|  |
| --- |
| import random from datetime import datetime, timedelta import pandas as pd from dateutil.parser import parse   class CreatingDB:  *"""  Class for creating random database  """* num\_people = 0 *# number of people to create* base\_date = None *# the base date of data* def \_\_init\_\_(self, num\_people, base\_date):  self.num\_people = num\_people  self.base\_date = base\_date   def generate\_incurred\_date(self):  *"""  function to create random incurred date  :return:  incurred\_date: string, the day of infection or contact  elapsed\_days: int, the difference between base date and incurred date  """* elapsed\_days = random.randint(0, 14) *# the valid day period is 0~14  # extracting the incurred day using periods and base date* incurred\_date = (self.base\_date - timedelta(days=elapsed\_days)).\  strftime(**"%Y %m %d"**)  return incurred\_date, elapsed\_days   def compute\_severity(self, status, elapsed\_days):  *"""  function to compute severity  :param status: string, 'contacted' or 'confirmed'  :param elapsed\_days: the period form the incurred date  :return: the computed severity using formula  """* if status == **'contacted'**: *# contacted person?* return 1 - (elapsed\_days \* 0.05)  else: *# confirmed person?* return (1 - (elapsed\_days \* 0.05)) \* 0.5   def generate\_address\_list(self):  *"""  function to get one address randomly from the adress list  :return: the randomly generated address list  """* with open(**'./Address\_Part.txt'**, **'r'**, encoding=**'utf-8'**) as add\_file:  *# add\_file = add\_file.encoding* address\_list = add\_file.readlines()   random\_address\_list = [] *# list to store addresses   # extract addresses as many as the number of recipients* for \_ in range(1, self.num\_people+1):  random\_address\_list.append(random.choice(address\_list))   return random\_address\_list   def generate\_csv\_data(self):  *"""  function to create .csv file with randomly generated records  :return: None  """* num\_healthy = round(self.num\_people / 3) *# 1/3 is healthy* num\_contacted = round(self.num\_people / 3) *# 1/3 is contacted  # 1/3 is confirmed* num\_confirmed = self.num\_people - num\_healthy - num\_contacted   id\_list = list(range(1, self.num\_people+1)) *# ID as many as people* random.shuffle(id\_list) *# shuffle list   # age records as many as people* age\_list = list(random.randint(1, 100)  for \_ in range(1, self.num\_people+1))  *# address records as many as people* address\_list = self.generate\_address\_list()   severity\_list = [] *# severity records as many as people* incurred\_date\_list = [] *# incurred date list including 'None'(healthy)* status\_list = [] *# status(Healthy, Contacted, and Confirmed) list   # Entire people num = healthy + contacted + confirmed  # Repeat as many healthy people* for \_ in range(num\_healthy):  severity\_list.append(0)  status\_list.append(**'Healthy'**)  incurred\_date\_list.append(**'None'**)   *# Repeat as many contacted people* for count in range(num\_contacted):  date, days = self.generate\_incurred\_date()  status\_list.append(**'Contacted'**)  severity\_list.append(round(self.compute\_severity(**'contacted'**, days), 2))  incurred\_date\_list.append(date)   *# Repeat as many confirmed people* for \_ in range(num\_confirmed):  date, days = self.generate\_incurred\_date()  status\_list.append(**'Confirmed'**)  severity\_list.append(round(self.compute\_severity(**'confirmed'**, days), 2))  incurred\_date\_list.append(date)   *# converting as pandas DataFrame data type to save .csv* df = pd.DataFrame({  **"ID"**: id\_list,  **"Age"**: age\_list,  **"Address"**: address\_list,  **"Covid Status"**: status\_list,  *# "Severity": severity\_list,* **"Incurred Date"**: incurred\_date\_list,  })  df = df.sort\_values([**'ID'**], ascending=[True])  df.reset\_index(drop=True, inplace=True)   *# saving as .csv file* df.to\_csv(**"corona\_data.csv"**, mode=**'w'**, encoding=**'utf-8-sig'**)   if \_\_name\_\_ == **'\_\_main\_\_'**:  *# require the number of people and base date* num\_people = int(input(**"Enter the number of people: "**))  date\_input = input(**"Enter the base date(Year-Month-Day): "**)  if date\_input == **''**:  print(**"The base date is set as today."**)  date = datetime.now().date()  else:  date = parse(date\_input).date()   cdb = CreatingDB(num\_people, date) *# creating instance* cdb.generate\_csv\_data() *# creating .csv file* |