Маратулы Темирболат

Используя рассмотренный на уроке пример, добавьте:

1. В класс Events дополнительную функциональность контейнера – словаря (время – ключ, событие – значение списка)
2. Реализовать программу, заданную на прошлом уроке, с использованием шаблона MVC.
3. В класс DiaryNotes дополнительную функциональность итератора, при итерации возвращать список строк в формат:

[[дата, [событие, событие, событие]], [дата, [, событие, событие]]]

class Events:

    def \_\_init\_\_(self):

        self.\_\_time = []

        self.\_\_details = []

    def get\_time\_details(self,time):

        if(time in self.\_\_time):

            key = self.\_\_time.index(time)

            return self.\_\_details[key]

        else:

            raise ValueError

    def \_\_getitem\_\_(self,time):

        try:

            return self.get\_time\_details(time)

        except ValueError:

            raise IndexError

    def \_\_setitem\_\_(self,key,value):

        try:

            key = self.\_\_time.index(key)

            self.\_\_details[key] = value

        except Exception:

            self.\_\_time.append(key)

            self.\_\_details.append(value)

    def \_\_str\_\_(self):

        i = 0

        comma = ', '

        str\_ = ""

        while i < len(self.\_\_time):

            if(i == len(self.\_\_time) - 1):

                comma = ' '

            str\_ += self.\_\_time[i] + \

                    " | " + \

                    self.\_\_details[i] + comma

            i+=1

        return str\_

    def \_\_iter\_\_(self):

        self.\_\_i = 0

        return self

    def \_\_next\_\_(self):

        if self.\_\_i > len(self.\_\_time)-1:

            raise StopIteration

        else:

            event = self.\_\_time[self.\_\_i] + \

                    "| " + \

                    self.\_\_details[self.\_\_i]

            self.\_\_i += 1

            return event

class DiaryNote:

    def \_\_init\_\_(self):

        self.\_\_dates = []

        self.\_\_events = []

    def add\_date(self,dates):

        try:

            return self.\_\_dates.index(dates)

        except ValueError:

            self.\_\_dates.append(dates)

            self.\_\_events.append(Events())

            return self.\_\_dates.index(dates)

    def add\_event(self,dates,time,detail):

        key = self.add\_date(dates)

        self.\_\_events[key][time] = detail

    def get\_dates(self):

        dates = ""

        for i in self.\_\_dates:

            dates += i + "\n"

        return dates

    def get\_day\_events(self,date):

        key = self.\_\_dates.index(date)

        # print(key)

        return self.\_\_events[key]

    def \_\_getitem\_\_(self,date):

        try:

            return self.get\_day\_events(date)

        except ValueError:

            raise IndexError

    def \_\_iter\_\_(self):

        self.\_\_index = 0

        return self

    def \_\_next\_\_(self):

        if(self.\_\_index > len(self.\_\_dates) - 1):

            raise StopIteration

        else:

            result = '[' + self.\_\_dates[self.\_\_index] + ', [' + str(self.\_\_events[self.\_\_index]) + ']'

            self.\_\_index += 1

            return result

    def \_\_len\_\_(self):

        return len(self.\_\_dates)

class View:

    def input\_date(self):

        return input('Пожалуйста, введите дату события: ')

    def input\_time(self):

        return input('Пожалуйста, введите время события: ')

    def input\_detail(self):

        return input('Пожалуйста, введите событие: ')

    def view\_options(self,options):

        print()

        option\_number = 1

        for option in options:

            print(option\_number,option)

            option\_number += 1

    def input\_user\_choice(self):

        return int(input('Ваш выбор: '))

    def view\_wrong\_message(self,error):

        print('Ошибка:',error)

    def view\_buy\_message(self):

        print('Спасибо за использование программы! До свидания!')

    def view\_date\_events(self,date,events):

        print('События для даты "{}":'.format(date))

        print(events)

    def view\_all\_dairy\_note\_info(self,dairy\_note):

        print('[',end = "")

        comma = ', '

        current\_date = 1

        for date in dairy\_note:

            if(current\_date == len(dairy\_note)):

                comma = ''

            print(date,end = "]{}".format(comma))

            current\_date += 1

        print(']')

class Controller:

    view = View()

    diary\_note = DiaryNote()

    menu\_options = ['Создать событие','Просмотреть события данного дня','Просмотреть все события','Выйти из программы']

    def create\_event(self):

        event\_date = self.view.input\_date()

        event\_time = self.view.input\_time()

        event\_details = self.view.input\_detail()

        self.diary\_note.add\_event(event\_date,event\_time,event\_details)

    def view\_date\_events(self):

        event\_date = self.view.input\_date()

        self.view.view\_date\_events(event\_date, self.diary\_note[event\_date])

    def view\_all\_dates\_events(self):

        self.view.view\_all\_dairy\_note\_info(self.diary\_note)

    menu\_functions = [create\_event,view\_date\_events,view\_all\_dates\_events]

    def main\_program(self):

        while(True):

            try:

                self.view.view\_options(self.menu\_options)

                user\_choice = self.view.input\_user\_choice()

                if(user\_choice <= 0 or user\_choice > len(self.menu\_options)):

                    self.view.view\_wrong\_message('такой опции нет!')

                    continue

                elif(user\_choice == len(self.menu\_options)):

                    self.view.view\_buy\_message()

                    break

                self.menu\_functions[user\_choice - 1](self)

            except ValueError as ex:

                # print(type(ex))

                self.view.view\_wrong\_message('Вам надо написать число, а не что то другое')

            except IndexError as ex:

                # print(type(ex))

                self.view.view\_wrong\_message('Извините, но таких данных нет')

            except Exception as ex:

                # print(type(ex))

                self.view.view\_wrong\_message(ex)

controller = Controller()

controller.main\_program()



