Документация

Для хранения справочника используется файл с расширением .csv, что позволяет работать с ним, как с таблицей данных, сам таблица представлена в виде словаря с ключами: «Name», «Surname», «Birth Date», «Age», «Mobile Phone», «Work Phone», «Home Phone». Обязательные поля для заполнения: имя, фамилия, мобильный телефон, остальные - опциональные.

```
Enter a command from the list:
view (1) - view the whole table
search (2) - search and output according to certain parameters
add (3) - add a new entry
edit (4) - edit an existing entry
view by age (5) - record output based on age
birthday list (6) - birthdays next month
delete by name and surname (7) - delete a record
delete by phone (8) - delete (a) record(s)
quit (9)
```

В начале работы программы пользователю высвечивается список всех возможных функций программы, в числе которых:

- **View**: позволяет вывести весь телефонный справочник
- **Search**: дает возможность поиска записей по одному или нескольким параметрам (имя, фамилия, дата рождения, возраст, мобильный, рабочий и домашние телефоны)
- **Add**: добавление новой записи в справочник
- **Edit**: изменение параметров (имени, фамилии, даты рождения и др.) существующей записи
- **View by age**: позволяет выводить записи, которые по возрасту младше/старше/равны некоторому количеству лет
- **Birthday list**: имена, фамилии и даты рождения тех, у кого в ближайшие 30 дней дни рождения
- **Delete by name and surname**: позволяет удалять запись по имени и фамилии
- **Delete by phone**: дает возможность удалить одну или сразу несколько записей по номеру телефона
- **Quit**: завершает работу программы

Теперь подробнее о каждой из этих функций.

View.

```
if command.casefold() == "view" or command == '1':
    view_database()
```

Функция вызывается из main, если при просьбе программы ввести команду, переменная command получила строку *view* (регистр неважен) или *1*.

```
database_path = "data.csv"
def view_database():
    df = pd.read_csv(database_path)
    pd.set_option('display.max_columns', 20)
    pd.set_option('display.width', 2000)
    print(df)
```

Тогда вызывается функция *view_database()* и выводит таблицу данных из файла с путем, хранящимся в *database_path* (переменная глобальная). *set_option* необходимы для того, чтобы таблица выводилась полностью.

Пример вывода:

```
view
          Name     Surname  Birth Date Age  Mobile Phone   Work Phone   Home Phone
0    Ekaterina  Konovalova  02/03/2001  19   89203099899            -  88314122663
1         Igor      Krylov           -   -   89038900777  89213029999            -
2         Olga  Petryakova  11/04/2003  17   89042300202            -            -
3         Petr      Ivanov  11/07/1990  30   89053211211            -  89042311212
4         Igor    Stepanov  07/09/1989  31   89765644444            -            -
5        Alena     Krylova  04/12/1980  39   89542099090            -  88314121632
6        Alina   Troyanova           -   -   89073211212            -  88319022323
7         Igor    Tregubov  12/12/2004  15   89700709898            -            -
8     Kristina   Troyanova  31/12/2009  10   89067099090            -            -
9         Egor      Petrov  01/01/1989  31   89500232323            -            -
10      Karina      Ponova  01/01/2003  17   89500232323            -            -
11      Sergey     Nechaev           -   -   89500232323            -            -
12      Alexey    Nikolaev  18/03/1992  28   89692032323            -            -
13      Karina     Gromova           -   -   89077655656  89056766556            -
14  Konstantin      Petrov  12/11/2001  19   89032021212            -            -
15       Boris      Petrov  11/06/1999  21   89600755454            -            -
16       Roman      Ivanov  11/03/2001  19   89032022222            -  88314121632
17        Olga     Krylova  11/08/2000  20   89200471122            -            -
```

Search.

```
elif command.casefold() == "search" or command == '2':
    search()
```

Функция вызывается из main, если при просьбе программы ввести команду, переменная command получила строку *search* (регистр неважен) или *2*.

```
def search():
    # выбор ключей
    ind = input("By what parameters to search?\nname (1), surname (2), birth date (3), age (4), mobile phone "
                "(5), work phone (6), home phone (7)\nUse , if you want to delete several (f.e. 2,7,8): ")
    while (ind.replace(',', '')).isdigit() == 0:
        ind = input("Wrong format, try again: ")
    list = ind.split(',')
```

Вызывается функция *search()* и просит пользователя через запятую (иначе будет ошибка и нужно будет вводить параметры заново) ввести номера параметров, по которым будет производится поиск записей.

После того, как пользователь правильно ввел номера, начинает работать цикл. Отрывок из цикла выглядит таким образом:

```
for i in list:
    if i == '1':
        name = input("Enter the name: ")
        name = name_correctness(name, 0)
        name = (name.lower()).title()
        df = df[df["Name"] == name]
```

В каждом случае программа будет писать, что пользователю нужно ввести, в данном случае ввести требуется имя (регистр неважен). Для того, чтобы проверить правильность введенного имени (использование запрещенных символов), используется функция *name_correctness()*. Затем из таблицы данных отбираются строки, которые удовлетворяют параметру.

Пример работы функции:

```
SEARCH
By what parameters to search?
name (1), surname (2), birth date (3), age (4), mobile phone (5), work phone (6), home phone (7)
Use , if you want to delete several (f.e. 2,7,8): 2,4
Enter the surname: PETROV
Enter the age: 31
   Name Surname  Birth Date Age  Mobile Phone Work Phone Home Phone
9  Egor  Petrov  01/01/1989  31   89500232323          -          -
```

Add.

```
elif command.casefold() == "add" or command == '3':
    name = input("Please, enter name: ")
    surname = input("Please, enter surname: ")
    add(name, surname)
```

Функция вызывается из main, если при просьбе программы ввести команду, переменная command получила строку *add* (регистр неважен) или *3*.
В этом случае пользователя просят ввести имя и фамилию.

```
def add(name, surname):
    # вызывает фукнцию проверки на уникальность комбинации имени и фамилии
    name, surname = exist(name, surname)
```

Имя и фамилия проверяются на уникальность с помощью функции *exist()*.

Функция *exist()* в свою очередь проверяет правильность имени и фамилии и использует функцию *in_row()*, которая возвращает -1 в случае, если такой комбинации в справочнике нет, либо номер строки, в которой комбинация содержится.
В случае, если такой человек уже есть, предлагается либо попробовать ввести новые имя и фамилию, либо изменить у существующего пользователя какие-то параметры.

```
def exist(name, surname):
    name = name_correctness(name, 0)
    surname = name_correctness(surname, 1)
    if in_row(name, surname) != -1:
        command = input("This person is already in the phonebook, do you want to try again? (Yes/No)")
        if command.casefold() == 'yes':
            name_extra = input("Please, enter name: ")
            surname_extra = input("Please, enter surname: ")
            name, surname = exist(name_extra, surname_extra)
            return name, surname
        else:
            command = input("Do you want to edit the record with these name and surname? (Yes/No)")
            if command.casefold() == 'yes':
                edit_row(name, surname)
            return 'no', 'addition'
    return name, surname
```

Если такого человека нет в справочнике, то пользователя просят ввести обязательные поля (мобильный телефон) и другие на выбор (выбор заключается в ответе на вопрос (да/нет), хотя на деле программа работает так: если не «да», то «нет»), один отрывок работы функции приведен на рисунке.
Все номера телефонов проверяются на правильность с помощью функции *phone_fmt()*. Разрешено вводить номера телефонов в форматах: XXXXXXXXXX, 8XXXXXXXXXX, +7XXXXXXXXXX, иначе нужно будет вводить номер еще раз. Правильность введенной даты рождения - с помощью *date_fmt(),* которая проверяет и на правильность формата (ДД/ММ/ГГГГ или ДДММГГГГ), и на правильность самой даты (т.е. нельзя ввести, например, 30 февраля).

```
answer = input("Do you want to write a birth date? (Yes/No) ")
if answer.casefold() == 'yes':
    birth_date = input("Enter the birth date (DD/MM/YYYY or DDMMYYYY): ")
    birth_date = date_fmt(birth_date)
    age = age_from(birth_date)
mob_phone = input("Enter the mobile phone, please: ")
mob_phone = phone_fmt(mob_phone)
```

Пример работы программы:

```
add
Please, enter name: 8__d
Please, enter surname: *d
You used wrong symbols in the name, try again
Please, enter name: ivan i
You used wrong symbols in the surname, try again
Please, enter surname: loganov
Do you want to write a birth date? (Yes/No) yes
Enter the birth date (DD/MM/YYYY or DDMMYYYY): 13/07/2000
Enter the mobile phone, please: 9064543232
Do you want to write a work phone? (Yes/No) no
Do you want to write a home phone? (Yes/No) no
```

Edit.

Функция вызывается из main, если при просьбе программы ввести команду, переменная command получила строку *edit* (регистр неважен) или *4*. Пользователя просят ввести имя и фамилию, чтобы затем изменить поля у этой записи. Затем проверяется, есть ли такая запись. Если запись не найдена, тогда предлагается сохранить в справочник нового человека или ввести имя и фамилию заново.

```
def edit_row(name, surname):
    i = in_row(name, surname)
    # если запись не найдена, тогда предлагаем пользователю сохранить в справочник нового человека
    # или ввести имя и фамилию заново
    if i == -1:
        answer = input("This person isn't in the phonebook, do you want to add him/her? (Yes/No): ")
        if answer.casefold() == 'yes':
            add(name, surname)
        else:
            answer = input("Do you want to try editing again? (Yes/No): ")
            if answer.casefold() == 'yes':
                name_extra = input("Please, enter name: ")
                surname_extra = input("Please, enter surname: ")
                edit_row(name_extra, surname_extra)
```

Если человек найден, запускается цикл и предлагается выбрать один параметр, который пользователь хочет изменить. Цикл работает до того момента, пока пользователь не введет *none* или *7*.

```
command = input("What do you want to edit? Choose one: name (1), surname (2), birth date (3), mobile phone (4),"
                " work phone (5), home phone (6), none (7)\n")
```

Например, одним из вариантов может быть изменение даты рождения. Проверяется правильность введенной даты, а также запускается функция *age_from()*, которая высчитывает, сколько человеку лет на сегодняшний день.
В конце работы функции все обновленные данные заново сохраняются.

```
elif command.casefold() == 'birth date' or command == '3':
    birth_date = input("Enter the birth date (DD/MM/YYYY or DDMMYYYY) or cancel (1): "
    birth_date = date_fmt(birth_date)
    age = age_from(birth_date)
    df.loc[i, "Birth Date"] = birth_date
    df.loc[i, "Age"] = age
```

Интересно, что в программе также присутствует функция *age_update()*. Она запускается вначале работы программы и обновляет возраст всех людей в таблице данных (если она не пустая).

```
def age_update():
    df = pd.read_csv(database_path)
    for index, row in df.iterrows():
        if row[2] != '-':
            df.loc[index, 'Age'] = age_from(row[2])
    df.to_csv(database_path, index=False)
```

Пример работы функции:

```
Please, enter name: karina
Please, enter surname: ponova
Name                Karina
Surname             Ponova
Birth Date      01/01/2003
Age                     17
Mobile Phone    89500232323
Work Phone                -
Home Phone                -
Name: 10, dtype: object
What do you want to edit? Choose one: name (1), surname (2), birth date (3), mobile phone (4), work phone (5), home phone (6), none (7)
6
Enter the home phone, please, or cancel (1): 89079088765
What do you want to edit? Choose one: name (1), surname (2), birth date (3), mobile phone (4), work phone (5), home phone (6), none (7)
```

View by age

Функция вызывается из main, если при просьбе программы ввести команду, переменная command получила строку *view by age* (регистр неважен) или *5*.

```python
def view_age():
    age = input("Enter the age: ")
    if '0' < age <= '9':
        age = '0' + age
    while age.isdigit() == 0:
        age = input("It is not a number, try again or cancel and turn back to the command menu (0): ")
    if age != '0':
        df = pd.read_csv(database_path)
        command = input("Do you want to view people who are: younger (1), older (2), this age (3): ")
        if command == '1' or command.casefold() == 'younger':
            print(df.loc[(df.Age < str(age)) & (df.Age != '-')])
        elif command == '2' or command.casefold() == 'older':
            print(df.loc[(df.Age > str(age)) & (df.Age != '-')])
        elif command == '3' or command.casefold() == 'this age':
            print(df[df["Age"] == str(age)])
```

Функция просит ввести возраст (отличный от нуля, так как он отвечает за выход из функции).
Если это не число, то запускается цикл, пока пользователь не введет *0* или число, с которым будет сравниваться возраст.
Переменная *age* является строкой, чтобы не возникало ошибки в сравнении, необходимо к возрасту от 1 до 9 прибавлять впереди 0, тогда функция работает корректно.
Далее пользователя просят ввести младше, старше или такого возраста людей из справочника нужно вывести.

Пример работы функции:

```
5
Enter the age: 18
Do you want to view people who are: younger (1), older (2), this age (3): 2
          Name     Surname  Birth Date Age  Mobile Phone Work Phone   Home Phone
0    Ekaterina  Konovalova  02/03/2001  19   89203099899          -  88314122663
3         Petr      Ivanov  11/07/1990  30   89053211211          -  89042311212
4         Igor    Stepanov  07/09/1989  31   89765644444          -           -
5        Alena     Krylova  04/12/1980  39   89542099090          -  88314121632
9         Egor      Petrov  01/01/1989  31   89500232323          -           -
12      Alexey    Nikolaev  18/03/1992  28   89692032323          -           -
14  Konstantin      Petrov  12/11/2001  19   89032021212          -           -
15       Boris      Petrov  11/06/1999  21   89600755454          -           -
16       Roman      Ivanov  11/03/2001  19   89032022222          -  88314121632
17        Olga     Krylova  11/08/2000  20   89200471122          -           -
```

Birthday list.

```python
def birthday_list():
    with open(database_path, 'r') as file:
        df = pd.read_csv(file)
        today = (datetime.datetime.today()).strftime('%Y%m%d')
        month_later = (datetime.datetime.today() + datetime.timedelta(days=30)).strftime('%Y%m%d')
        year = (datetime.datetime.today()).strftime('%Y')
        ind = []
        for index, row in df.iterrows():
            if row[2] != '-':
                if today <= year + row[2][3:5] + row[2][0:2] < month_later:
                    ind.append(index)
        print(df.loc[ind, "Name":"Birth Date"])
```

Функция вызывается из main, если при просьбе программы ввести команду, переменная command получила строку *birthday list* (регистр неважен) или *6*.
Программа выводит всех, у кого день рождения в ближайшие 30 дней. Для этого сравниваются даты рождения людей с сегодняшним днем и датой, которая будет через месяц. Важно учесть, что на стыке двух лет важно учитывать еще и год, поэтому я прибавляю сегодняшний год к месяцу и дню рождения пользователей.

Пример вывода программы (выводится имя, фамилия и дата рождения):

```
6
       Name    Surname  Birth Date
6      Igor   Tregubov  12/12/2004
7  Kristina  Troyanova  31/12/2009
```

Delete by name and surname.

Функция вызывается из main, если при просьбе программы ввести команду, переменная command получила строку *delete by name and surname* (регистр неважен) или *7*.

```
def delete_row(name, surname):
    i = in_row(name, surname)
    if i == -1:
        answer = input("There is not anyone with these identifiers. Do you want to try again? (Yes/No): ")
        if answer.casefold() == 'yes':
            name_extra = input("Please, enter name: ")
            surname_extra = input("Please, enter surname: ")
            delete_row(name_extra, surname_extra)
    else:
        df = pd.read_csv(database_path)
        df = df.drop(i)
        df.to_csv(database_path, index=False)
        print(df)
```

Пользователя просят ввести имя и фамилию, вызывается функция *delete_row()*, а за ней *in_row()*, которая вернет номер строки с такими параметрами.
Если такой строки нет, пользователя спрашивают, хочет ли он попробовать еще раз.
Если есть, тогда это строка удаляется и выводится обновленная таблица. Затем таблица переписывается и индексы после удаленной строки сдвигаются.

Пример:

```
7
Please, enter name: na
Please, enter surname: name
There is not anyone with these identifiers. Do you want to try again? (Yes/No): yes
Please, enter name: alena
Please, enter surname: krylova
          Name     Surname  Birth Date Age  Mobile Phone   Work Phone   Home Phone
0    Ekaterina  Konovalova  02/03/2001  19   89203099899            -  88314122663
1         Igor      Krylov           -   -   89038900777  89213029999            -
2         Olga  Petryakova  11/04/2003  17   89042300202            -            -
3         Petr      Ivanov  11/07/1990  30   89053211211            -  89042311212
4         Igor    Stepanov  07/09/1989  31   89765644444            -            -
6        Alina   Troyanova           -   -   89073211212            -  88319022323
7         Igor    Tregubov  12/12/2004  15   89700709898            -            -
8     Kristina   Troyanova  31/12/2009  10   89067099090            -            -
9         Egor      Petrov  01/01/1989  31   89500232323            -            -
10      Karina      Ponova  01/01/2003  17   89500232323  89066666643  89079088765
11      Sergey     Nechaev           -   -   89500232323            -            -
12      Alexey    Nikolaev  18/03/1992  28   89692032323            -            -
13      Karina     Gromova           -   -   89077655656  89056766556            -
14  Konstantin      Petrov  12/11/2001  19   89032021212            -            -
15       Boris      Petrov  11/06/1999  21   89600755454            -            -
16       Roman      Ivanov  11/03/2001  19   89032022222            -  88314121632
17        Olga     Krylova  11/08/2000  20   89200471122            -            -
```

Delete by phone.

Функция вызывается из main, если при просьбе программы ввести команду, переменная command получила строку *delete by phone* (регистр неважен) или *8*. Пользователя просят ввести мобильный телефон, а затем вызывается сама функция.
Функция delete_by_phone() вызывает phone_in(), которая возвращает индексы всех строк с таким мобильным телефоном.
Затем пользователю показывают все строки с данным телефоном и предлагают выбрать индексы тех строк, которые нужно удалить (выбирать индексы можно только те, которые выведены на экране).
Пример функции:

```
def delete_by_phone(mob_phone):
    list = phone_in(mob_phone)
    df = pd.read_csv(database_path)
    # выводит все записи с данным номером и предлагает выбрать, какие удалить
    print(df.loc[list])
    if len(list) != 0:
        ind = input("Which rows do you want to delete? Enter indexes of these rows\n"
                    "Use , if you want to delete several (f.e. 2,7,8): ")
        while (ind.replace(',', '')).isdigit() == 0:
            ind = input("Wrong format, try again: ")
        ind = ind.split(',')
        for i in ind:
            if int(i) in list:
                df = df.drop(int(i))
                print("Successfully delete the", i, "row")
            else:
                print("Cannot delete", i, "row because it is not in the given table")
        df.to_csv(database_path, index=False)
        print(df)
```

```
Which rows do you want to delete? Enter indexes of these rows
Use , if you want to delete several (f.e. 2,7,8): 8,10
Successfully delete the, 8 row
Successfully delete the, 10 row
          Name     Surname  Birth Date Age  Mobile Phone   Work Phone   Home Phone
0    Ekaterina  Konovalova  02/03/2001  19   89203099899            -  88314122663
1         Igor      Krylov           -   -   89038900777  89213029999            -
2         Olga  Petryakova  11/04/2003  17   89042300202            -            -
3         Petr      Ivanov  11/07/1990  30   89053211211            -  89042311212
4         Igor    Stepanov  07/09/1989  31   89765644444            -            -
5        Alina   Troyanova           -   -   89073211212            -  88319022323
6         Igor    Tregubov  12/12/2004  15   89700709898            -            -
7     Kristina   Troyanova  31/12/2009  10   89067099090            -            -
9       Karina      Ponova  01/01/2003  17   89500232323  89066666643  89079088765
11      Alexey    Nikolaev  18/03/1992  28   89692032323            -            -
12      Karina     Gromova           -   -   89077655656  89056766556            -
13  Konstantin      Petrov  12/11/2001  19   89032021212            -            -
14       Boris      Petrov  11/06/1999  21   89600755454            -            -
15       Roman      Ivanov  11/03/2001  19   89032022222            -  88314121632
16        Olga     Krylova  11/08/2000  20   89200471122            -            -
```

Quit.

Завершает работу цикла, а также самой программы.

```
quit

Process finished with exit code 0
```