Junioraufgabe 2: Treffsicherheit

Team-ID: 00025

Team: Christian

Bearbeiter dieser Aufgabe:
Christian Krause

2. November 2021

Inhaltsverzeichnis

Lösungsidee

Das Ziel der Aufgabe ist es, anhand von einer Präferenztabelle, indem die Mitglieder einer Gruppe eingetragen haben, welche Termine ihnen gefallen, herauszufinden, wie viele Einträge in der Präferenztabelle wenigstens verändert werden müssen, sodass der entsprechende Termin ein allseits beliebter Termin wird. Ein Termin ist allseits beliebt, wenn es für jeden Teilnehmer keinen besseren Termin gibt. Um diesen allseits beliebten Termin zu finden wird als erstes für jeden Teilnehmer bestimmt, welche Termine die besten des entsprechenden Teilnehmers sind. Dann wird überprüft, welcher Termin bei den meisten Teilnehmern bereits der bestmögliche Termin ist. Dieser Termin ist der Termin, bei dem am wenigsten Einträge verändert werden müssen, um diesen Termin zu einem allseits beliebten Termin zu machen. Die Anzahl der Teilnehmer, für die dieser Termin noch nicht der bestmögliche Termin ist, entspricht der Anzahl der Veränderungen, da einfach der entsprechende Termin für jeden Teilnehmer, bei dem dieser Termin nicht einer der bestmöglichen Termine ist, auf grün verändert werden muss, da dieser Termin dann zwingend einer der bestmöglichen für den Teilnehmer ist.

Umsetzung

Meine Lösung ist in Python programmiert.

```
import dataloader  # Ein eigenes Modul um Beispieldateien herunterzuladen und
# abzuspeichern.

class Data(dataloader.Dataloader):  # Data Klasse, die von der Dataloader-KLasse des
    # dataloader-Moduls erbt.
    def __init__(self):
        self.Data = {}  # Data-Dictionary, in das die Beispieldateien geladen werden
        super().__init__("Praeferenzen",
           "https://bwinf.de/fileadmin/user_upload/praeferenzen", 6, "Praeferenztabelle")
        # Die Oberklasse wird initialisiert...
        self.check_local_files()  # ...und die Beispieldateien werden geladen.

    def load_file(self, filename, ident):
        """
        Die load_file-Funktion überschreibt eine Funktion der Oberklasse. Die load_file
        Funktion wird aufgerufen, um die Beispieldateien in das Data-Dictionary zu laden.
        :param filename: Dateiname der Beispieldatei
        :param ident: Nummer der Beispieldatei
        """
        f = open(filename)  # Die entsprechende Beispieldatei wird geöffnet...
        file = f.read() # ...geladen...
        filelist = file.split("\n") # ...und zeilenweise in eine Liste aufgeteilt.
        filelist = [[int(j) for j in i.split()] for i in filelist if i != ""]  # Die
        # Elemente dieser Liste werden dann weiter aufgeteilt und die Strings werden zu
        # Zahlenwerten konvertiert.
        members = filelist[0][0]  # Die Anzahl der Mitglieder der Clique...
        Termine = filelist[0][1]  # ...und die Anzahl der Termine wird eingelesen.
        Kalender = {}  # Dann wird ein Kalender-Dictionary erstellt.
        for i in range(members):
            Kalender[i] = filelist[i + 1]  # In dieses Dictionary werden dann die
            # Terminpräferenzen der Mitglieder geschrieben.
        self.Data[ident] = Table(members=members, Termine=Termine, Kalender=Kalender)
        # Dann wird ein Table-Objekt erstellt und in das Data-Dictionary geschrieben.
        f.close()  # Als letztes wird die Beispieldatei wieder geschlossen.
```

Als Erstes wird das dataloader-Modul importiert, das die Beispieldateien herunterläd und abspeichert. Dann wird die Data-Klasse definiert, die das herunterladen und abspeichern der Beispieldateien einleitet und die Beispieldateien im Data-Dictionary mithilfe der Table-Klasse abspeichert.

```
class Table:
    """
    Mit der Table-Klasse können die Informationen der Beispieldateien gespeichert werden.
    Diese Klasse wird später für jede Beispieldatei initialisiert.
    """
    def __init__(self, members, Termine, Kalender):
        self.members = members  # Anzahl der Mitglieder, für die ein Termin gefunden werden
        # muss
        self.Termine = Termine  # Anzahl der Termine, von denen der beste bestimmt werden soll
        self.Kalender = Kalender  # Ein Dictionary, in dem für jeden Teilnehmer eine Liste
        # hinterlegt ist, in der die Terminpräferenzen des Teilnehmers stehen
        self.temp_Kalender = {i[0]: i[1].copy() for i in Kalender.items()}
```

```
        self.beste_Termine = {}  # Ein Dictionary, indem für jeden Teilnehmer dessen
        # bestmögliche Termine hinterlegt sind

    def __getitem__(self, item):
        """
        Diese Funktion ermöglicht es mit dieser Schreibweise 'Table[item]'
        Einträge aus dem Dictionary Kalender abzurufen.
        """
        return self.Kalender[item]

    def check_praeferenzen(self):
        """
        Mit dieser Funktion wird für jeden Teilnehmer mithilfe der 'get_mins()' Funktion
        das 'beste_Temine' Dictionary vervollständigt
        """
        for i in range(self.members):
            self.beste_Termine[i] = get_mins(list(self.Kalender[i]))

    def get_best_Termin(self):
        """
        Diese Funktion ermittelt den Termin, der für die meisten Teilnehmer einer ihrer besten
        Termine ist. Außerdem wird zurückgegeben, für welche (also auch für wie viele)
        Teilnehmer dieser Termin noch nicht einer der bestmöglichen ist.
        """
        count = []  # Zählvariable, um den besten Termin zu ermitteln
        changecount = []  # Liste, in der gespeichert wird, welche Mitglieder ihre
        #  Terminpräferenz für diesen Terminändern müssten, sodass dieser Termin ein allseits
        # beliebter Termin wird.
        for i in range(self.Termine):
            count.append(0)
            changecount.append([])
            for j in self.beste_Termine:
                if i in self.beste_Termine[j][1]:  # Wenn dieser Termin einer der
                    # bestmöglichen des Teilnehmers ist
                    count[i] += 1  # ...dann wird der Zähler für diesen Termin erhöht

                else:  # Wenn dieser Termin nicht einer der bestmöglichen des Teilnehmers ist
                    changecount[i].append(j)  # ...dann wird der 'changecount' Liste an der
                    # Position des aktuellenTermins der index des Teilnehmers angehängt,
                    # sodass dieser Termin gegebenenfalls später verändert werden kann.
        return count.index(max(count)), changecount[count.index(max(count))]  # Der beste
        # Termin und die Veränderungen,die dafür nötig sind, dass der Termin zu einem allseits
        # beliebten Termin wird, werden zurückgegeben.

    def solve(self):
        """
        Diese Funktion findet mithilfe der vorherigen Funktionen den Termin, der für die
        meisten Teilnehmer einer der bestmöglichen Termine ist und ändert die Termine der
        anderen Teilnehmer, sodass der Termin für alle der bestmögliche ist.
        """
        self.check_praeferenzen()  # Zuerst wird die check_praeferenzen-Funktion aufgerufen.
        best_Termin, to_change = self.get_best_Termin()  # Dann wird die get_best_Termin-
        # Funktion aufgerufen.
        for i in to_change:  # Für jeden Teilnehmer in der to_change-Liste...
            self.Kalender[i][best_Termin] = 0  # ...wird der entsprechende Termin auf 0 (also
            # "Passt gut") gesetzt, da dieser dann zwingend einer der bestmöglichen Termine
```

```
        # dieses Teilnehmers ist.
        return best_Termin + 1, len(to_change)  # Hier wird die 'Lösung' der Aufgabe
        # zurückgegeben, also der bestmögliche Termin und die Anzahl der Veränderungen.
```

Für jede Beispieldatei wird eine Instanz der Table-Klasse erstellt, in der die Informationen der Beispieldatei gespeichert sind. Außerdem werden die Funktionen definiert, die für die Lösung des Problems benötigt werden. Als Erstes wird die check_praeferenzen-Funktion definiert, die für jeden Teilnehmer die bestmöglichen Termine ermittelt und abspeichert. Die get_best_Termin-Funktion ermittelt mithilfe diesre Informationen den Termin, der für die meisten Teilnehemer einer der bestmöglichen Termine ist. In der solve-Funktion werden diese Funktionen aufgerufen und es werden die nötigen Änderungen am Kalender vorgenommen.

```
if __name__ == '__main__':
    D = Data()  # Als erstes wird die Data-Klasse initialisiert, also die Dateien werden
    # geladen.
    for i in range(6):  # Dann wird für jede Beispieldatei...
        best_Termin, changes = D.Data[i].solve()  # ...die "solve" Funktion aufgerufen...
        tempsave = "".join([str(D.Data[i][j]) + "\n" for j in range(len(D.Data[i].Kalender))])
        D.load_file(f"Praeferenzen/Praeferenztabelle{i}.txt", i)
        tempsave2 = "".join([str(D.Data[i][j]) +"\n" for j in range(len(D.Data[i].Kalender))])
        save = f"\nBespiel {i}:\n\nKalender urspruenglich: \n{tempsave2} \n\n Kalender“ \
            f"veraendert: \n{tempsave}"
        save += f"\nDer beste Termin ist Termin {best_Termin}.\nDafuer wurden {changes}" \
            f“Eintraege im Kalender veraendert.\n\n"
        # ...und eine Ausgabe erstellt, die dann ausgegeben und abgespeichert wird.
        print(save)
        file = open(f"Ergebnis/praeferenzen{i}.txt", "w")
        file.write(save)
        file.close()
```

In der main-Funtkion wird zuerst die Data-Klasse initialisiert und dann wird für jede Beispieldatei der beste Termin und die dafür nötigen Änderungen ermittelt und es wird eine Ausgabe erstellt, in der auch ersichtlich ist, welche Termine im Kalender verändert werden müssen, um den besten Termin zu einem allseits beliebten Termin zu machen.

Beispiele

0: passt gut; 1: passt nur mäßig; 2: passt gar nicht

Eigene Beispiele

Keine Veränderung nötig

Präferenztabelle:

```
6 7
0 0 0 0 0 0 0
1 0 0 1 1 0 0
2 2 2 1 2 2 1
2 1 1 1 2 1 1
0 1 2 2 1 0 0
1 2 1 2 0 1 0
```

Termin 7 ist bereits ein allsiets beliebter Termin.

Ausgabe:

```
Bespiel 0:

Kalender urspruenglich:
[0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 1, 0, 0]
[2, 2, 2, 1, 2, 2, 1]
[2, 1, 1, 1, 2, 1, 1]
[0, 1, 2, 2, 1, 0, 0]
[1, 2, 1, 2, 0, 1, 0]


 Kalender veraendert:
[0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 1, 0, 0]
[2, 2, 2, 1, 2, 2, 1]
[2, 1, 1, 1, 2, 1, 1]
[0, 1, 2, 2, 1, 0, 0]
[1, 2, 1, 2, 0, 1, 0]

Der beste Termin ist Termin 7.
Dafuer wurden 0 Eintraege im Kalender veraendert.
```

Mehrere allseits beliebte Termine mit gleicher Veränderungsanzahl

Präferenztabelle:

```
6 7
0 0 0 0 0 0 0
1 0 0 1 1 0 0
2 2 2 2 2 2 2
2 1 1 1 2 1 1
0 1 2 2 1 0 0
1 2 1 2 0 1 1
```

Für Termin 6 und 7 ist jeweils eine Veränderung notwendig, um die Termine zu einem allseits beliebten Termin zu machen. In solchen Fällen wird der erste (von links) allseits beliebte Termin mit den wenigsten Veränderungen ausgegeben.

Ausgabe:

```
Bespiel 0:

Kalender urspruenglich:
[0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 1, 0, 0]
[2, 2, 2, 2, 2, 2, 2]
[2, 1, 1, 1, 2, 1, 1]
[0, 1, 2, 2, 1, 0, 0]
[1, 2, 1, 2, 0, 1, 1]


Kalender veraendert:
```

```
[0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 1, 0, 0]
[2, 2, 2, 2, 2, 2, 2]
[2, 1, 1, 1, 2, 1, 1]
[0, 1, 2, 2, 1, 0, 0]
[1, 2, 1, 2, 0, 0, 1]

Der beste Termin ist Termin 6.
Dafuer wurden 1 Eintraege im Kalender veraendert.
```

Beipsiele von der Bwinf-Webseite

Beispieldatei 0 *(https://bwinf.de/fileadmin/user_upload/praeferenzen0.txt):*

```
Bespiel 0:

Kalender urspruenglich:
[0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 1, 0, 0]
[2, 2, 2, 1, 2, 2, 2]
[2, 1, 1, 1, 2, 1, 2]
[0, 1, 2, 2, 1, 0, 0]
[1, 2, 1, 2, 0, 1, 1]


Kalender veraendert:
[0, 0, 0, 0, 0, 0, 0]
[1, 0, 0, 1, 1, 0, 0]
[2, 2, 2, 1, 2, 0, 2]
[2, 1, 1, 1, 2, 1, 2]
[0, 1, 2, 2, 1, 0, 0]
[1, 2, 1, 2, 0, 0, 1]

Der beste Termin ist Termin 6.
Dafuer wurden 2 Eintraege im Kalender veraendert.
```

Beispieldatei 1 (https://bwinf.de/fileadmin/user_upload/praeferenzen1.txt):

```
Bespiel 1:

Kalender urspruenglich:
[0, 0, 0, 0, 0]
[1, 1, 2, 2, 1]
[2, 1, 1, 1, 1]
[2, 2, 0, 1, 1]
[2, 0, 0, 1, 1]


Kalender veraendert:
[0, 0, 0, 0, 0]
[1, 1, 2, 2, 1]
[2, 1, 1, 1, 1]
```

```
[2, 0, 0, 1, 1]
[2, 0, 0, 1, 1]

Der beste Termin ist Termin 2.
Dafuer wurden 1 Eintraege im Kalender veraendert.
```

Beispieldatei 2 (https://bwinf.de/fileadmin/user_upload/praeferenzen2txt):

```
Bespiel 2:

Kalender urspruenglich:
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[2, 2, 2, 0, 1, 1, 2, 0, 0, 1]
[1, 1, 0, 0, 1, 1, 1, 0, 0, 1]
[2, 2, 2, 1, 2, 1, 1, 2, 1, 1]
[1, 1, 1, 1, 2, 2, 1, 1, 2, 1]
[1, 1, 2, 0, 1, 1, 1, 1, 1, 2]
[2, 1, 2, 0, 2, 2, 2, 2, 0, 2]
[1, 2, 1, 0, 2, 1, 2, 1, 2, 2]


Kalender veraendert:
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[2, 2, 2, 0, 1, 1, 2, 0, 0, 1]
[1, 1, 0, 0, 1, 1, 1, 0, 0, 1]
[2, 2, 2, 1, 2, 1, 1, 2, 1, 1]
[1, 1, 1, 1, 2, 2, 1, 1, 2, 1]
[1, 1, 2, 0, 1, 1, 1, 1, 1, 2]
[2, 1, 2, 0, 2, 2, 2, 2, 0, 2]
[1, 2, 1, 0, 2, 1, 2, 1, 2, 2]

Der beste Termin ist Termin 4.
Dafuer wurden 0 Eintraege im Kalender veraendert.
```

Beispieldatei 3 (https://bwinf.de/fileadmin/user_upload/praeferenzen3.txt):

```
Bespiel 3:

Kalender urspruenglich:
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 2, 1, 1, 2, 1, 2, 2, 1, 2, 0, 2, 1, 2, 2, 1]
[2, 2, 1, 0, 1, 1, 1, 0, 2, 1, 0, 1, 1, 1, 2, 0, 0, 2, 2, 1]
[1, 1, 0, 1, 2, 2, 2, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 1, 2]
[2, 2, 1, 1, 2, 0, 0, 2, 0, 0, 1, 2, 2, 1, 2, 1, 0, 1, 1, 2]
[1, 2, 1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 0, 2, 0]
[0, 2, 2, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 1, 1, 2, 1, 2]
[1, 1, 2, 2, 1, 1, 1, 2, 0, 1, 0, 1, 1, 1, 2, 1, 1, 0, 0, 1]
[0, 0, 2, 1, 2, 0, 1, 2, 2, 1, 1, 2, 1, 1, 2, 0, 1, 2, 0, 1]
[1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 2, 2, 2]
```

```
[2, 1, 1, 1, 2, 1, 1, 1, 0, 2, 1, 2, 0, 1, 0, 1, 0, 0, 2, 0]
[2, 1, 1, 2, 1, 1, 0, 1, 2, 1, 1, 1, 2, 1, 1, 0, 1, 0, 2, 0]
[0, 1, 0, 1, 2, 1, 1, 2, 2, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1]
[0, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 2, 1, 0, 2]


Kalender veraendert:
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 1, 0, 2, 1, 1, 2, 1, 2, 2, 1, 2, 0, 2, 1, 0, 2, 1]
[2, 2, 1, 0, 1, 1, 1, 0, 2, 1, 0, 1, 1, 1, 2, 0, 0, 0, 2, 1]
[1, 1, 0, 1, 2, 2, 2, 2, 1, 1, 1, 2, 1, 1, 1, 1, 1, 0, 1, 2]
[2, 2, 1, 1, 2, 0, 0, 2, 0, 0, 1, 2, 2, 1, 2, 1, 0, 0, 1, 2]
[1, 2, 1, 1, 2, 1, 2, 2, 1, 2, 1, 2, 2, 1, 1, 1, 1, 0, 2, 0]
[0, 2, 2, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 2, 2, 1, 1, 0, 1, 2]
[1, 1, 2, 2, 1, 1, 1, 2, 0, 1, 0, 1, 1, 1, 2, 1, 1, 0, 0, 1]
[0, 0, 2, 1, 2, 0, 1, 2, 2, 1, 1, 2, 1, 1, 2, 0, 1, 0, 0, 1]
[1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 0, 2, 2]
[2, 1, 1, 1, 2, 1, 1, 1, 0, 2, 1, 2, 0, 1, 0, 1, 0, 0, 2, 0]
[2, 1, 1, 2, 1, 1, 0, 1, 2, 1, 1, 1, 2, 1, 1, 0, 1, 0, 2, 0]
[0, 1, 0, 1, 2, 1, 1, 2, 2, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1]
[0, 1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 2, 2, 1, 1, 1, 2, 0, 0, 2]

Der beste Termin ist Termin 18.
Dafuer wurden 7 Eintraege im Kalender veraendert.
```

Beispieldatei 4 (https://bwinf.de/fileadmin/user_upload/praeferenzen4.txt):

```
Bespiel 4:

Der beste Termin ist Termin 22.
Dafuer wurden 14 Eintraege im Kalender veraendert.
```

(Ausgabe des Kalenders zu groß)

Beispieldatei 5 (https://bwinf.de/fileadmin/user_upload/praeferenzen5.txt):

```
Bespiel 5:

Der beste Termin ist Termin 31.
Dafuer wurden 34 Eintraege im Kalender veraendert.
```

(Ausgabe des Kalenders zu groß)

Quellcode

Laufzeit: ca.30 ms (ohne herunterladen der Beispieldateien)(Pycharm cProfile profiler)

```python
import dataloader  # Ein eigenes Modul um Beispieldateien herunterzuladen und abzuspeichern.


class Data(dataloader.Dataloader):  # Data Klasse, die von der Dataloader-KLasse des dataloader-Moduls erbt.
    def __init__(self):
```

```
        self.Data = {}  # Data-Dictionary, in das die Beispieldateien geladen werden
        super().__init__("Praeferenzen", "https://bwinf.de/fileadmin/user_upload/praeferenzen", 6,
            "Praeferenztabelle")
        # Die Oberklasse wird initialisiert...
        self.check_local_files()  # ...und die Beispieldateien werden geladen.

    def load_file(self, filename, ident):
        """
        Die load_file-Funktion überschreibt eine Funktion der Oberklasse. Die load_file Funktion wird aufgerufen, um
        die Beispieldateien in das Data-Dictionary zu laden.
        :param filename: Dateiname der Beispieldatei
        :param ident: Nummer der Beispieldatei
        """
        f = open(filename)  # Die entsprechende Beispieldatei wird geöffnet...
        file = f.read() # ...geladen...
        filelist = file.split("\n") # ...und zeilenweise in eine Liste aufgeteilt.
        filelist = [[int(j) for j in i.split()] for i in filelist if i != ""]  # Die Elemente dieser Liste werden
        # dann weiter aufgeteilt und die Strings werden zu Zahlenwerten konvertiert.
        members = filelist[0][0]  # Die Anzahl der Mitglieder der Clique...
        Termine = filelist[0][1]  # ...und die Anzahl der Termine wird eingelesen.
        Kalender = {}  # Dann wird ein Kalender-Dictionary erstellt.
        for i in range(members):
            Kalender[i] = filelist[i + 1]  # In dieses Dictionary werden dann die Terminpräferenzen der Mitglieder
            # geschrieben.
        self.Data[ident] = Table(members=members, Termine=Termine, Kalender=Kalender)  # Dann wird ein Table-Objekt
        # erstellt und in das Data-Dictionary geschrieben.
        f.close()  # Als letztes wird die Beispieldatei wieder geschlossen.


def get_mins(List):
    """
    Die Funktion get_mins ist dafür zuständig, die Positionen der kleinsten Zahlen (→ Beste Termine) in einer Liste
    zu bestimmen.
    :param List: Liste an Terminpräferenzen
    :return: Liste mit den Positionen der besten Termine in dieser Liste
    """
    mn = min(List)  # Der kleinste Wert, der in der Liste vorkommt wird bestimmt.
    ret = [mn,  # Die Liste 'ret' wird später zurückgegeben, sie enthält den Wert der kleinsten Zahlen
           []]  # und eine Liste der Positionen der kleinsten Zahlen
    for i in range(len(List)):
        try:  # Hier wird versucht mit der '.index' Funktion die Position der kleinsten Werte zu bestimmen
            index = List.index(
                mn)  # Die '.index' geht von links nach rechts durch die Liste und gibt die erste Position
            # von einem der kleinen Werte zurück
            ret[1].append(index)  # Die Position des kleinsten Wertes wird dann der zweiten Liste in ret angehängt
            List[index] = None  # Diese Position wird dann in der temporären Liste auf 'None' gesetzt,
            # sodass diese Position nicht nochmal ermittelt wird.
        except ValueError:  # Wenn es diesen Wert in der Liste nicht mehr gibt, wird der Fehler hier abgefangen
            break  # und die Liste wird beendet.
    return ret  # Die fertige Liste aus dem minimum und dessen Positionen wird zurückgegeben.


class Table:
    """
    Mit der Table-Klasse können die Informationen der Beispieldateien gespeichert werden.
    Diese Klasse wird später für jede Beispieldatei initialisiert.
    """
    def __init__(self, members, Termine, Kalender):
        self.members = members  # Anzahl der Mitglieder, für die ein Termin gefunden werden muss
        self.Termine = Termine  # Anzahl der Termine, von denen der beste bestimmt werden soll
        self.Kalender = Kalender  # Ein Dictionary, in dem für jeden Teilnehmer eine Liste hinterlegt ist,
        # in der die Terminpräferenzen des Teilnehmers stehen
        self.temp_Kalender = {i[0]: i[1].copy() for i in Kalender.items()}
```

```
        self.beste_Termine = {}  # Ein Dictionary, indem für jeden Teilnehmer dessen bestmögliche Termine hinterlegt
        # sind

    def __getitem__(self, item):
        """
        Diese Funktion ermöglicht es mit dieser Schreibweise 'Table[item]'
        Einträge aus dem Dictionary Kalender abzurufen.
        """
        return self.Kalender[item]

    def check_praeferenzen(self):
        """
        Mit dieser Funktion wird für jeden Teilnehmer mithilfe der 'get_mins()' Funktion
        das 'beste_Temine' Dictionary vervollständigt
        """
        for i in range(self.members):
            self.beste_Termine[i] = get_mins(list(self.Kalender[i]))

    def get_best_Termin(self):
        """
        Diese Funktion ermittelt den Termin, der für die meisten Teilnehmer einer ihrer besten Termine ist.
        Außerdem wird zurückgegeben, für welche (also auch für wie viele) Teilnehmer dieser Termin noch nicht einer
        der bestmöglichen ist.
        """
        count = []  # Zählvariable, um den besten Termin zu ermitteln
        changecount = []  # Liste, in der gespeichert wird, welche Mitglieder ihre Terminpräferenz für diesen Termin
        # ändern müssten, sodass dieser Termin ein allseits beliebter Termin wird.
        for i in range(self.Termine):
            count.append(0)
            changecount.append([])
            for j in self.beste_Termine:
                if i in self.beste_Termine[j][1]:  # Wenn dieser Termin einer der bestmöglichen des Teilnehmers ist
                    count[i] += 1  # ...dann wird der Zähler für diesen Termin erhöht

                else:  # Wenn dieser Termin nicht einer der bestmöglichen des Teilnehmers ist
                    changecount[i].append(j)  # ...dann wird der 'changecount' Liste an der Position des aktuellen
                    # Termins der index des Teilnehmers angehängt, sodass dieser Termin gegebenenfalls später
                    # verändert werden kann.
        return count.index(max(count)), changecount[count.index(max(count))]  # Der beste Termin und die
        # Veränderungen,die dafür nötig sind, dass der Termin zu einem allseits beliebten Termin wird, werden
        # zurückgegeben.

    def solve(self):
        """
        Diese Funktion findet mithilfe der vorherigen Funktionen den Termin, der für die meisten Teilnehmer einer der
        bestmöglichen Termine ist und ändert die Termine der anderen Teilnehmer, sodass der Termin für alle der
        bestmögliche ist.
        """
        self.check_praeferenzen()  # Zuerst wird die check_praeferenzen-Funktion aufgerufen.
        best_Termin, to_change = self.get_best_Termin()  # Dann wird die get_best_Termin-Funktion aufgerufen

        for i in to_change:  # Für jeden Teilnehmer in der to_change-Liste...
            self.Kalender[i][best_Termin] = 0  # ...wird der entsprechende Termin auf 0 (also "Passt gut") gesetzt,
            # da dieser dann zwingend einer der bestmöglichen Termine dieses Teilnehmers ist.
        return best_Termin + 1, len(to_change)  # Hier wird die 'Lösung' der Aufgabe zurückgegeben,
        # also der bestmögliche Termin und die Anzahl der Veränderungen.


if __name__ == '__main__':
    D = Data()  # Als erstes wird die Data-Klasse initialisiert, also die Dateien werden geladen.
    for i in range(6):  # # Dann wird für jede Beispieldatei...
        best_Termin, changes = D.Data[i].solve()  # ...die "solve" Funktion aufgerufen...
        tempsave = "".join([str(D.Data[i][j]) + "\n" for j in range(len(D.Data[i].Kalender))])
```

```
    D.load_file(f"Praeferenzen/Praeferenztabelle{i}.txt", i)
    tempsave2 = "".join([str(D.Data[i][j]) + "\n" for j in range(len(D.Data[i].Kalender))])
    save = f"\nBespiel {i}:\n\nKalender urspruenglich: \n{tempsave2} \n\nKalender veraendert: \n{tempsave}"
    save += f"\nDer beste Termin ist Termin {best_Termin}.\nDafuer wurden {changes} Eintraege im Kalender " \
            f"veraendert.\n\n"
    # ...und eine Ausgabe erstellt, die dann ausgegeben und abgespeichert wird.
    print(save)
    file = open(f"Ergebnis/praeferenzen{i}.txt", "w")
    file.write(save)
    file.close()
```