```
from __future__ import division
import sys
import random
import math
import numpy as np
from models import *
from searchers import *
from options import *
from utilities import *
from sk import *
sys.dont_write_bytecode = True
#Dr.M
rand=  random.random # generate nums 0..1
any=   random.choice # pull any from list
sqrt= math.sqrt  #square root function

def display(modelName,searcher,runTimes,scores,historyhi=[],historylo=[]):
  assert(len(runTimes) ≡ len(scores)),'Ouch! it hurts'
  print "=============================================================="
  print "Model Name: %s"%modelName
  print "Searcher Name: %s"%searcher
  print "Options Used: ",
  print myoptions[searcher]
  import time
  print ("Date: %s"%time.strftime("%d/%m/%Y"))
  print "Average running time: %f " %np.mean(runTimes)
  if(len(historyhi)≠0):
    for x in xrange(myModelobjf[modelName]):
      print "Objective No. %d: High: %f Low: %f"%(x+1,historyhi[x],historylo[x])
  #for i in range(0,len(runTimes)):
  #  print "RunNo: %s RunTime: %s Score: %s"%(i+1,runTimes[i],scores[i])
  #print scores
  print xtile(scores,width=25,show=" %1.6f")
  print "=============================================================="



def multipleRun():
   from collections import defaultdict
   r = 2
   for klass in [Fonseca]:#Schwefel,Fonseca,Viennet,Kursawe,ZDT1,ZDT3,Schaffer,DTLZ7]:
     print "Model Name: %s"%klass.__name__
     eraCollector=defaultdict(list)
     for searcher in [DE]:#,GA,SA,MaxWalkSat]:
       n = 0.0
       listTimeTaken = []
       listScores = []
       random.seed(6)
       historyhi=[−9e10 for count in xrange(myModelobjf[klass.__name__])]
       historylo=[9e10 for count in xrange(myModelobjf[klass.__name__])]
       for _ in range(r):
         test = searcher(klass(),"display2")
         print ".",
         import time
         t1 = time.time()
         solution,score,model = test.evaluate()
         #print score,model.minVal,model.maxVal
         for x in xrange(model.objf):
           #print len(model.past[x].listing)
           #print x
           historyhi[x]=max(model.past[x].historyhi,historyhi[x])
           historylo[x]=min(model.past[x].historylo,historylo[x])
           sys.stdout.flush()
         timeTaken = (time.time() − t1) * 1000
         listTimeTaken.append(timeTaken)
         listScores.append(score)
         eraCollector[searcher.__name__]=listScores
         #print "Score: %f"%(score)
       print
     callrdivdemo(eraCollector)

def step2():
    rdivDemo([
      ["Romantic",385,214,371,627,579],
      ["Action",480,566,365,432,503],
      ["Fantasy",324,604,326,227,268],
      ["Mythology",377,288,560,368,320]])


def callrdivdemo(eraCollector):
  #print eraCollector
  #print "callrdivdemo %d"%len(eraCollector.keys())
  keylist = eraCollector.keys()
  #print keylist
  variant = len(keylist)
  #print variant
  rdivarray=[]
  for y in xrange(variant):
      #print "Length of array: %f"%len(eraCollector[keylist[y]][x])
      temp = eraCollector[keylist[y]]
      #print temp
      temp.insert(0,str(keylist[y]))
      #print temp
      rdivarray.append(temp)
  rdivDemo(rdivarray)



def testDE():
  for klass in [Viennet]:
    random.seed(6)
    test = DE(klass(),"display2")
    print test.evaluate()
```

```
if __name__ ≡ '__main__':
 # random.seed(1)
 # nums = [random.random()**2 for _ in range(100)]
 # print xtile(nums,lo=0,hi=1.0,width=25,show=" %3.2f")
 # model = ZDT1()
 # model.testgx()
 # for klass in [ZDT1]:
 #   print klass.__name__
 multipleRun()
 #testDE()
 #part6()
 #step2()
```