Water Utility Billing Data Converter – Version 1.0

Requires Python at Minimum 3.11 and Pandas Plugin

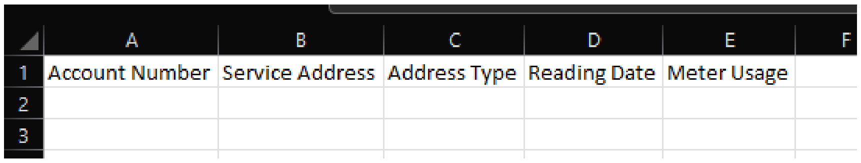
Python Download
https://www.python.org/downloads/release/python-3121/

Pandas Installation Instructions:
https://pandas.pydata.org/getting_started.html

Instructions

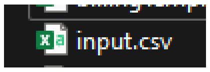
1. Evaluate Water Utility Billing Data

2. Paste Relevant Billing Data into billingTemplate.csv



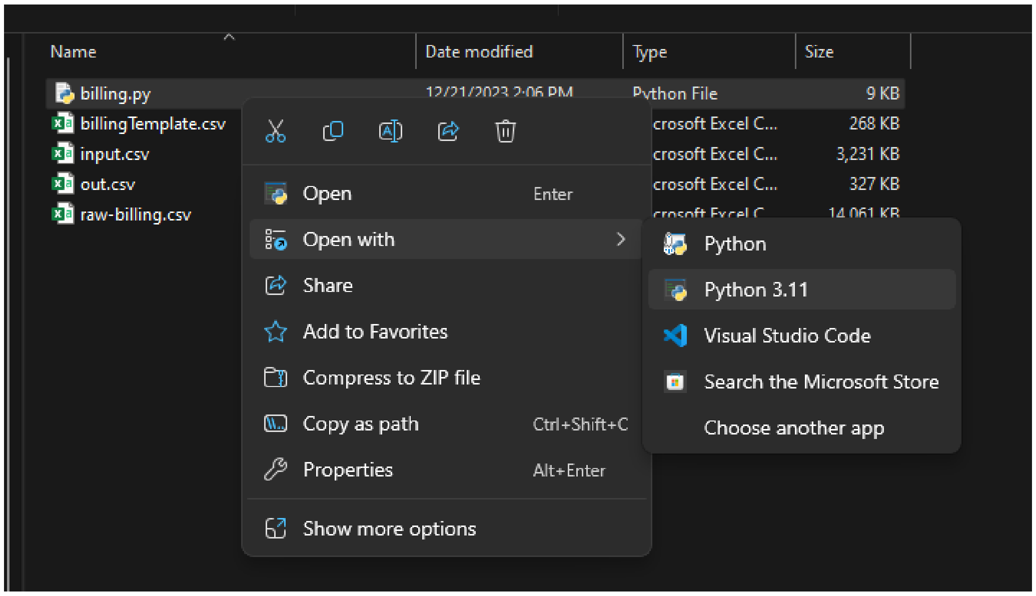
	A	B	C	D	E	F
1	Account Number	Service Address	Address Type	Reading Date	Meter Usage	
2						
3						

3. Change File Name to input.csv



input.csv

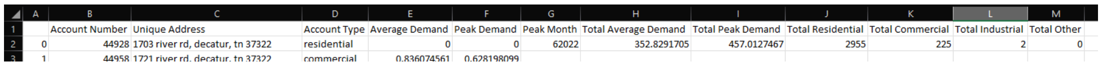
4. Right Click billing.py and Select Open with Python 3.11



Name	Date modified	Type	Size
billing.py	12/21/2023 2:06 PM	Python File	9 KB
billingTemplate.csv		crosoft Excel C...	268 KB
input.csv		crosoft Excel C...	3,231 KB
out.csv		crosoft Excel C...	327 KB
raw-billing.csv		crosoft Excel C	14,061 KB

5. out.csv Will Populate when the Program Has Finished

out.csv



	A	B	C	D	E	F	G	H	I	J	K	L	M
1		Account Number	Unique Address	Account Type	Average Demand	Peak Demand	Peak Month	Total Average Demand	Total Peak Demand	Total Residential	Total Commercial	Total Industrial	Total Other
2	0	44928	1703 river rd, decatur, tn 37322	residential	0	0	62022	352.8291705	457.0127467	2955	225	2	0
3	1	44958	1721 river rd, decatur, tn 37322	commercial	0.836074561	0.628198099							

```
import pandas as pd
from datetime import datetime

billingData = pd.read_csv(r"C:\Users\tgalyon\Desktop\water_billing\input.csv")
columns = list(billingData)

#Variable initialization for input column indexes
accountNumberIndex = 0
nameIndex = 0
addressIndex = 0
typeIndex = 0
readingDateIndex = 0
usageReadingIndex = 0

#Change i = value to match the column name in the input CSV file
for i in columns:
  if i == "Account Number":
    accountNumberIndex = columns.index(i)
  #elif i == "name":
    #nameIndex = columns.index(i)
  elif i == "Service Address":
    addressIndex = columns.index(i)
  elif i == "Address Type":
    typeIndex = columns.index(i)
  elif i == "Reading Date":
    readingDateIndex = columns.index(i)
  elif i == "Meter Usage":
    usageReadingIndex = columns.index(i)

#Variable initialization for extracting lists from pandas dataframe
accountNumber = billingData.iloc[:,accountNumberIndex]
#name = billingData.iloc[:,nameIndex]
address = billingData.iloc[:,addressIndex]
accountType = billingData.iloc[:,typeIndex]
readingDate = billingData.iloc[:,readingDateIndex]
meterUsage = billingData.iloc[:,usageReadingIndex]

#Python list variable intialization
accountNumberList = []
meterUsageList = []
date = []
month = []
year = []
monthYear = []
uniqueMonthYear = []
uniqueAccountNumber = []
uniqueType = []
#nameList = []
addressList = []
accountTypeList = []
formatDate = '%m/%d/%Y'

#Extracts data from pandas dataframe and changes to Python lists
for i in readingDate:
  date.append(datetime.strptime(i, formatDate).date())
```

```
for i in date:
    monthYear.append(str(i.month) + str(i.year))

for i in monthYear:
    if i not in uniqueMonthYear:
        uniqueMonthYear.append(i)

for i in accountNumber:
    accountNumberList.append(i)
    if i not in uniqueAccountNumber:
        uniqueAccountNumber.append(i)

for i in meterUsage:
    meterUsageList.append(i)

for i in accountType:
    accountTypeList.append(i)

#for i in name:
    #nameList.append(i)

for i in address:
    addressList.append(i)

#Function section
def getAverageList(uniqueList, nonUniqueList, usageData):
    tempSumList = []
    listLocation = 0
    global averageUsage
    averageUsage = []
    for i in uniqueList:
        for n in nonUniqueList:
            if n == i:
                if listLocation == len(nonUniqueList) - 1:
                    if len(tempSumList) == 0:
                        averageUsage.append(0)
                        tempAverage = 0
                        listLocation = 0
                        tempSumList.clear()
                    else:
                        tempAverage = (sum(tempSumList) / len(tempSumList))
                        averageUsage.append(tempAverage / (30.4*24*60))
                        tempAverage = 0
                        listLocation = 0
                        tempSumList.clear()
                else:
                    tempSumList.append(usageData[listLocation])
                    listLocation += 1
            else:
                if listLocation == len(nonUniqueList) - 1:
                    if len(tempSumList) == 0:
                        averageUsage.append(0)
                        tempAverage = 0
                        listLocation = 0
                        tempSumList.clear()
                    else:
                        tempAverage = (sum(tempSumList) / len(tempSumList))
                        averageUsage.append(tempAverage / (30.4*24*60))
                        tempAverage = 0
                        listLocation = 0
                        tempSumList.clear()
                else:
                    listLocation += 1

def getPeakList(uniqueList, nonUniqueList, usageData):
    tempMaxList = []
    tempMaxValue = 0
    listLocation = 0
    global maxMonth
    global peakUsage
    global peakDict
    peakUsage = []
    peakDict = {}
    for i in uniqueList:
        for n in nonUniqueList:
            if n == i:
                if listLocation == len(nonUniqueList) - 1:
                    tempMaxList.append(usageData[listLocation])
                    tempMaxValue = sum(tempMaxList)
                    peakUsage.append(tempMaxValue/(30.4*24*60))
                    listLocation = 0
                    tempMaxList.clear()
                    tempMaxValue = 0
                else:
                    tempMaxList.append(usageData[listLocation])
                    listLocation += 1
            else:
                if listLocation == len(nonUniqueList) - 1:
                    tempMaxValue = sum(tempMaxList)
                    peakUsage.append(tempMaxValue/(30.4*24*60))
                    listLocation = 0
                    tempMaxList.clear()
                    tempMaxValue = 0
                else:
                    listLocation += 1
    peakDict = dict(zip(uniqueList,peakUsage))
    maxMonth = (max(peakDict, key = peakDict.get))

def getPeakMonth(uniqueList, nonUniqueList, usageData, dateList):
    global peakData
    tempValue = 0
    peakData = []
    for i in uniqueList:
        for idn, n in enumerate(nonUniqueList):
            if idn == (len(nonUniqueList)- 1):
                peakData.append(0)
            else:
                if n == i:
                    if dateList[idn] == maxMonth:
```

```
                    tempValue = usageData[idn]
                    peakData.append(tempValue/(30.4*24*60))
                    break

def getAddressType(nonUniqueList, uniqueList, typeList):
    global uniqueType
    uniqueType = []
    for i in uniqueList:
        for idn, n in enumerate(nonUniqueList):
            if i == n:
                uniqueType.append((typeList[idn]).lower())
                break

def classifyType(typeList):
    global residential
    global commercial
    global industrial
    global other
    residential = 0
    commercial = 0
    industrial = 0
    other = 0
    for i in typeList:
        if i == "residential":
            residential += 1
        elif i =="commercial":
            commercial += 1
        elif i == "industrial":
            industrial += 1
        else:
            other += 1

#def getName(nonUniqueList, uniqueList, typeList):
    #global uniqueName
    #uniqueName = []
    #for i in uniqueList:
        #for idn, n in enumerate(nonUniqueList):
            #if i == n:
                #uniqueType.append((typeList[idn]).lower())
                #break

def getAddress(nonUniqueList, uniqueList, typeList):
    global uniqueAddress
    uniqueAddress = []
    for i in uniqueList:
        for idn, n in enumerate(nonUniqueList):
            if i == n:
                uniqueAddress.append((typeList[idn]).lower())
                break


#Function Calls
getAverageList(uniqueAccountNumber, accountNumberList, meterUsageList)
getPeakList(uniqueMonthYear, monthYear, meterUsageList)
getPeakMonth(uniqueAccountNumber, accountNumberList, meterUsageList, monthYear)
```

```
getAddressType(accountNumberList, uniqueAccountNumber, accountTypeList)
classifyType(uniqueType)
#getName(accountNumberList, uniqueAccountNumber, nameList)
#print(maxMonth)
getAddress(accountNumberList, uniqueAccountNumber, addressList)
#Exports Python lists to pandas dataframe
data = {
    "Account Number": pd.Series(uniqueAccountNumber),
    "Unique Address": pd.Series(uniqueAddress),
    #"uniqueName": uniqueName,
    "Account Type": pd.Series(uniqueType),
    "Average Demand": pd.Series(averageUsage),
    "Peak Demand": pd.Series(peakData),
    "Peak Month": pd.Series(maxMonth),
    "Total Average Demand": pd.Series(sum(averageUsage)),
    "Total Peak Demand": pd.Series(sum(peakData)),
    "Total Residential": pd.Series(residential),
    "Total Commercial": pd.Series(commercial),
    "Total Industrial": pd.Series(industrial),
    "Total Other": pd.Series(other)
    }

df = pd.DataFrame(data)

#Exports pandas dataframe to output CSV
df.to_csv(r"C:\Users\tgalyon\Desktop\water_billing\out.csv")

#Written and developed by Tyler C. Galyon
```