RECAP OF BASICS

- Lists

```
list = [['a', 'b', 'c'], [1, 2, 3], [8, 9, 10]]
list[0][1] #prints ‘b’
list[-1][2] #prints 10
list[0:2] #returns [[‘a’, ‘b’, ‘c’], [1,2,3]]
list[1:] #evaluates from index 1 till the end
list[:2] #evaluates from index 0 till index 1
```

RECAP OF BASICS

```
names = ['mel', 'nel', ‘vasanth']
for name in range(len(names)):
        print('Name at ' + str(name) + ' is ' + names[name])
#Strings can also do many of the same operations as lists but are immutable


mel, nel, vasanth = names
#will assign the respective values to the variables on the left


one, two, three = 'one', 'two', 'three'
#also works in the same way
```

RECAP OF BASICS

names.index('mel') #will return the index of the first occurence or raise an exception

names.append('prince') #will add prince to the end of the list

names.insert(1, 'prince') #will insert prince at index 1 of the list

names.remove('nel') #removes nel from the list

names.sort() #will sort the list in ASCII- betical

names.sort(key=str.lower) #will sort in true alphabetical order

names.sort(reverse=True) #will reverse sort the list

RECAP OF BASICS

Exercise – Write a Python program to implement a ‘Stack’.

RECAP OF BASICS

Sample solution –

```
def push(item):
    global stack
    stack.append(item)

def pop():
    global stack
    return stack.pop()

def display():
    print(stack)
```

```
def doOperations():
    op = input('Press 1 to push. 2 
to pop. 3 to print ')
    if 1 == int(op):
        item = input('Enter what 
you want to push ')
        push(item)
    elif 2 == int(op):
        length = len(stack)
        if length > 0:
            top = pop()
            print(top, ' was returned')
        else: print('Cannot pop 
because stack is empty')
    elif 3 == int(op):
        display()
```

```
stack = []

ch = ''
while ch != 'q':
    doOperations()
    ch = input('Press q to quit or 
any other key to continue ')
```

RECAP OF BASICS

```
numbers = {"one" : 1, "two" : 2, "three" : 3, 4 : "four"}
numbers.keys() #returns keys
numbers.values() #returns values
numbers.items() #returns items as a tuple list

for k,v in numbers.items():
        print(k,v)

names.get(five', '') #returns a value if the key is found or returns the default

names.default(five', 5) #will set a key-value pair in the dictionary if the key doesn't exist
```

RECAP OF BASICS

```
import os
os.getcwd()
#returns the current working directory

import pprint
pprint.pprint(names)
#will pretty print any collection

import copy
namescopy = copy.deepcopy(names)
#will create a new copy and not just assign the reference of names
```

RECAP OF BASICS

```
def Func(*args, **kwargs):
    for arg in args:
        print(arg)
    for item in kwargs.items():
        print(item)

Func('hello', 'hi', x = 1, y = 2)
```

THREADS

```
import threading
import time

def loop(count, sleeptime):
    for i in range(1, count+1):
        time.sleep(sleeptime) #seconds
        print(i)
        print(threading.current_thread())

threading.Thread(name='t1', target=loop, args=(10,0.2)).start()
threading.Thread(name='t2', target=loop, args=(6,0.1)).start()
```

CURSES

```
import curses
import time

screen = curses.initscr() #initialize the screen
curses.noecho() #don’t display the key entered
curses.start_color() #enable colors

#your code

screen.getch() #get key input
curses.endwin() #end program
```

CURSES

```
import curses
import time
screen = curses.initscr()
dim = screen.getmaxyx() #returns a tuple of the dimensions (y,x)
for z in range(dim[1]-12):
        screen.clear()
        screen.addstr(dim[0]/2,z, 'Hello')
        screen.refresh()
        time.sleep(0.5)
screen.getch()
curses.endwin()
```