Домашня робота №11/12

Sergiy Ponomarenko

26 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

```
# Домашнє завдання №10


- Записи `Record` у `AddressBook` зберігаються як значення у словнику.
  В якості ключів використовується значення `Record.name.value`.
- `Record` зберігає об'єкт <Name> в окремому атрибуті.
- `Record` зберігає список об'єктів `Phone` в окремому атрибуті.
- `Record` реалізує методи додавання/видалення/редагування об'єктів `Phone`.
- `AddressBook` реалізує метод `add_record`, який додає <Record> у `self.data`.


## Команди
- `hello` --- чат вітається.
- `set birthday` -- встановлює дату народження контакуту у форматі  `DD.MM.YYY`, наприклад `set birthday Sergiy 12.12.1978`.
- `birthday of` -- Виводить на екран дату вказаного контакту, наприклад `birthday of Sergiy`.
- `add` --- чат додає ім'я і телефон, приклад `add Sergiy 0936564532`.
- `chage` --- чат змінює номер для відповідного контакту, приклад `change Sergiy 0936564532 0634564545`.
- `phones` --- чат виводить номери телефонів контакту, приклад `phone Sergiy`.
- `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.
- `remove` --- чат видаляє запис з вказаним іменем, приклад `remove Sergiy`.
- `good bye`, `good`, `exit` --- чат прощається і завершує роботу і зберігає контакти у файл `contacts.json`.
- `.` --- чат перериває свою роботу без попереджень і зберігає контакти у файл `contacts.json`.
- `save` --- зберігає контакти у файл `.json`, наприклад `save contacts`.
- `load` --- завантажує книгу з контактами з файлу `.json` в чат, наприклад `load contacts`.
- `search` -- здійснює пошук за ключовою фразою, частиною номеру телефона чи дні народження, наприклад `search 123`, або `search Beth`.


```
COMMANDS = {
    "hello": hello,
    "set birthday": set_birthday,
    "birthday of": birthday,
    "add": add,
    "change": change,
    "phones of": phones,
    "show all": show_all,
    "remove": remove,
    "good bye": good_bye,
    "close": good_bye,
    "exit": good_bye,
    "save": save,
    "load": load,
    "search": search,
}
```
```

3 Реалізація класів

```python
import re
from collections import UserDict

import pickle
from datetime import datetime


# ================================== Classes ==================================#


"""
- Записи <Record> у <AddressBook> зберігаються як значення у словнику.
  В якості ключів використовується значення <Record.name.value>.
```

```
- <Record> зберігає об'єкт <Name> в окремому атрибуті.
- <Record> зберігає список об'єктів <Phone> в окремому атрибуті.
- <Record> реалізує методи додавання/видалення/редагування об'єктів <Phone>.
- <AddressBook> реалізує метод <add_record>, який додає <Record> у <self.data>.

"""


class Field:
    """Клас є батьківським для всіх полів, у ньому реалізується логіка,
    загальна для всіх полів."""

    def __init__(self, value: str):
        self.__value = value
        self.value = value

    @property
    def value(self):
        return self.__value

    def __eq__(self, other):
        return self.value == other.value


class Name(Field):
    """Клас --- обов'язкове поле з ім'ям."""

    @Field.value.setter
    def value(self, value):
        self.__value = value


class Phone(Field):
    """Клас -- необов'язкове поле з телефоном та таких один записів (Record)
    може містити кілька."""

    @Field.value.setter
    def value(self, value):
        if not bool(re.match(r"\d{10}", value)):
            raise ValueError("Phone number must be 10 digits")
        self.__value = value


class Birthday(Field):
    """Клас -- необов'язкове поле з датою народження."""

    @Field.value.setter
    def value(self, value):
        try:
            date = datetime.strptime(value, "%d.%m.%Y")
        except (TypeError, ValueError):
            raise ValueError("Invalid date format. Please use DD.MM.YYYY")
        if date > datetime.today():
            raise ValueError("Date cannot be in the future")
        self.__value = date


class Record:
    """Клас відповідає за логіку додавання/видалення/редагування
    необов'язкових полів та зберігання обов'язкового поля Name."""

    def __init__(
        self,
```

```
        name: Name,
        phones: list[Phone] = None,
        birthday: Birthday = None,
    ):

        self.name = name  # Name --- атрибут ля зберігання об'єкту Name
        self.phones = phones or []
        self.birthday = birthday

    def add_birthday(self, birthday: Birthday):
        """Метод додає об'єкт день народження до запису."""

        self.birthday = birthday

    def add_phone(self, phone: Phone):
        """Метод додає об'єкт телефон до запису."""

        self.phones.append(phone)

    def remove_phone(self, phone: Phone):
        """Метод видаляє об'єкт телефон із запису."""

        self.phones.remove(phone)

    def change_phone(self, old_phone: Phone, new_phone: Phone) -> bool:
        """Метод змінює об'єкт телефон в записі на новий."""

        for phone in self.phones:
            if phone == old_phone:
                self.phones.remove(phone)
                self.phones.append(new_phone)
                return True
            return False

    def show_phones(self):

        phones = ", ".join(phone.value for phone in self.phones) or "-"
        return phones

    def show_birthday(self):

        birthday = getattr(self.birthday, "value", None) or "-"
        return birthday

    def days_to_birthday(self) -> int:
        """Метод повертає кількість днів до наступного дня народження контакту."""

        if not self.birthday:
            return None

        today = datetime.today()
        dt_birthday = datetime.strptime(self.birthday.value, "%d.%m.%Y")
        next_birthday = dt_birthday.replace(year=today.year)

        if next_birthday < today:
            next_birthday = dt_birthday.replace(year=today.year + 1)

        return (next_birthday - today).days


class AddressBook(UserDict):
    """Клас містить логіку пошуку за записами до цього класу."""

```

```
    def add_record(self, record: Record):
        """Метод додає запис до списку контактів."""

        self.data[record.name.value] = record

    def save_contacts(self, filename):
        with open(filename, "wb") as file:
            pickle.dump(list(self.data.items()), file)

    def load_contacts(self, filename):
        with open(filename, "rb") as file:
            data = pickle.load(file)
            self.clear()
            self.update(data)

    def search(self, search_string):
        """Метод шукає записи  по ключовому слову."""

        if not search_string:
            search_string = ''

        results = AddressBook()
        for key in self.data:
            record = self.data[key]
            if (
                search_string in record.name.value
                or (
                    getattr(record.birthday, "value", False)
                    and search_string in record.birthday.value
                )
                or any(search_string in phone.value for phone in record.phones)
            ):
                results.add_record(record)
        return results

    def iterator(self, n: int):
        """Метод ітерується по записам і виводить їх частинами по n-штук."""

        data_items = list(self.data.items())
        for i in range(0, len(data_items), n):
            data_slice = dict(data_items[i: i + n])
            yield data_slice
            if i + n < len(data_items):
                yield "continue"
```

4 Реалізація основної програми

```
import re
from prettytable import PrettyTable
from botmodule import Name, Phone, Birthday, Record, AddressBook

# ============================ Tables decoration ============================#


def build_table(data):
    """Функция строит PrettyTable для заданного списка записей."""
    table = PrettyTable()
```

```
    table.field_names = ["Name", "Birthday", "Phones"]
    table.min_width.update({"Name": 20, "Birthday": 12, "Phones": 40})
    data = AddressBook(data)
    for key in data:
        record = data[key]
        name = record.name.value
        birthday = record.show_birthday()
        phones = record.show_phones()
        table.add_row([name, birthday, phones])
    return table


# ================================== Decorator ==================================#


def input_error(func):
    def wrapper(*func_args, **func_kwargs):
        try:
            return func(*func_args, **func_kwargs)
        except KeyError as error:
            return "\033[31m{}\033[0m".format(str(error).strip("'"))
        except ValueError as error:
            return f"\033[31m{str(error)}\033[0m"
        except TypeError as error:
            return f"\033[31m{str(error)}\033[0m"
        except FileNotFoundError:
            return "\033[31mFile not found\033[0m"

    return wrapper


# ================================== handlers ==================================#


def hello(*args):
    return "\033[32mHow can I help you?\033[0m"


def good_bye(*args):
    contacts.save_contacts("contacts")
    return "\033[32mGood bye!\033[0m"


def undefined(*args):
    return "\033[32mWhat do you mean?\033[0m"


def show_all(*args):
    """Функция-handler показує книгу контактів."""
    return f"Address book contain {len(contacts)} contacts"


@input_error
def save(*args):
    contacts.save_contacts(args[0])
    return f"File {args[0]} saved"


@input_error
def load(*args):
    contacts.load_contacts(args[0])
    return f"File {args[0]} loaded"

```

```python

@input_error
def set_birthday(*args):
    """Функція-handler додає день народження до контакту."""

    if not args[0]:
        raise KeyError("Give me a name, please")
    if not args[1]:
        raise ValueError("Give me a date, please")

    name, birthday = Name(args[0]), Birthday(args[1])

    if name.value in contacts.data:
        record = contacts.data[name.value]
    else:
        record = Record(name)
        contacts.add_record(record)
    record.add_birthday(birthday)

    return f"I added a birthday {args[1]} to contact {args[0]}"


@input_error
def add(*args):
    """Добавляет телефонный номер в контакт по имени."""

    if not args[0]:
        raise KeyError("Give me a name, please")

    if not args[1]:
        raise ValueError("Give me a phone, please")

    name, phone = Name(args[0]), Phone(args[1])

    if name.value in contacts.data:
        record = contacts.data[name.value]
    else:
        record = Record(name)
        contacts.add_record(record)
    record.add_phone(phone)

    return f"I added a phone {args[1]} to contact {args[0]}"


@input_error
def phones(*args):
    """Функція-handler показує телефонні номери відповідного контакту."""

    if not args[0]:
        raise KeyError("Give me a name, please")

    table = PrettyTable()
    table.field_names = ["Name", "Phones"]
    table.min_width.update({"Name": 20, "Phones": 55})

    phones = contacts.get(args[0]).show_phones() or "-"
    table.add_row([args[0], phones])

    return table


@input_error
def birthday(*args):
```

```
137     """Функція-handler показує день народження та кількість днів до наступного."""
138
139     if not args[0]:
140         raise KeyError("Give me a name, please")
141
142     table = PrettyTable()
143     table.field_names = ["Name", "Birthday", "Days to next Birthday"]
144     table.min_width.update(
145         {"Name": 20, "Birthday": 12, "Days to next Birthday": 40}
146     )
147
148     days_to_next_birthday = contacts.data[args[0]].days_to_birthday() or "-"
149     birthday = contacts.get(args[0]).show_birthday() or "-"
150
151     table.add_row([args[0], birthday, days_to_next_birthday])
152
153     return table
154
155     # return "No such contach founded"
156
157
158 def search(*args):
159     return "Here are the found contacts"
160
161
162 @input_error
163 def remove(*args):
164     """Функція-handler видаляє запис з книги."""
165
166     if not args[0]:
167         raise KeyError("Give me a name, please")
168
169     name = Name(args[0])
170
171     del contacts[name.value]
172
173     return f"Contact {name.value} was removed"
174
175
176 @input_error
177 def change(*args):
178     """Функція-handler змінює телефон контакту."""
179
180     if not args[0]:
181         raise KeyError("Give me a name, please")
182
183     if not args[1]:
184         raise ValueError("Old phone number is required")
185
186     if not args[2]:
187         raise ValueError("New phone number is required")
188
189     name = Name(args[0])
190     old_phone = Phone(args[1])
191     new_phone = Phone(args[2])
192
193     if name.value not in contacts:
194         return f"Contact {name.value} not found"
195
196     contact_list = contacts[name.value].phones
197     for number in contact_list:
198         if number == old_phone:
199             idx = contact_list.index(number)
```

```
            contact_list[idx] = new_phone
            break
        return f"Phone {old_phone.value} not found for {name.value}"

    return f"Contact {name.value} with phone number {old_phone.value} was updated with new phone number {new_phone.value}"


# ================================ handler loader ==============================#

COMMANDS = {
    "hello": hello,
    "set birthday": set_birthday,
    "birthday of": birthday,
    "add": add,
    "change": change,
    "phones of": phones,
    "show all": show_all,
    "remove": remove,
    "good bye": good_bye,
    "close": good_bye,
    "exit": good_bye,
    "save": save,
    "load": load,
    "search": search,
}


def get_handler(*args):
    """Функція викликає відповідний handler."""
    return COMMANDS.get(args[0], undefined)


# ================================ main function ==============================#

contacts = AddressBook()  # Global variable for storing contacts


def main():

    table = PrettyTable()
    table.field_names = ["Name", "Birthday", "Phones"]
    table.min_width.update({"Name": 20, "Birthday": 12, "Phones": 40})

    command_pattern = "|".join(COMMANDS.keys())
    pattern = re.compile(
        r"\b(\.|" + command_pattern + r")\b"
        r"(?:\s+([a-zA-Z0-9\.]+))?"
        r"(?:\s+(\d+|\d{1,2}\.\d{1,2}\.\d{4}(?:\.\d{2})?))?"
        r"(?:\s+(\d+)?)?",
        re.IGNORECASE,
    )

    load("contacts")
    while True:
        # waiting for nonempty input
        while True:
            inp = input(">>> ").strip()
            if inp == "":
                continue
            break

        text = pattern.search(inp)
```

```
262
263         params = (
264             tuple(
265                 map(
266                     # Made a commands to be a uppercase
267                     lambda x: x.lower() if text.groups().index(x) == 0 else x,
268                     text.groups(),
269                 )
270             )
271             if text
272             else (None, 0, 0)
273         )
274         handler = get_handler(*params)
275         response = handler(*params[1:])
276         if inp.strip() == ".":
277             contacts.save_contacts("contacts")
278             return
279         if params[0] in ("show all", "search"):
280             param = (
281                 int(params[1])
282                 if params[1] is not None
283                 and isinstance(params[1], str)
284                 and params[1].isdigit()
285                 else 100
286             )
287             if params[0] == "show all":
288                 entry = contacts
289             elif params[0] == "search" and params[0] is not None:
290                 entry = contacts.search(params[1])
291             for tab in entry.iterator(param):
292                 if tab == "continue":
293                     input("Press <Enter> to continue...")
294                 else:
295                     print(build_table(tab))
296
297         print(response)
298         if response == "Good bye!":
299             return
300
301
302 # ================================ main programm ============================ #
303
304 if __name__ == "__main__":
305
306     main()
```