", "%F"% area_of_pie, "square inches in area")
```

The status bar at the bottom right of the terminal window shows "Ln: 12 Col: 46".

Print_example_1.py



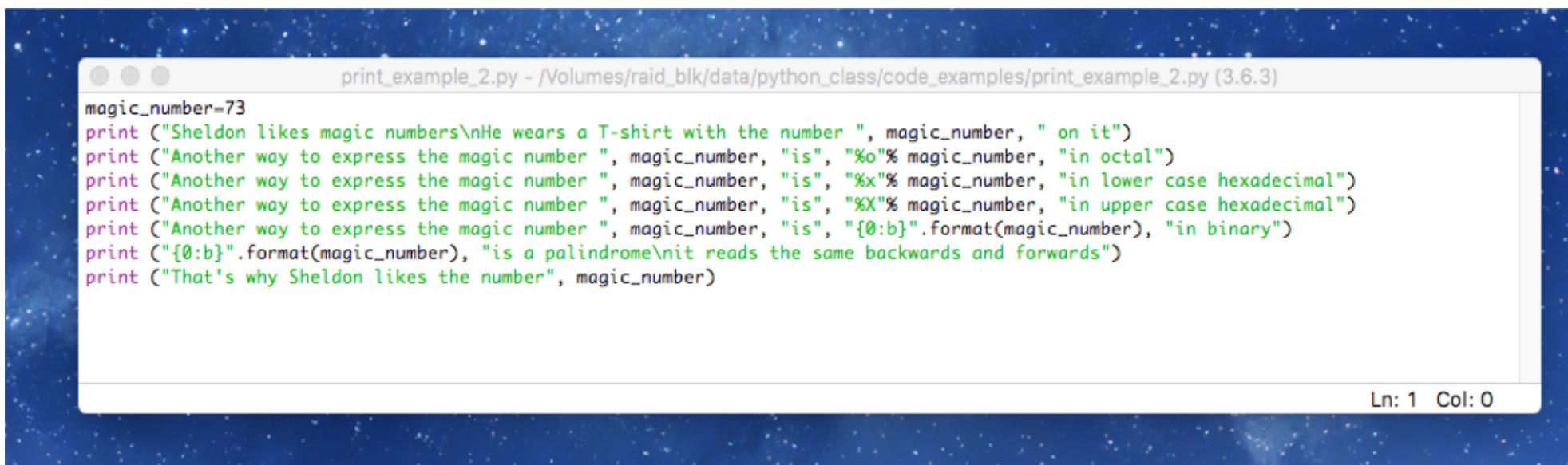
The screenshot shows a Mac OS X window titled "Python 3.6.3 Shell". The window contains the following text:

```
Python 3.6.3 (v3.6.3:2c5fed86e0, Oct  3 2017, 00:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.

RESTART: /Volumes/raid_blk/data/python_class/code_examples/print_example_1.py
we have 3 pies
we have 3 pies 10 inches in diameter
we have 3 pies      79 square inches in area
we have 3 pies      78.5 square inches in area
we have 3 pies      78.54 square inches in area
we have 3 pies      78.540 square inches in area
we have 3 pies      78.5398 square inches in area
we have 3 pies 78.5398 square inches in area
we have 3 pies 78.539815 square inches in area
>>> |
```

At the bottom right of the window, it says "Ln: 17 Col: 4".

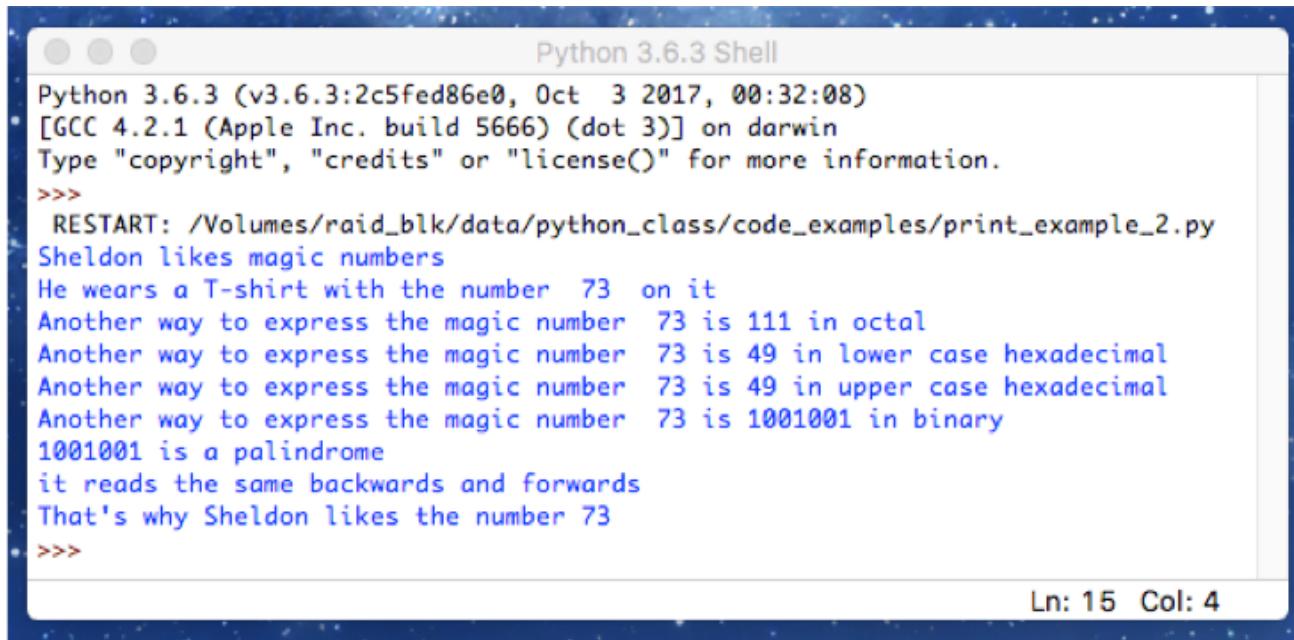
print_example_2.py



```
print_example_2.py - /Volumes/raid_blk/data/python_class/code_examples/print_example_2.py (3.6.3)
magic_number=73
print ("Sheldon likes magic numbers\nHe wears a T-shirt with the number ", magic_number, " on it")
print ("Another way to express the magic number ", magic_number, "is", "%o"% magic_number, "in octal")
print ("Another way to express the magic number ", magic_number, "is", "%x"% magic_number, "in lower case hexadecimal")
print ("Another way to express the magic number ", magic_number, "is", "%X"% magic_number, "in upper case hexadecimal")
print ("Another way to express the magic number ", magic_number, "is", "{0:b}".format(magic_number), "in binary")
print ("{0:b}".format(magic_number), "is a palindrome\nit reads the same backwards and forwards")
print ("That's why Sheldon likes the number", magic_number)

Ln: 1 Col: 0
```

print_example_2.py



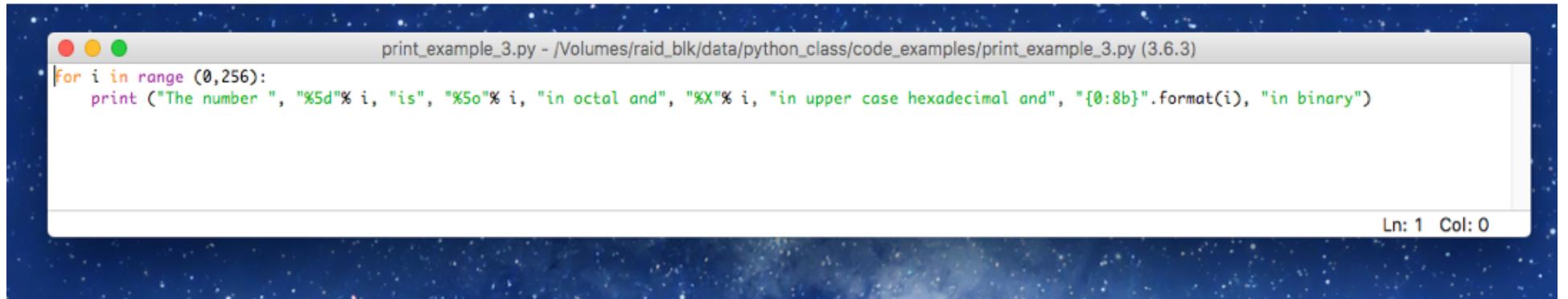
Python 3.6.3 Shell

```
Python 3.6.3 (v3.6.3:2c5fed86e0, Oct  3 2017, 00:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.

>>>
RESTART: /Volumes/raid_blk/data/python_class/code_examples/print_example_2.py
Sheldon likes magic numbers
He wears a T-shirt with the number 73 on it
Another way to express the magic number 73 is 111 in octal
Another way to express the magic number 73 is 49 in lower case hexadecimal
Another way to express the magic number 73 is 49 in upper case hexadecimal
Another way to express the magic number 73 is 1001001 in binary
1001001 is a palindrome
it reads the same backwards and forwards
That's why Sheldon likes the number 73
>>>
```

Ln: 15 Col: 4

example_3.py



A screenshot of a terminal window titled "print_example_3.py - /Volumes/raid_blk/data/python_class/code_examples/print_example_3.py (3.6.3)". The window contains the following Python code:

```
For i in range (0,256):
    print ("The number ", "%5d"% i, "is", "%5o"% i, "in octal and", "%X"% i, "in upper case hexadecimal and", "{0:8b}".format(i), "in binary")
```

The status bar at the bottom right shows "Ln: 1 Col: 0".

print example 3 nv

The image shows two overlapping Python 3.6.3 Shell windows. The top window displays the output of the script `print_example_3.py`, which prints numbers from 0 to 7 and their octal, uppercase hexadecimal, and binary representations. The bottom window shows the same script running again, this time with numbers from 244 to 255.

```
>>>
RESTART: /Volumes/raid_blk/data/python_class/code_examples/print_example_3.py
The number    0 is      0 in octal and 0 in upper case hexadecimal and      0 in binary
The number    1 is      1 in octal and 1 in upper case hexadecimal and      1 in binary
The number    2 is      2 in octal and 2 in upper case hexadecimal and     10 in binary
The number    3 is      3 in octal and 3 in upper case hexadecimal and     11 in binary
The number    4 is      4 in octal and 4 in upper case hexadecimal and    100 in binary
The number    5 is      5 in octal and 5 in upper case hexadecimal and    101 in binary
The number    6 is      6 in octal and 6 in upper case hexadecimal and    110 in binary
The number    7 is      7 in octal and 7 in upper case hexadecimal and    111 in binary
Ln: 262 Col: 4

The number 244 is 364 in octal and F4 in upper case hexadecimal and 11110100 in binary
The number 245 is 365 in octal and F5 in upper case hexadecimal and 11110101 in binary
The number 246 is 366 in octal and F6 in upper case hexadecimal and 11110110 in binary
The number 247 is 367 in octal and F7 in upper case hexadecimal and 11110111 in binary
The number 248 is 370 in octal and F8 in upper case hexadecimal and 11111000 in binary
The number 249 is 371 in octal and F9 in upper case hexadecimal and 11111001 in binary
The number 250 is 372 in octal and FA in upper case hexadecimal and 11111010 in binary
The number 251 is 373 in octal and FB in upper case hexadecimal and 11111011 in binary
The number 252 is 374 in octal and FC in upper case hexadecimal and 11111100 in binary
The number 253 is 375 in octal and FD in upper case hexadecimal and 11111101 in binary
The number 254 is 376 in octal and FE in upper case hexadecimal and 11111110 in binary
The number 255 is 377 in octal and FF in upper case hexadecimal and 11111111 in binary
>>> |                                         Ln: 262 Col: 4
```

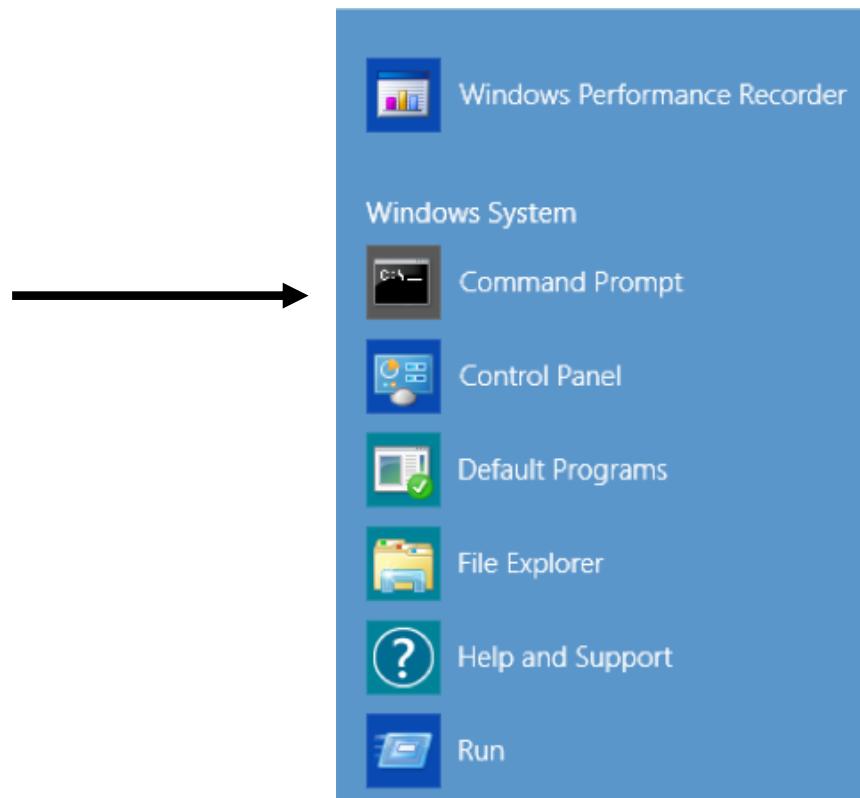
Calls to the OS

- Sometimes the program needs information from the operating system
- The os statement will execute a command to the os and can read the output of the commands to the os
- The commands that can be executed are those that can be run from:
 - Terminal window on OS X
 - Terminal window on Linux
 - Command Window on Windows
- The commands for each OS are different so the resulting code will be OS dependent
 - It is possible to fingerprint the OS and add code specific for each OS
- Example:
 - ifconfig in Linux and OS X gets the state of the network ports
 - ipconfig in Windows gets the state of the network ports

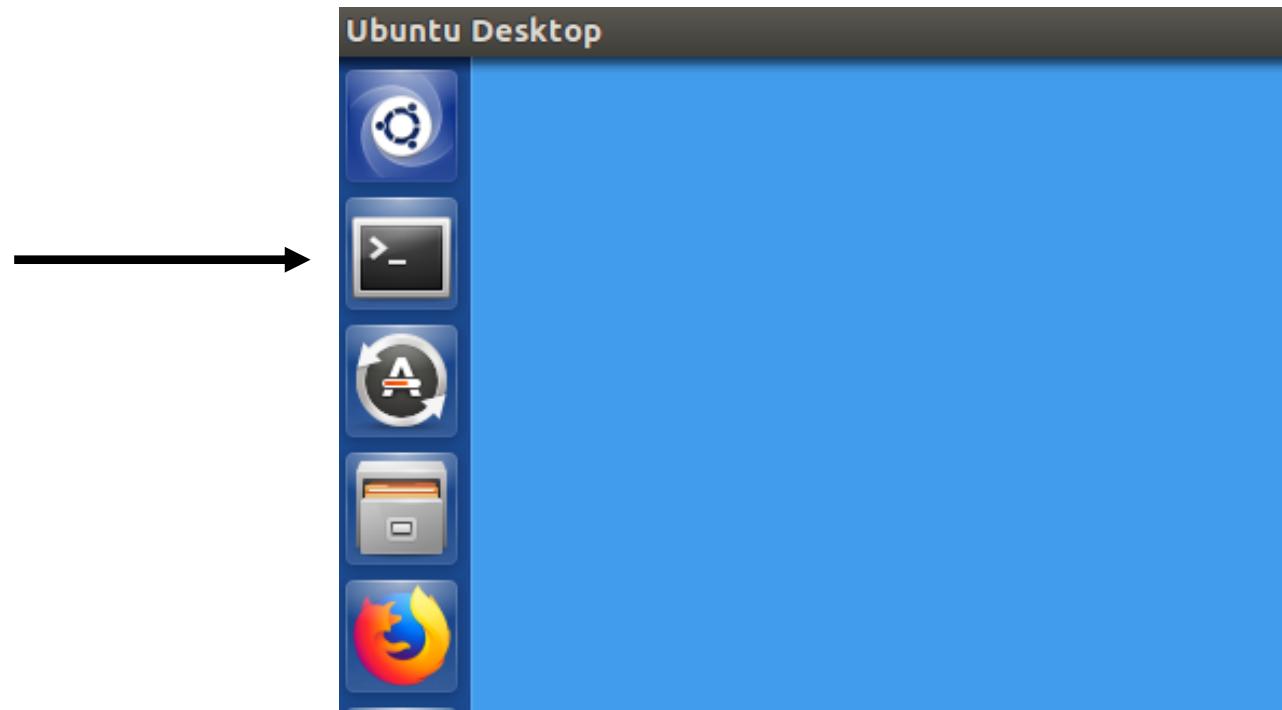
Terminal window in OS X



Command Prompt in Windows



Terminal Window in Linux



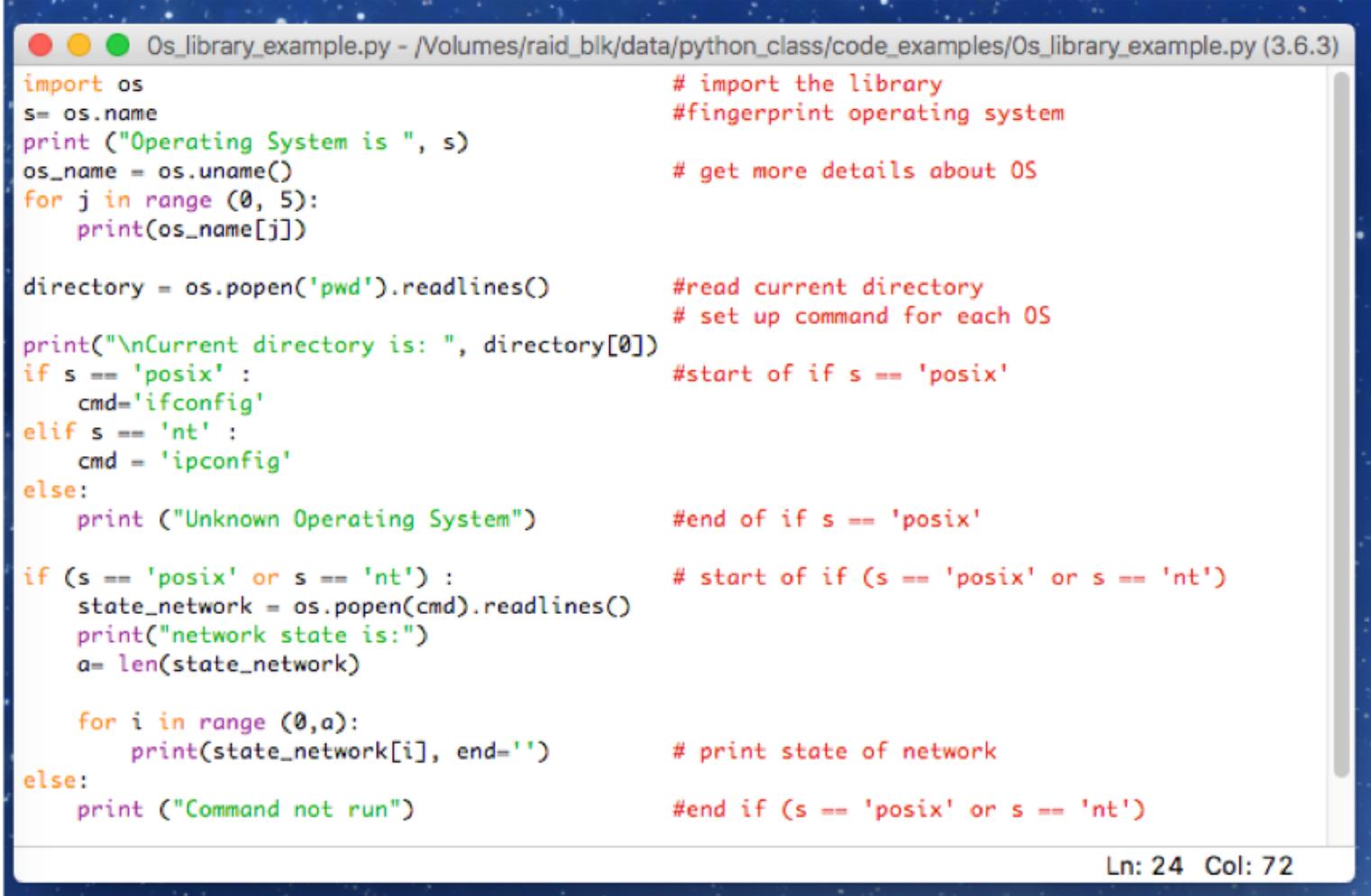
Some OS Calls

- `os.system()` Executing a shell command
- `os.environ()` Get the users environment
- `os.getcwd()` Returns the current working directory.
- `os.getgid()` Return the real group id of the current process.
- `os.getuid()` Return the current process's user id.
- `os.getpid()` Returns the real process ID of the current process.
- `os.umask(mask)` Set the current numeric umask and return the previous umask.
- `os.uname()` Return information identifying the current operating system.

Some OS Calls (Con't)

- `os.chroot(path)` Change the root directory of the current process to path
- `os.listdir(path)` Return a list of the entries in the directory given by path.
- `os.mkdir(path)` Create a directory named path
- `os.makedirs(path)` Recursive directory creation function.
- `os.remove(path)` Remove (delete) the file path.
- `os.removedirs(path)` Remove directories recursively.
- `os.rename(src, dst)` Rename the file or directory src to dst.
- `os.rmdir(path)` Remove (delete) the directory path.
- `os.popen('command').readlines()` Execute 'command'; read the results

Example use of os library



```
Os_library_example.py - /Volumes/raid_blk/data/python_class/code_examples/Os_library_example.py (3.6.3)
import os                                     # import the library
s= os.name                                    #fingerprint operating system
print ("Operating System is ", s)
os_name = os.uname()
for j in range (0, 5):
    print(os_name[j])

directory = os.popen('pwd').readlines()          #read current directory
                                                # set up command for each OS
print("\nCurrent directory is: ", directory[0])
if s == 'posix' :                               #start of if s == 'posix'
    cmd='ifconfig'
elif s == 'nt' :
    cmd = 'ipconfig'
else:
    print ("Unknown Operating System")          #end of if s == 'posix'

if (s == 'posix' or s == 'nt') :
    state_network = os.popen(cmd).readlines()
    print("network state is:")
    a= len(state_network)

    for i in range (0,a):
        print(state_network[i], end='')         # print state of network
else:
    print ("Command not run")                  #end if (s == 'posix' or s == 'nt')

Ln: 24 Col: 72
```

Output of example

```
Python 3.6.3 Shell
Python 3.6.3 (v3.6.3:2c5fed86e0, Oct  3 2017, 00:32:08)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: /Volumes/raid_blk/data/python_class/code_examples/08_library_example.py
Operating System is  posix
Darwin
josephs-iMac.local
16.7.0
Darwin Kernel Version 16.7.0: Mon Nov 13 21:56:25 PST 2017; root:xnu-3789.72.11~1/RELEASE_X86_64
x86_64

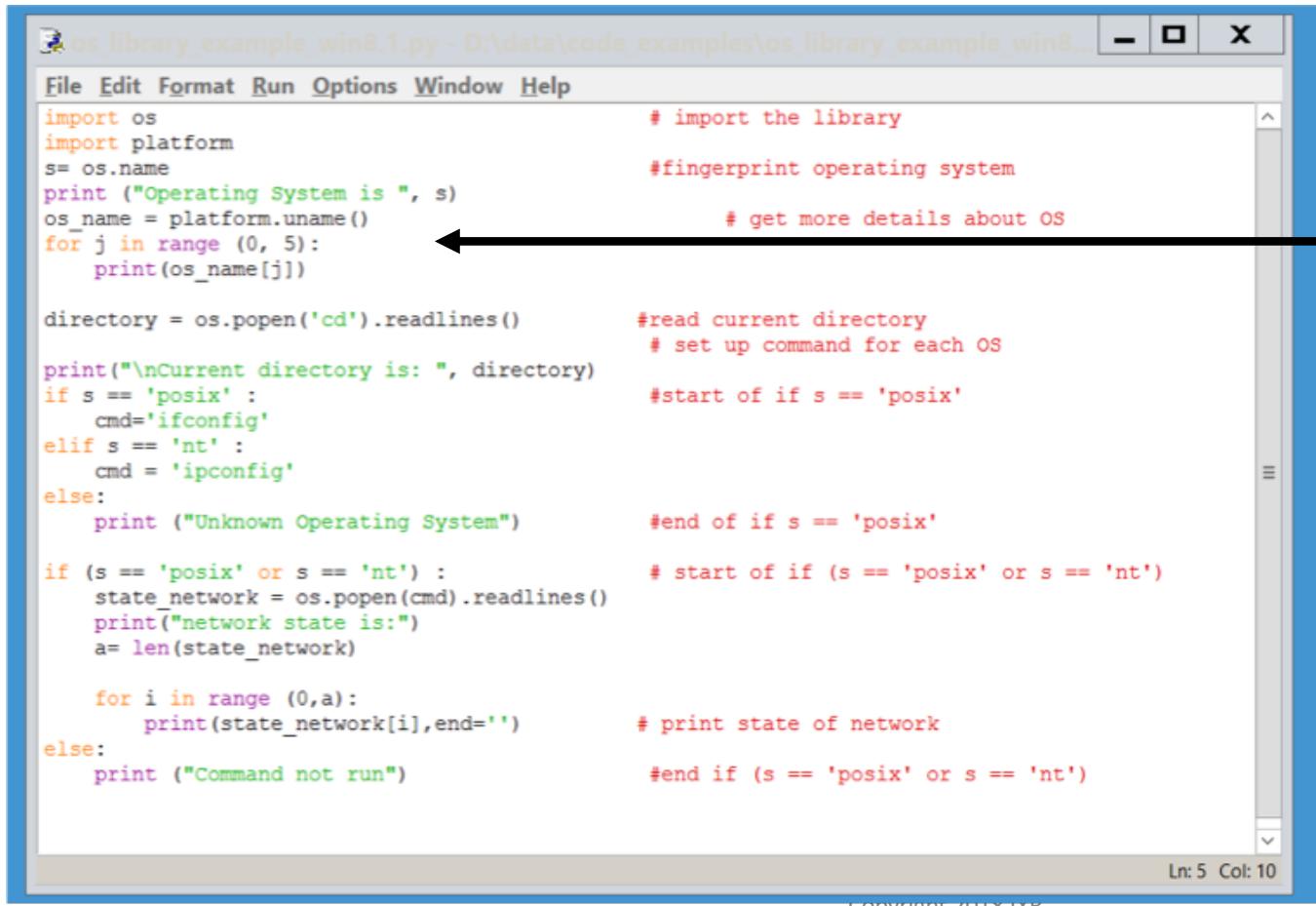
Current directory is: /Volumes/raid_blk/data/python_class/code_examples

network state is:
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
    inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
            inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
                nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=10b<RXCSUM,TXCSUM,VLAN_HWTAGGING,AV>
    ether a8:60:b6:3c:e0:58
        inet6 fe80::109c:104c:e982:c1eb%en0 prefixlen 64 secured scopeid 0x4
        inet 192.168.201.101 netmask 0xffffffff broadcast 192.168.201.255
            nd6 options=201<PERFORMNUD,DAD>
            media: autoselect (100baseTX <full-duplex,flow-control,energy-efficient-ethernet>)
            status: active
Ln: 74 Col: 4
```

Rest of Output

```
Python 3.6.3 Shell
en1: flags=8823<UP,BROADCAST,SMART,SIMPLEX,MULTICAST> mtu 1500
      ether 98:9e:63:26:39:1c
      nd6 options=201<PERFORMNUD,DAD>
      media: autoselect (<unknown type>)
      status: inactive
en2: flags=963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX> mtu 1500
      options=68<TS04,TS06>
      ether aa:00:d9:68:82:00
      media: autoselect <full-duplex>
      status: inactive
en3: flags=963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX> mtu 1500
      options=68<TS04,TS06>
      ether aa:00:d9:68:82:01
      media: autoselect <full-duplex>
      status: inactive
bridge0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
      options=63<RXCSUM,TXCSUM,TS04,TS06>
      ether aa:00:d9:68:82:00
      Configuration:
          id 0:0:0:0:0:0 priority 0 fwddelay 0
          maxage 0 holdcnt 0 proto stp maxaddr 100 timeout 1200
          root id 0:0:0:0:0:0 priority 0 ifcost 0 port 0
          ipfilter disabled flags 0x2
      member: en2 flags=3<LEARNING,DISCOVER>
          ifmaxaddr 0 port 6 priority 0 path cost 0
      member: en3 flags=3<LEARNING,DISCOVER>
          ifmaxaddr 0 port 7 priority 0 path cost 0
      nd6 options=201<PERFORMNUD,DAD>
      media: <unknown type>
      status: inactive
p2p0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 2304
      ether 0a:9e:63:26:39:1c
      media: autoselect
      status: inactive
awdl0: flags=8902<BROADCAST,PROMISC,SIMPLEX,MULTICAST> mtu 1484
      ether 9e:af:22:b0:89:fd
      nd6 options=201<PERFORMNUD,DAD>
      media: autoselect
      status: inactive
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2000
      inet6 fe80::de1e:7466:6c00:bfce%utun0 prefixlen 64 scopeid 0xb
      nd6 options=201<PERFORMNUD,DAD>
>>>                                         Ln: 74 Col: 4
```

Examples of OS calls Windows



```
File Edit Format Run Options Window Help
import os                                # import the library
import platform                          #fingerprint operating system
s= os.name                               # get more details about os
print ("Operating System is ", s)
os_name = platform.uname()
for j in range (0, 5):
    print(os_name[j])

directory = os.popen('cd').readlines()      #read current directory
                                                # set up command for each OS
print("\nCurrent directory is: ", directory)
if s == 'posix':
    cmd='ifconfig'
elif s == 'nt' :
    cmd = 'ipconfig'
else:
    print ("Unknown Operating System")      #end of if s == 'posix'

if (s == 'posix' or s == 'nt') :
    state_network = os.popen(cmd).readlines()
    print("network state is:")
    a= len(state_network)

    for i in range (0,a):
        print(state_network[i],end='')     # print state of network
else:
    print ("Command not run")              #end if (s == 'posix' or s == 'nt')
```

Command os.uname does not work in Windows
Platform.uname() does work

Results

```
Command Prompt

D:\data\code_examples>c:\Users\joe\AppData\Local\Programs\Python\Python36-32\python.exe os_library_example_win8.1.py
Operating System is nt
Windows
joe-sdr-desktop
8.1
6.3.9600
AMD64

Current directory is: ['D:\\data\\code_examples\\n']
network state is:

Windows IP Configuration

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix . . . .
    Link-local IPv6 Address . . . . . : fe80::e4b7:2f75:c29a:4f3d%9
    IPv4 Address . . . . . : 192.168.201.116
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.201.1

Ethernet adapter Bluetooth Network Connection:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . . .

Ethernet adapter Local Area Connection 2:

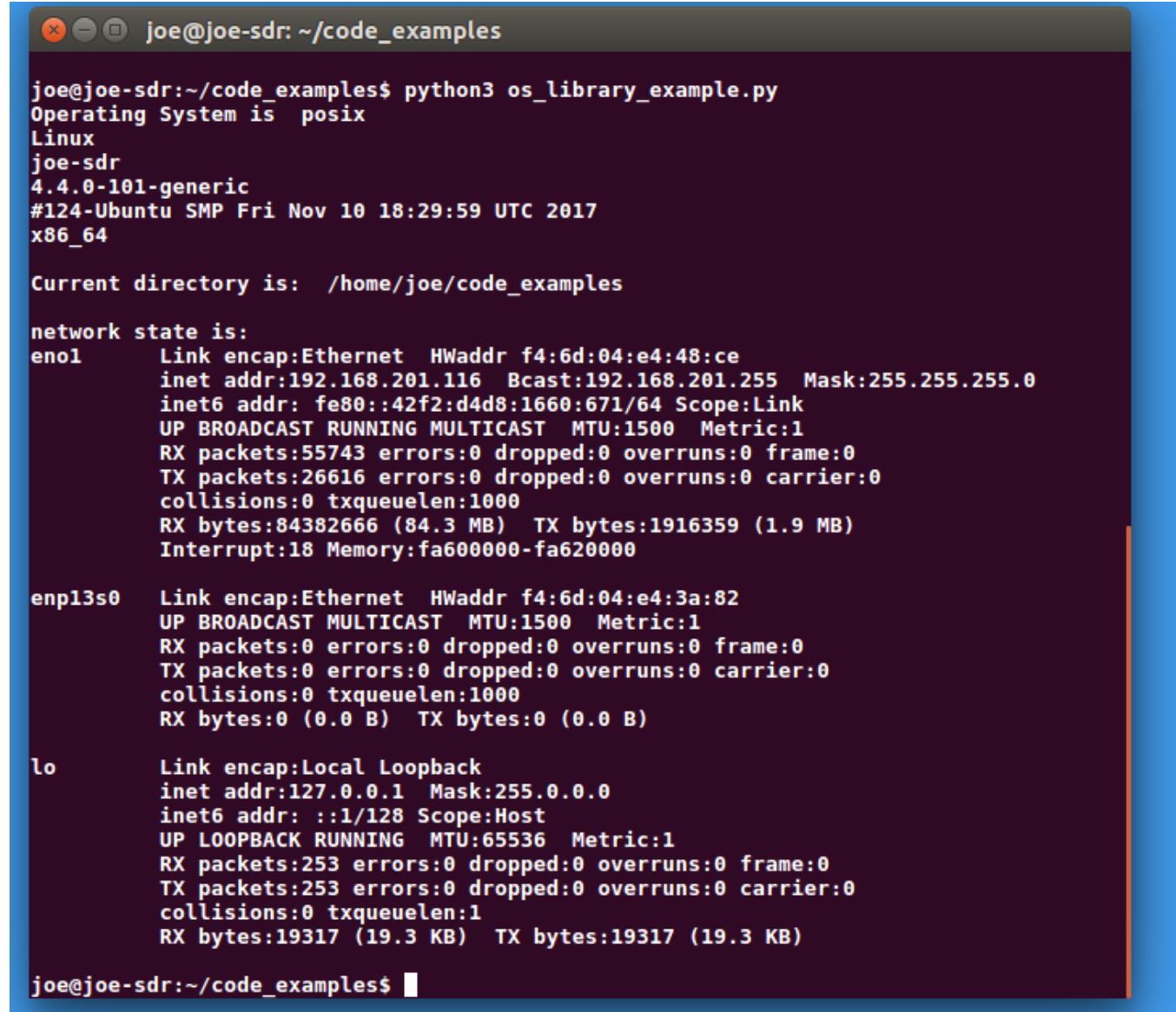
    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . . .

Tunnel adapter isatap.(75C72278-DB33-4746-B9F3-118C170DC00D):

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix . . .

D:\data\code_examples>
```

Example of OS calls in Linux



joe@joe-sdr:~/code_examples\$ python3 os_library_example.py
Operating System is posix
Linux
joe-sdr
4.4.0-101-generic
#124-Ubuntu SMP Fri Nov 10 18:29:59 UTC 2017
x86_64

Current directory is: /home/joe/code_examples

network state is:
en0 Link encap:Ethernet HWaddr f4:6d:04:e4:48:ce
inet addr:192.168.201.116 Bcast:192.168.201.255 Mask:255.255.255.0
inet6 addr: fe80::42f2:d4d8:1660:671/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:55743 errors:0 dropped:0 overruns:0 frame:0
TX packets:26616 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:84382666 (84.3 MB) TX bytes:1916359 (1.9 MB)
Interrupt:18 Memory:fa600000-fa620000

enp13s0 Link encap:Ethernet HWaddr f4:6d:04:e4:3a:82
UP BROADCAST MULTICAST MTU:1500 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:253 errors:0 dropped:0 overruns:0 frame:0
TX packets:253 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1
RX bytes:19317 (19.3 KB) TX bytes:19317 (19.3 KB)

joe@joe-sdr:~/code_examples\$

Home Work

- `time.time()` returns the time in seconds since January 1, 1970, 00:00:00 (UTC). When will this no longer work. i.e., at what year/date/hour will the clock reset to Jan 1, 1970
- The example pygame program goes off the screen with the circle is moved multiple times.
 - Change the program so when the circle goes off the screen it moves to the other side (up, down, left, right) like some old video games
 - This is called an unbounded universe
 - Bonus: Change the program so the direction of move changes when the circle hits the edge of the screen
 - This is called a bounded universe
 - Question of the ages: Is our real universe bounded or unbounded?
- Modify the os example to make one program for OS X/Linux/Windows