#1

Є список, кожен елемент якого є словником з контактами користувача наступного виду:

{

"name": "Allen Raymond",

"email": "nulla.ante@vestibul.co.uk",

"phone": "(992) 914-3792",

"favorite": False,

}

Словник містить ім'я користувача name, його email, телефонний номер phone та властивість favorite - обраний контакт чи ні.

Розробіть дві функції для серіалізації та десеріалізації списку контактів за допомогою пакета pickle та зберігання отриманих даних у бінарному файлі.

Перша функція write\_contacts\_to\_file приймає два параметри: filename - ім'я файлу, contacts - список контактів. Вона зберігає вказаний список у файл, використовуючи метод dump пакету pickle.

Друга функція read\_contacts\_from\_file читає та повертає зазначений список contacts з файлу filename, використовуючи метод load пакету pickle.

import pickle

def write\_contacts\_to\_file(filename, contacts):

with open(filename, "wb") as fh:

pickle.dump(contacts, fh)

def read\_contacts\_from\_file(filename):

with open(filename, "rb") as fh:

return pickle.load(fh)

#2

Є список, кожен елемент якого є словником з контактами користувача наступного виду:

{

"name": "Allen Raymond",

"email": "nulla.ante@vestibul.co.uk",

"phone": "(992) 914-3792",

"favorite": False,

}

Словник містить ім'я користувача name, його email, телефонний номер phone та властивість favorite - обраний контакт чи ні.

Розробіть дві функції для серіалізації та десеріалізації списку контактів за допомогою пакету json та зберігання отриманих даних у текстовому файлі.

Перша функція write\_contacts\_to\_file приймає два параметри: filename - ім'я файлу, contacts - список контактів. Вона зберігає вказаний список у файл, використовуючи метод dump пакету json.

Структура json файлу має бути такою:

{

"contacts": [

{

"name": "Allen Raymond",

"email": "nulla.ante@vestibul.co.uk",

"phone": "(992) 914-3792",

"favorite": false

},

...

]

}

Тобто список контактів повинен зберігатися за ключем "contacts", а не просто зберегти список у файл.

Друга функція read\_contacts\_from\_file читає та повертає зазначений список contacts з файлу filename, збереженого раніше функцією write\_contacts\_to\_file, використовуючи метод load пакету json.

import json

def write\_contacts\_to\_file(filename, contacts):

with open(filename, "w") as fh:

copy\_con = {"contacts": contacts}

json.dump(copy\_con, fh)

def read\_contacts\_from\_file(filename):

with open(filename, "r") as fh:

unpacked = json.load(fh)

result = unpacked["contacts"]

return result

#3

Є список, кожен елемент якого є словником з контактами користувача наступного виду:

{

"name": "Allen Raymond",

"email": "nulla.ante@vestibul.co.uk",

"phone": "(992) 914-3792",

"favorite": False,

}

Словник містить ім'я користувача name, його email, телефонний номер phone та властивість favorite - обраний контакт чи ні.

Розробіть дві функції для серіалізації та десеріалізації списку контактів за допомогою пакету csv та зберігання отриманих даних у текстовому файлі.

Перша функція write\_contacts\_to\_file приймає два параметри: filename - ім'я файлу, contacts - список контактів. Вона зберігає вказаний список у файлі формату csv.

Структура csv файлу має бути такою:

name,email,phone,favorite

Allen Raymond,nulla.ante@vestibul.co.uk,(992) 914-3792,False

Chaim Lewis,dui.in@egetlacus.ca,(294) 840-6685,False

Kennedy Lane,mattis.Cras@nonenimMauris.net,(542) 451-7038,True

Wylie Pope,est@utquamvel.net,(692) 802-2949,False

Cyrus Jackson,nibh@semsempererat.com,(501) 472-5218,True

Зверніть увагу, першим рядком у файлі йдуть заголовки – це назви ключів.

Друга функція read\_contacts\_from\_file читає, виконує перетворення даних та повертає вказаний список contacts із файлу filename, збереженого раніше функцією write\_contacts\_to\_file.

Примітка: При читанні файлу csv ми отримуємо властивість словника favorite у вигляді рядка, тобто. наприклад favorite='False' . Необхідно його привести до логічного виразу назад, щоб стало favorite=False.

import csv

def write\_contacts\_to\_file(filename, contacts):

with open(filename, 'w', newline='') as fh:

print(contacts)

field\_names = [cont for cont in contacts[0]]

writer = csv.DictWriter(fh, fieldnames=field\_names)

writer.writeheader()

[writer.writerow(cont) for cont in contacts]

#return contacts

def read\_contacts\_from\_file(filename):

with open(filename, 'r', newline='') as fh:

reader = csv.DictReader(fh)

contats\_new = []

for row in reader:

for value in row.values():

if value == "True":

row["favorite"] = True

elif value == "False":

row["favorite"] = False

contats\_new.append(row)

return contats\_new

#4

Розробіть клас Person. Він має чотири властивості: ім'я користувача name, його email, телефонний номер phone та властивість favorite - обраний контакт чи ні.

Приклад створення екземпляра класу:

Person(

"Allen Raymond",

"nulla.ante@vestibul.co.uk",

"(992) 914-3792",

False,

)

Розробіть клас Contacts. Він повинен ініціалізувати через конструктор дві властивості: filename - ім'я файлу для пакування об'єкта класу Contacts, contacts - список контактів, екземплярів класу Person.

Приклад створення екземпляра класу:

contacts = [

Person(

"Allen Raymond",

"nulla.ante@vestibul.co.uk",

"(992) 914-3792",

False,

),

Person(

"Chaim Lewis",

"dui.in@egetlacus.ca",

"(294) 840-6685",

False,

),

]

persons = Contacts("user\_class.dat", contacts)

Розробіть два методи для серіалізації та десеріалізації екземпляра класу Contacts за допомогою пакету pickle та зберігання даних у бінарному файлі.

Перший метод save\_to\_file зберігає екземпляр класу Contacts у файл, використовуючи метод dump пакету pickle. Ім'я файлу зберігається в атрибуті filename.

Другий метод read\_from\_file читає та повертає екземпляр класу Contacts з файлу filename, використовуючи метод load пакету pickle.

Приклад роботи:

persons = Contacts("user\_class.dat", contacts)

persons.save\_to\_file()

person\_from\_file = persons.read\_from\_file()

print(persons == person\_from\_file) # False

print(persons.contacts[0] == person\_from\_file.contacts[0]) # False

print(persons.contacts[0].name == person\_from\_file.contacts[0].name) # True

print(persons.contacts[0].email == person\_from\_file.contacts[0].email) # True

print(persons.contacts[0].phone == person\_from\_file.contacts[0].phone) # True

import pickle

*class* Person:

*def* \_\_init\_\_(*self*, *name*: str, *email*: str, *phone*: str, *favorite*: bool):

*self*.name = *name*

*self*.email = *email*

*self*.phone = *phone*

*self*.favorite = *favorite*

*self*.data = [*self*.name, *self*.email, *self*.phone, *self*.favorite]

*def* \_\_repr\_\_(*self*):

        return *f*"{*self*.data}"

*class* Contacts:

*def* \_\_init\_\_(*self*, *filename*: str, *contacts*: list[Person] = None):

        if *contacts* is None:

*contacts* = []

*self*.filename = *filename*

*self*.contacts = *contacts*

*def* save\_to\_file(*self*):

        with open(*self*.filename, "wb") as fh:

            pickle.dump(*self*, fh)

*def* read\_from\_file(*self*):

        with open(*self*.filename, "rb") as fh:

            return pickle.load(fh)

*def* \_\_repr\_\_(*self*):

        return *f*"{*self*.contacts}"

contacts = [

    Person(

        "Allen Raymond",

        "nulla.ante@vestibul.co.uk",

        "(992) 914-3792",

        False,

    ),

    Person(

        "Chaim Lewis",

        "dui.in@egetlacus.ca",

        "(294) 840-6685",

        False,

    ),

]

print("i just print contacts:", contacts)

persons = Contacts("user\_class.bin", contacts)

# print(persons, type(persons))

persons.save\_to\_file()

person\_from\_file = persons.read\_from\_file()

print(person\_from\_file, "boo", type(person\_from\_file))

# print(persons == person\_from\_file)  # False

print(persons.contacts[1] == person\_from\_file.contacts[1])  # False

print(persons.contacts[1].name == person\_from\_file.contacts[1].name)

print(persons.contacts[1].name)

print(person\_from\_file.contacts[0].name)  # True

print(persons.contacts[1].email == person\_from\_file.contacts[1].email)

print(persons.contacts[1].email)  # True

print(person\_from\_file.contacts[1].email)  # True

# True

print(persons.contacts[1].phone == person\_from\_file.contacts[1].phone)  # True

print(persons.contacts[1].phone)

print(person\_from\_file.contacts[1].phone)

#5

Ми продовжимо розширювати приклад попереднього завдання. Додайте до класу Contacts атрибут count\_save, за замовчуванням він повинен мати значення 0. Реалізуйте магічний метод \_\_getstate\_\_ для класу Contacts. При упаковуванні екземпляра метод класу повинен збільшувати значення атрибута count\_save на одиницю. Таким чином, ця властивість - лічильник повторних операцій пакування екземпляра класу

Приклад роботи коду:

persons = Contacts("user\_class.dat", contacts)

persons.save\_to\_file()

first = persons.read\_from\_file()

first.save\_to\_file()

second = first.read\_from\_file()

second.save\_to\_file()

third = second.read\_from\_file()

print(persons.count\_save) # 0

print(first.count\_save) # 1

print(second.count\_save) # 2

print(third.count\_save) # 3

import pickle

class Person:

def \_\_init\_\_(self, name: str, email: str, phone: str, favorite: bool):

self.name = name

self.email = email

self.phone = phone

self.favorite = favorite

class Contacts:

def \_\_init\_\_(self, filename: str, contacts: list[Person] = None):

if contacts is None:

contacts = []

self.filename = filename

self.contacts = contacts

self.count\_save = 0

def save\_to\_file(self):

with open(self.filename, "wb") as file:

pickle.dump(self, file)

def read\_from\_file(self):

with open(self.filename, "rb") as file:

content = pickle.load(file)

return content

def \_\_getstate\_\_(self):

attributes = self.\_\_dict\_\_.copy()

attributes['count\_save'] += 1

return attributes

#6

Продовжуємо розширювати приклад із попереднього завдання. Додайте до класу Contacts атрибут is\_unpacking, за замовчуванням він повинен мати значення False. Реалізуйте магічний метод \_\_setstate\_\_ для класу Contacts. При розпаковуванні екземпляра класу метод повинен змінювати значення атрибута is\_unpacking на значення True. Таким чином, ця властивість визначатиме, що екземпляр класу отримано внаслідок розпакування.

Приклад роботи коду:

persons = Contacts("user\_class.dat", contacts)

persons.save\_to\_file()

person\_from\_file = persons.read\_from\_file()

print(persons.is\_unpacking) # False

print(person\_from\_file.is\_unpacking) # True

import pickle

class Person:

def \_\_init\_\_(self, name: str, email: str, phone: str, favorite: bool):

self.name = name

self.email = email

self.phone = phone

self.favorite = favorite

class Contacts:

def \_\_init\_\_(self, filename: str, contacts: list[Person] = None):

if contacts is None:

contacts = []

self.filename = filename

self.contacts = contacts

self.count\_save = 0

self.is\_unpacking = False

def save\_to\_file(self):

with open(self.filename, "wb") as file:

pickle.dump(self, file)

def read\_from\_file(self):

with open(self.filename, "rb") as file:

content = pickle.load(file)

return content

def \_\_getstate\_\_(self):

attributes = self.\_\_dict\_\_.copy()

attributes["count\_save"] = attributes["count\_save"] + 1

return attributes

def \_\_setstate\_\_(self, value):

self.\_\_dict\_\_ = value

self.is\_unpacking = True

#7

Для копіювання екземпляра класу Person із попереднього прикладу реалізуйте функцію copy\_class\_person. Як параметр вона приймає екземпляр класу person, та повертає "поверхневу" копію об'єкта за допомогою функції copy із пакета copy.

Приклад коду:

person = Person(

"Allen Raymond",

"nulla.ante@vestibul.co.uk",

"(992) 914-3792",

False,

)

copy\_person = copy\_class\_person(person)

print(copy\_person == person) # False

print(copy\_person.name == person.name) # True

...

import copy

class Person:

def \_\_init\_\_(self, name: str, email: str, phone: str, favorite: bool):

self.name = name

self.email = email

self.phone = phone

self.favorite = favorite

def copy\_class\_person(person):

copy\_instance = copy.copy(person)

return copy\_instance

#8

Як ви вже зрозуміли, для класу Contacts створення поверхневої копії екземпляра класу не увінчається успіхом через те, що ми маємо атрибут contacts, який є списком екземплярів об'єктів класу Person, а отже, всі вони будуть передані за посиланням. Тому необхідно використовувати глибоке копіювання методом deepcopy з пакета copy

Для класу Contacts реалізуйте функцію copy\_class\_contacts. Як параметр вона приймає екземпляр класу Contacts і повертає глибоку копію об'єкта за допомогою функції deepcopy з пакета copy.

Приклад коду:

persons = Contacts("user\_class.dat", contacts)

new\_persons = copy\_class\_contacts(persons)

new\_persons.contacts[0].name = "Another name"

print(persons.contacts[0].name) # Allen Raymond

print(new\_persons.contacts[0].name) # Another name

import copy

import pickle

*class* Person:

*def* \_\_init\_\_(*self*, *name*: str, *email*: str, *phone*: str, *favorite*: bool):

*self*.name = *name*

*self*.email = *email*

*self*.phone = *phone*

*self*.favorite = *favorite*

*def* copy\_class\_person(*person*):

    return copy.copy(*person*)

*class* Contacts:

*def* \_\_init\_\_(*self*, *filename*: str, *contacts*: list[Person] = None):

        if *contacts* is None:

*contacts* = []

*self*.filename = *filename*

*self*.contacts = *contacts*

*self*.is\_unpacking = False

*self*.count\_save = 0

*def* save\_to\_file(*self*):

        with open(*self*.filename, "wb") as file:

            pickle.dump(*self*, file)

*def* read\_from\_file(*self*):

        with open(*self*.filename, "rb") as file:

            content = pickle.load(file)

        return content

*def* \_\_getstate\_\_(*self*):

        attributes = *self*.\_\_dict\_\_.copy()

        attributes["count\_save"] = attributes["count\_save"] + 1

        return attributes

*def* \_\_setstate\_\_(*self*, *value*):

*self*.\_\_dict\_\_ = *value*

*self*.is\_unpacking = True

*def* copy\_class\_contacts(*contacts*):

    copy\_instance = copy.deepcopy(*contacts*)

    return copy\_instance

contacts = [

    Person(

        "Allen Raymond",

        "nulla.ante@vestibul.co.uk",

        "(992) 914-3792",

        False,

    ),

    Person(

        "Chaim Lewis",

        "dui.in@egetlacus.ca",

        "(294) 840-6685",

        False,

    ),

]

persons = Contacts("user\_class.bin", contacts)

# print(persons, type(persons))

persons.save\_to\_file()

person\_from\_file = persons.read\_from\_file()

print(persons == person\_from\_file)  # False

print(persons.contacts[1] == person\_from\_file.contacts[1])  # False

print(persons.contacts[1].name == person\_from\_file.contacts[1].name)

# True

print(persons.contacts[1].email == person\_from\_file.contacts[1].email)

# True

# True

print(persons.contacts[1].phone == person\_from\_file.contacts[1].phone)  # True

persons = Contacts("user\_class.dat", contacts)

persons.save\_to\_file()

person\_from\_file = persons.read\_from\_file()

print(persons.is\_unpacking)  # False

print(person\_from\_file.is\_unpacking)  # True

new\_persons = copy\_class\_contacts(persons)

new\_persons.contacts[0].name = "Another name"

print(persons.contacts[0].name)  # Allen Raymond

print(new\_persons.contacts[0].name)  # Another name

#9

Реалізуйте метод \_\_copy\_\_ для класу Person.

Реалізуйте методи \_\_copy\_\_ та \_\_deepcopy\_\_ для класу Contacts.

from copy import deepcopy, copy

import pickle

class Person:

def \_\_init\_\_(self, name: str, email: str, phone: str, favorite: bool):

self.name = name

self.email = email

self.phone = phone

self.favorite = favorite

def \_\_copy\_\_(self):

copy\_obj = Person(None, None, None, None)

copy\_obj.name = copy(self.name)

copy\_obj.email = copy(self.email)

copy\_obj.phone = copy(self.phone)

copy\_obj.favorite = copy(self.favorite)

return copy\_obj

class Contacts:

def \_\_init\_\_(self, filename: str, contacts: list[Person] = None):

if contacts is None:

contacts = []

self.filename = filename

self.contacts = contacts

self.is\_unpacking = False

self.count\_save = 0

def save\_to\_file(self):

with open(self.filename, "wb") as file:

pickle.dump(self, file)

def read\_from\_file(self):

with open(self.filename, "rb") as file:

content = pickle.load(file)

return content

def \_\_getstate\_\_(self):

attributes = self.\_\_dict\_\_.copy()

attributes["count\_save"] = attributes["count\_save"] + 1

return attributes

def \_\_setstate\_\_(self, value):

self.\_\_dict\_\_ = value

self.is\_unpacking = True

def \_\_copy\_\_(self):

copy\_obj = Contacts(None, None)

copy\_obj.filename = copy(self.filename)

copy\_obj.contacts = copy(self.contacts)

copy\_obj.is\_unpacking = copy(self.is\_unpacking)

copy\_obj.count\_save = copy(self.count\_save)

return copy\_obj

def \_\_deepcopy\_\_(self, memo):

copy\_obj = Contacts(None, None)

memo[id(copy\_obj)] = copy\_obj

copy\_obj.filename = deepcopy(self.filename)

copy\_obj.contacts = deepcopy(self.contacts)

copy\_obj.is\_unpacking = deepcopy(self.is\_unpacking)

copy\_obj.count\_save = deepcopy(self.count\_save)

return copy\_obj