Week 6 (Week 7 on BB) File I/O and program arguments

Part 1 File I/O

Exercise 1 - File input and output

1,2

Using the open method, the file HighScoreList.txt is opened in read mode.
The file is composed of line which we can iterate through to store the data either in a dictionary or an array

In [1]:

```
HighScoreTxt = open("HighScoreList.txt","r")
```

In [2]:

```
HighScoreTable = []
HighScoreDict = {}
for Line in  HighScoreTxt:
    if(Line != None):
        Data = Line.split(',') # splits data at the comma
        HighScoreTable.append([int(Data[0]),Data[1].replace("\n","")])
        HighScoreDict[Data[1].replace("\n","")] = int(Data[0])
print(HighScoreTable)
print(HighScoreDict)
```

```
[[575, 'Mit Hobson'], [500, 'Janice Waterfell'], [450, 'Tony Robber'], [35
0, 'Clark Kent'], [180, 'James Hunter']]
{'Mit Hobson': 575, 'Janice Waterfell': 500, 'Tony Robber': 450, 'Clark Ke
nt': 350, 'James Hunter': 180}
```

Now the data is store both in a dictionary (where the keys are the players' name and the value their respective Highscore) and a table
To print the data in the correct formatting function is used that will print depending on the type of the argument

In [3]:

```
def print_scores(data):
    if isinstance(data,dict) : # for dictionaries
        for keys in data.keys():
            print("Name: "+ keys +" Score: "+ str(data[keys]))
    elif isinstance(data,list): # for lists
        for entries in data:
            print("Name: "+ entries[1] +" Score: "+ str(entries[0]))
    else:
        print("Please input either a table or a dictionary")
print_scores(HighScoreTable)
print()
print_scores(HighScoreDict)
```

```
Name: Mit Hobson Score: 575
Name: Janice Waterfell Score: 500
Name: Tony Robber Score: 450
Name: Clark Kent Score: 350
Name: James Hunter Score: 180

Name: Mit Hobson Score: 575
Name: Janice Waterfell Score: 500
Name: Tony Robber Score: 450
Name: Clark Kent Score: 350
Name: James Hunter Score: 180
```

2,3

Depending on desired format we need to make sure that the user inputs the data in the correct format.

In [4]:

```
## input a new score
new_entry = []
while True:
    userinput = input("Please enter first name surname and score")
    parsedinput = userinput.split(',')
    if len(parsedinput) == 1:
        print("Please input the score in the format: [Score],[Playert Name]")
        continue
    try:
        score = int(parsedinput[0])
        new_entry.append(int(parsedinput[0]))
        new_entry.append(parsedinput[1].strip())
        break
    except:
        print("Please Input a valid Score")


print(new_entry)
```

```
[123, 'John Smith']
```

3.1, 3.2

Now that we have a new entry we need to compare to the score with the current table. Since the next exercise relates to appending vs overwriting let's define two entries one containing a new lowest score and a new highest score
Note on dictionary vs list: Using a dictionary makes it easier to determine duplicate entries when updating high scores this edge case will not be covered here

In [6]:

```
new_lowest = [179,'John Smith']
new_highest = [599,'John Tron']
#listofscores = [(i[0],c) for c,i in enumerate(HighScoreTable) ]
#print(listofscores)
```

In [14]:

```
# In this case the list of high score is already sorted to accessing the lowest
# element is the same as access the last one
lowest_score_table = HighScoreTable[-1][0]
print(lowest_score_table)
# For the dictionary we can access dictionary values
lowest_score_dict = min(HighScoreDict.values())
print(lowest_score_dict)
```

```
180
180
```

For the lowest score we need to append to the file as it isn't necessary to overwrite it
For this example we will use the new_lowest entry

In [8]:

```
new_entry = new_lowest
newScore = new_entry[0]
if newScore <= lowest_score_table:
    Update = open("HighScoreList.txt","a")
    Update.write(str(new_entry[0])+","+new_entry[1]) #you might need to add a return ch
aracter "\n" depending on how your file is set up
    Update.close()
```

Now we expand this in the case were the new score belongs anywhere but the end of the table in which case overwrite the file but first we need to figure out how to sort the new entry.

In [9]:

```
# for a table
new_entry = new_highest
newScore = new_entry[0]
HighScoreTable.append(new_entry)
print(HighScoreTable)
HighScoreTable.sort(reverse=True)
print(HighScoreTable)
```

```
[[575, 'Mit Hobson'], [500, 'Janice Waterfell'], [450, 'Tony Robber'], [35
0, 'Clark Kent'], [180, 'James Hunter'], [599, 'John Tron']]
[[599, 'John Tron'], [575, 'Mit Hobson'], [500, 'Janice Waterfell'], [450,
'Tony Robber'], [350, 'Clark Kent'], [180, 'James Hunter']]
```

In [10]:

```
#for a dictionary
HighScoreDict[new_entry[1]] = new_entry[0]
HighScoreFromDict  = sorted(HighScoreDict.items(), key=lambda x:x[1], reverse=True)
print(HighScoreFromDict)
```

```
[('John Tron', 599), ('Mit Hobson', 575), ('Janice Waterfell', 500), ('Ton
y Robber', 450), ('Clark Kent', 350), ('James Hunter', 180)]
```

As you can see this becomes a list. However it is possible to use dictionary comprehension

In [12]:

```
HighScoreDict2 = {key: val for key, val in sorted(HighScoreDict.items(), key = lambda e
ntry: entry[1], reverse = True)}
print(HighScoreDict2)
```

```
{'John Tron': 599, 'Mit Hobson': 575, 'Janice Waterfell': 500, 'Tony Robbe
r': 450, 'Clark Kent': 350, 'James Hunter': 180}
```

In [17]:

```
if newScore <= lowest_score_table:
    Update = open("HighScoreList.txt","a") #notice the a option to append
    Update.write(str(new_entry[0])+","+new_entry[1])
    Update.close()
else:
    Update = open("HighScoreList.txt","w") #notice the w option to overwrite
    for line in HighScoreTable: # This is done using the list but can easily be done be
accessing the dictionary using the key
        Update.write(str(line[0])+","+line[1]+'\n')
    Update.close()
```

In [ ]:

```

```