

Домашня робота №11/12

Sergiy Ponomarenko

27 лютого 2023 р.

<https://www.edu.goit.global/ru/learn/8016550/5594293/5615318/homework>

1 Вигляд роботи програми

```
>>> hello
How can I help you?
>>> search S
```

Name	Birthday	Phones
Sergiy	12.02.1569	2341241323
Skirt	02.02.1991	9431287135
Snack	-	3897016452
Spade	-	8249367105
Sable	19.07.2008	7201634985

```
Here are the found contacts
>>> show all
```

Name	Birthday	Phones
Roast	01.01.1990	7219542064
Skirt	02.02.1991	9431287135
Chord	03.03.1992	5871693280
Tasty	04.04.1993	3798124615
Gloom	05.05.1994	8024793119
Fluke	06.06.1995	2146518723
Tease	-	2159834871
Bumpy	-	5019478326
Brisk	-	3576012389
Plush	-	6782934015

```
Press <Enter> to continue...
```

Name	Birthday	Phones
Haste	-	8304695127
Snack	-	3897016452
Blunt	-	1629405837
Spade	-	8249367105
Tonic	-	5983721046
Chive	16.04.2005	2708193541
Flair	17.05.2006	6859247310
Quell	18.06.2007	4681739250
Sable	19.07.2008	7201634985
Vigor	20.08.2009	8352406179

```
Address book contain 30 contacts
>>>
```

2 Файл README.MD

```
1 # Домашнє завдання №12
2
3
4 - Записи `Record` у `AddressBook` зберігаються як значення у словнику.
5 В якості ключів використовується значення `Record.name.value`.
6 - `Record` зберігає об'єкт <Name> в окремому атрибуті.
7 - `Record` зберігає список об'єктів `Phone` в окремому атрибуті.
8 - `Record` реалізує методи додавання/видалення/редагування об'єктів `Phone`.
9 - `AddressBook` реалізує метод `add_record`, який додає <Record> у `self.data`.
10
11
12 ## Команди
13 - `hello` --- чат вітається.
14 - `set birthday` -- встановлює дату народження контакту у форматі `DD.MM.YYY`, наприклад `set birthday Sergiy`
   ↳ `12.12.1978`.
15 - `birthday of` -- Виводить на екран дату вказаного контакту, наприклад `birthday of Sergiy`.
16 - `add` --- чат додає ім'я і телефон, приклад `add Sergiy 0936564532`.
17 - `chage` --- чат змінює номер для відповідного контакту, приклад `change Sergiy 0936564532 0634564545`.
18 - `phones` --- чат виводить номери телефонів контакту, приклад `phone Sergiy`.
19 - `show all N` --- чат показує усі контакти та їх номери, приклад `show all 10`. Необов'язковий параметр `N` -
   ↳ число записів, що виводяться за одну ітерацію.
20 - `remove` --- чат видаляє запис з вказаним іменем, приклад `remove Sergiy`.
21 - `good bye`, `good`, `exit` --- чат прощається і завершує роботу і зберігає контакти у файл `contacts.json`.
22 - `.` --- чат перериває свою роботу без попереджень і зберігає контакти у файл `contacts`.
23 - `save` --- зберігає контакти у файл `.json`, наприклад `save contacts`.
24 - `load` --- завантажує книгу з контактами з файлу в чат, наприклад `load contacts`.
25 - `search` -- здійснює пошук за ключовою фразою, частиною номеру телефона чи дні народження, наприклад `search`
   ↳ `123`, або `search Beth`.
26 - `export` -- Експортує дані в формат `csv`, приклад `export somefile`.
27 - `import` -- Імпортує дані з формату `csv`, приклад `impott somefile`.
28
29
30
31
32 COMMANDS = {
33     "hello": hello,
34     "set birthday": set_birthday,
35     "birthday of": birthday,
36     "add": add,
37     "change": change,
38     "phones of": phones,
39     "show all": show_all,
40     "remove": remove,
41     "good bye": good_bye,
42     "close": good_bye,
43     "exit": good_bye,
44     "save": save,
45     "load": load,
46     "search": search,
47     "export": export_to_csv,
48     "import": import_from_csv,
49 }
```

3 Реалізація класів

```
1 import re
2 from collections import UserDict
3
4 import pickle
5 import csv
6 from datetime import datetime
7
8
9 # ===== Classes =====#
10
```

```

11
12 """
13 - Записи <Record> у <AddressBook> зберігаються як значення у словнику.
14 - В якості ключів використовується значення <Record.name.value>.
15 - <Record> зберігає об'єкт <Name> в окремому атрибуті.
16 - <Record> зберігає список об'єктів <Phone> в окремому атрибуті.
17 - <Record> реалізує методи додавання/видалення/редагування об'єктів <Phone>.
18 - <AddressBook> реалізує метод <add_record>, який додає <Record> у <self.data>.
19
20 """
21
22
23 class Field:
24     """Клас є батьківським для всіх полів, у ньому реалізується логіка,
25     загальна для всіх полів."""
26
27     def __init__(self, value: str):
28         self.__value = value
29         self.value = value
30
31     @property
32     def value(self):
33         return self.__value
34
35     def __eq__(self, other):
36         return self.value == other.value
37
38
39 class Name(Field):
40     """Клас --- обов'язкове поле з ім'ям."""
41
42     @Field.value.setter
43     def value(self, value):
44         self.__value = value
45
46
47 class Phone(Field):
48     """Клас -- необов'язкове поле з телефоном та таких один записів (Record)
49     може містити кілька."""
50
51     @Field.value.setter
52     def value(self, value):
53         if not bool(re.match(r"\d{10}", value)) or len(value) > 10:
54             raise ValueError("Phone number must be 10 digits")
55         self.__value = value
56
57
58 class Birthday(Field):
59     """Клас -- необов'язкове поле з датою народження."""
60
61     @Field.value.setter
62     def value(self, value):
63         try:
64             date = datetime.strptime(value, "%d.%m.%Y")
65         except (TypeError, ValueError):
66             raise ValueError("Invalid date format. Please use DD.MM.YYYY")
67         if date > datetime.today():
68             raise ValueError("Date cannot be in the future")
69         self.__value = date
70
71
72 class Record:
73     """Клас відповідає за логіку додавання/видалення/редагування

```

```

74     необов'язкових полів та зберігання обов'язкового поля Name. """
75
76     def __init__(
77         self,
78         name: Name,
79         phones: list[Phone] = None,
80         birthday: Birthday = None,
81     ):
82
83         self.name = name # Name --- атрибут для зберігання об'єкту Name
84         self.phones = phones or []
85         self.birthday = birthday
86
87     def add_birthday(self, birthday: Birthday):
88         """Метод додає об'єкт день народження до запису. """
89
90         self.birthday = birthday
91
92     def add_phone(self, phone: Phone):
93         """Метод додає об'єкт телефон до запису. """
94
95         self.phones.append(phone)
96
97     def remove_phone(self, phone: Phone):
98         """Метод видаляє об'єкт телефон із запису. """
99
100        self.phones.remove(phone)
101
102    def change_phone(self, old_phone: Phone, new_phone: Phone) -> bool:
103        """Метод змінює об'єкт телефон в записі на новий. """
104
105        for phone in self.phones:
106            if phone == old_phone:
107                self.phones.remove(phone)
108                self.phones.append(new_phone)
109                return True
110            return False
111
112    def show_phones(self):
113
114        phones = ", ".join(phone.value for phone in self.phones) or "-"
115        return phones
116
117    def show_birthday(self):
118
119        birthday = getattr(self.birthday, "value", None) or "-"
120        return birthday
121
122        name = getattr(self.name, "value", False)
123        return name
124
125    def days_to_birthday(self) -> int:
126        """Метод повертає кількість днів до наступного дня народження контакту. """
127
128        if not self.birthday:
129            return None
130
131        today = datetime.today()
132        dt_birthday = datetime.strptime(self.birthday.value, "%d.%m.%Y")
133        next_birthday = dt_birthday.replace(year=today.year)
134
135        if next_birthday < today:
136            next_birthday = dt_birthday.replace(year=today.year + 1)

```

```

137         return (next_birthday - today).days
138
139
140
141 class AddressBook(UserDict):
142     """Клас містить логіку пошуку за записами до цього класу."""
143
144     def add_record(self, record: Record):
145         """Метод додає запис до списку контактів."""
146
147         self.data[record.name.value] = record
148
149     def save_contacts(self, filename):
150         with open(filename, "wb") as file:
151             pickle.dump(list(self.data.items()), file)
152
153     def load_contacts(self, filename):
154         with open(filename, "rb") as file:
155             data = pickle.load(file)
156             self.clear()
157             self.update(data)
158
159     def search(self, search_string):
160         """Метод шукає записи по ключовому слову."""
161
162         results = AddressBook()
163         for key in self.data:
164             record = self.data[key]
165             if (
166                 search_string in record.name.value
167                 or (
168                     getattr(record.birthday, "value", False)
169                     and search_string in record.birthday.value
170                 )
171                 or any(search_string in phone.value for phone in record.phones)
172             ):
173                 results.add_record(record)
174         return results
175
176     def export_to_csv(self, filename):
177         """Експорт записів із AddressBook в CSV-файл"""
178
179         with open(filename, "w", newline="") as csvfile:
180             fieldnames = ["name", "birthday", "phones"]
181             writer = csv.DictWriter(
182                 csvfile, fieldnames=fieldnames, delimiter="|"
183             )
184             writer.writeheader()
185             for record in self.data.values():
186                 writer.writerow(
187                     {
188                         "name": record.name.value,
189                         "phones": record.show_phones(),
190                         "birthday": record.show_birthday(),
191                     }
192                 )
193
194     def import_from_csv(self, filename):
195         """Імпорт записів із CSV-файлу в AddressBook"""
196
197         with open(filename, newline="") as csvfile:
198             reader = csv.DictReader(csvfile, delimiter="|")
199             for row in reader:

```

```

200         name = row["name"]
201         phones = [
202             Phone(phone.strip())
203             for phone in row["phones"].split(",")
204             if phone.strip() and row["phones"] != "-"
205         ]
206         birthday = (
207             Birthday(row["birthday"])
208             if row["birthday"] != "-"
209             else None
210         )
211         record = Record(
212             name=Name(name), phones=phones, birthday=birthday
213         )
214         self.add_record(record)
215
216     def iterator(self, n: int = 10):
217         """Метод ітерується по записам і виводить їх частинами по n-штук."""
218
219         data_items = list(self.data.items())
220         for i in range(0, len(data_items), n):
221             data_slice = dict(data_items[i : i + n])
222             yield data_slice
223             if i + n < len(data_items):
224                 yield "continue"

```

4 Реалізація основної програми

```

1 import re
2 from prettytable import PrettyTable
3 from botmodule import Name, Phone, Birthday, Record, AddressBook
4
5 # ===== Tables decoration =====#
6
7
8 def build_table(data):
9     """Функція строить PrettyTable для заданного списка записей."""
10    table = PrettyTable()
11    table.field_names = ["Name", "Birthday", "Phones"]
12    table.min_width.update({"Name": 20, "Birthday": 12, "Phones": 40})
13    data = AddressBook(data)
14    for key in data:
15        record = data[key]
16        name = record.name.value
17        birthday = record.show_birthday()
18        phones = record.show_phones()
19        table.add_row([name, birthday, phones])
20    return table
21
22
23 # ===== Decorator =====#
24
25
26 def input_error(func):
27     def wrapper(*func_args, **func_kwargs):
28         try:
29             return func(*func_args, **func_kwargs)

```

```

30     except KeyError as error:
31         return "\033[1;31m{}\033[0m".format(str(error).strip("'"))
32     except ValueError as error:
33         return f"\033[1;31m{str(error)}\033[0m"
34     except TypeError as error:
35         return f"\033[1;31m{str(error)}\033[0m"
36     except FileNotFoundError:
37         return "\033[1;31mFile not found\033[0m"
38
39     return wrapper
40
41
42 # ===== handlers =====#
43
44
45 def hello(*args):
46     return "\033[32mHow can I help you?\033[0m"
47
48
49 def good_bye(*args):
50     contacts.save_contacts("contacts")
51     return "Good bye!"
52
53
54 def undefined(*args):
55     return "\033[32mWhat do you mean?\033[0m"
56
57
58 def show_all(*args):
59     """Функція-handler показує книгу контактів."""
60     return f"Address book contain {len(contacts)} contacts"
61
62
63 @input_error
64 def save(*args):
65     contacts.save_contacts(args[0])
66     return f"File {args[0]} saved"
67
68
69 @input_error
70 def load(*args):
71     contacts.load_contacts(args[0])
72     return f"File {args[0]} loaded"
73
74
75 @input_error
76 def set_birthday(*args):
77     """Функція-handler додає день народження до контакту."""
78
79     if not args[0] or args[0].isdigit():
80         raise KeyError("Give me a name, please")
81     if not args[1]:
82         raise ValueError("Give me a date, please")
83
84     name, birthday = Name(args[0]), Birthday(args[1])
85
86     if name.value in contacts.data:
87         record = contacts.data[name.value]
88     else:
89         record = Record(name)
90         contacts.add_record(record)
91     record.add_birthday(birthday)
92

```

```

93     return f"I added a birthday {args[1]} to contact {args[0]}"
94
95
96 @input_error
97 def add(*args):
98     """Додає телефонний номер в контакт по імені."""
99
100     if not args[0]:
101         raise KeyError("Give me a name, please")
102
103     if not args[1]:
104         raise ValueError("Give me a phone, please")
105
106     name, phone = Name(args[0]), Phone(args[1])
107
108     if name.value in contacts.data:
109         record = contacts.data[name.value]
110     else:
111         record = Record(name)
112         contacts.add_record(record)
113     record.add_phone(phone)
114
115     return f"I added a phone {args[1]} to contact {args[0]}"
116
117
118 @input_error
119 def phones(*args):
120     """Функція-handler показує телефонні номери відповідного контакту."""
121
122     if not args[0]:
123         raise KeyError("Give me a name, please")
124
125     if args[0] not in contacts.keys():
126         raise ValueError("Contact does not in AddressBook")
127
128     table = PrettyTable()
129     table.field_names = ["Name", "Phones"]
130     table.min_width.update({"Name": 20, "Phones": 55})
131
132     phones = contacts.get(args[0]).show_phones() or "-"
133     table.add_row([args[0], phones])
134
135     return f"\033[0m{table}"
136
137
138 @input_error
139 def birthday(*args):
140     """Функція-handler показує день народження та кількість днів до наступного."""
141
142     if not args[0]:
143         raise KeyError("Give me a name, please")
144
145     table = PrettyTable()
146     table.field_names = ["Name", "Birthday", "Days to next Birthday"]
147     table.min_width.update(
148         {"Name": 20, "Birthday": 12, "Days to next Birthday": 40}
149     )
150
151     if args[0] not in contacts.keys():
152         raise ValueError("Contact does not in AddressBook")
153
154     days_to_next_birthday = contacts.data[args[0]].days_to_birthday() or "-"
155     birthday = contacts.get(args[0]).show_birthday() or "-"

```



```

156     table.add_row([args[0], birthday, days_to_next_birthday])
157
158
159     return f"\033[0m{table}"
160
161
162 @input_error
163 def search(*args):
164
165     if not args[0]:
166         raise KeyError("Give me a some string, please")
167
168     results = contacts.search(args[0])
169
170     if results:
171         return f"\033[0m{build_table(results)}"
172     return "By your request found nothing"
173
174
175 @input_error
176 def remove(*args):
177     """Функція-handler видаляє запис з книги."""
178
179     if not args[0]:
180         raise KeyError("Give me a name, please")
181
182     name = Name(args[0])
183
184     del contacts[name.value]
185
186     return f"Contact {name.value} was removed"
187
188
189 @input_error
190 def change(*args):
191     """Функція-handler змінює телефон контакту."""
192
193     if not args[0]:
194         raise KeyError("Give me a name, please")
195
196     if not args[1]:
197         raise ValueError("Old phone number is required")
198
199     if not args[2]:
200         raise ValueError("New phone number is required")
201
202     name = Name(args[0])
203     old_phone = Phone(args[1])
204     new_phone = Phone(args[2])
205
206     if name.value not in contacts:
207         return f"Contact {name.value} not found"
208
209     contact_list = contacts[name.value].phones
210     for number in contact_list:
211         if number == old_phone:
212             idx = contact_list.index(number)
213             contact_list[idx] = new_phone
214             break
215     return f"Phone {old_phone.value} not found for {name.value}"
216
217     return f"Contact {name.value} with phone number {old_phone.value} was updated with new
↵ phone number {new_phone.value}"

```

```

218
219
220 @input_error
221 def export_to_csv(*args):
222     if not args[0]:
223         raise TypeError("Set file name, please")
224     contacts.export_to_csv(args[0])
225     return f"File {args[0]} exported to csv"
226
227
228 @input_error
229 def import_from_csv(*args):
230     contacts.import_from_csv(args[0])
231     return f"File {args[0]} imported from csv"
232
233
234 def help_commands(*args):
235     """Функція показує перелік всіх команд."""
236
237     table = PrettyTable()
238     table.field_names = ["Command"]
239     table.min_width.update({"Command": 20})
240
241     for command in COMMANDS.keys():
242         table.add_row([command])
243
244     return f"\033[0m{table}"
245
246
247 # ===== handler loader =====#
248
249 COMMANDS = {
250     "help": help_commands,
251     "hello": hello,
252     "set birthday": set_birthday,
253     "birthday of": birthday,
254     "add": add,
255     "change": change,
256     "phones of": phones,
257     "show all": show_all,
258     "remove": remove,
259     "good bye": good_bye,
260     "close": good_bye,
261     "exit": good_bye,
262     "save": save,
263     "load": load,
264     "search": search,
265     "export": export_to_csv,
266     "import": import_from_csv,
267 }
268
269
270 def get_handler(*args):
271     """Функція викликає відповідний handler."""
272     return COMMANDS.get(args[0], undefined)
273
274
275 # ===== main function =====#
276
277 contacts = AddressBook() # Global variable for storing contacts
278
279
280 def main():

```

```

281 table = PrettyTable()
282 table.field_names = ["Name", "Birthday", "Phones"]
283 table.min_width.update({"Name": 20, "Birthday": 12, "Phones": 40})
284
285 command_pattern = "|".join(COMMANDS.keys())
286 pattern = re.compile(
287     r"\b(\.|" + command_pattern + r")\b"
288     r"(?:\s+([a-zA-Z0-9\.]�)?)?"
289     r"(?:\s+(\d{1,2}\.\d{1,2}\.\d{4}|\d{1,})?)?"
290     r"(?:\s+(\d+)?)?",
291     re.IGNORECASE,
292 )
293
294
295 load("contacts")
296 while True:
297     # waiting for nonempty input
298     while True:
299         inp = input(">>> ").strip()
300         if inp == "":
301             continue
302         break
303
304     text = pattern.search(inp)
305
306     params = (
307         tuple(
308             map(
309                 # Made a commands to be a uppercase
310                 lambda x: x.lower() if text.groups().index(x) == 0 else x,
311                 text.groups(),
312             )
313         )
314         if text
315         else (None, 0, 0)
316     )
317     handler = get_handler(*params)
318     response = handler(*params[1:])
319     if inp.strip() == ".":
320         contacts.save_contacts("contacts")
321         return
322     if params[0] == "show all":
323         param = (
324             int(params[1])
325             if params[1] is not None
326             and isinstance(params[1], str)
327             and params[1].isdigit()
328             else 100
329         )
330         for tab in contacts.iterator(param):
331             if tab == "continue":
332                 input("\033[1;32mPress <Enter> to continue...\033[0m")
333             else:
334                 print(build_table(tab))
335
336     print(f"\033[1;32m{response}\033[0m")
337     if response == "Good bye!":
338         return
339
340
341 # ===== main program ===== #
342
343 if __name__ == "__main__":

```

344

345

```
main()
```