Python Classes and Objects

Classes and Objects

class MyClass:

variable = "blah"

def function(self):

print("This is a message inside the class.")

myobjectx = MyClass()

myobjectx.function()

Inheritance –

# Example file for working with classes

class myClass():

def method1(self):

print ("Guru99")

def method2(self,someString):

print ("Software Testing:" + someString)

class childClass(myClass):

#def method1(self):

#myClass.method1(self);

#print "Child Class Method1"

def method2(self):

print ("childClass method2")

def main():

# exercise the class methods

c2 = childClass()

c2.method1()

c2.method2()

if \_\_name\_\_== "\_\_main\_\_":

main()

Date and time

from datetime import date

from datetime import time

from datetime import datetime

def main():

##DATETIME OBJECTS

#Get today's date from datetime class

today=datetime.now()

#print today

# Get the current time

#t = datetime.time(datetime.now())

#print "The current time is", t

#weekday returns 0 (monday) through 6 (sunday)

wd = date.weekday(today)

#Days start at 0 for monday

days= ["monday","tuesday","wednesday","thursday","friday","saturday","sunday"]

print "Today is day number %d" % wd

print "which is a " + days[wd]

print now.strftime(“%a,%d,%b,%y”)

if \_\_name\_\_== "\_\_main\_\_":

main()

Calender

import calendar

#Create a plain text calendar

c= calendar.TextCalendar(calendar.THURSDAY)

str= c.formatmonth(2015,1,0,0)

print str

#Create an HTML formatted calendar

hc = calendar.HTMLCalendar(calendar.THURSDAY)

str = hc.formatmonth(2015, 1)

print str

#loop over the days of a month

#zeroes indicate that the day of the week is in a next month or overlapping month

for i in c.itermonthdays(2015,4):

print i

#The calendar can give info based on local such a names of days and months (full and abbreviated forms)

for name in calendar.month\_name:

print name

for day in calendar.day\_name:

print day

#calculate days based on a rule: For instance an audit day on the second Monday of every month

#Figure out what days that would be for each month, we can use the script as shown here

for month in range(1,13):

# It retrieves a list of weeks that represent the month

mycal = calendar.monthcalendar(2020, month)

# The second MONDAY has to be within the first two weeks

week1 = mycal[1]

week2 = mycal[2]

if week1[calendar.MONDAY] != 0:

auditday = week1[calendar.MONDAY]

else:

# if the second MONDAY isn't in the first week, it must be in the second week

auditday = week2[calendar.MONDAY]

print "%10s %2d" % (calendar.month\_name[month], auditday)