Project Report

IFT 458 - PD 3

Spring 2018

Ashley Mendoza

Vincent Li

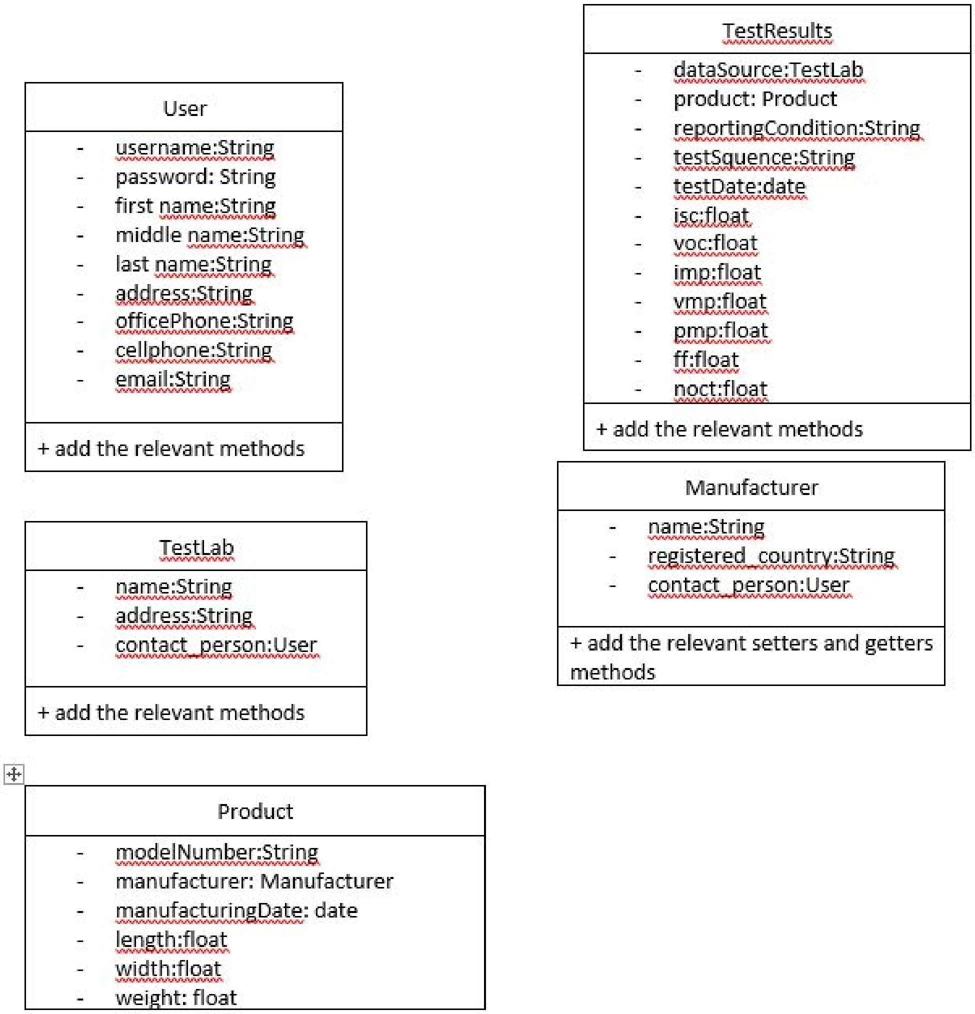
Dr. Kuitche

Feb 5, 2018

Problem Definition

Introduction

In this project deliverable part 3 we will add python scripts to aid data migration from excel or csv files into the previously created MySQL tables. In order to do this, we will create multiple files that will implement classes designed from a uml diagram as well as another file that will take information from a .csv file to read and return the contents of the file. This will be tested on a sample .csv file. Users will have the ability to add or register a new PV module from a specifications form. These users will be registered through a portal with the necessary registration data stored in a python dictionary. We will model the python classes on the following diagrams.



User

- username:String
- password: String
- first name:String
- middle name:String
- last name:String
- address:String
- officePhone:String
- cellphone:String
- email:String

+ add the relevant methods

TestResults

- dataSource:TestLab
- product: Product
- reportingCondition:String
- testSquence:String
- testDate:date
- isc:float
- voc:float
- imp:float
- vmp:float
- pmp:float
- ff:float
- noct:float

+ add the relevant methods

TestLab

- name:String
- address:String
- contact_person:User

+ add the relevant methods

Manufacturer

- name:String
- registered_country:String
- contact_person:User

+ add the relevant setters and getters methods

Product

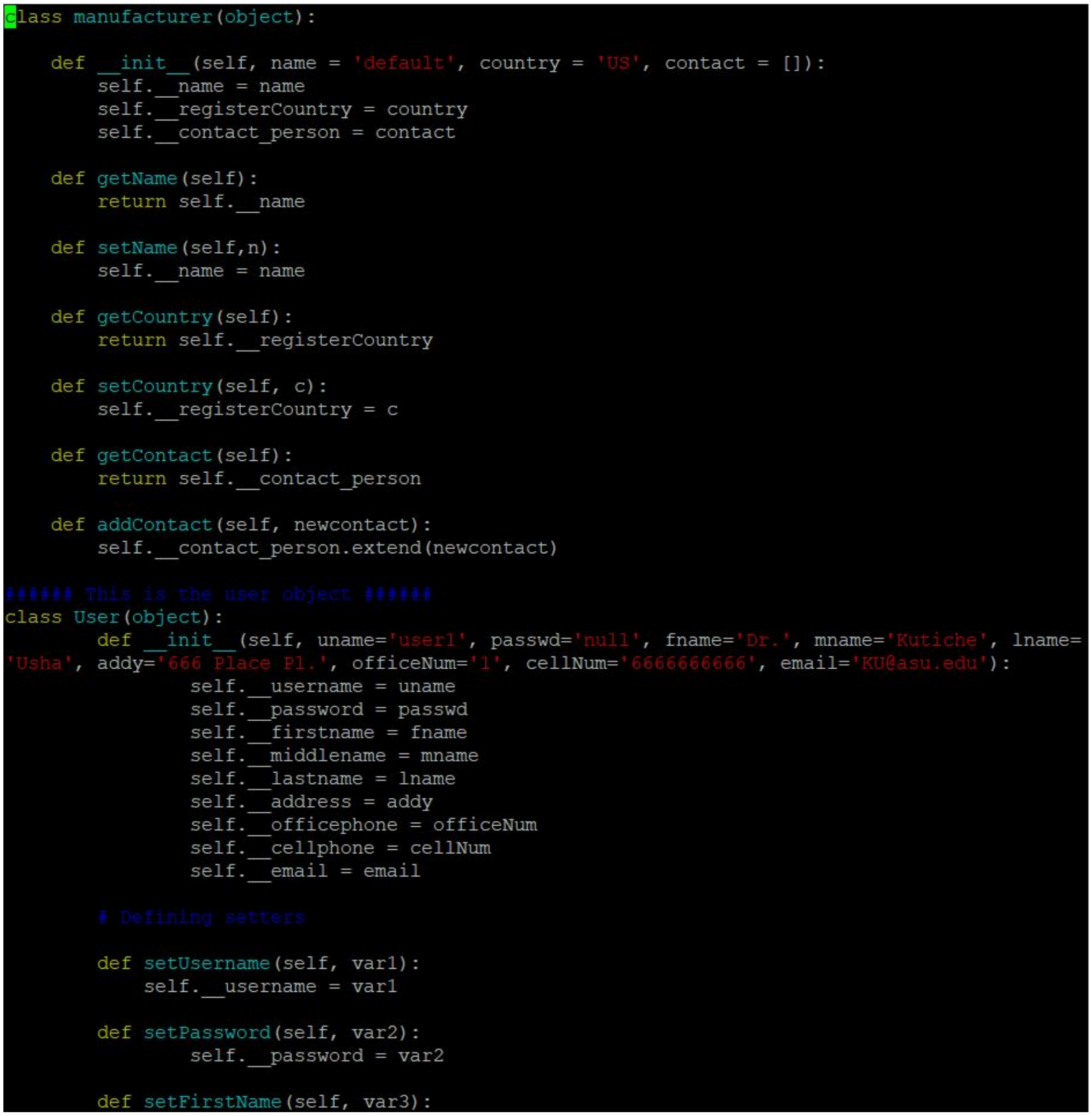
- modelNumber:String
- manufacturer: Manufacturer
- manufacturingDate: date
- length:float
- width:float
- weight: float

Description of Work

A. Implement all the classes from the class diagram above. Save them in a file called **MyClasses.py**.

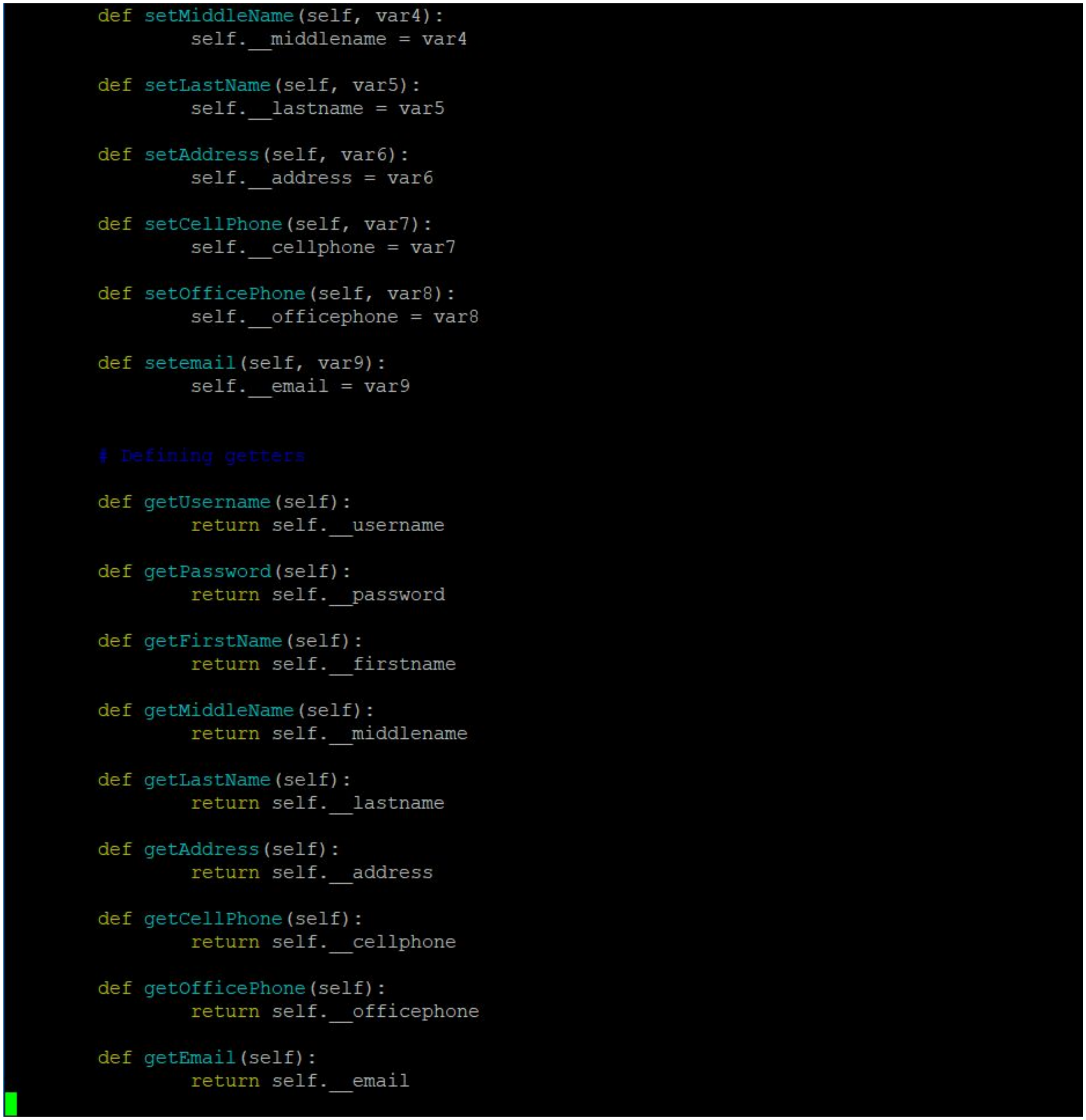
Figures 1 - 11 below demonstrate the code written to implement the classes (constructors, getters, setters) from the diagram above (Manufacturer, User, Test Lab, Product, and Test Results). The UML diagram above categorizes all of the attributes as private (-), thus the attributes were implemented with two underscores to establish the private permission. Additionally, the constructors contain parameters with default values. Overall, the goal of these classes are to allow us to instantiate objects and retrieve their values easily and effectively.

Figure 1: Class Manufacturer and User



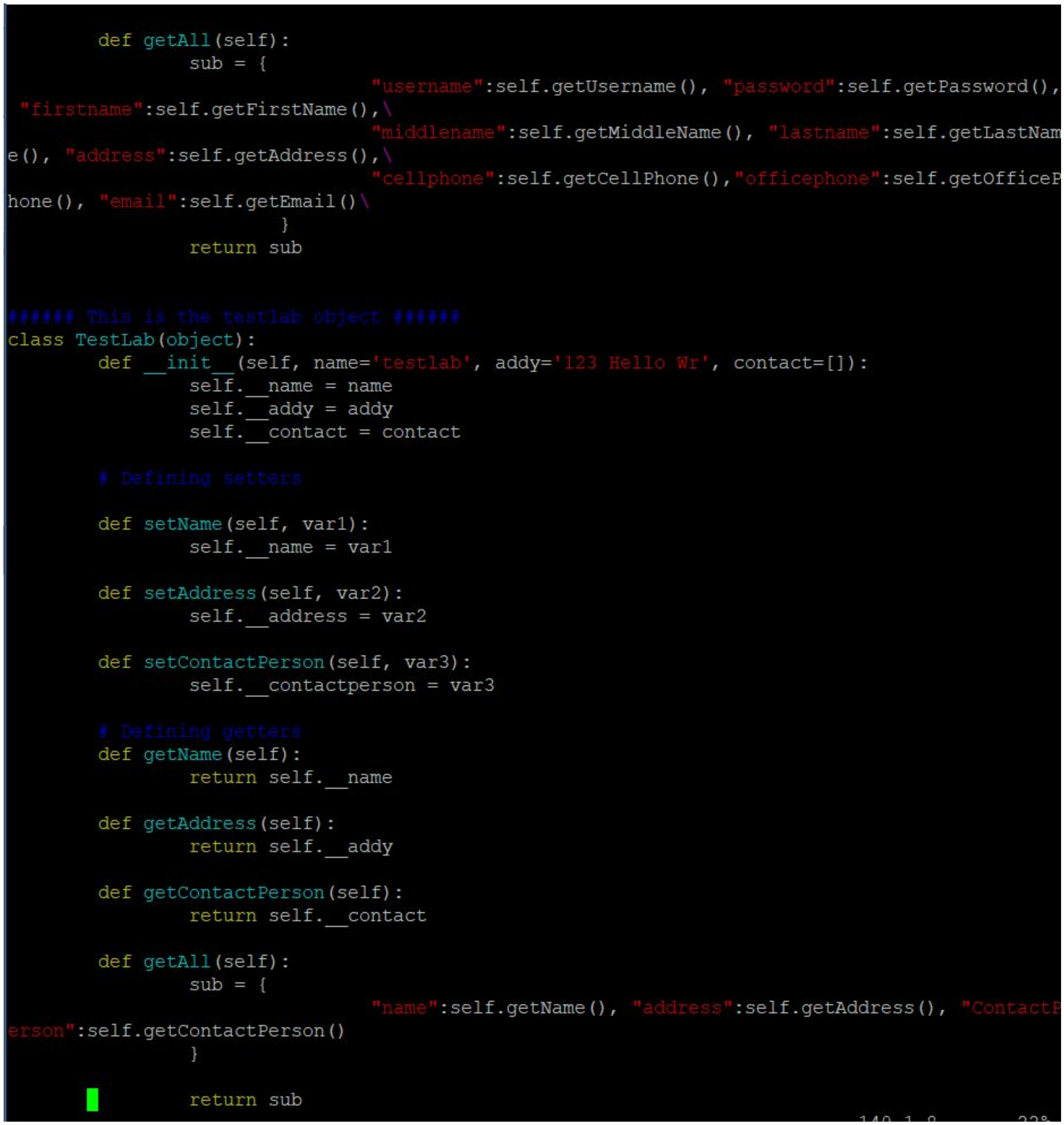
```
class manufacturer(object):

    def __init__(self, name = 'default', country = 'US', contact = []):
        self.__name = name
        self.__registerCountry = country
        self.__contact_person = contact

    def getName(self):
        return self.__name

    def setName(self,n):
        self.__name = name

    def getCountry(self):
        return self.__registerCountry

    def setCountry(self, c):
        self.__registerCountry = c

    def getContact(self):
        return self.__contact_person

    def addContact(self, newcontact):
        self.__contact_person.extend(newcontact)

###### This is the user object ######
class User(object):
        def __init__(self, uname='user1', passwd='null', fname='Dr.', mname='Kutiche', lname=
'Usha', addy='666 Place Pl.', officeNum='1', cellNum='6666666666', email='KU@asu.edu'):
                self.__username = uname
                self.__password = passwd
                self.__firstname = fname
                self.__middlename = mname
                self.__lastname = lname
                self.__address = addy
                self.__officephone = officeNum
                self.__cellphone = cellNum
                self.__email = email

        # Defining setters

        def setUsername(self, var1):
            self.__username = var1

        def setPassword(self, var2):
                self.__password = var2

        def setFirstName(self, var3):
```

Figure 2: Class User continued



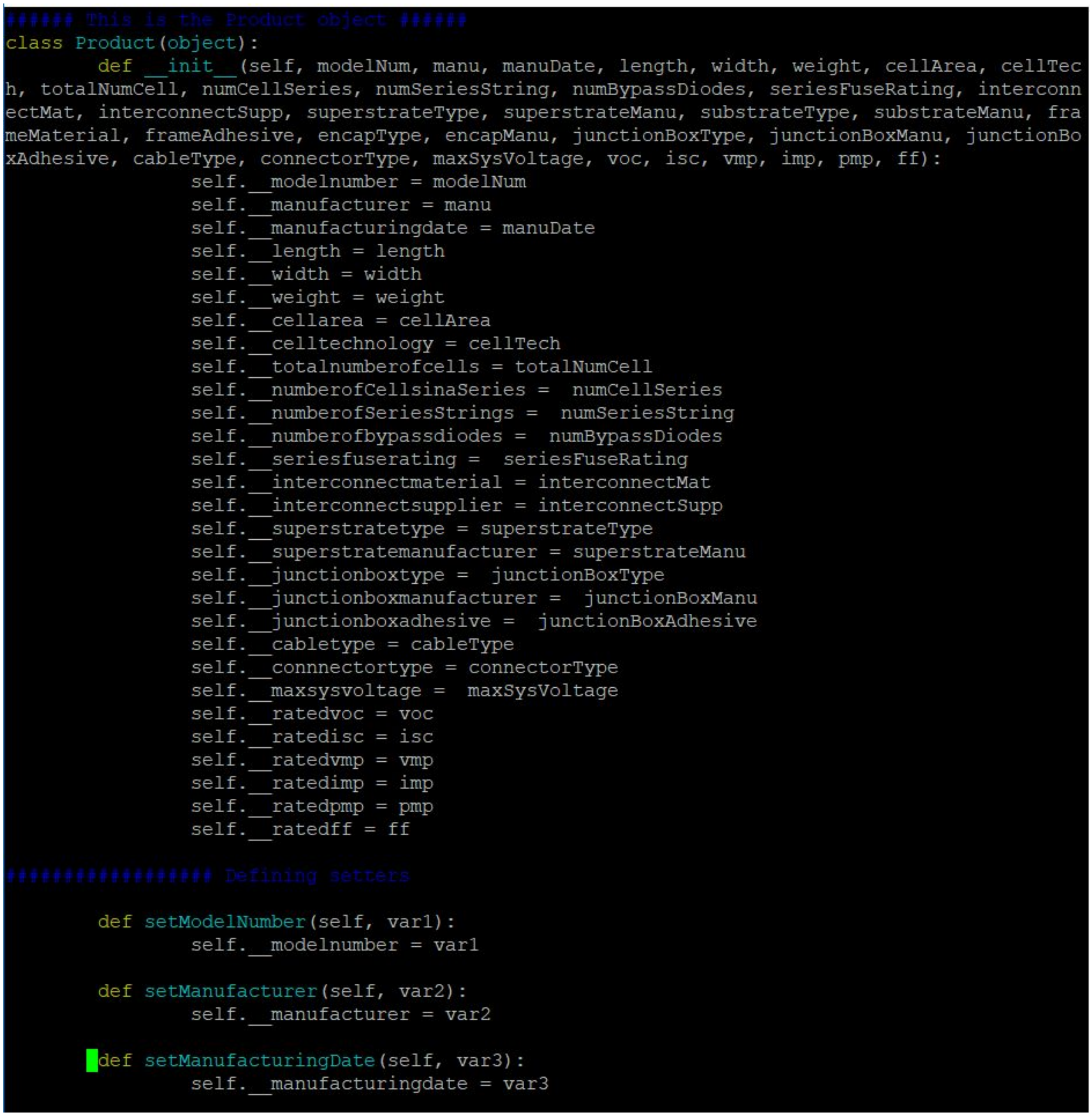
```
        def setMiddleName(self, var4):
                self.__middlename = var4

        def setLastName(self, var5):
                self.__lastname = var5

        def setAddress(self, var6):
                self.__address = var6

        def setCellPhone(self, var7):
                self.__cellphone = var7

        def setOfficePhone(self, var8):
                self.__officephone = var8

        def setemail(self, var9):
                self.__email = var9


        # Defining getters

        def getUsername(self):
                return self.__username

        def getPassword(self):
                return self.__password

        def getFirstName(self):
                return self.__firstname

        def getMiddleName(self):
                return self.__middlename

        def getLastName(self):
                return self.__lastname

        def getAddress(self):
                return self.__address

        def getCellPhone(self):
                return self.__cellphone

        def getOfficePhone(self):
                return self.__officephone

        def getEmail(self):
                return self.__email
```

Figure 3: Class User Continued and Class TestLab



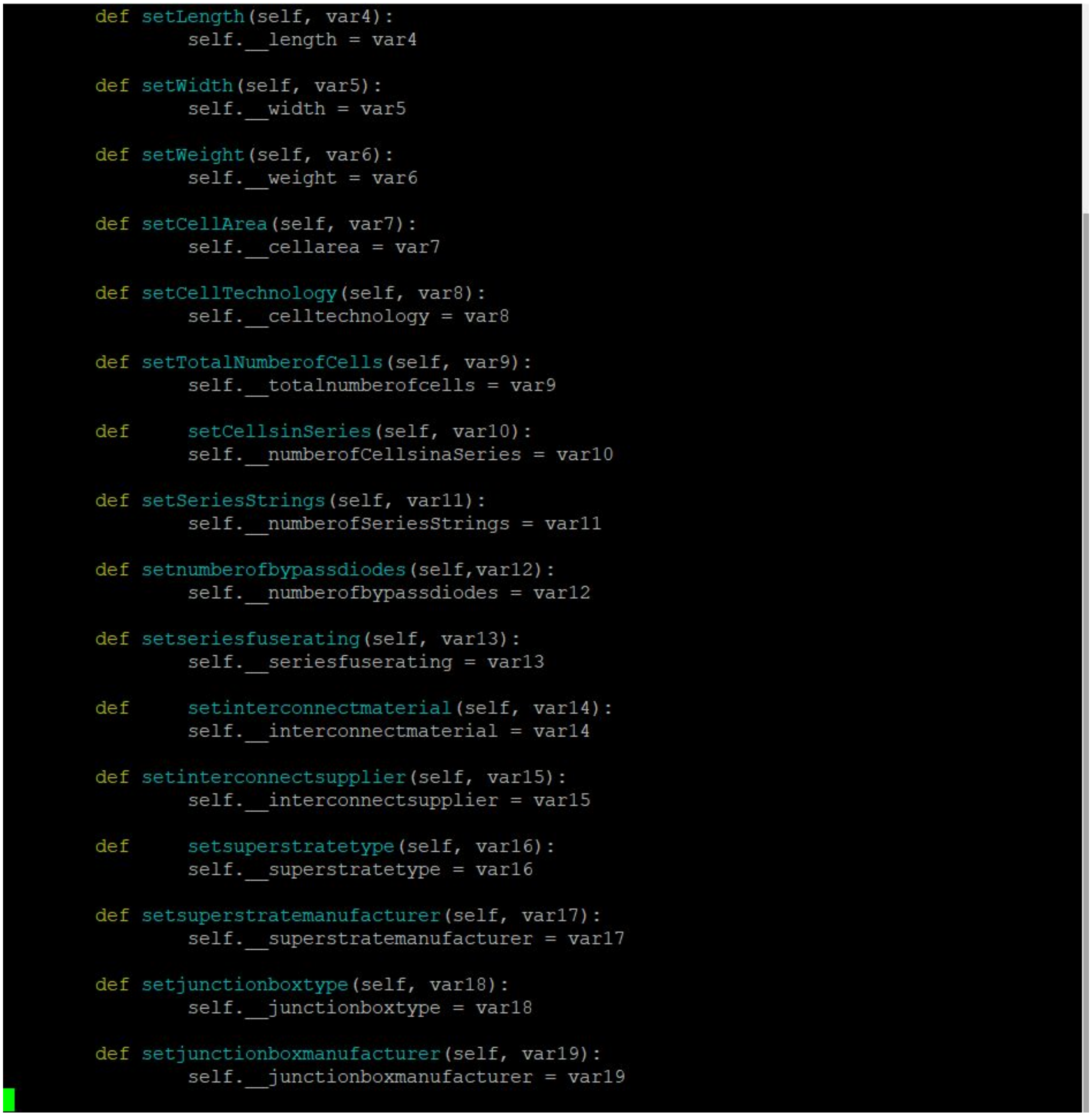
```
        def getAll(self):
                sub = {
                                "username":self.getUsername(), "password":self.getPassword(),
 "firstname":self.getFirstName(),\
                                "middlename":self.getMiddleName(), "lastname":self.getLastNam
e(), "address":self.getAddress(),\
                                "cellphone":self.getCellPhone(),"officephone":self.getOfficeP
hone(), "email":self.getEmail()\
                        }
                return sub


###### This is the testlab object ######
class TestLab(object):
        def __init__(self, name='testlab', addy='123 Hello Wr', contact=[]):
                self.__name = name
                self.__addy = addy
                self.__contact = contact

        # Defining setters

        def setName(self, var1):
                self.__name = var1

        def setAddress(self, var2):
                self.__address = var2

        def setContactPerson(self, var3):
                self.__contactperson = var3

        # Defining getters
        def getName(self):
                return self.__name

        def getAddress(self):
                return self.__addy

        def getContactPerson(self):
                return self.__contact

        def getAll(self):
                sub = {
                                "name":self.getName(), "address":self.getAddress(), "ContactP
erson":self.getContactPerson()
                }

                return sub
                                                                      140,1-8       23%
```

Figure 4: Class Product



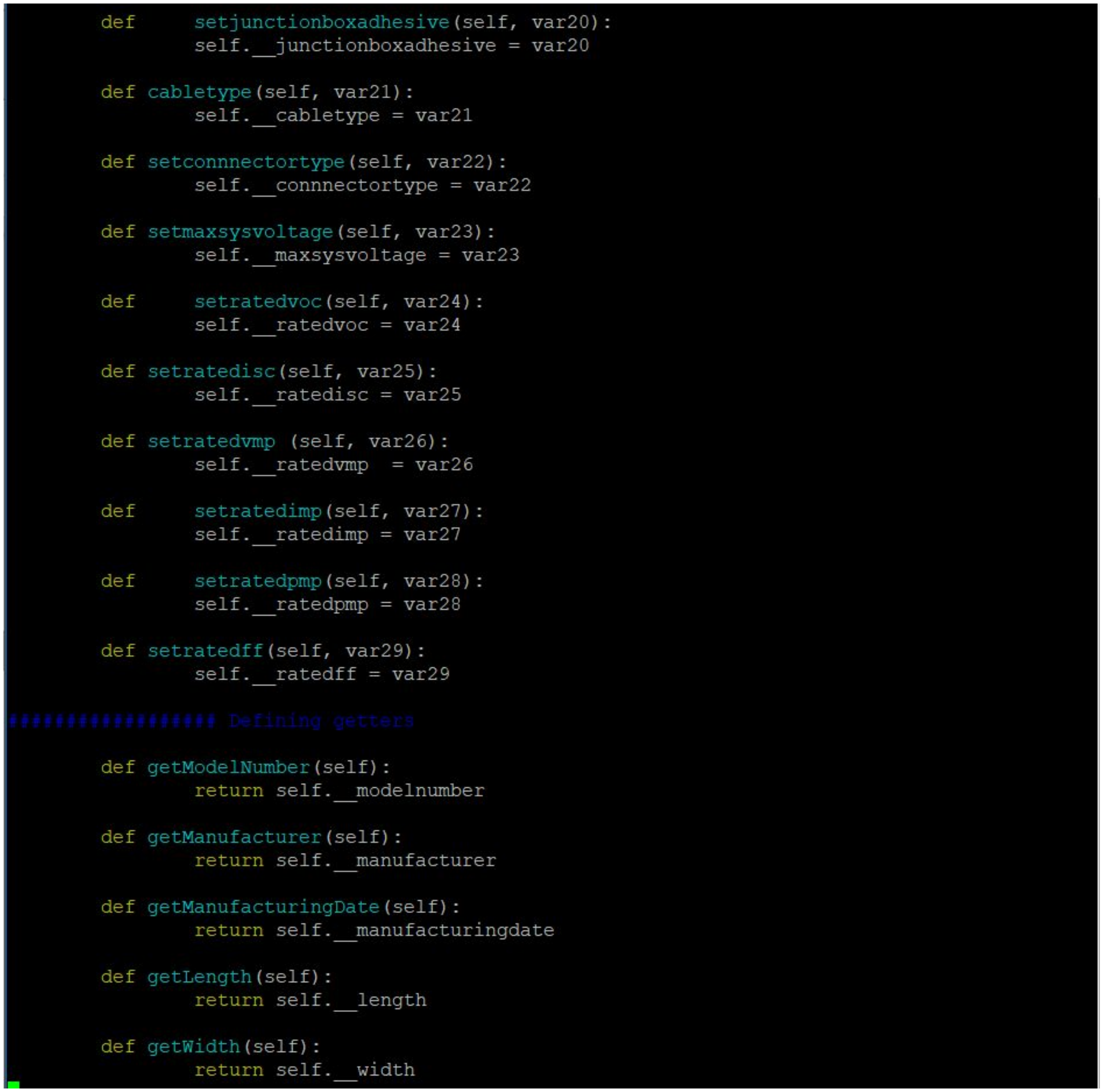
```
###### This is the Product object ######
class Product(object):
        def __init__(self, modelNum, manu, manuDate, length, width, weight, cellArea, cellTec
h, totalNumCell, numCellSeries, numSeriesString, numBypassDiodes, seriesFuseRating, interconn
ectMat, interconnectSupp, superstrateType, superstrateManu, substrateType, substrateManu, fra
meMaterial, frameAdhesive, encapType, encapManu, junctionBoxType, junctionBoxManu, junctionBo
xAdhesive, cableType, connectorType, maxSysVoltage, voc, isc, vmp, imp, pmp, ff):
                self.__modelnumber = modelNum
                self.__manufacturer = manu
                self.__manufacturingdate = manuDate
                self.__length = length
                self.__width = width
                self.__weight = weight
                self.__cellarea = cellArea
                self.__celltechnology = cellTech
                self.__totalnumberofcells = totalNumCell
                self.__numberofCellsinaSeries =  numCellSeries
                self.__numberofSeriesStrings =  numSeriesString
                self.__numberofbypassdiodes =  numBypassDiodes
                self.__seriesfuserating =  seriesFuseRating
                self.__interconnectmaterial = interconnectMat
                self.__interconnectsupplier = interconnectSupp
                self.__superstratetype = superstrateType
                self.__superstratemanufacturer = superstrateManu
                self.__junctionboxtype =  junctionBoxType
                self.__junctionboxmanufacturer =  junctionBoxManu
                self.__junctionboxadhesive =  junctionBoxAdhesive
                self.__cabletype = cableType
                self.__connnectortype = connectorType
                self.__maxsysvoltage =  maxSysVoltage
                self.__ratedvoc = voc
                self.__ratedisc = isc
                self.__ratedvmp = vmp
                self.__ratedimp = imp
                self.__ratedpmp = pmp
                self.__ratedff = ff

################## Defining setters

        def setModelNumber(self, var1):
                self.__modelnumber = var1

        def setManufacturer(self, var2):
                self.__manufacturer = var2

        def setManufacturingDate(self, var3):
                self.__manufacturingdate = var3
```

Figure 5: Class Product Continued



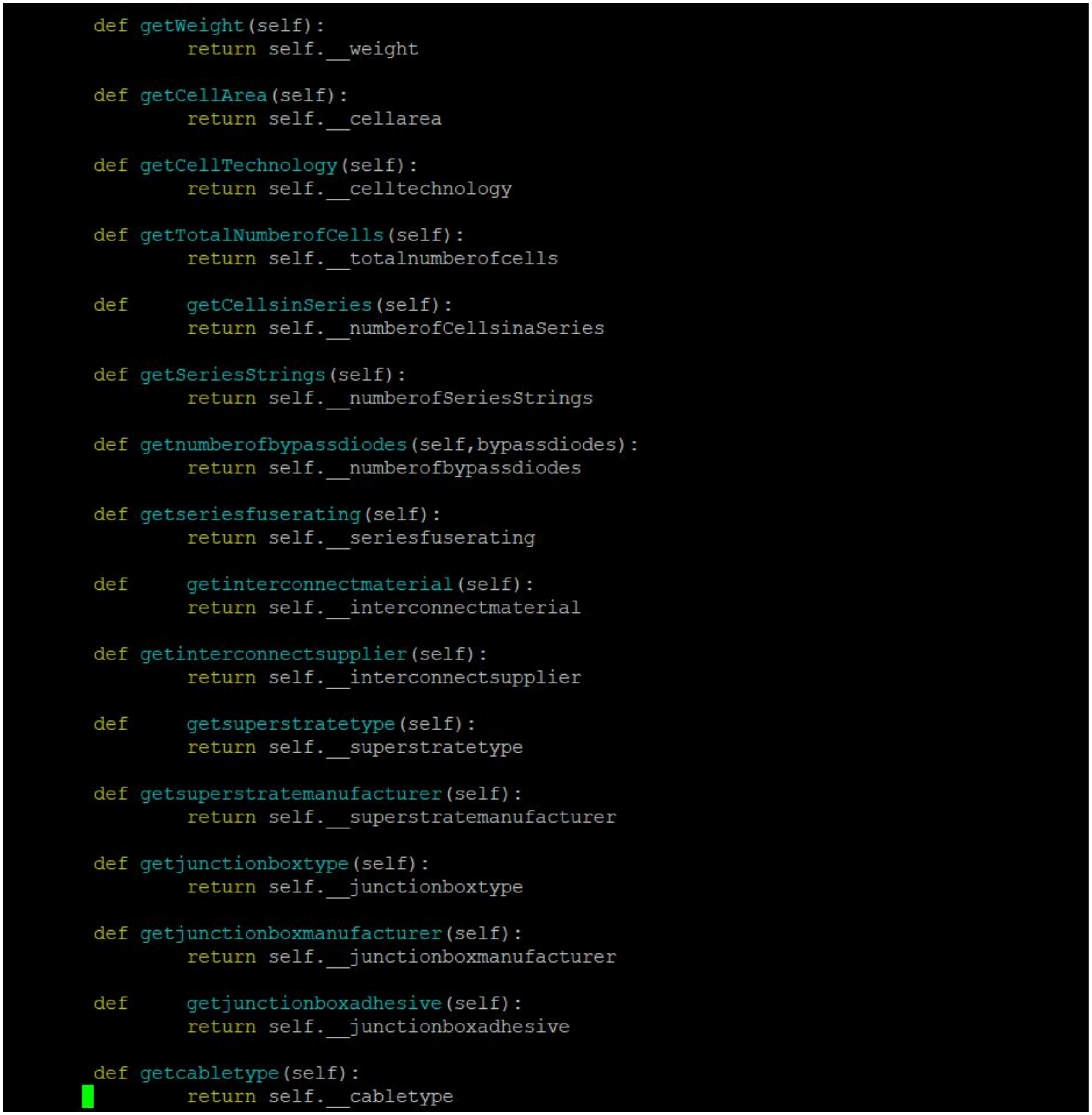
```
        def setLength(self, var4):
                self.__length = var4

        def setWidth(self, var5):
                self.__width = var5

        def setWeight(self, var6):
                self.__weight = var6

        def setCellArea(self, var7):
                self.__cellarea = var7

        def setCellTechnology(self, var8):
                self.__celltechnology = var8

        def setTotalNumberofCells(self, var9):
                self.__totalnumberofcells = var9

        def     setCellsinSeries(self, var10):
                self.__numberofCellsinaSeries = var10

        def setSeriesStrings(self, var11):
                self.__numberofSeriesStrings = var11

        def setnumberofbypassdiodes(self,var12):
                self.__numberofbypassdiodes = var12

        def setseriesfuserating(self, var13):
                self.__seriesfuserating = var13

        def     setinterconnectmaterial(self, var14):
                self.__interconnectmaterial = var14

        def setinterconnectsupplier(self, var15):
                self.__interconnectsupplier = var15

        def     setsuperstratetype(self, var16):
                self.__superstratetype = var16

        def setsuperstratemanufacturer(self, var17):
                self.__superstratemanufacturer = var17

        def setjunctionboxtype(self, var18):
                self.__junctionboxtype = var18

        def setjunctionboxmanufacturer(self, var19):
                self.__junctionboxmanufacturer = var19
```

Figure 6: Class Product Continued



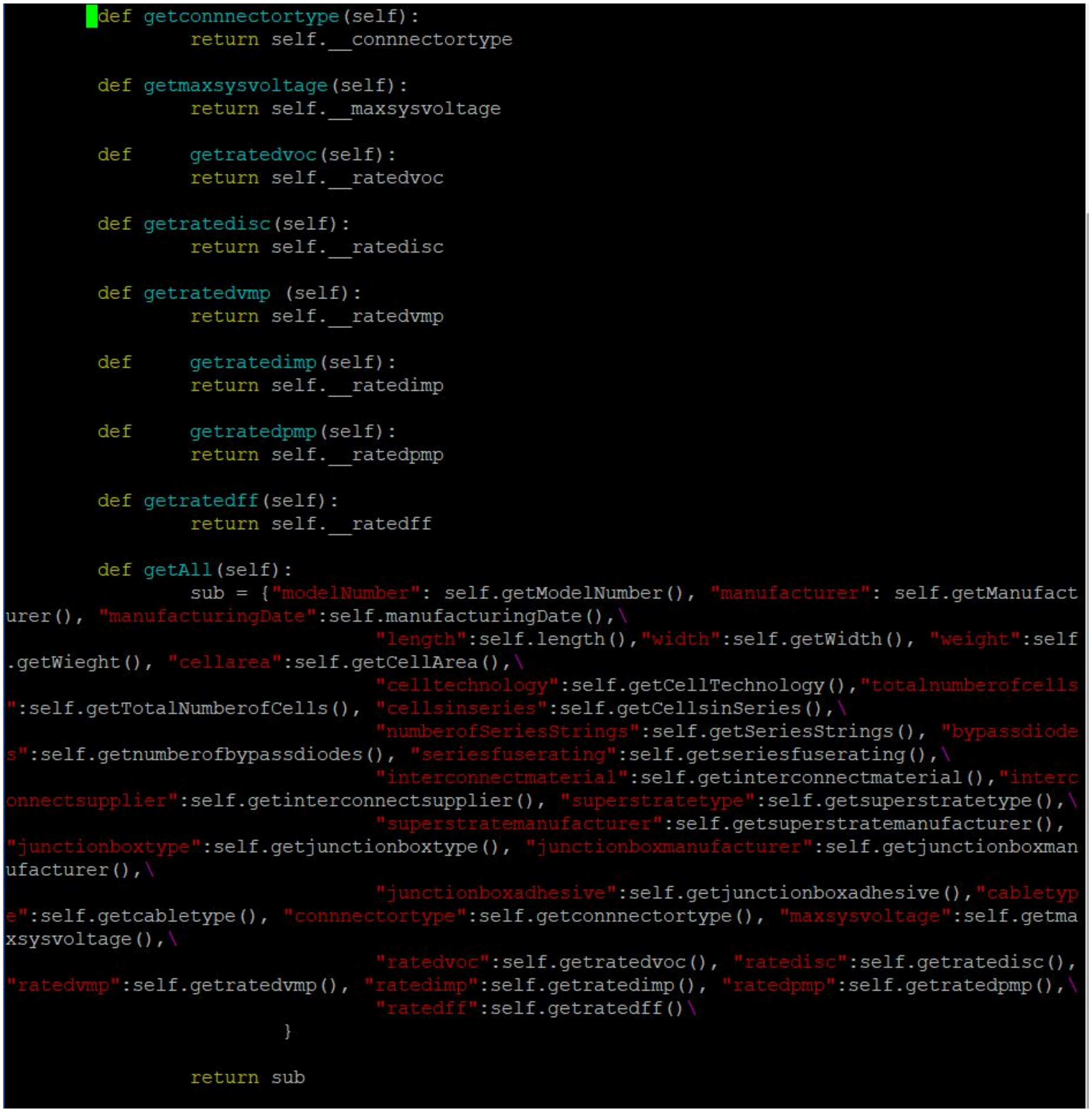
```
        def     setjunctionboxadhesive(self, var20):
                self.__junctionboxadhesive = var20

        def cabletype(self, var21):
                self.__cabletype = var21

        def setconnnectortype(self, var22):
                self.__connnectortype = var22

        def setmaxsysvoltage(self, var23):
                self.__maxsysvoltage = var23

        def     setratedvoc(self, var24):
                self.__ratedvoc = var24

        def setratedisc(self, var25):
                self.__ratedisc = var25

        def setratedvmp (self, var26):
                self.__ratedvmp  = var26

        def     setratedimp(self, var27):
                self.__ratedimp = var27

        def     setratedpmp(self, var28):
                self.__ratedpmp = var28

        def setratedff(self, var29):
                self.__ratedff = var29

################## Defining getters

        def getModelNumber(self):
                return self.__modelnumber

        def getManufacturer(self):
                return self.__manufacturer

        def getManufacturingDate(self):
                return self.__manufacturingdate

        def getLength(self):
                return self.__length

        def getWidth(self):
                return self.__width
```

Figure 7: Class Product Continued



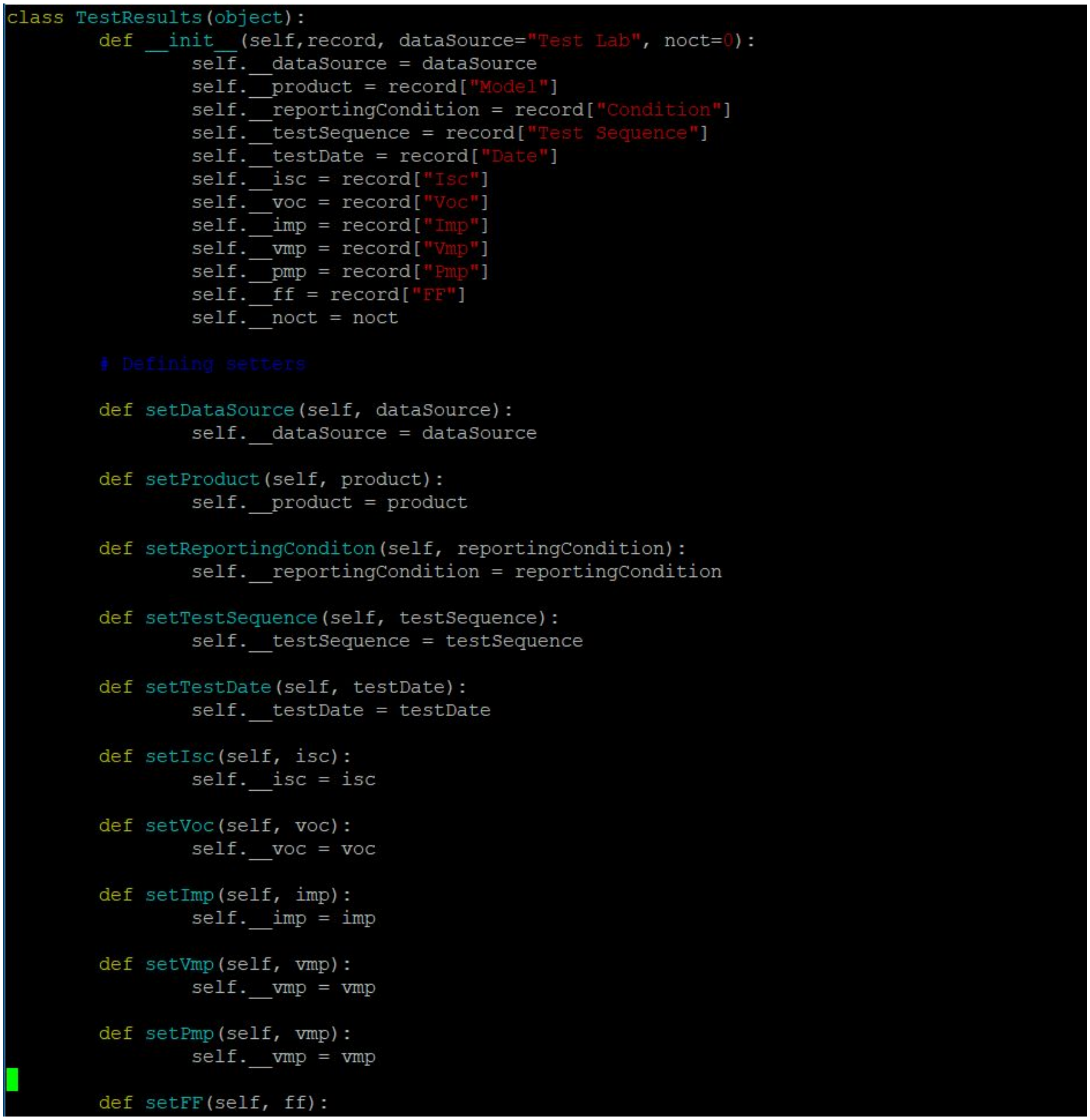
```
    def getWeight(self):
            return self.__weight

    def getCellArea(self):
            return self.__cellarea

    def getCellTechnology(self):
            return self.__celltechnology

    def getTotalNumberofCells(self):
            return self.__totalnumberofcells

    def     getCellsinSeries(self):
            return self.__numberofCellsinaSeries

    def getSeriesStrings(self):
            return self.__numberofSeriesStrings

    def getnumberofbypassdiodes(self,bypassdiodes):
            return self.__numberofbypassdiodes

    def getseriesfuserating(self):
            return self.__seriesfuserating

    def     getinterconnectmaterial(self):
            return self.__interconnectmaterial

    def getinterconnectsupplier(self):
            return self.__interconnectsupplier

    def     getsuperstratetype(self):
            return self.__superstratetype

    def getsuperstratemanufacturer(self):
            return self.__superstratemanufacturer

    def getjunctionboxtype(self):
            return self.__junctionboxtype

    def getjunctionboxmanufacturer(self):
            return self.__junctionboxmanufacturer

    def     getjunctionboxadhesive(self):
            return self.__junctionboxadhesive

    def getcabletype(self):
            return self.__cabletype
```

Figure 8: Class Product Continued



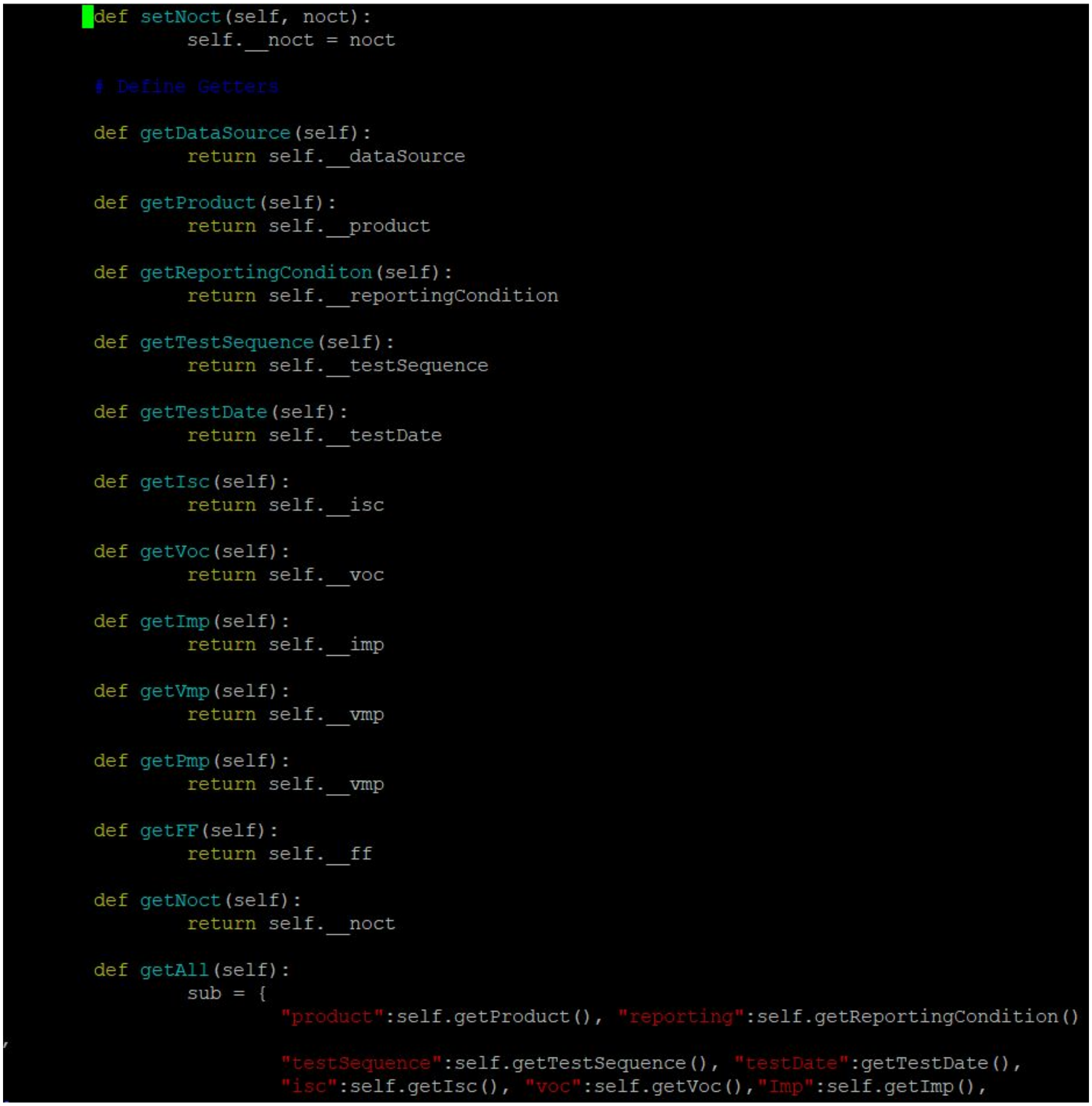
```
        def getconnnectortype(self):
                return self.__connnectortype

        def getmaxsysvoltage(self):
                return self.__maxsysvoltage

        def     getratedvoc(self):
                return self.__ratedvoc

        def getratedisc(self):
                return self.__ratedisc

        def getratedvmp (self):
                return self.__ratedvmp

        def     getratedimp(self):
                return self.__ratedimp

        def     getratedpmp(self):
                return self.__ratedpmp

        def getratedff(self):
                return self.__ratedff

        def getAll(self):
                sub = {"modelNumber": self.getModelNumber(), "manufacturer": self.getManufact
urer(), "manufacturingDate":self.manufacturingDate(),\
                                "length":self.length(),"width":self.getWidth(), "weight":self
.getWieght(), "cellarea":self.getCellArea(),\
                                "celltechnology":self.getCellTechnology(),"totalnumberofcells
":self.getTotalNumberofCells(), "cellsinseries":self.getCellsinSeries(),\
                                "numberofSeriesStrings":self.getSeriesStrings(), "bypassdiode
s":self.getnumberofbypassdiodes(), "seriesfuserating":self.getseriesfuserating(),\
                                "interconnectmaterial":self.getinterconnectmaterial(),"interc
onnectsupplier":self.getinterconnectsupplier(), "superstratetype":self.getsuperstratetype(),\
                                "superstratemanufacturer":self.getsuperstratemanufacturer(),
"junctionboxtype":self.getjunctionboxtype(), "junctionboxmanufacturer":self.getjunctionboxman
ufacturer(),\
                                "junctionboxadhesive":self.getjunctionboxadhesive(),"cabletyp
e":self.getcabletype(), "connnectortype":self.getconnnectortype(), "maxsysvoltage":self.getma
xsysvoltage(),\
                                "ratedvoc":self.getratedvoc(), "ratedisc":self.getratedisc(),
"ratedvmp":self.getratedvmp(), "ratedimp":self.getratedimp(), "ratedpmp":self.getratedpmp(),\
                                "ratedff":self.getratedff()\
                        }

                return sub
```

Figure 9: Class Test Results



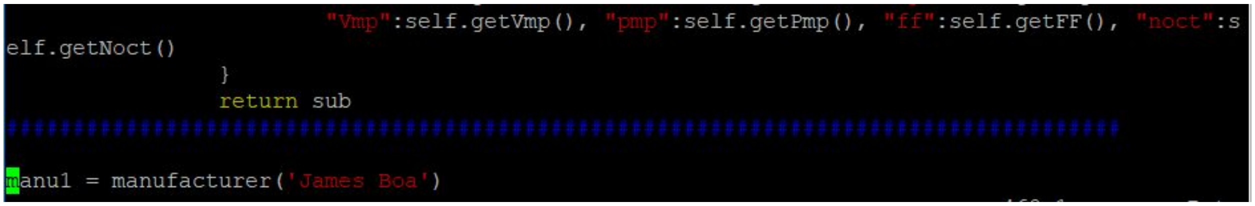
```
class TestResults(object):
        def __init__(self,record, dataSource="Test Lab", noct=0):
                self.__dataSource = dataSource
                self.__product = record["Model"]
                self.__reportingCondition = record["Condition"]
                self.__testSequence = record["Test Sequence"]
                self.__testDate = record["Date"]
                self.__isc = record["Isc"]
                self.__voc = record["Voc"]
                self.__imp = record["Imp"]
                self.__vmp = record["Vmp"]
                self.__pmp = record["Pmp"]
                self.__ff = record["FF"]
                self.__noct = noct

        # Defining setters

        def setDataSource(self, dataSource):
                self.__dataSource = dataSource

        def setProduct(self, product):
                self.__product = product

        def setReportingConditon(self, reportingCondition):
                self.__reportingCondition = reportingCondition

        def setTestSequence(self, testSequence):
                self.__testSequence = testSequence

        def setTestDate(self, testDate):
                self.__testDate = testDate

        def setIsc(self, isc):
                self.__isc = isc

        def setVoc(self, voc):
                self.__voc = voc

        def setImp(self, imp):
                self.__imp = imp

        def setVmp(self, vmp):
                self.__vmp = vmp

        def setPmp(self, vmp):
                self.__vmp = vmp

        def setFF(self, ff):
```

Figure 10: Class TestResults Continued



```
        def setNoct(self, noct):
                self.__noct = noct

        # Define Getters

        def getDataSource(self):
                return self.__dataSource

        def getProduct(self):
                return self.__product

        def getReportingConditon(self):
                return self.__reportingCondition

        def getTestSequence(self):
                return self.__testSequence

        def getTestDate(self):
                return self.__testDate

        def getIsc(self):
                return self.__isc

        def getVoc(self):
                return self.__voc

        def getImp(self):
                return self.__imp

        def getVmp(self):
                return self.__vmp

        def getPmp(self):
                return self.__vmp

        def getFF(self):
                return self.__ff

        def getNoct(self):
                return self.__noct

        def getAll(self):
                sub = {
                        "product":self.getProduct(), "reporting":self.getReportingCondition()
,
                        "testSequence":self.getTestSequence(), "testDate":getTestDate(),
                        "isc":self.getIsc(), "voc":self.getVoc(),"Imp":self.getImp(),
```

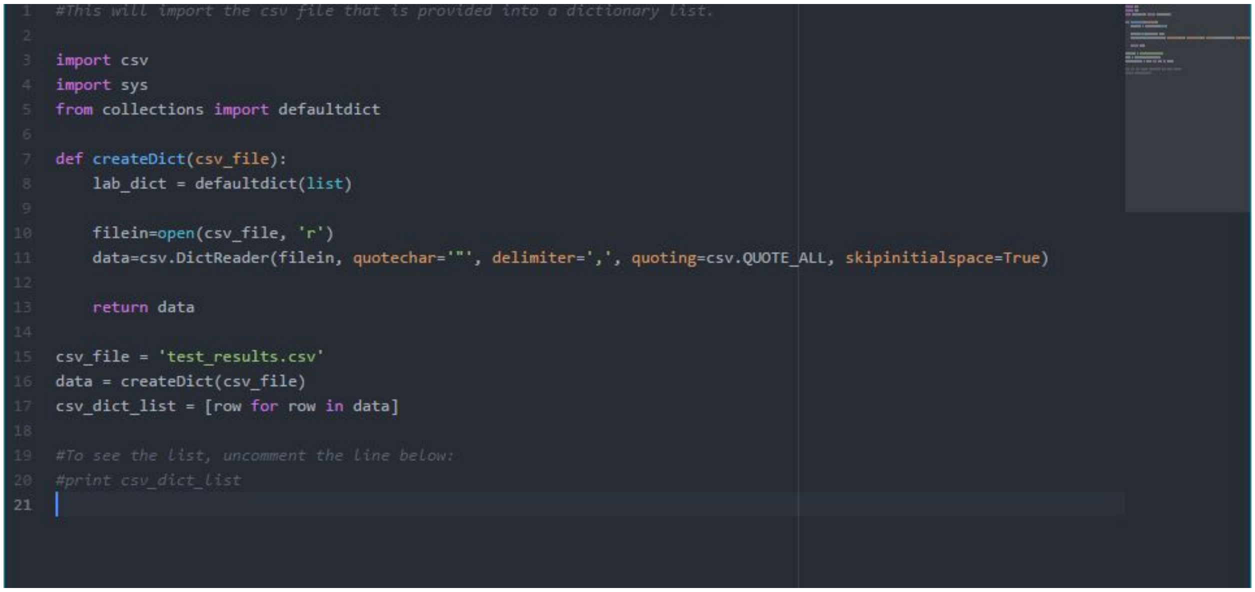
Figure 11: Class TestResults Continued



```
                        "Vmp":self.getVmp(), "pmp":self.getPmp(), "ff":self.getFF(), "noct":s
elf.getNoct()
                }
                return sub
##################################################################################

manu1 = manufacturer('James Boa')
```

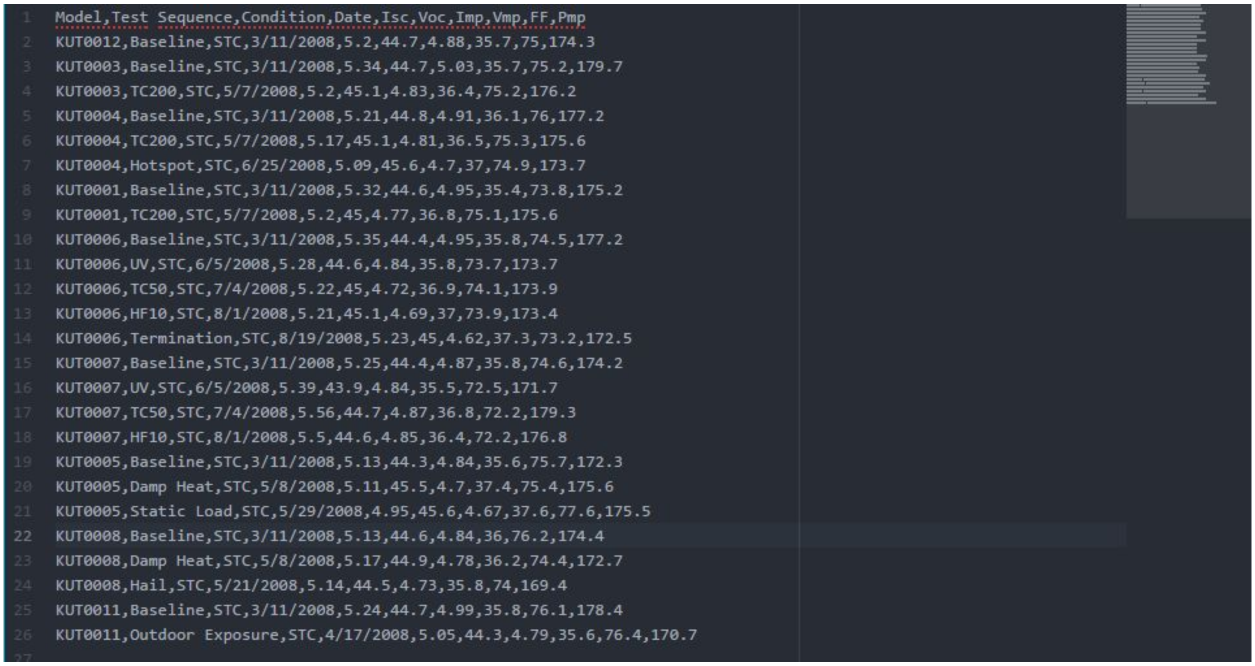
B. Create a new file called **pd3.py** for the tasks below:

1) We'll assume that test labs will upload the test results in csv files. So we need a function that takes as argument a csv file, reads and returns the contents of those files. A sample CSV test results file for use in this PD is attached.



```python
#This will import the csv file that is provided into a dictionary list.

import csv
import sys
from collections import defaultdict

def createDict(csv_file):
    lab_dict = defaultdict(list)

    filein=open(csv_file, 'r')
    data=csv.DictReader(filein, quotechar='"', delimiter=',', quoting=csv.QUOTE_ALL, skipinitialspace=True)

    return data

csv_file = 'test_results.csv'
data = createDict(csv_file)
csv_dict_list = [row for row in data]

#To see the list, uncomment the line below:
#print csv_dict_list

```

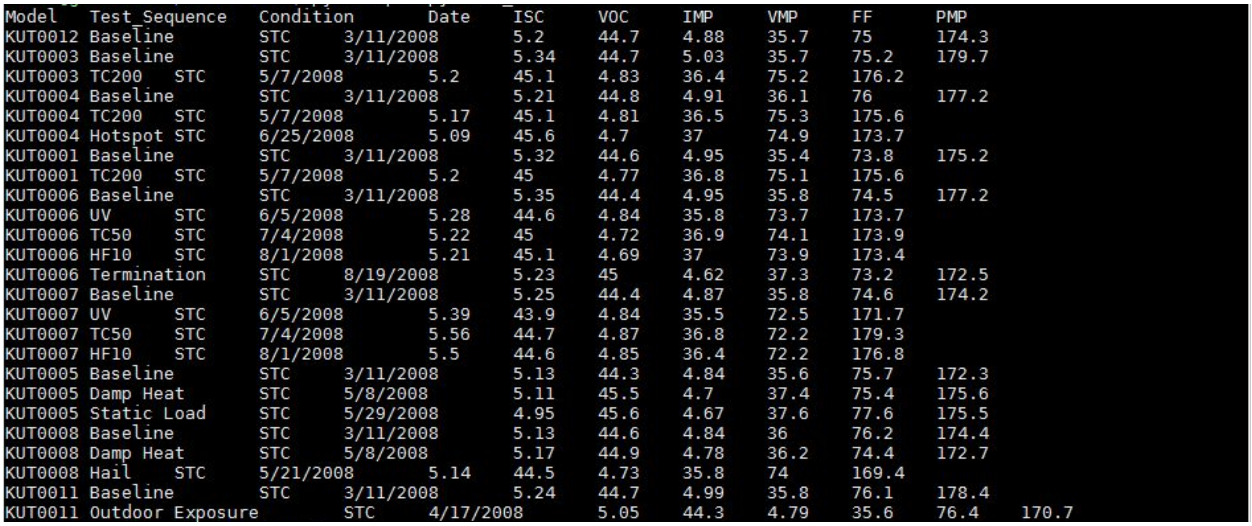
This section of pd3.py file will take a csv file provided and convert it into a dictionary list so that migrating data will be simple when we need to import this data into the mySQL server that was created in our previous project deliverable. The data this will import looks like the screenshot below:



```
Model,Test Sequence,Condition,Date,Isc,Voc,Imp,Vmp,FF,Pmp
KUT0012,Baseline,STC,3/11/2008,5.2,44.7,4.88,35.7,75,174.3
KUT0003,Baseline,STC,3/11/2008,5.34,44.7,5.03,35.7,75.2,179.7
KUT0003,TC200,STC,5/7/2008,5.2,45.1,4.83,36.4,75.2,176.2
KUT0004,Baseline,STC,3/11/2008,5.21,44.8,4.91,36.1,76,177.2
KUT0004,TC200,STC,5/7/2008,5.17,45.1,4.81,36.5,75.3,175.6
KUT0004,Hotspot,STC,6/25/2008,5.09,45.6,4.7,37,74.9,173.7
KUT0001,Baseline,STC,3/11/2008,5.32,44.6,4.95,35.4,73.8,175.2
KUT0001,TC200,STC,5/7/2008,5.2,45,4.77,36.8,75.1,175.6
KUT0006,Baseline,STC,3/11/2008,5.35,44.4,4.95,35.8,74.5,177.2
KUT0006,UV,STC,6/5/2008,5.28,44.6,4.84,35.8,73.7,173.7
KUT0006,TC50,STC,7/4/2008,5.22,45,4.72,36.9,74.1,173.9
KUT0006,HF10,STC,8/1/2008,5.21,45.1,4.69,37,73.9,173.4
KUT0006,Termination,STC,8/19/2008,5.23,45,4.62,37.3,73.2,172.5
KUT0007,Baseline,STC,3/11/2008,5.25,44.4,4.87,35.8,74.6,174.2
KUT0007,UV,STC,6/5/2008,5.39,43.9,4.84,35.5,72.5,171.7
KUT0007,TC50,STC,7/4/2008,5.56,44.7,4.87,36.8,72.2,179.3
KUT0007,HF10,STC,8/1/2008,5.5,44.6,4.85,36.4,72.2,176.8
KUT0005,Baseline,STC,3/11/2008,5.13,44.3,4.84,35.6,75.7,172.3
KUT0005,Damp Heat,STC,5/8/2008,5.11,45.5,4.7,37.4,75.4,175.6
KUT0005,Static Load,STC,5/29/2008,4.95,45.6,4.67,37.6,77.6,175.5
KUT0008,Baseline,STC,3/11/2008,5.13,44.6,4.84,36,76.2,174.4
KUT0008,Damp Heat,STC,5/8/2008,5.17,44.9,4.78,36.2,74.4,172.7
KUT0008,Hail,STC,5/21/2008,5.14,44.5,4.73,35.8,74,169.4
KUT0011,Baseline,STC,3/11/2008,5.24,44.7,4.99,35.8,76.1,178.4
KUT0011,Outdoor Exposure,STC,4/17/2008,5.05,44.3,4.79,35.6,76.4,170.7
```

Our script will take the first line as the dictionary keys and the corresponding datas will be placed into a list according to the dictionary key.

This data is then displayed in a loop with headings to make it easier for users to visualize the data from the csv file. Below is an example output.



```
Model   Test_Sequence   Condition       Date    ISC     VOC     IMP     VMP     FF      PMP
KUT0012 Baseline        STC     3/11/2008       5.2     44.7    4.88    35.7    75      174.3
KUT0003 Baseline        STC     3/11/2008       5.34    44.7    5.03    35.7    75.2    179.7
KUT0003 TC200   STC     5/7/2008        5.2     45.1    4.83    36.4    75.2    176.2
KUT0004 Baseline        STC     3/11/2008       5.21    44.8    4.91    36.1    76      177.2
KUT0004 TC200   STC     5/7/2008        5.17    45.1    4.81    36.5    75.3    175.6
KUT0004 Hotspot STC     6/25/2008       5.09    45.6    4.7     37      74.9    173.7
KUT0001 Baseline        STC     3/11/2008       5.32    44.6    4.95    35.4    73.8    175.2
KUT0001 TC200   STC     5/7/2008        5.2     45      4.77    36.8    75.1    175.6
KUT0006 Baseline        STC     3/11/2008       5.35    44.4    4.95    35.8    74.5    177.2
KUT0006 UV      STC     6/5/2008        5.28    44.6    4.84    35.8    73.7    173.7
KUT0006 TC50    STC     7/4/2008        5.22    45      4.72    36.9    74.1    173.9
KUT0006 HF10    STC     8/1/2008        5.21    45.1    4.69    37      73.9    173.4
KUT0006 Termination     STC     8/19/2008       5.23    45      4.62    37.3    73.2    172.5
KUT0007 Baseline        STC     3/11/2008       5.25    44.4    4.87    35.8    74.6    174.2
KUT0007 UV      STC     6/5/2008        5.39    43.9    4.84    35.5    72.5    171.7
KUT0007 TC50    STC     7/4/2008        5.56    44.7    4.87    36.8    72.2    179.3
KUT0007 HF10    STC     8/1/2008        5.5     44.6    4.85    36.4    72.2    176.8
KUT0005 Baseline        STC     3/11/2008       5.13    44.3    4.84    35.6    75.7    172.3
KUT0005 Damp Heat       STC     5/8/2008        5.11    45.5    4.7     37.4    75.4    175.6
KUT0005 Static Load     STC     5/29/2008       4.95    45.6    4.67    37.6    77.6    175.5
KUT0008 Baseline        STC     3/11/2008       5.13    44.6    4.84    36      76.2    174.4
KUT0008 Damp Heat       STC     5/8/2008        5.17    44.9    4.78    36.2    74.4    172.7
KUT0008 Hail    STC     5/21/2008       5.14    44.5    4.73    35.8    74      169.4
KUT0011 Baseline        STC     3/11/2008       5.24    44.7    4.99    35.8    76.1    178.4
KUT0011 Outdoor Exposure        STC     4/17/2008       5.05    44.3    4.79    35.6    76.4    170.7
```

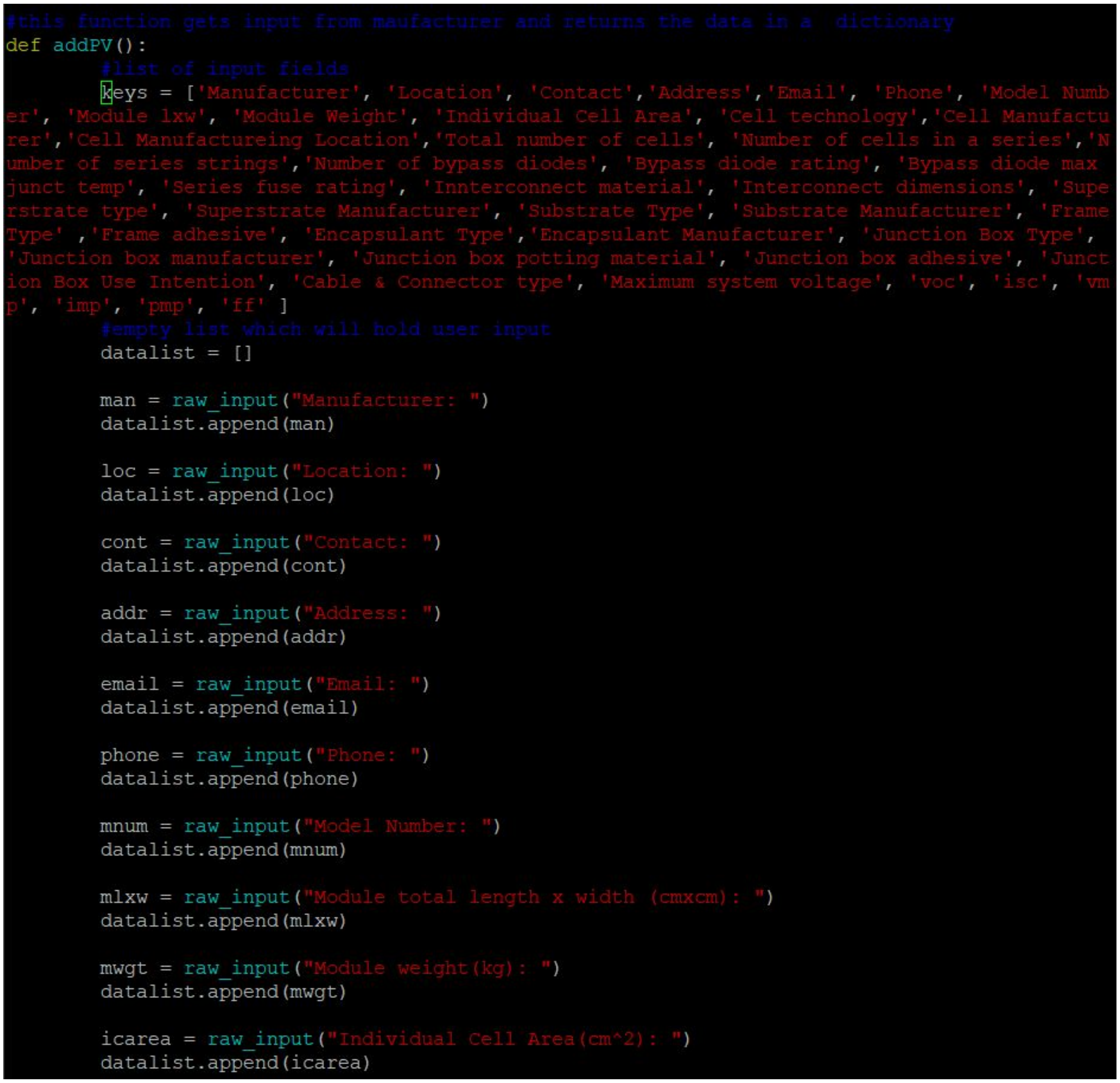
2) To add or register a new PV module, a manufacturer must fill out and submit a module detailed specifications (MDS) form. A sample to use in this PD is attached. Write a function that prompts the user to enter data from the MDS, then returns the dictionary of these data.

To accomplish creating a function that returns a dictionary of the user input in the MDS file, we first created a list of *keys* which contain the MDS input fields. We then created an empty list called "*datalist"*, which will contain a list of the users inputs in the MDS form. This was accomplished by prompting the user for input then appending the input from the user to the list: *datalist*. In order to create the dictionary of user input, we zipped both list together using the zip method. Lastly, we returned this dictionary to the user. This will allow the user to call the function addPV() and the user will retrieve the dictionary created by the function. *Figures 12 - 15*, shows the steps illustrated above.

Figure 16 demonstrates the execution of the function, taking the use input given in the MDS example form. *Figure 17* shows the proper output of the dictionary.

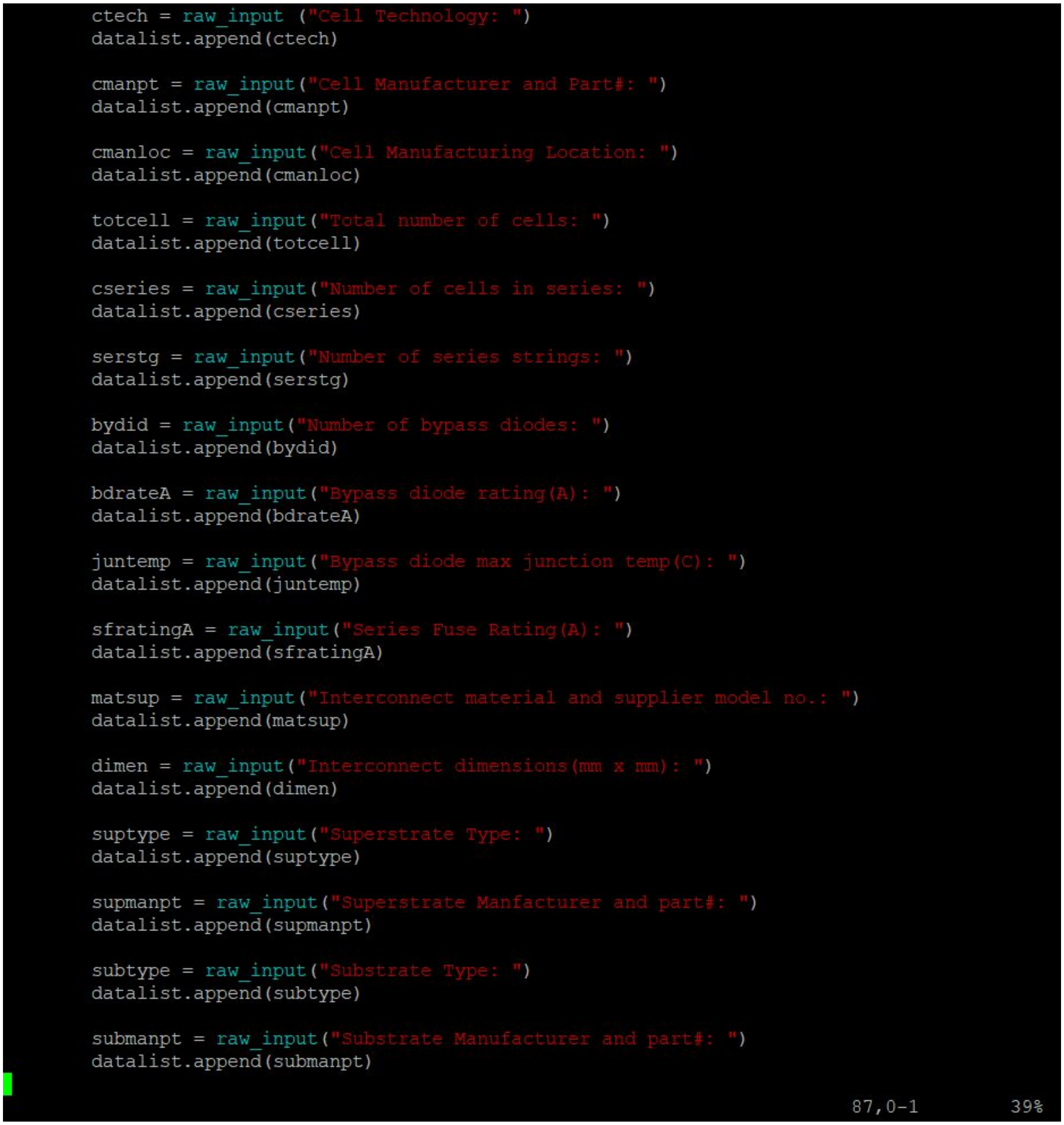
Note: The function is written independently to demonstrate proof of concept and proper execution, thus it contains its' own main() function. This is later removed when implemented in PD3.py.

Figure 12: addPV() function - MDS form



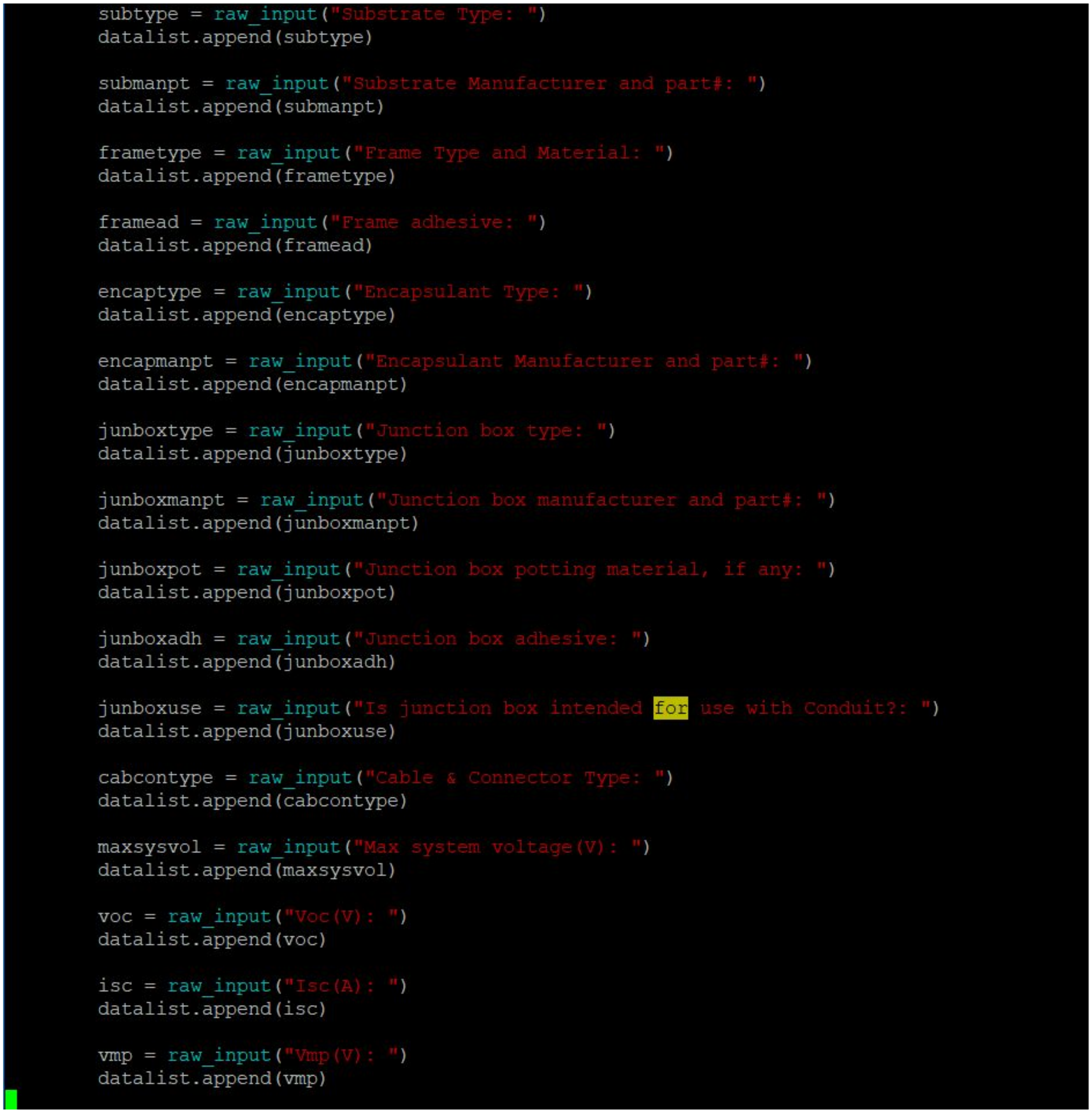
```
#this function gets input from maufacturer and returns the data in a  dictionary
def addPV():
        #list of input fields
        keys = ['Manufacturer', 'Location', 'Contact','Address','Email', 'Phone', 'Model Numb
er', 'Module lxw', 'Module Weight', 'Individual Cell Area', 'Cell technology','Cell Manufactu
rer','Cell Manufactureing Location','Total number of cells', 'Number of cells in a series','N
umber of series strings','Number of bypass diodes', 'Bypass diode rating', 'Bypass diode max
junct temp', 'Series fuse rating', 'Innterconnect material', 'Interconnect dimensions', 'Supe
rstrate type', 'Superstrate Manufacturer', 'Substrate Type', 'Substrate Manufacturer', 'Frame
Type' ,'Frame adhesive', 'Encapsulant Type','Encapsulant Manufacturer', 'Junction Box Type',
'Junction box manufacturer', 'Junction box potting material', 'Junction box adhesive', 'Junct
ion Box Use Intention', 'Cable & Connector type', 'Maximum system voltage', 'voc', 'isc', 'vm
p', 'imp', 'pmp', 'ff' ]
        #empty list which will hold user input
        datalist = []

        man = raw_input("Manufacturer: ")
        datalist.append(man)

        loc = raw_input("Location: ")
        datalist.append(loc)

        cont = raw_input("Contact: ")
        datalist.append(cont)

        addr = raw_input("Address: ")
        datalist.append(addr)

        email = raw_input("Email: ")
        datalist.append(email)

        phone = raw_input("Phone: ")
        datalist.append(phone)

        mnum = raw_input("Model Number: ")
        datalist.append(mnum)

        mlxw = raw_input("Module total length x width (cmxcm): ")
        datalist.append(mlxw)

        mwgt = raw_input("Module weight(kg): ")
        datalist.append(mwgt)

        icarea = raw_input("Individual Cell Area(cm^2): ")
        datalist.append(icarea)
```

Figure 13: addPV() function - MDS form



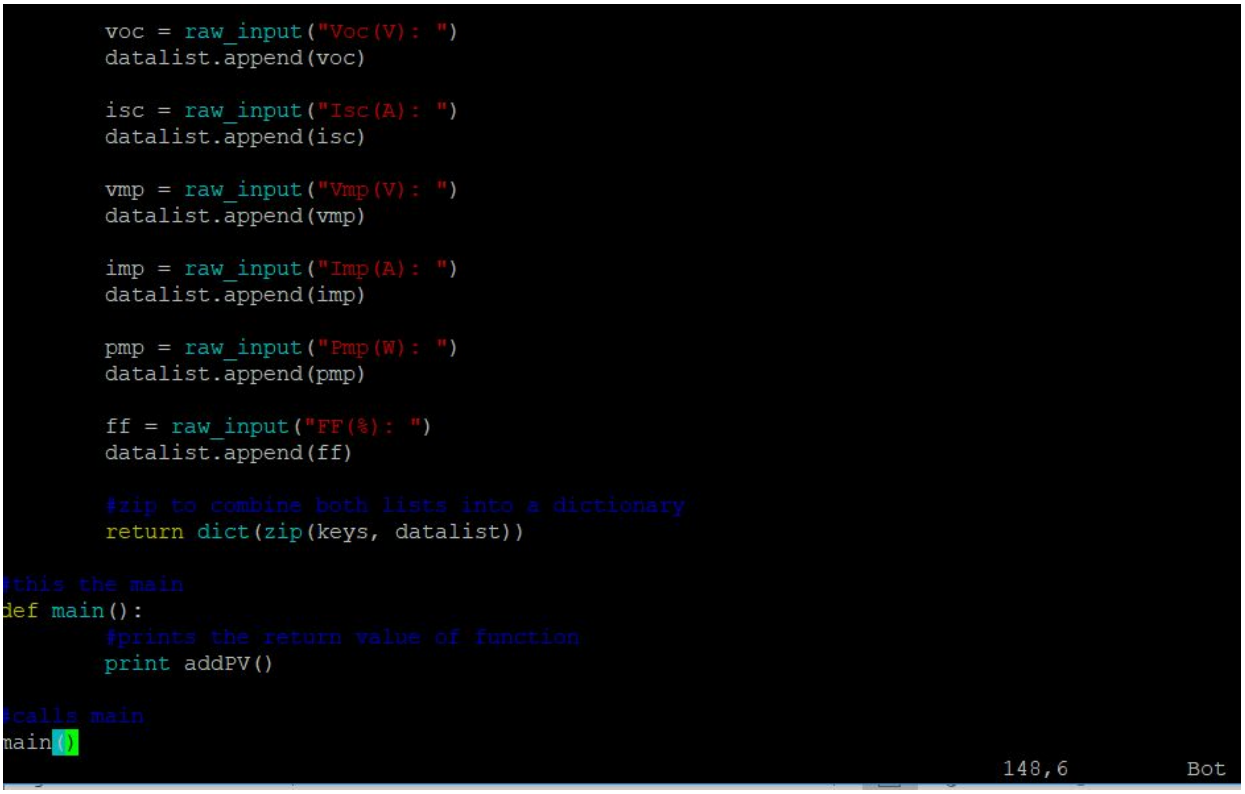
```
        ctech = raw_input ("Cell Technology: ")
        datalist.append(ctech)

        cmanpt = raw_input("Cell Manufacturer and Part#: ")
        datalist.append(cmanpt)

        cmanloc = raw_input("Cell Manufacturing Location: ")
        datalist.append(cmanloc)

        totcell = raw_input("Total number of cells: ")
        datalist.append(totcell)

        cseries = raw_input("Number of cells in series: ")
        datalist.append(cseries)

        serstg = raw_input("Number of series strings: ")
        datalist.append(serstg)

        bydid = raw_input("Number of bypass diodes: ")
        datalist.append(bydid)

        bdrateA = raw_input("Bypass diode rating(A): ")
        datalist.append(bdrateA)

        juntemp = raw_input("Bypass diode max junction temp(C): ")
        datalist.append(juntemp)

        sfratingA = raw_input("Series Fuse Rating(A): ")
        datalist.append(sfratingA)

        matsup = raw_input("Interconnect material and supplier model no.: ")
        datalist.append(matsup)

        dimen = raw_input("Interconnect dimensions(mm x mm): ")
        datalist.append(dimen)

        suptype = raw_input("Superstrate Type: ")
        datalist.append(suptype)

        supmanpt = raw_input("Superstrate Manfacturer and part#: ")
        datalist.append(supmanpt)

        subtype = raw_input("Substrate Type: ")
        datalist.append(subtype)

        submanpt = raw_input("Substrate Manufacturer and part#: ")
        datalist.append(submanpt)

                                                                          87,0-1         39%
```

Figure 14: addPV() function - MDS form Continued



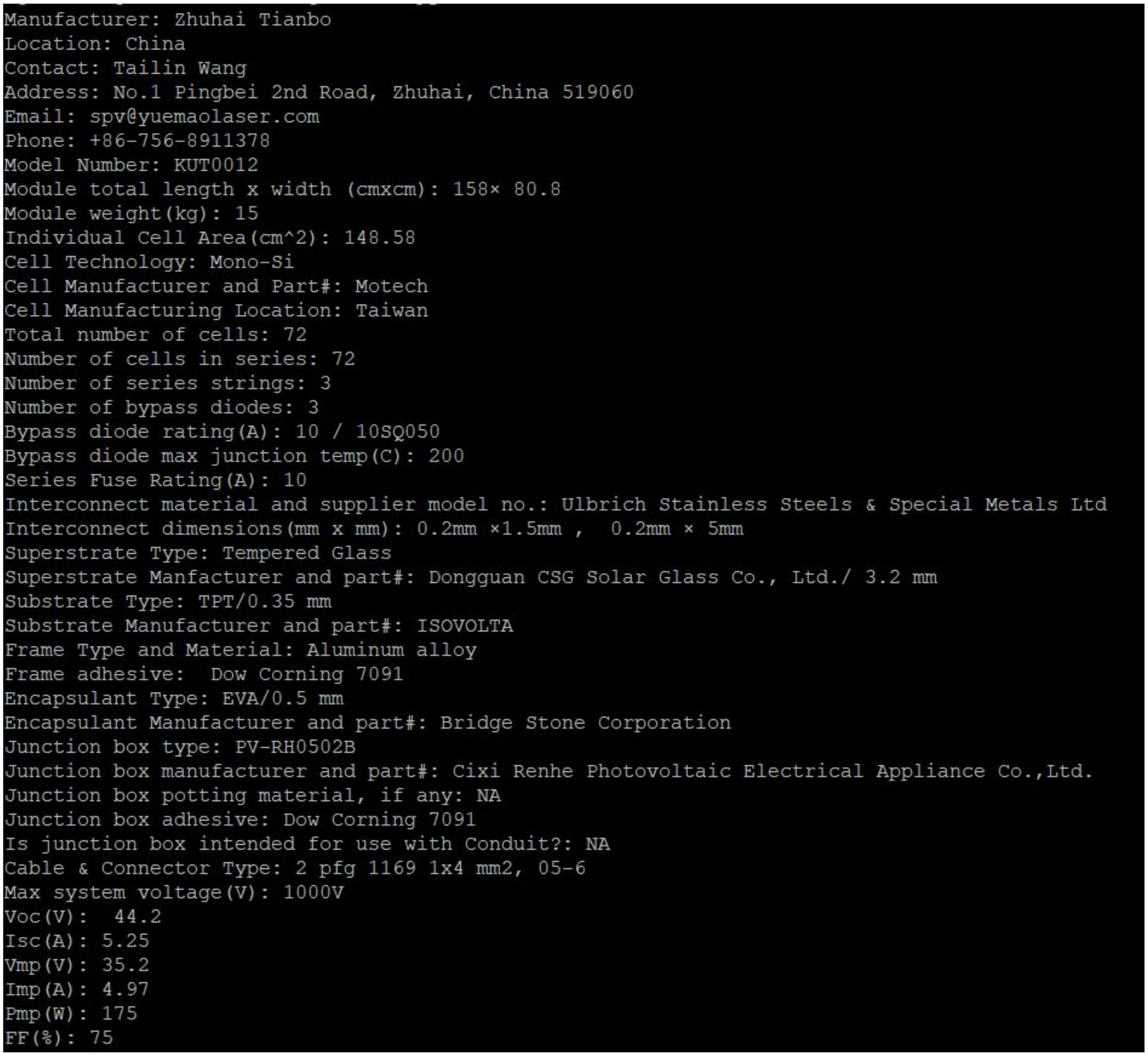
```
        subtype = raw_input("Substrate Type: ")
        datalist.append(subtype)

        submanpt = raw_input("Substrate Manufacturer and part#: ")
        datalist.append(submanpt)

        frametype = raw_input("Frame Type and Material: ")
        datalist.append(frametype)

        framead = raw_input("Frame adhesive: ")
        datalist.append(framead)

        encaptype = raw_input("Encapsulant Type: ")
        datalist.append(encaptype)

        encapmanpt = raw_input("Encapsulant Manufacturer and part#: ")
        datalist.append(encapmanpt)

        junboxtype = raw_input("Junction box type: ")
        datalist.append(junboxtype)

        junboxmanpt = raw_input("Junction box manufacturer and part#: ")
        datalist.append(junboxmanpt)

        junboxpot = raw_input("Junction box potting material, if any: ")
        datalist.append(junboxpot)

        junboxadh = raw_input("Junction box adhesive: ")
        datalist.append(junboxadh)

        junboxuse = raw_input("Is junction box intended for use with Conduit?: ")
        datalist.append(junboxuse)

        cabcontype = raw_input("Cable & Connector Type: ")
        datalist.append(cabcontype)

        maxsysvol = raw_input("Max system voltage(V): ")
        datalist.append(maxsysvol)

        voc = raw_input("Voc(V): ")
        datalist.append(voc)

        isc = raw_input("Isc(A): ")
        datalist.append(isc)

        vmp = raw_input("Vmp(V): ")
        datalist.append(vmp)
```

Figure 15: addPV() function - MDS for Continued



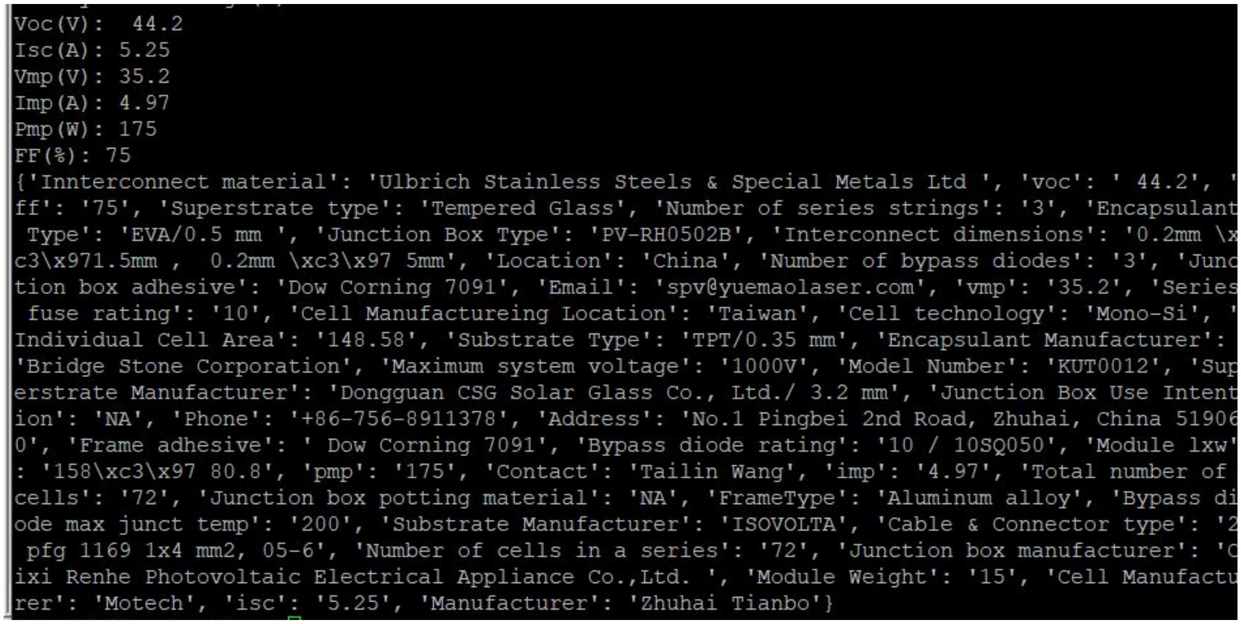
```
        voc = raw_input("Voc(V): ")
        datalist.append(voc)

        isc = raw_input("Isc(A): ")
        datalist.append(isc)

        vmp = raw_input("Vmp(V): ")
        datalist.append(vmp)

        imp = raw_input("Imp(A): ")
        datalist.append(imp)

        pmp = raw_input("Pmp(W): ")
        datalist.append(pmp)

        ff = raw_input("FF(%): ")
        datalist.append(ff)

        #zip to combine both lists into a dictionary
        return dict(zip(keys, datalist))

#this the main
def main():
        #prints the return value of function
        print addPV()

#calls main
main()
                                                              148,6          Bot
```

Figure 16: Execution of addPV() function



```
Manufacturer: Zhuhai Tianbo
Location: China
Contact: Tailin Wang
Address: No.1 Pingbei 2nd Road, Zhuhai, China 519060
Email: spv@yuemaolaser.com
Phone: +86-756-8911378
Model Number: KUT0012
Module total length x width (cmxcm): 158× 80.8
Module weight(kg): 15
Individual Cell Area(cm^2): 148.58
Cell Technology: Mono-Si
Cell Manufacturer and Part#: Motech
Cell Manufacturing Location: Taiwan
Total number of cells: 72
Number of cells in series: 72
Number of series strings: 3
Number of bypass diodes: 3
Bypass diode rating(A): 10 / 10SQ050
Bypass diode max junction temp(C): 200
Series Fuse Rating(A): 10
Interconnect material and supplier model no.: Ulbrich Stainless Steels & Special Metals Ltd
Interconnect dimensions(mm x mm): 0.2mm ×1.5mm ,  0.2mm × 5mm
Superstrate Type: Tempered Glass
Superstrate Manfacturer and part#: Dongguan CSG Solar Glass Co., Ltd./ 3.2 mm
Substrate Type: TPT/0.35 mm
Substrate Manufacturer and part#: ISOVOLTA
Frame Type and Material: Aluminum alloy
Frame adhesive:  Dow Corning 7091
Encapsulant Type: EVA/0.5 mm
Encapsulant Manufacturer and part#: Bridge Stone Corporation
Junction box type: PV-RH0502B
Junction box manufacturer and part#: Cixi Renhe Photovoltaic Electrical Appliance Co.,Ltd.
Junction box potting material, if any: NA
Junction box adhesive: Dow Corning 7091
Is junction box intended for use with Conduit?: NA
Cable & Connector Type: 2 pfg 1169 1x4 mm2, 05-6
Max system voltage(V): 1000V
Voc(V):  44.2
Isc(A): 5.25
Vmp(V): 35.2
Imp(A): 4.97
Pmp(W): 175
FF(%): 75
```

Figure 17: Output of addPV() function



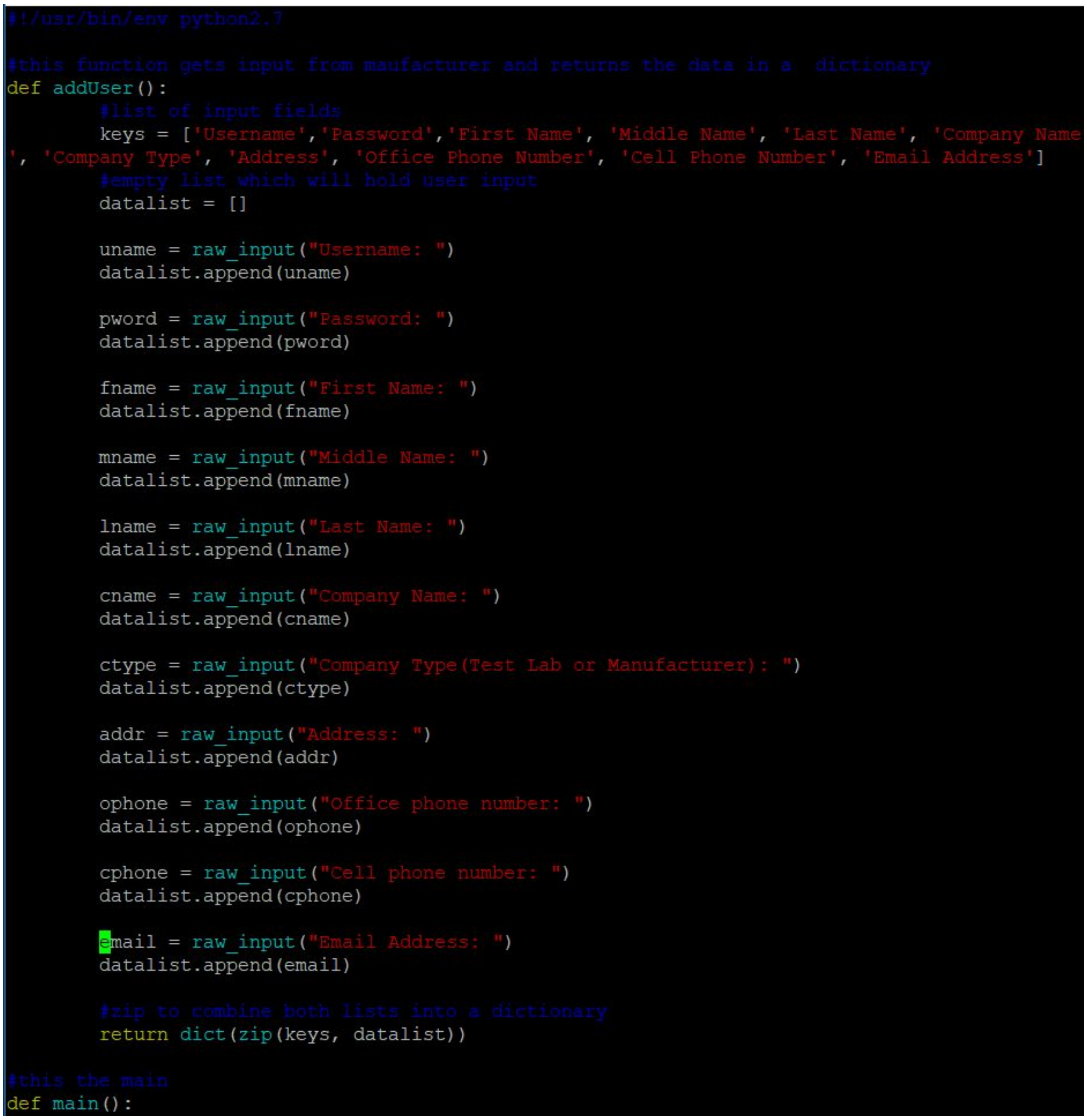
```
Voc(V):  44.2
Isc(A): 5.25
Vmp(V): 35.2
Imp(A): 4.97
Pmp(W): 175
FF(%): 75
{'Innterconnect material': 'Ulbrich Stainless Steels & Special Metals Ltd ', 'voc': ' 44.2', '
ff': '75', 'Superstrate type': 'Tempered Glass', 'Number of series strings': '3', 'Encapsulant
 Type': 'EVA/0.5 mm ', 'Junction Box Type': 'PV-RH0502B', 'Interconnect dimensions': '0.2mm \x
c3\x971.5mm ,  0.2mm \xc3\x97 5mm', 'Location': 'China', 'Number of bypass diodes': '3', 'Junc
tion box adhesive': 'Dow Corning 7091', 'Email': 'spv@yuemaolaser.com', 'vmp': '35.2', 'Series
 fuse rating': '10', 'Cell Manufactureing Location': 'Taiwan', 'Cell technology': 'Mono-Si', '
Individual Cell Area': '148.58', 'Substrate Type': 'TPT/0.35 mm', 'Encapsulant Manufacturer':
'Bridge Stone Corporation', 'Maximum system voltage': '1000V', 'Model Number': 'KUT0012', 'Sup
erstrate Manufacturer': 'Dongguan CSG Solar Glass Co., Ltd./ 3.2 mm', 'Junction Box Use Intent
ion': 'NA', 'Phone': '+86-756-8911378', 'Address': 'No.1 Pingbei 2nd Road, Zhuhai, China 51906
0', 'Frame adhesive': ' Dow Corning 7091', 'Bypass diode rating': '10 / 10SQ050', 'Module lxw'
: '158\xc3\x97 80.8', 'pmp': '175', 'Contact': 'Tailin Wang', 'imp': '4.97', 'Total number of
cells': '72', 'Junction box potting material': 'NA', 'FrameType': 'Aluminum alloy', 'Bypass di
ode max junct temp': '200', 'Substrate Manufacturer': 'ISOVOLTA', 'Cable & Connector type': '2
 pfg 1169 1x4 mm2, 05-6', 'Number of cells in a series': '72', 'Junction box manufacturer': 'C
ixi Renhe Photovoltaic Electrical Appliance Co.,Ltd. ', 'Module Weight': '15', 'Cell Manufactu
rer': 'Motech', 'isc': '5.25', 'Manufacturer': 'Zhuhai Tianbo'}
```

3) Each user must register with the portal. A user could be a Manufacturer, a Testing Lab, or any interested party. Write a function that takes a user registration and returns a dictionary of data. To register, a user provides the following information:

Similar to problem 2, to accomplish creating a function that returns a dictionary of the user input in the User Registration form, we first created a list of *keys* which contain the User Registration input fields. We then created an empty list called "*datalist"*, which will contain a list of the users inputs in the Registration form. This was accomplished by prompting the user for input then appending the input from the user to the list: *datalist*. In order to create the dictionary of user input, we zipped both list together using the zip method. Lastly, we returned this dictionary to the user. This will allow the user to call the function addUser() and the user will retrieve the dictionary created by the function. *Figure 18* shows the steps illustrated above. *Figure 19* demonstrates functionality of the addUser() function by outputting proper information.

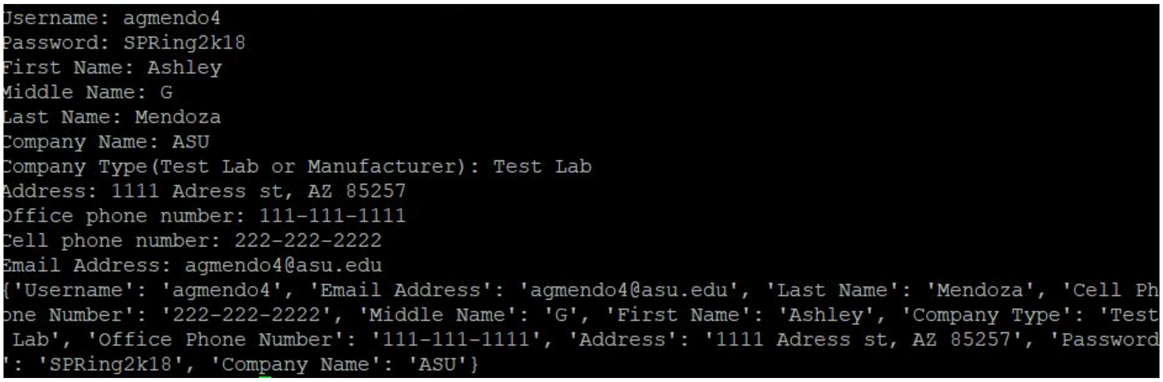
Note: The function is written independently to demonstrate proof of concept and proper execution, thus it contains its' own main() function. This is later removed when implemented in PD3.py.

Figure 18: Function addUser()



```
#!/usr/bin/env python2.7

#this function gets input from maufacturer and returns the data in a  dictionary
def addUser():
        #list of input fields
        keys = ['Username','Password','First Name', 'Middle Name', 'Last Name', 'Company Name
', 'Company Type', 'Address', 'Office Phone Number', 'Cell Phone Number', 'Email Address']
        #empty list which will hold user input
        datalist = []

        uname = raw_input("Username: ")
        datalist.append(uname)

        pword = raw_input("Password: ")
        datalist.append(pword)

        fname = raw_input("First Name: ")
        datalist.append(fname)

        mname = raw_input("Middle Name: ")
        datalist.append(mname)

        lname = raw_input("Last Name: ")
        datalist.append(lname)

        cname = raw_input("Company Name: ")
        datalist.append(cname)

        ctype = raw_input("Company Type(Test Lab or Manufacturer): ")
        datalist.append(ctype)

        addr = raw_input("Address: ")
        datalist.append(addr)

        ophone = raw_input("Office phone number: ")
        datalist.append(ophone)

        cphone = raw_input("Cell phone number: ")
        datalist.append(cphone)

        email = raw_input("Email Address: ")
        datalist.append(email)

        #zip to combine both lists into a dictionary
        return dict(zip(keys, datalist))

#this the main
def main():
```

Figure 19: Execution and Output of function addUser()



```
Username: agmendo4
Password: SPRing2k18
First Name: Ashley
Middle Name: G
Last Name: Mendoza
Company Name: ASU
Company Type(Test Lab or Manufacturer): Test Lab
Address: 1111 Adress st, AZ 85257
Office phone number: 111-111-1111
Cell phone number: 222-222-2222
Email Address: agmendo4@asu.edu
{'Username': 'agmendo4', 'Email Address': 'agmendo4@asu.edu', 'Last Name': 'Mendoza', 'Cell Ph
one Number': '222-222-2222', 'Middle Name': 'G', 'First Name': 'Ashley', 'Company Type': 'Test
 Lab', 'Office Phone Number': '111-111-1111', 'Address': '1111 Adress st, AZ 85257', 'Password
': 'SPRing2k18', 'Company Name': 'ASU'}
```

4) Main function:

a. Get the MDS data and instantiate:

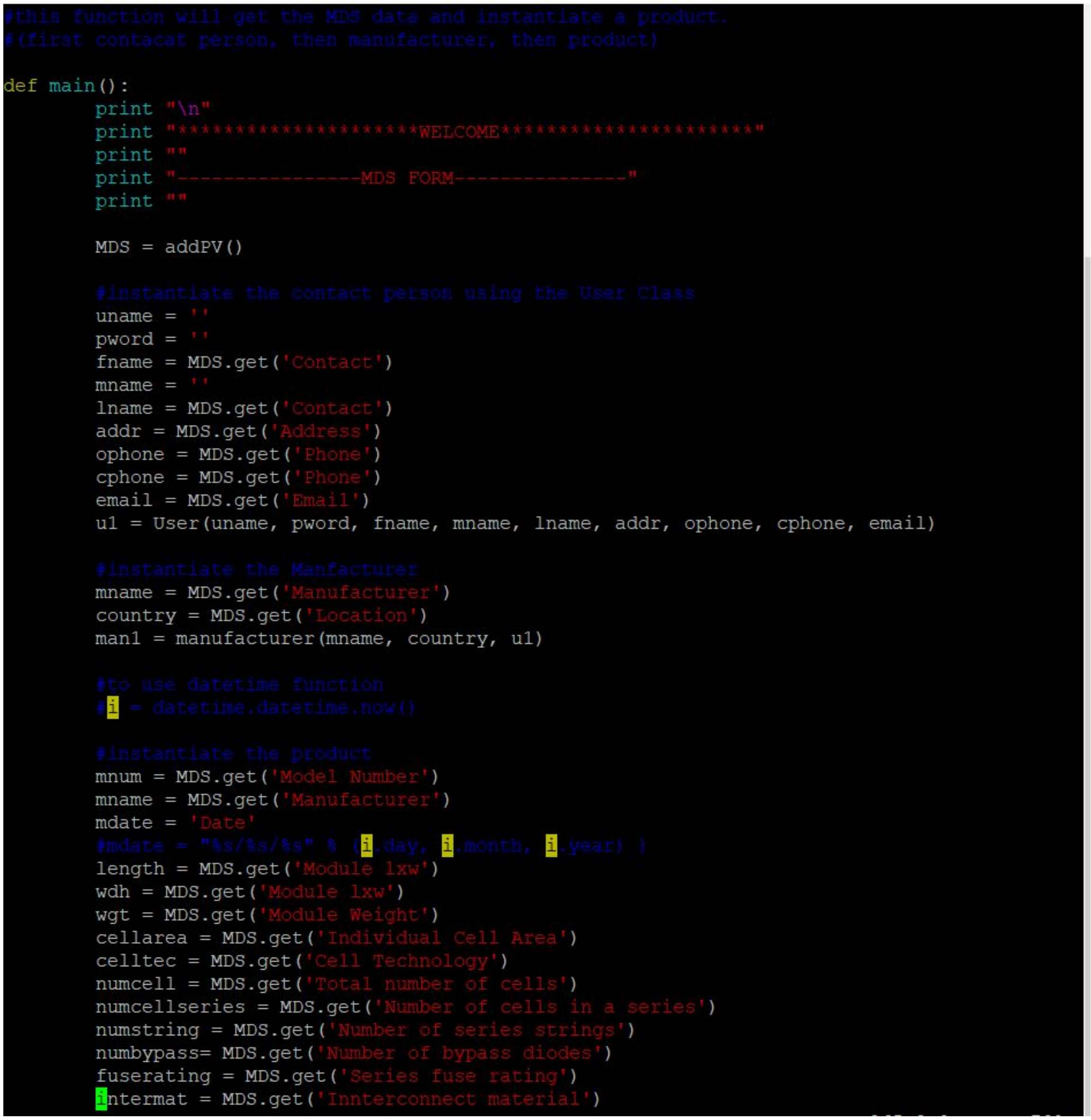
- A product (i.e. a new PV module) with the relevant data.
  Note that it would require you to instantiate a manufacturer; which in turn will also require you to instantiate a contact person.
- Display the following information about the product:
  Manufacturer name, Contact name, Contact Email, Model Number, Cell Technology, System voltage, Rated Power (Pmp)

In order to construct the main function outlined in problem 4a, we first had to instantiate a contact person using the MDS data. We created the contact person by using the class User from the myClasses.py file to initialize the object u1. Information not contained in the MDS form was passed an empty string to allow initialization. Next, we instantiated the manufacturer. Similar to user, we initialized it using the constructor, pass relevant data from the MDS form and the object u1 previously created. Finally, we instantiated the product using either the MDS form or manufacturer object and methods. Lastly, to display the proper information, we called the objects' getters to return proper values. Reference *Figure 20 - 21* for implementation and *Figures 23 -24* for output.

b. Get the test results data and display only the data for the test sequence “Baseline” on the screen.

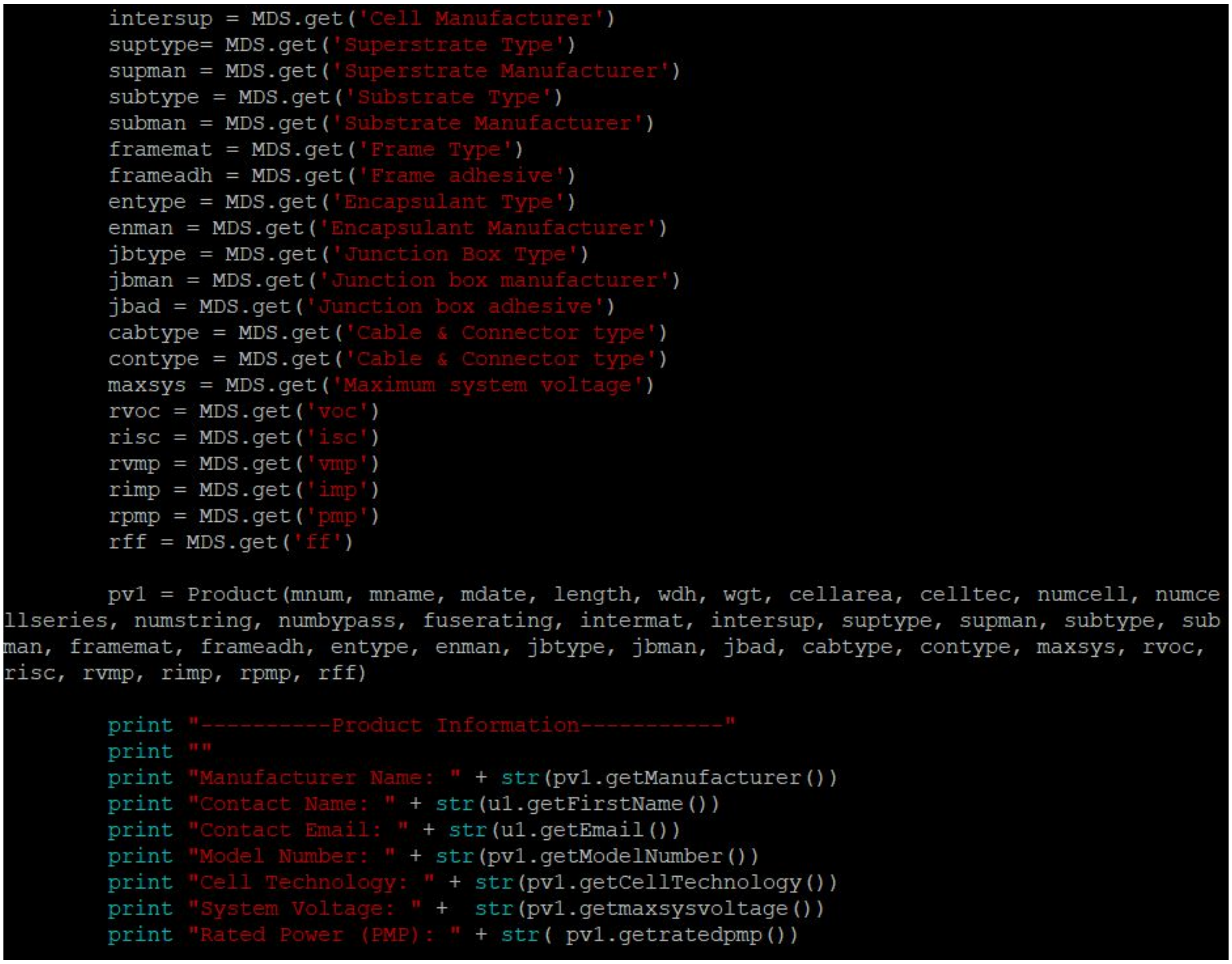
In order to print the Baseline results contained in the .csv file provided, we created a nest for loop which looked at every record with key ‘Baseline’ and printed the values specified using the methods in the TestResults class. The implementation is demonstrated in *Figure* 22 and its' output is shown in *Figures 24-26*.

Figure 20: main() function - Instantiations



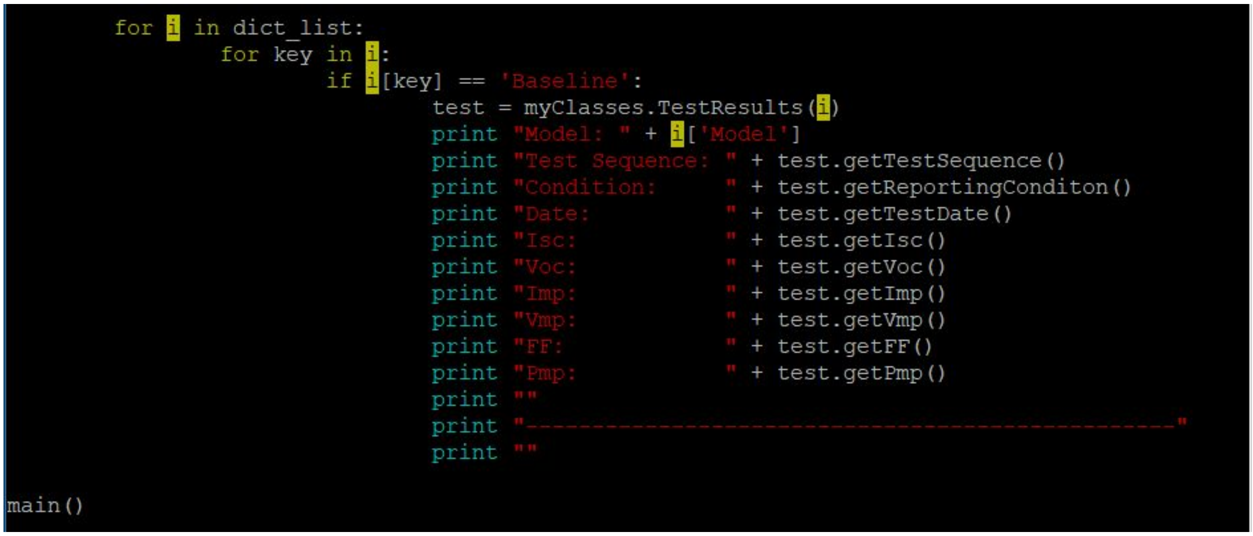
```
#this function will get the MDS data and instantiate a product.
#(first contacat person, then manufacturer, then product)

def main():
        print "\n"
        print "*********************WELCOME***********************"
        print ""
        print "----------------MDS FORM---------------"
        print ""

        MDS = addPV()

        #instantiate the contact person using the User Class
        uname = ''
        pword = ''
        fname = MDS.get('Contact')
        mname = ''
        lname = MDS.get('Contact')
        addr = MDS.get('Address')
        ophone = MDS.get('Phone')
        cphone = MDS.get('Phone')
        email = MDS.get('Email')
        u1 = User(uname, pword, fname, mname, lname, addr, ophone, cphone, email)

        #instantiate the Manfacturer
        mname = MDS.get('Manufacturer')
        country = MDS.get('Location')
        man1 = manufacturer(mname, country, u1)

        #to use datetime function
        #i = datetime.datetime.now()

        #instantiate the product
        mnum = MDS.get('Model Number')
        mname = MDS.get('Manufacturer')
        mdate = 'Date'
        #mdate = "%s/%s/%s" % (i.day, i.month, i.year) )
        length = MDS.get('Module lxw')
        wdh = MDS.get('Module lxw')
        wgt = MDS.get('Module Weight')
        cellarea = MDS.get('Individual Cell Area')
        celltec = MDS.get('Cell Technology')
        numcell = MDS.get('Total number of cells')
        numcellseries = MDS.get('Number of cells in a series')
        numstring = MDS.get('Number of series strings')
        numbypass= MDS.get('Number of bypass diodes')
        fuserating = MDS.get('Series fuse rating')
        intermat = MDS.get('Innterconnect material')
```

Figure 21: main() function - Instantiations Continued and Product Information



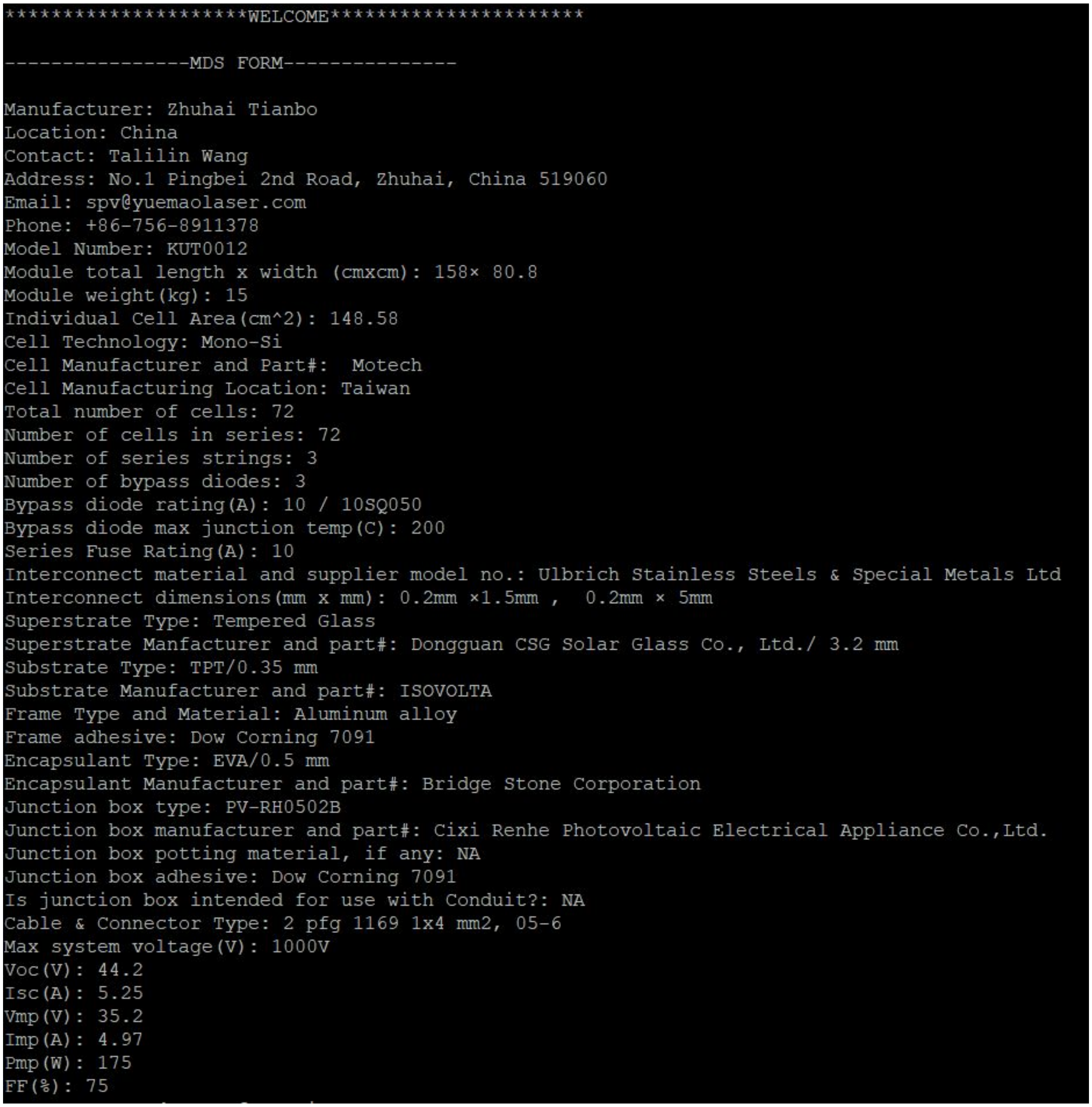
```
        intersup = MDS.get('Cell Manufacturer')
        suptype= MDS.get('Superstrate Type')
        supman = MDS.get('Superstrate Manufacturer')
        subtype = MDS.get('Substrate Type')
        subman = MDS.get('Substrate Manufacturer')
        framemat = MDS.get('Frame Type')
        frameadh = MDS.get('Frame adhesive')
        entype = MDS.get('Encapsulant Type')
        enman = MDS.get('Encapsulant Manufacturer')
        jbtype = MDS.get('Junction Box Type')
        jbman = MDS.get('Junction box manufacturer')
        jbad = MDS.get('Junction box adhesive')
        cabtype = MDS.get('Cable & Connector type')
        contype = MDS.get('Cable & Connector type')
        maxsys = MDS.get('Maximum system voltage')
        rvoc = MDS.get('voc')
        risc = MDS.get('isc')
        rvmp = MDS.get('vmp')
        rimp = MDS.get('imp')
        rpmp = MDS.get('pmp')
        rff = MDS.get('ff')

        pv1 = Product(mnum, mname, mdate, length, wdh, wgt, cellarea, celltec, numcell, numce
llseries, numstring, numbypass, fuserating, intermat, intersup, suptype, supman, subtype, sub
man, framemat, frameadh, entype, enman, jbtype, jbman, jbad, cabtype, contype, maxsys, rvoc,
risc, rvmp, rimp, rpmp, rff)

        print "----------Product Information-----------"
        print ""
        print "Manufacturer Name: " + str(pv1.getManufacturer())
        print "Contact Name: " + str(u1.getFirstName())
        print "Contact Email: " + str(u1.getEmail())
        print "Model Number: " + str(pv1.getModelNumber())
        print "Cell Technology: " + str(pv1.getCellTechnology())
        print "System Voltage: " +  str(pv1.getmaxsysvoltage())
        print "Rated Power (PMP): " + str( pv1.getratedpmp())
```

Figure 22: main() function - Baseline Test Results



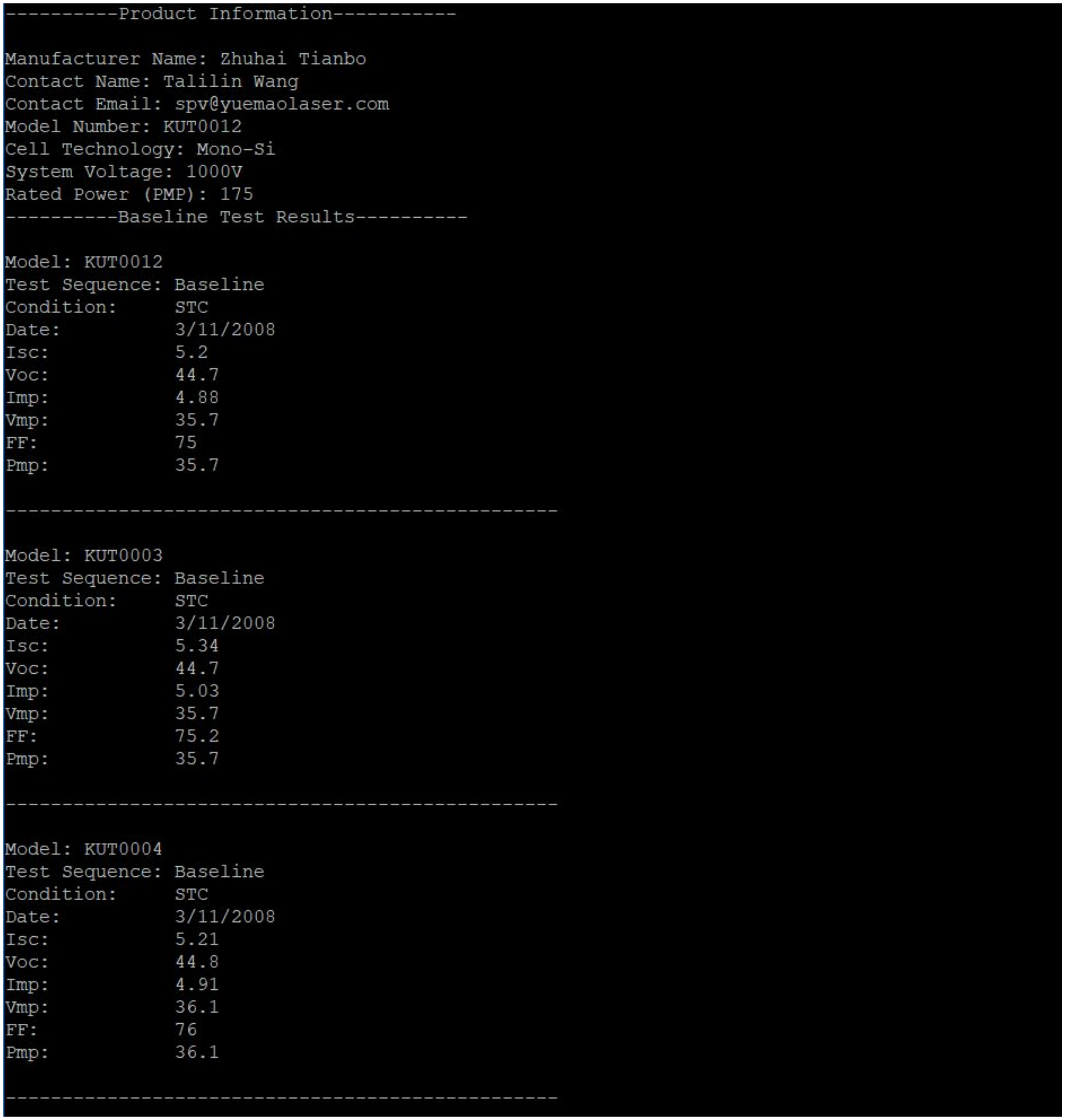
```
        for i in dict_list:
                for key in i:
                        if i[key] == 'Baseline':
                                test = myClasses.TestResults(i)
                                print "Model: " + i['Model']
                                print "Test Sequence: " + test.getTestSequence()
                                print "Condition:     " + test.getReportingConditon()
                                print "Date:          " + test.getTestDate()
                                print "Isc:           " + test.getIsc()
                                print "Voc:           " + test.getVoc()
                                print "Imp:           " + test.getImp()
                                print "Vmp:           " + test.getVmp()
                                print "FF:            " + test.getFF()
                                print "Pmp:           " + test.getPmp()
                                print ""
                                print "-------------------------------------------------"
                                print ""

main()
```

Figure 23: Input MDS Form with sample data



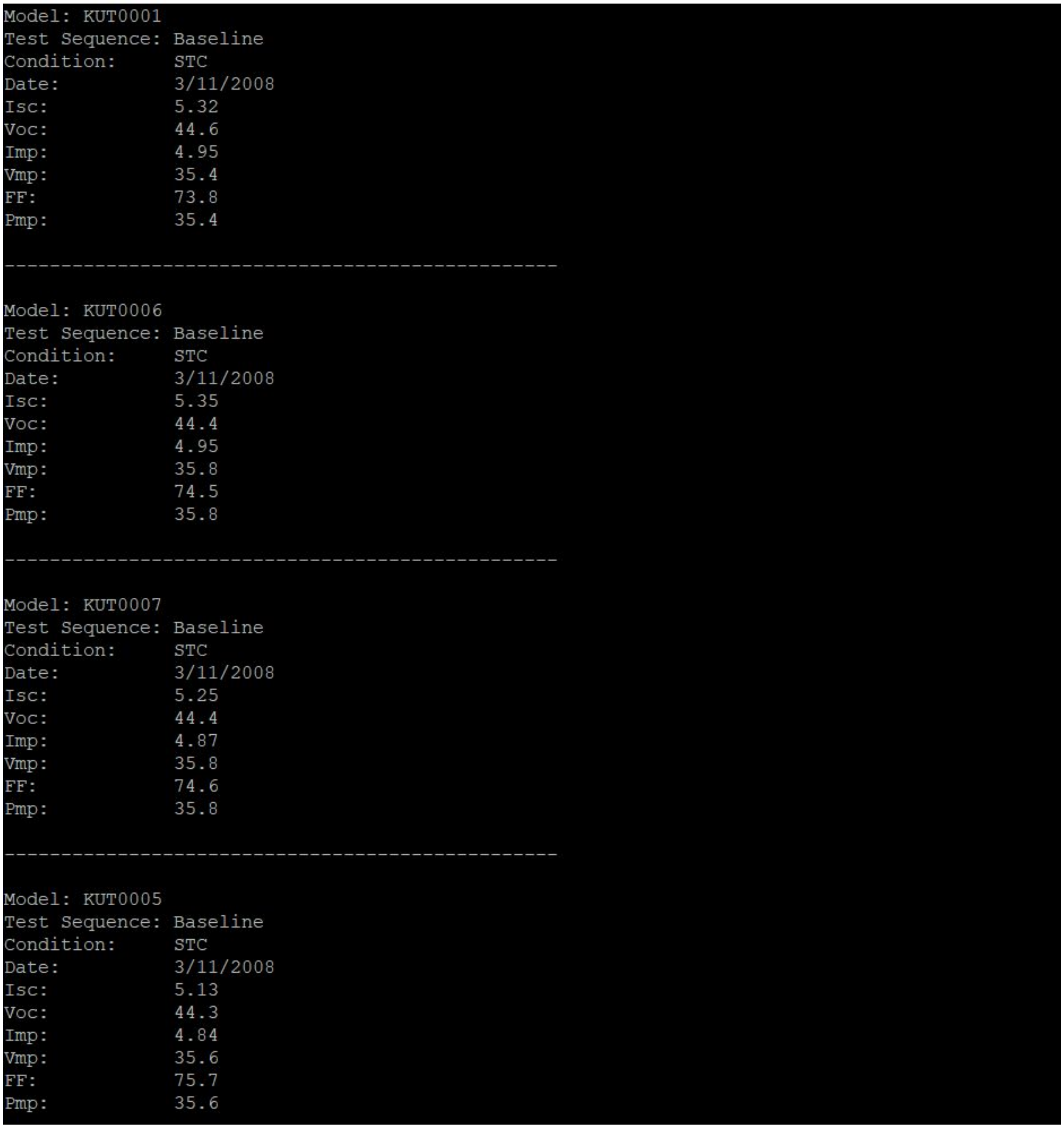
```
*********************WELCOME**********************

----------------MDS FORM---------------

Manufacturer: Zhuhai Tianbo
Location: China
Contact: Talilin Wang
Address: No.1 Pingbei 2nd Road, Zhuhai, China 519060
Email: spv@yuemaolaser.com
Phone: +86-756-8911378
Model Number: KUT0012
Module total length x width (cmxcm): 158× 80.8
Module weight(kg): 15
Individual Cell Area(cm^2): 148.58
Cell Technology: Mono-Si
Cell Manufacturer and Part#:  Motech
Cell Manufacturing Location: Taiwan
Total number of cells: 72
Number of cells in series: 72
Number of series strings: 3
Number of bypass diodes: 3
Bypass diode rating(A): 10 / 10SQ050
Bypass diode max junction temp(C): 200
Series Fuse Rating(A): 10
Interconnect material and supplier model no.: Ulbrich Stainless Steels & Special Metals Ltd
Interconnect dimensions(mm x mm): 0.2mm ×1.5mm ,  0.2mm × 5mm
Superstrate Type: Tempered Glass
Superstrate Manfacturer and part#: Dongguan CSG Solar Glass Co., Ltd./ 3.2 mm
Substrate Type: TPT/0.35 mm
Substrate Manufacturer and part#: ISOVOLTA
Frame Type and Material: Aluminum alloy
Frame adhesive: Dow Corning 7091
Encapsulant Type: EVA/0.5 mm
Encapsulant Manufacturer and part#: Bridge Stone Corporation
Junction box type: PV-RH0502B
Junction box manufacturer and part#: Cixi Renhe Photovoltaic Electrical Appliance Co.,Ltd.
Junction box potting material, if any: NA
Junction box adhesive: Dow Corning 7091
Is junction box intended for use with Conduit?: NA
Cable & Connector Type: 2 pfg 1169 1x4 mm2, 05-6
Max system voltage(V): 1000V
Voc(V): 44.2
Isc(A): 5.25
Vmp(V): 35.2
Imp(A): 4.97
Pmp(W): 175
FF(%): 75
```

Figure 24: Required Output Data - Product Info and Baseline Results



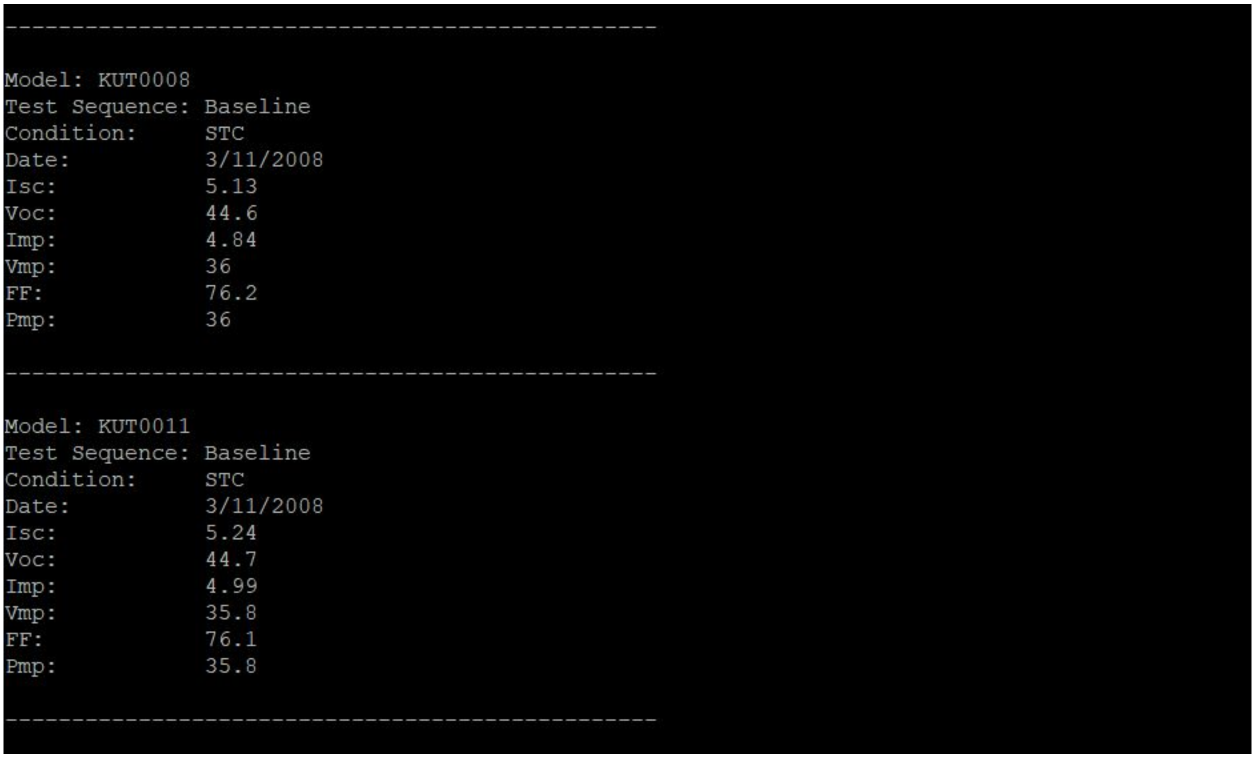
```
----------Product Information-----------

Manufacturer Name: Zhuhai Tianbo
Contact Name: Talilin Wang
Contact Email: spv@yuemaolaser.com
Model Number: KUT0012
Cell Technology: Mono-Si
System Voltage: 1000V
Rated Power (PMP): 175
----------Baseline Test Results----------

Model: KUT0012
Test Sequence: Baseline
Condition:     STC
Date:          3/11/2008
Isc:           5.2
Voc:           44.7
Imp:           4.88
Vmp:           35.7
FF:            75
Pmp:           35.7

-------------------------------------------------

Model: KUT0003
Test Sequence: Baseline
Condition:     STC
Date:          3/11/2008
Isc:           5.34
Voc:           44.7
Imp:           5.03
Vmp:           35.7
FF:            75.2
Pmp:           35.7

-------------------------------------------------

Model: KUT0004
Test Sequence: Baseline
Condition:     STC
Date:          3/11/2008
Isc:           5.21
Voc:           44.8
Imp:           4.91
Vmp:           36.1
FF:            76
Pmp:           36.1

-------------------------------------------------
```

Figure 25: Required Output Data - Baseline Results Continued



```
Model: KUT0001
Test Sequence: Baseline
Condition:     STC
Date:          3/11/2008
Isc:           5.32
Voc:           44.6
Imp:           4.95
Vmp:           35.4
FF:            73.8
Pmp:           35.4

-------------------------------------------------

Model: KUT0006
Test Sequence: Baseline
Condition:     STC
Date:          3/11/2008
Isc:           5.35
Voc:           44.4
Imp:           4.95
Vmp:           35.8
FF:            74.5
Pmp:           35.8

-------------------------------------------------

Model: KUT0007
Test Sequence: Baseline
Condition:     STC
Date:          3/11/2008
Isc:           5.25
Voc:           44.4
Imp:           4.87
Vmp:           35.8
FF:            74.6
Pmp:           35.8

-------------------------------------------------

Model: KUT0005
Test Sequence: Baseline
Condition:     STC
Date:          3/11/2008
Isc:           5.13
Voc:           44.3
Imp:           4.84
Vmp:           35.6
FF:            75.7
Pmp:           35.6
```

Figure 26: Required Output Data - Baseline Results Continued



```
-------------------------------------------------

Model: KUT0008
Test Sequence: Baseline
Condition:     STC
Date:          3/11/2008
Isc:           5.13
Voc:           44.6
Imp:           4.84
Vmp:           36
FF:            76.2
Pmp:           36

-------------------------------------------------

Model: KUT0011
Test Sequence: Baseline
Condition:     STC
Date:          3/11/2008
Isc:           5.24
Voc:           44.7
Imp:           4.99
Vmp:           35.8
FF:            76.1
Pmp:           35.8

-------------------------------------------------
```

User Manual

To execute these scripts, ensure that the .zip file provided is downloaded and all files are kept in the same folder. Next, you can open the file and right click on pd3.py and execute it. If this does not work, you can upload the file to the asu general using the “myfiles” GUI feature on the ASU homepage. Then you can open a terminal that allows access to the general (putty) and login using your asurite username and password. Next, you need to give the file execute permissions by running the command `chmod u+x pd3.py`. Lastly, you may run the program by executing the command `python pd3.py`.

Conclusion

In conclusion, completing this portion of the project allowed us to review the concepts learned in IFT 394 (IFT383) and served mostly as a refresher for python object oriented. We did, however, learn different variations on reading in files, taking arguments from the command line and overall manipulation of objects using methods. Additionally, we learned that the value of dictionaries and how to properly create them from user input. The most challenging task we faced was simplifying the project in order to understand the steps (algorithm) that would be implemented in code. We did overcome this hurdle by reading the PDF provided for this topic, listening to the online lectures, discussing with each other our thoughts, and finally from revisions from Dr. Kuitche. There are two ways that these scripts could be improved: 1) by reducing the amount of code used, making it more efficient to process and 2) by implementing a main menu feature. However, both of those improvements were beyond the scope of the deliverable.