Kathleen Wong

November 30, 2021

Intro to Programming

Assignment 07

[GitHub Link]

Pickling

Introduction

This week, I learned about pickling and try/except. After learning these two topics, I had developed my own script, incorporating the lessons learned this week. For my assignment, I had developed a script, asking the user to provide some groceries and the cost and then pickling their results and saving them to a binary file.

Pickling is a process to convert a data structure into a linear form that can be stored or transmitted over a network (https://realpython.com/python-pickle-module/) (External Site). This allows the computer to compartmentalize the data, making the computer capable of transferring the data over a disk or network. In other words, the process changes the format of the data, making it illegible for humans to read but easy for computers to read and transfer. Despite it being illegible to the human eye, it is The reverse process is called unpickling.

Try and except allows programmers to catch and handle exceptions (https://realpython.com/python-exceptions/#raising-an-exception) (External Site). The try statement executes the expected part of the program. The except statement handles any exceptions raised.

Assignment Background

This week I had more freedom in deciding what script to write. Unlike my recent assignments, I get to develop my own script by scratch. The only requirements are to include a way to showcase try/except and pickling. I decided to develop a script, requesting the user's grocery list.

Pickling

The first step to pickling any item is to import the pickle library. As the library name suggests, the library allows programmers to pickle and unpickle their data.

```
# ------------------------------------------------- #
# Title: Assignment 07
# Description: A simple try-catch demo with pickling
# ChangeLog: (Who, When, What)
# KWong,11.30.2021,Created Script
# ------------------------------------------------- #
import pickle
```

Variables

The next step is to set up the variables I will be using throughout my script. In this script, I will be using a .dat file. I will also be setting up a dictionary within a dictionary to complete my task.

```
# declare variables and constants
file_name = "GroceryList.dat" # An object that represents a file
dicRow = {}    # A row of data separated into elements of a dictionary {item,
cost}
strChoice = '' # A Capture the user option selection
```

<u>try/except/pickling</u>

The try and except statements allow programmers to handle exceptions. The block of code under the try statement is executed first if possible. If not, the block of code under the except statement is executed.

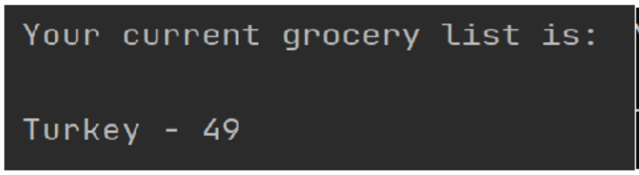
I want to allow the user to read the data in the file and add it to the file automatically. However, because someone can accidentally mess up exporting the file, I set it up into a try/except situation. Under my try statement, the script is opening the GroceryList file and reading the file, loading the dictionary loaded into the file and printing out the formatted dictionary before closing the file. And because I expected the data to be pickled, I unpickled the data with the pickle.load() function.

Within the except statement, I expect a potential FileNotFoundError error message and set up an exception code to that. If the file does not exist, so the user cannot complete the try statement, the script will request the item and cost information before setting them both up into a key : value relationship. Afterwards, the file will open and write whatever the user had written and pickle the data (with the pickle.dump() function) and dump the data into the file and close.

```
#try to open file and unpickle the data
try:
    strFile = open(file_name, "rb")
    dicRow = pickle.load(strFile) #unpickling
    print("Your current grocery list is: \n")
    for item, cost in dicRow.items():
        print(item, '-', cost)
    strFile.close()

# create file and add data to file
except FileNotFoundError as e:
    print("No data found! Let's get some data!")
    item = str(input("What item do you need? "))
    cost = str(input("How much does this item cost? " ))
    dicRow[item] = cost
    strFile = open(file_name, "wb")
    pickled_grocery = pickle.dump(dicRow, strFile) #pickle the data
    strFile.close()
```

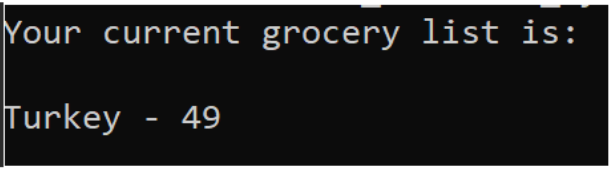
Try:

PyCharm



```
Your current grocery list is:

Turkey - 49
```

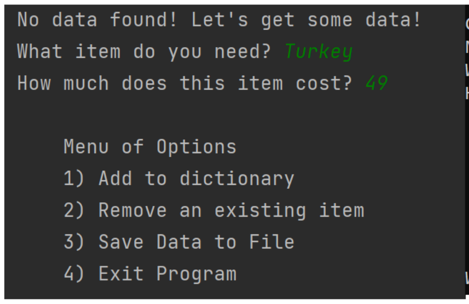
Command Prompt



```
Your current grocery list is:

Turkey - 49
```

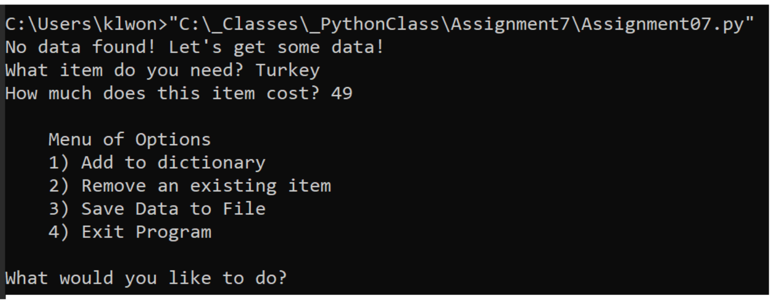
Except:

PyCharm



```
No data found! Let's get some data!
What item do you need? Turkey
How much does this item cost? 49

    Menu of Options
    1) Add to dictionary
    2) Remove an existing item
    3) Save Data to File
    4) Exit Program
```

Command Prompt



```
C:\Users\klwon>"C:\_Classes\_PythonClass\Assignment7\Assignment07.py"
No data found! Let's get some data!
What item do you need? Turkey
How much does this item cost? 49

    Menu of Options
    1) Add to dictionary
    2) Remove an existing item
    3) Save Data to File
    4) Exit Program

What would you like to do?
```

While Statement

I previously learned how to set up a while statement, wanting to continuously allow users to provide more information if necessary/wanted. In this case, I set up a while True statement, intending to keep the script continuing to offer the same menu of options until it reaches a stop (with 5 as a break).

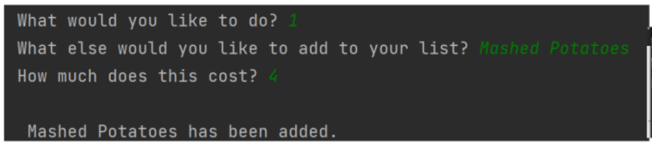
```
while True:
    print("""
    Menu of Options
    1) Add to dictionary
    2) Remove an existing item
    3) Save Data to File
    4) Exit Program
    """)
    strChoice = str(input("What would you like to do? "))
```

Adding Data

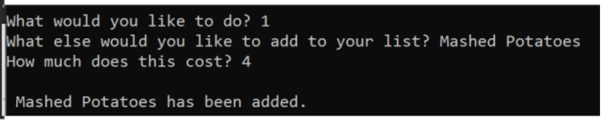
If the item is not already used as a key in the dictionary, I set the script to ask for the cost. Otherwise, it will print the item is already in the list.

```
if item not in dicRow:
    cost = str(input("How much does this cost? "))
    dicRow[item] = cost
    print('\n', item, "has been added.")
else:
    print("This item is already on your list! ")
continue
```

PyCharm

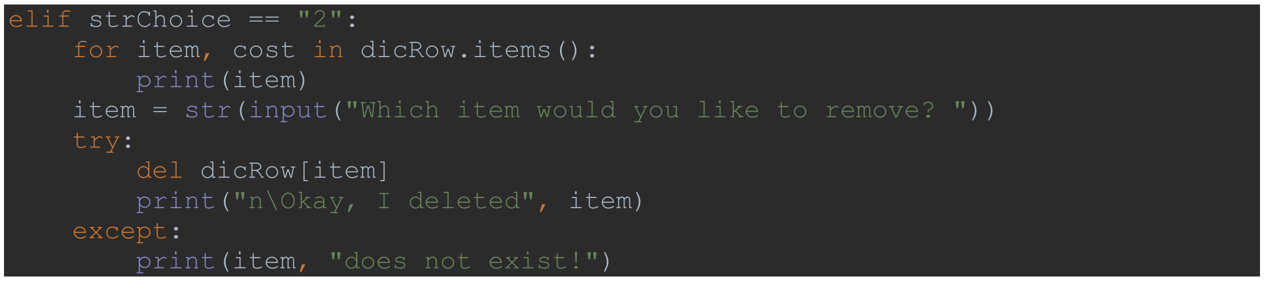

```
What would you like to do? 1
What else would you like to add to your list? Mashed Potatoes
How much does this cost? 4


 Mashed Potatoes has been added.
```

Command Prompt


```
What would you like to do? 1
What else would you like to add to your list? Mashed Potatoes
How much does this cost? 4

 Mashed Potatoes has been added.
```

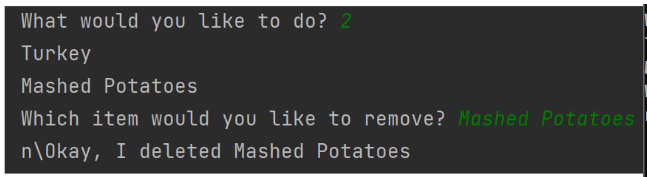
Deleting Data

Next, I'd like to provide the option to delete the items from the list. Because I know there's a chance the user may accidentally input an item not actually in the list, I decided to set up another try/except scenario. It's simple here. The computer will first try to execute the try statement. The code here tells the computer to delete the item from the dictionary and inform the user. Otherwise, it will execute the except statement, which simply states to print that the item is not in the dictionary.

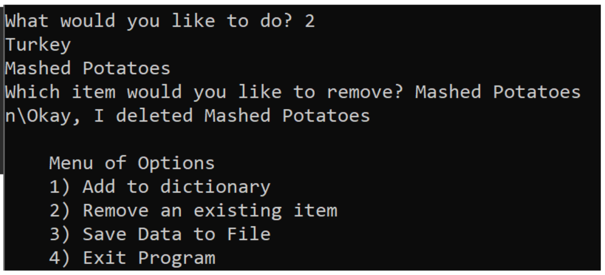

```
elif strChoice == "2":
    for item, cost in dicRow.items():
        print(item)
    item = str(input("Which item would you like to remove? "))
    try:
        del dicRow[item]
        print("n\Okay, I deleted", item)
    except:
        print(item, "does not exist!")
```

Try:

PyCharm

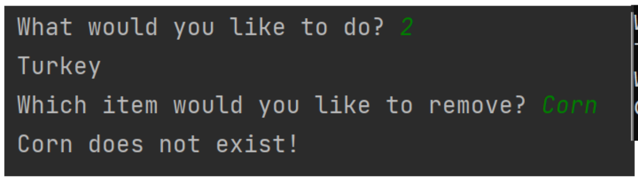

```
What would you like to do? 2
Turkey
Mashed Potatoes
Which item would you like to remove? Mashed Potatoes
n\Okay, I deleted Mashed Potatoes
```

Command Prompt

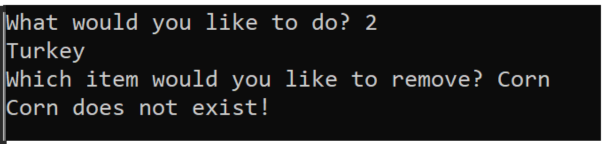

```
What would you like to do? 2
Turkey
Mashed Potatoes
Which item would you like to remove? Mashed Potatoes
n\Okay, I deleted Mashed Potatoes

    Menu of Options
    1) Add to dictionary
    2) Remove an existing item
    3) Save Data to File
    4) Exit Program
```

Except:

PyCharm

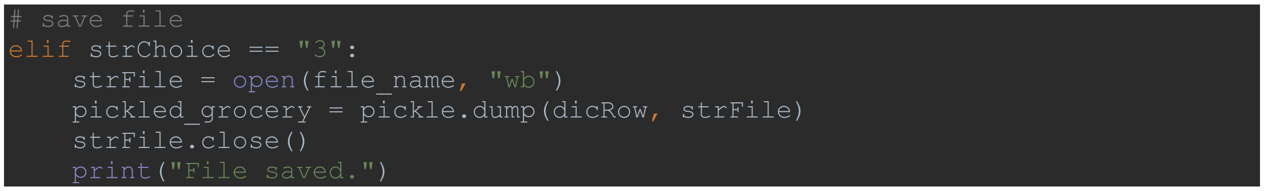

```
What would you like to do? 2
Turkey
Which item would you like to remove? Corn
Corn does not exist!
```

Command Prompt


```
What would you like to do? 2
Turkey
Which item would you like to remove? Corn
Corn does not exist!
```

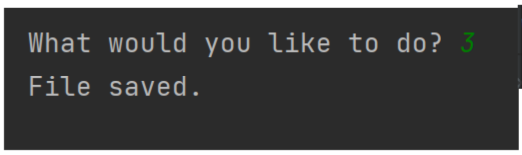
Saving the File

Because I want to compress my data, I pickled the data and wrote it to a .dat file. Pickle.dump() saves the pickled data into the file. Afterwards, I closed the file to follow best practice.



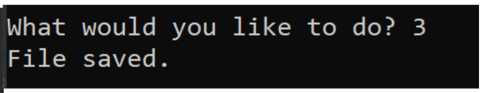
```
# save file
elif strChoice == "3":
    strFile = open(file_name, "wb")
    pickled_grocery = pickle.dump(dicRow, strFile)
    strFile.close()
    print("File saved.")
```

PyCharm



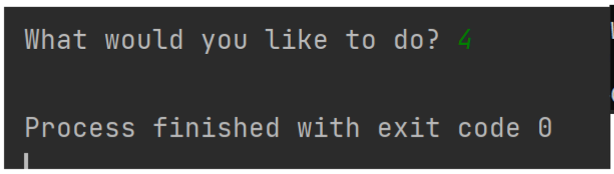
```
What would you like to do? 3
File saved.
```

Command Prompt



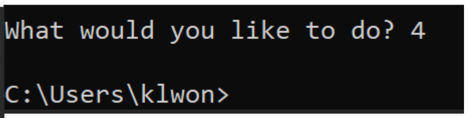
```
What would you like to do? 3
File saved.
```

PyCharm



```
What would you like to do? 4

Process finished with exit code 0
```

Command Prompt



```
What would you like to do? 4

C:\Users\klwon>
```

Conclusion

This week I was given the task to apply pickling and try/except into a new script that I've developed on my own. I decided to execute this task by setting up a while loop statement, continuously asking for more information and using a try/except to ensure the user will be able to take an action. I then set up my data to be pickled to compress the file.