```
from __future__ import division
import sys
import random
import math
import numpy as np
sys.dont_write_bytecode = True

sqrt=math.sqrt

class ModelBasic():
  def returnMin(self,num):
    if(num<self.minVal):
      self.minVal=num
      return num
    else:
      return self.minVal

  def returnMax(self,num):
    if(num>self.maxVal):
      self.maxVal=num
      return num
    else:
      return self.maxVal

  def evaluate(self,listpoint):
    energy = self.f1(listpoint)+ self.f2(listpoint)
    return (energy-self.minVal)/(self.maxVal-self.minVal)



  def  neighbour(self,minN,maxN):
    return minN + (maxN-minN)*random.random()

class Fonseca(ModelBasic):
  maxVal=-10000
  minVal=10000

  def __init__(self,minR=-4,maxR=4,n=3):
    self.minR=minR
    self.maxR=maxR
    self.n=n
    self.minVal=10000000
    self.maxVal=-1e6

  def f1(self,listpoint):
    n=len(listpoint)
    rootn=(n**0.5)**-1
    sum=0
    for i in range(0,n):
        sum+=(listpoint[i]-rootn)**2
    return (1 - math.exp(-sum))

  def f2(self,listpoint):
    n=len(listpoint)
    rootn=(n**0.5)**-1
    sum=0
    for i in range(0,n):
        sum+=(listpoint[i]+rootn)**2
    return (1 - math.exp(-sum))

  def info(self):
    return "Fonseca~"

  def baseline(self,minR,maxR):
    for x in range(0,50000):
      solution = [(minR + random.random()*(maxR-minR)) for z in range(0,3)]
      self.returnMax(self.f1(solution)+ self.f2(solution))
      self.returnMin(self.f1(solution)+ self.f2(solution))


class Kursawe(ModelBasic):
  def __init__(self,minR=-5,maxR=5,n=3):
    self.minR=minR
```



```
    self.maxR=maxR
    self.n=n
    self.minVal=10000000
    self.maxVal=-1e6


  def f1(self,listpoint):
    n=len(listpoint)
    #inspired by 'theisencr'
    return np.sum([-10*math.exp(-0.2*(np.sqrt(listpoint[i]**2 + listpoint[i+1]**
2))) for i in range (0, n-1)])
    return sum

  def f2(self,listpoint):
    a=0.8
    b=3
    n=len(listpoint)
    #inspired by 'theisencr'
    return np.sum([math.fabs(listpoint[i])**a + 5*np.sin(listpoint[i])**b for i
in range (0, n)])

  def info(self):
    return "Kursawe~"

  def baseline(self,minR,maxR):
    for x in range(0,50000):
      solution = [(minR + random.random()*(maxR-minR)) for z in range(0,3)]
      self.returnMax(self.f1(solution)+ self.f2(solution))
      self.returnMin(self.f1(solution)+ self.f2(solution))

class ZDT1(ModelBasic):
  maxVal=-10000
  minVal=10000

  def __init__(self,minR=0,maxR=1,n=30):
    self.minR=minR
    self.maxR=maxR
    self.n=n

  def f1(self,lst):
    assert(len(lst)==self.n),"Something's Messed up"
    return lst[0]

  def gx(self,lst):
    n=self.n
    assert(len(lst) == n),"Something's Messed up"
    return (1+ 9*np.sum([lst[i] for i in range(1,n)])/(n-1))

  def f2(self,lst):
    n=self.n
    assert(len(lst)==n),"Something's Messed up"
    gx=self.gx(lst)
    assert(gx!=0),"Ouch! it hurts"
    return gx * (1- sqrt(lst[0]/gx))


  def baseline(self,minR=0,maxR=1):
    for x in range(0,90000):
      solution = [(minR + random.random()*(maxR-minR)) for z in range(0,30)]
      self.returnMax(self.f1(solution)+ self.f2(solution))
      self.returnMin(self.f2(solution)+ self.f2(solution))

  def info(self):
    return "ZDT1~"


class Schaffer(ModelBasic):

  def __init__(self,minR=-1e4,maxR=1e4,n=1):
    self.minR=minR
    self.maxR=maxR
    self.n=n
    self.minVal=10000000
```

```
145       self.maxVal=-1e6

      def evaluate(self,listpoint):
        assert(len(listpoint) == 1),"Something's Messed up"
        var=listpoint[0]
150     rawEnergy = (var**2 +(var-2)**2)
        energy = (rawEnergy -self.minVal)/(self.maxVal-self.minVal)
        return energy


155   def info(self):
        return "Schaffer~"

      def baseline(self,minR,maxR):
        low = self.minR
160     high = self.maxR
        for index in range(0,1000000):
          inputRand =(low + (high-low)*random.random())
          #print "inputRand: %s"%inputRand
          temp = (inputRand**2 +(inputRand-2)**2)
165       self.minVal=self.returnMin(temp)
          self.maxVal=self.returnMax(temp)
        print("Max: %d Min: %d"%(self.maxVal,self.minVal))
```