提出日：2024/5/24

プログラミング演習　第６回演習レポート

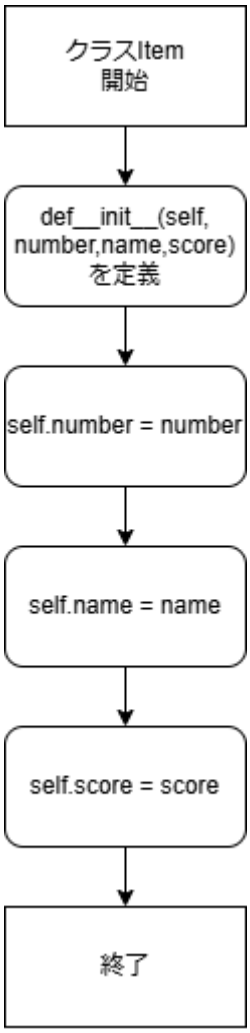
担当教員：杉本　千佳先生
所属：理工学部　数物・電子情報系学科
電子情報システムEP
学年・クラス：２年　Fe1
学籍番号：2364092
氏名：熊田　真歩

（1）課題番号：基本課題4

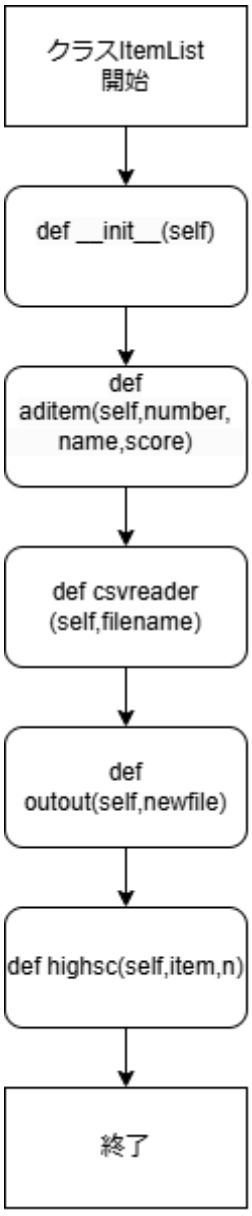
課題名：得点が同じ場合には学籍番号順（ 昇順 ）に出力する

（2）プログラムのフローチャート

・クラス Item について

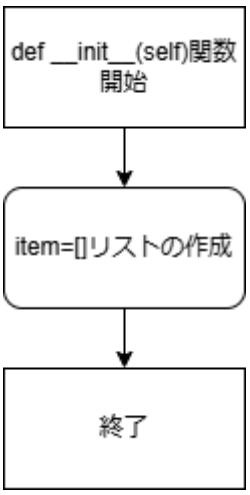


・クラス ItemList について

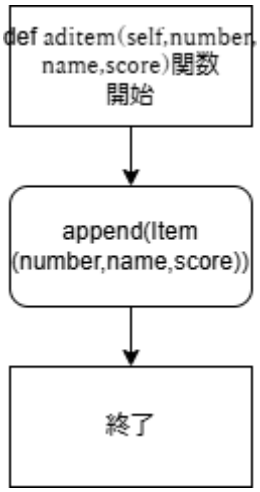


クラス ItemList 内部の関数について

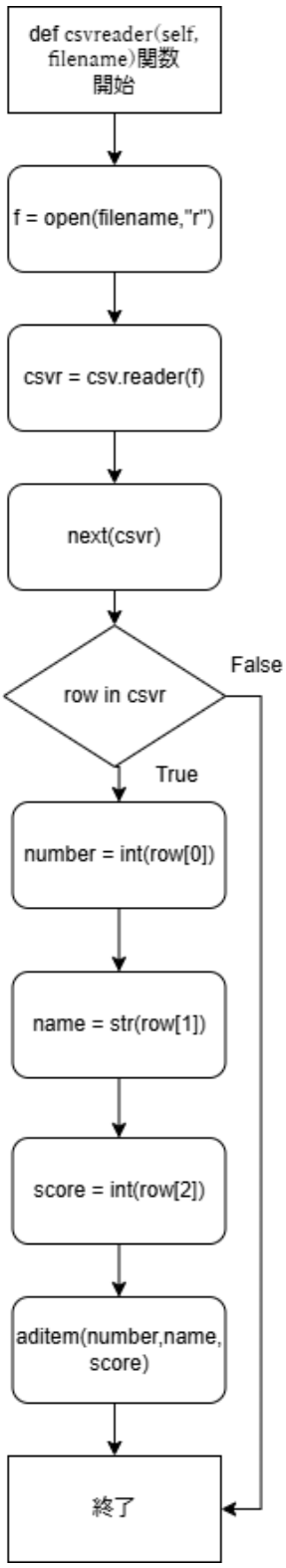
・__init__(self)関数



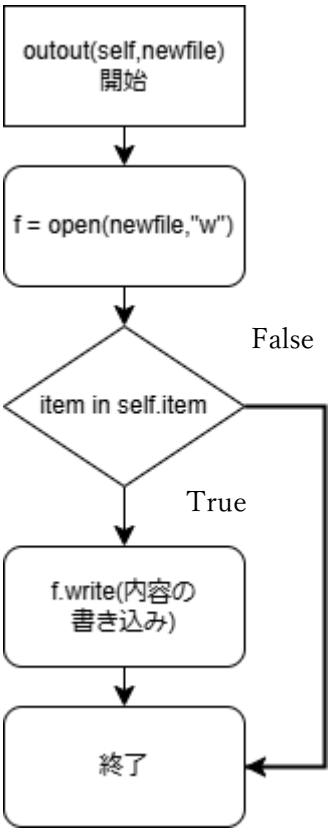
・aditem(self,number,name,score)関数



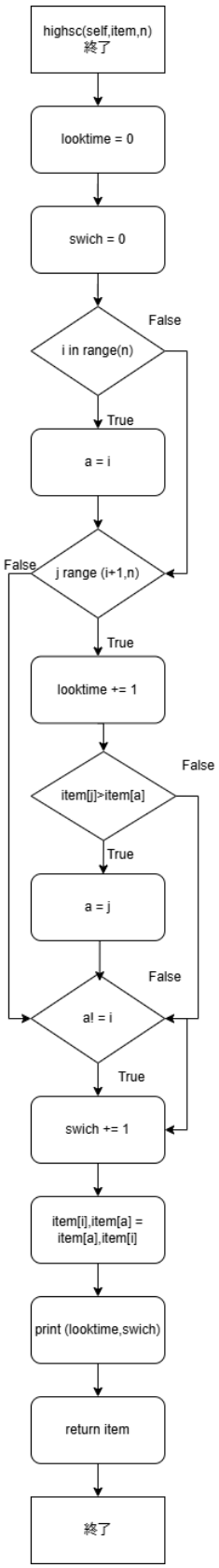
・csvreader(self,filename:str)関数



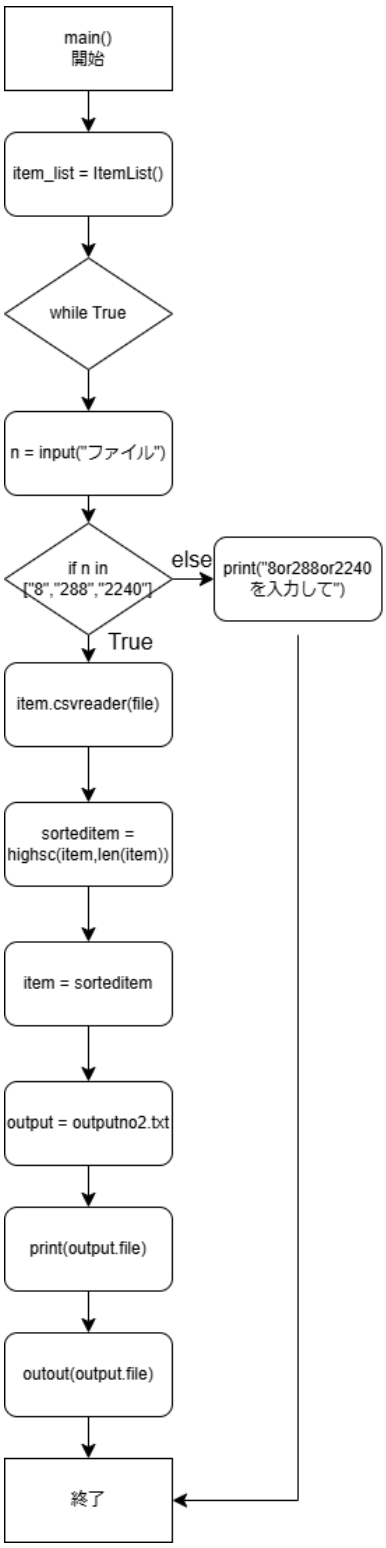
・outout(self, newfile:str)関数



・highsc(self,item,n)関数



・main()関数


main()
開始
item_list = ItemList()
while True
n = input("ファイル")
if n in
["8","288","2240"]
else
print("8or288or2240
を入力して")
True
item.csvreader(file)
sorteditem =
highsc(item,len(item))
item = sorteditem
output = outputno2.txt
print(output.file)
outout(output.file)
終了

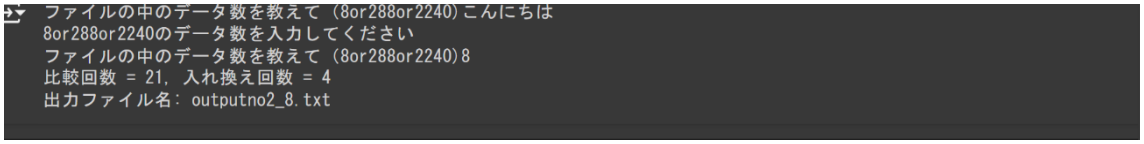
（3）アルゴリズムが「正しいこと」である説明

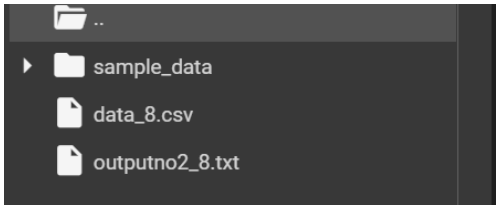
本プログラムは存在する学籍番号、名前、スコアが書かれた存在するある CSV ファイルに対してスコア順、更には同スコアの生徒に関しては学籍番号順に並び替えるというものである。

まず、すべての入力に対してプログラムは停止することを検証する。本プログラム data_8.csv、data_288.csv、data_2240.csv のいずれかのファイルが存在するという条件下で作成した。よって入力する値を 8or288or2240 というデータ数とした。入力時にループに入り、8,288, 2240 以外、すなわち存在しないファイルに関してはもう一度入力を求めるように作成した。以下にファイルが存在しない場合の実行例を示す。

更に以下のようにファイルのデータ数を聞かれ、数字で答えずに文字を入力した場合についても再びファイル名の入力を促すようになっており、エラーが発生しないことが分かる。

次に、すべての入力に対して「正しい答え」を出力することを検証する。上記で示したようにファイルが存在しない場合は再びファイル名の入力を促す。すなわちここでは入力されたファイル名が存在する場合に正しく出力結果を示すかを検証すればよい。以下のようにデータ数に 8 を入力し、data_8.csv ファイルについて処理する場合、プログラムは正常 j に停止し、正しい出力結果を生じすると共に以下に示すように outputno2_8.txt というファイルを作成し、テキストファイルの中もスコア順更に学籍番号順になっており、正しく処理できていると言える。ここでデータ数が増えても同様に正しい処理ができることも確認した。





outputno2_8.txt の内容

```
1439,Hurley          ,  95
 767,Warbeck         ,  76
1823,Morton          ,  64
1919,Walter          ,  54
1063,Edison          ,  41
1511,Nicholls        ,  41
  87,Ryle            ,   0
```

（4）ソース・プログラムの説明

以下に一つ一つの動作に対する説明を表示する。

```
import csv                          #csv ファイルを用いる時は必要
class Item:                         #クラス Item の定義
    def __init__(self,number:int, name: str,score:int):#クラスの定義、変数指定についての関数
        self.number = number                            #学籍番号について
        self.name = name                                #名前について
        self.score = score                              #スコアについて
class ItemList:                     #クラスを用いたリストの作成
    def __init__(self):             #クラスの定義についての関数
        self.item = []              #クラスの中に item のリストを作成
    def aditem(self,number:int,name:str,score:int):  #上記で作成したリストに内容を加える関数
        self.item.append(Item(number, name, score))  #append 関数を用いてリストに配列順に値を代入
    def csvreader(self,filename:str):                #CSV ファイルを読み込むための関数ファイル名は str 型
        with open(filename,'r') as f:                #読み取り専用でファイルを開く
            csvr = csv.reader(f)                     #変数 csvr にファイル内の内容を読み込む
            next(csvr)                               #先頭ラベルをスキップして読み込み
            for row in csvr:                         #CSV ファイルの行それぞれに対して
                number = int(row[0])                 #1 列目の内容は学籍番号について
                name = str(row[1])                   #2 列目の内容は名前について
                score = int(row[2])                  #③列目の内容は成績について
                self.aditem(number,name,score)       #上記のリストに内容を加える関数の呼び出し
    def outout(self, newfile:str):                   #内容を出力する outout()関数の定義
       f = open(newfile,'w')                         #ファイルを並び替えた内容を書き込むファイルの作成
       for item in self.item:                        #クラス item の中身 1 行ずつについて
         f.write(f"{item.number:4},{item.name:16},{item.score:3}¥n")#内容をファイルに書き込む
    def highsc(self,item,n):                         #スコア順更に学籍番号順に並び替える関数(引数は配列
                                                       とデータ数)
```

```
        looktime = 0                                    #比較回数について初期値 0
        swich = 0                                       #入れ替え回数について初期値 0
        for i in range(0,n):                            #i(比較する元の番号)を 0 からデータ数まで繰り返す
            a = i                                       #i の値を a に代入
            for j in range(i+1,n):                      #j(比較対象の番号)の値を i+1 からデータ数まで繰り返す
                looktime += 1                           #比較回数は＋1 回
                if item[j].score > item[a].score or(item[j].score == item[a].score and
item[j].number<item[a].number):
                    a = j                               #順番を変える必要がある時には a に j の値を代入
            if a!=i:                                    #比較する元の番号と比較対象の番号が異なる時
                swich += 1                              #入れ替え回数は＋1 回
            item[i],item[a] = item[a],item[i]           #順番の入れ替えを行う
        print(f"比較回数 = {looktime}, 入れ換え回数 = {swich}")#比較回数と入れ替え回数の値を出力
        return item                                     #item に値を返す


def main():                                             #main()関数についての定義
  item_list = ItemList()                                #変数に上記定義のクラスリストを格納
  while True:                                           #ファイル名が正しく入力されるまでのループ
    n = input("ファイルの中のデータ数を教えて(8or288or2240)")#ファイル名を聞くときにデータ数のみを入力すればよ
                                                                いようにした
    if n  in ["8" , "288" ,"2240"]:                             #ファイルが存在する時
     item_list.csvreader("data_"+str(n)+".csv")             #入力されたデータ数をもとにファイル名を上記関数に格納
     sorteditem = item_list.highsc(item_list.item, len(item_list.item)) #順序を並び替える関数の呼び
                                                                        出し
     item_list.item = sorteditem                                    #並び替えた item について扱う
     output_filename = "outputno2_"+str(n)+".txt"                   #データを出力するファイルの名前を
                                                                    入力されたデータ数を用いて定義
     print(f"出力ファイル名: {output_filename}")                    #出力ファイル名の出力
     item_list.outout(output_filename)                              #outout()関数の呼び出し
     break                                                          #ファイルが存在するものであったの
                                                                    でループから抜ける
    else:                                                           #ファイルが存在しない時
     print("8or288or2240 のデータ数を入力してください")                #注意文を出しもう一度ファイル名の入力


main()                                                                  #main()の実行
```

（5）考察

本プログラムは存在する学籍番号、名前、スコアが書かれた存在するある CSV ファイルに対してスコア順、更には同スコアの生徒に関しては学籍番号順に並び替えるというものである。ここで、指定があったため、クラスを用いたリストを使用して一人ひとりのデータの並び替え、出力を行った。また、それぞれの関数をクラス内部の関数とすることでよりスラスでの構造が明確になり分かりやすいアルゴリズムとなることを図った。しかし実際は関数をクラスの内部に入れても外部でも大差はなかったと考える。結局関数を呼び出し計算する回数は同様であるからだ。クラスは適応するデータ数が多いほどその効果を発揮する。どんなにデータ数が多くてもクラスを用いることで計算量や時間を抑えつつ正しくデータの処理ができることが今回のプログラムからも明らかである。

データをテキストファイルに入れる前に並び替えを行ったが、ここでは繰り返し構文を用いた。比較する元となるデータに対してすべてのデータを繰り返し構文により比較するというものである。このように比較の大本も比較対象もどっちも繰り返し構文に入れて比較を行ったため､少なくとも 2 回は同じ値通しでの比較を行っていることが明らかである。ここでより効率的なアルゴリズムにするためにここの比較部分を改良し比較回数を減らすことが可能である可能性が高いがそのアルゴリズムを考えることはできなかった。

更に､ファイル名の入力を行う input 関数においては存在しないファイル名を入力してもエラーが発生しないようにループ構文に落とし込んで処理を行うアルゴリズムを作成した。ここでこの処理がないアルゴリズムに比べ効率は下がるが､処理は一瞬であり､アルゴリズムを正しいものとするために不可欠な過程であると考えたため採用した。

（7）参考文献

・クラスについて
①アルゴリズムのサンプルプログラム
②TRAINCAMP「Python のクラス（class）の基本を徹底解説、具体的な書き方も」
https://camp.trainocate.co.jp/magazine/python-class/　2024/5/24 アクセス
③ITC　Media　「【完全版】Python クラスとは？使い方までをわかりやすく解説」
https://itc.tokyo/python/what-is-class/#google_vignette　2024/5/24
④Qiita　「Python の class の使い方をコード例を用いて解説」
https://qiita.com/shi_ei/items/f1c4801bbe1a8bf9ccbf　2024/5/24 アクセス

・ファイルについて
①アルゴリズムのサンプルプログラム
②note.nkmk.me「Python でファイルの読み込み、書き込み（作成・追記）」

https://note.nkmk.me/python-file-io-open-with/　2024/5/24 アクセス
③Qiita「【Python3】ファイル操作の基礎を理解する」
https://qiita.com/tanktop-kun/items/ff5f21ce46c107616692　2024/5/24 アクセス

（６）感想

今回は初めてクラスの内部に関数を導入するというアルゴリズムを採用した。この試みは私にとって恥ずかしながら初でありプログラムの作成に多大な時間がかかったがこの時間が自分自身のプログラミング言語の能力を高めたと自負しておく。このプログラムを書く上で self という言葉を書く場所、そもそも書くのかという事に相当悩まされた。次回以降クラスの導入を行う際はよりスムーズに書けるよう努めたい。また、関数をクラス内に入れるという行為は慣れていないのもあり大変難しかったため、今回は自分に甘えて関数をクラスの外に出すプログラムが妥当だったのではないかと思わざるを得なかった。

また、繰り返し構文を２重で用いるプログラムは簡単なはずだが、時間がかかってしまったのでより練習を積んでいこうと思った。