#1

Нехай ми маємо текстовий файл, який містить дані з місячною заробітною платою по кожному розробнику компанії.

Приклад файлу:

Alex Korp,3000

Nikita Borisenko,2000

Sitarama Raju,1000

Як бачимо, структура файлу – це прізвище розробника та значення його заробітної плати, розділеної комою.

Розробіть функцію total\_salary(path) (параметр path - шлях до файлу), яка буде розбирати текстовий файл і повертати загальну суму заробітної плати всіх розробників компанії.

Вимоги до завдання:

* функція total\_salary повертає значення типу float
* для читання файлу функція total\_salary використовує лише метод readline
* ми поки що не використовуємо менеджер контексту with

import re

*def* total\_salary(*path*):

    fh = open(*path*, 'r')

    all\_file = fh.read()

    nums = re.findall("\d{1,}", all\_file)

    total = float(sum(int(item) for item in nums))

    fh.close()

    return total

print(total\_salary(*r*"C:\Python\Go\_IT\Lessons\Modul\_6\salary.txt"))

#print(pathlib.Path(sys.argv[0]).parent / 'salary.txt')

#2

У компанії є кілька відділів. Список працівників для кожного відділу має такий вигляд:

['Robert Stivenson,28', 'Alex Denver,30']

Це список рядків із прізвищем та віком співробітника, розділеними комами.

Реалізуйте функцію запису даних про співробітників у файл, щоб інформація про кожного співробітника починалася з нового рядка.

Функція запису в файл write\_employees\_to\_file(employee\_list, path), де:

* path – шлях до файлу.
* employee\_list - список зі списками співробітників по кожному відділу, як у прикладі нижче:

[['Robert Stivenson,28', 'Alex Denver,30'], ['Drake Mikelsson,19']]

Вимоги:

* запишіть вміст employee\_list у файл, використовуючи режим "w".
* ми поки що не використовуємо менеджер контексту with
* кожен співробітник повинен бути записаний з нового рядка — тобто для попереднього списку вміст файлу має бути наступним:

Robert Stivenson,28

Alex Denver,30

Drake Mikelsson,19

*def* write\_employees\_to\_file(*employee\_list*, *path*):

    fh = open(*path*, 'w')

    for sets in *employee\_list*:

        for item in sets:

            item += "\n"

            fh.write(item)

    fh.close()

write\_employees\_to\_file([['Robert Stivenson,28', 'Alex Denver,30'], [

                        'Drake Mikelsson,19']], *r*"C:\Python\Go\_IT\Lessons\Modul\_6\employee\_list.txt")

#3

У попередній задачі ми записали співробітників у файл у такому вигляді:

Robert Stivenson,28

Alex Denver,30

Drake Mikelsson,19

Виконаємо тепер зворотнє завдання і створимо функцію read\_employees\_from\_file(path), яка читатиме дані з файлу та повертатиме список співробітників у вигляді:

['Robert Stivenson,28', 'Alex Denver,30', 'Drake Mikelsson,19']

Пам'ятайте про наявність символу кінця рядка \n під час читання даних із файлу. Його необхідно прибирати при додаванні прочитаного рядка до списку.

Вимоги:

* прочитайте вміст файлу за допомогою режиму "r".
* ми поки що не використовуємо менеджер контексту with
* поверніть із функції список співробітників із файлу
* import re
* *def* read\_employees\_from\_file(*path*):
* fh = open(*path*, "r")
* line = []
* while True:
* item = fh.readline()
* clear\_item = re.findall(*r*"\w{2,}[ ]\w{2,}[,]\d{2,}", item)
* line.extend(clear\_item)
* if not item:
* break
* fh.close()
* print(line)
* read\_employees\_from\_file(*r*"C:\Python\Go\_IT\Lessons\Modul\_6\employee\_list.txt")

#4

Реалізуйте функцію add\_employee\_to\_file(record, path), яка у файл (параметр path - шлях до файлу) буде додавати співробітника (параметр record) у вигляді рядка "Drake Mikelsson,19".

Вимоги:

* параметр record - співробітник у вигляді рядка виду "Drake Mikelsson,19"
* кожен запис у файл має починатися з нового рядка.
* необхідно щоб стара інформація у файлі теж зберігалася, тому при роботі з файлом відкрийте файл у режимі 'a', додайте співробітника record у файл і закрийте його
* ми поки що не використовуємо менеджер контексту with

*def* add\_employee\_to\_file(*record*, *path*):

    fh = open(*path*, "a")

    fh.write("\n")

    fh.write(*record*)

    fh.close()

add\_employee\_to\_file("Drake Mikelsson,19",

*r*"C:\Python\Go\_IT\Lessons\Modul\_6\employee\_list.txt")

#5

Ми маємо таку структуру файлу:

60b90c1c13067a15887e1ae1,Tayson,3

60b90c2413067a15887e1ae2,Vika,1

60b90c2e13067a15887e1ae3,Barsik,2

60b90c3b13067a15887e1ae4,Simon,12

60b90c4613067a15887e1ae5,Tessi,5

Кожен запис складається з трьох частин і починається з нового рядка. Наприклад, для першого запису початок 60b90c1c13067a15887e1ae1 — це первинний ключ бази даних MongoDB. Він завжди містить 12 байтів або рівно 24 символи. Далі ми бачимо прізвисько кота Tayson та його вік 3. Всі частини запису розділені символом кома ,

Розробіть функцію get\_cats\_info(path), яка повертатиме список словників із даними котів у вигляді:

[

{"id": "60b90c1c13067a15887e1ae1", "name": "Tayson", "age": "3"},

{"id": "60b90c2413067a15887e1ae2", "name": "Vika", "age": "1"},

{"id": "60b90c2e13067a15887e1ae3", "name": "Barsik", "age": "2"},

{"id": "60b90c3b13067a15887e1ae4", "name": "Simon", "age": "12"},

{"id": "60b90c4613067a15887e1ae5", "name": "Tessi", "age": "5"},

]

Параметри функції:

* path - шлях до файлу

Вимоги:

* прочитайте вміст файлу за допомогою режиму "r".
* ми використовуємо менеджер контексту with
* поверніть із функції список котів із файлу у потрібному форматі
* *def* get\_cats\_info(*path*):
* with open(*path*, "r") as fh:
* lines = ["id", "name", "age"]
* cats\_list = []
* cats\_dict = []
* while True:
* item = fh.readline()
* if len(item) > 0:
* clear\_item = item.removesuffix('\n')
* cats\_list.extend(clear\_item.split(","))
* cats\_dict.append(dict(zip(lines, cats\_list)))
* cats\_list.clear()
* if not item:
* break
* return cats\_dict
* get\_cats\_info(*r*"C:\Python\Go\_IT\Lessons\Modul\_6\cats\_info.txt")

#6

**ЗАВДАННЯ - ШУКАЄМО РЕЦЕПТ**

**ЗАВДАННЯ**

Нагадаємо, що у 4 модулі ми для кулінарного блогу писали функцію format\_ingredients, яка приймала на вхід список з інгредієнтами рецепта.

Ми продовжимо працювати в цьому напрямку та створимо функцію, яка шукатиме рецепт у файлі та повертатиме знайдений рецепт у вигляді словника певної форми.

У вас є файл, який містить рецепти у вигляді:

60b90c1c13067a15887e1ae1,Herbed Baked Salmon,4 lemons,1 large red onion,2 tablespoons chopped fresh basil

60b90c2413067a15887e1ae2,Lemon Pancakes,2 tablespoons baking powder,1 cup vanilla-flavored almond milk,1 lemon

60b90c2e13067a15887e1ae3,Chicken and Cold Noodles,6 ounces dry Chinese noodles,1 tablespoon sesame oil,3 tablespoons soy sauce

60b90c3b13067a15887e1ae4,Watermelon Cucumber Salad,1 large seedless watermelon,12 leaves fresh mint,1 cup crumbled feta cheese

60b90c4613067a15887e1ae5,State Fair Lemonade,6 lemons,1 cups white sugar,5 cups cold water

Кожен рецепт записаний з нового рядка (не забуваємо під час вирішення завдання про кінець рядка). Кожен запис починається з первинного ключа бази даних MongoDB. Далі через кому, йде назва рецепта, а потім через кому, йде перелік інгредієнтів рецепта.

Вам необхідно реалізувати функцію, котра буде отримувати інформацію про рецепт у вигляді словника для кожної страви що шукається. Створіть функцію get\_recipe(path, search\_id), яка повертатиме словник для рецепта із зазначеним ідентифікатором MongoDB.

Де параметри функції:

* path — шлях до файлу.
* search\_id — первинний ключ MongoDB для пошуку рецепта

Вимоги:

* Використовуйте менеджер контексту with для читання з файлу
* Якщо рецепта із зазначеним search\_id у файлі немає, функція повинна повернути None

Приклад: для файлу, вказаного вище, виклик функції у вигляді

get\_recipe("./data/ingredients.csv", "60b90c3b13067a15887e1ae4")

Повинен знайти у файлі рядок 60b90c3b13067a15887e1ae4,Watermelon Cucumber Salad,1 large seedless watermelon,12 leaves fresh mint,1 cup crumbled feta cheese і повернути результат у вигляді словника такої структури:

{

"id": "60b90c3b13067a15887e1ae4",

"name": "Watermelon Cucumber Salad",

"ingredients": [

"1 large seedless watermelon",

"12 leaves fresh mint",

"1 cup crumbled feta cheese",

],

*def* get\_recipe(*path*, *search\_id*):

    with open(*path*, "r") as fh:

        keys = ["id", "name", "ingredients"]

        cats\_list = []

        cats\_dict = []

        while True:

            item = fh.readline()

            if len(item) > 0:

                clear\_item = item.removesuffix('\n')

                cats\_list.extend(clear\_item.split(",", 2))

                cats\_list[2] = list(cats\_list[2].split(","))

                cats\_dict.append(dict(zip(keys, cats\_list)))

                cats\_list.clear()

            if not item:

                break

        for item in cats\_dict:

            for key in item.values():

                if *search\_id* == key:

                    print(item)

get\_recipe(*r*"C:\Python\Go\_IT\Lessons\Modul\_6\recipe.txt",

           "60b90c3b13067a15887e1ae4")

#7

**РОБОТА З ДВОМА ФАЙЛАМИ**

**ЗАВДАННЯ**

Розробіть функцію sanitize\_file(source, output), що переписує у текстовий файл output вміст текстового файлу source, очищений від цифр.

Вимоги:

* прочитайте вміст файлу source, використовуючи менеджер контексту with та режим "r".
* запишіть новий очищений від цифр вміст файлу output, використовуючи менеджер контексту with та режим "w"
* запис нового вмісту файлу output має бути одноразовий і використовувати метод write

import re

*def* sanitize\_file(*source*, *output*):

    with open(*source*, "r") as fh:

        source\_content = fh.read()

        output\_content = re.sub(*r*"\d", "", source\_content)

    with open(*output*, "w") as fh:

        fh.write(output\_content)

sanitize\_file(*r*"recipe.txt", *r*"sanitized.txt")

#8

**ЗАВДАННЯ: СТРУКТУРУЄМО ЗАПИС У ФАЙЛ**

**ЗАВДАННЯ**

Задано відомість абітурієнтів, які склали вступні іспити до університету. Структура даних щодо абітурієнтів подана у вигляді наступного списку:

[

{

"name": "Kovalchuk Oleksiy",

"specialty": 301,

"math": 175,

"lang": 180,

"eng": 155,

},

{

"name": "Ivanchuk Boryslav",

"specialty": 101,

"math": 135,

"lang": 150,

"eng": 165,

},

{

"name": "Karpenko Dmitro",

"specialty": 201,

"math": 155,

"lang": 175,

"eng": 185,

},

]

У кожному словнику цього списку записано прізвище абітурієнта — ключ name, код спеціальності, на яку він поступив — ключ specialty, та отримані ним бали з окремих дисциплін — ключі math (математика), lang ( українська мова) та eng (англійська мова)

Розробіть функцію save\_applicant\_data(source, output), яка буде вказаний список із параметра source зберігати у файл із параметра output

Структура файлу для зберігання повинна бути наступною. У кожному новому рядку файлу повинні бути записані через кому без прогалин прізвище абітурієнта, код спеціальності, на яку він поступив, та отримані ним бали за окремими дисциплінами.

Kovalchuk Oleksiy,301,175,180,155

Ivanchuk Boryslav,101,135,150,165

Karpenko Dmitro,201,155,175,185

Вимоги:

* відкрийте файл output для запису, використовуючи менеджер контексту with та режим w.
* запис нового вмісту файлу output має бути або за допомогою методу writelines, або використовувати метод write

def save\_applicant\_data(source, output): # output

item\_list = []

string = ""

for item in source:

for v in item.values():

string += str(v) + ","

string = string[0:-1]

item\_list.append(string)

string = ""

print(item\_list)

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

for item in item\_list:

item += "\n"

fh.write(item)

#9

Є два рядки у різних кодуваннях - "utf-8" та "utf-16". Нам необхідно зрозуміти, чи дорівнюються рядки між собою.

Реалізуйте функцію is\_equal\_string(utf8\_string, utf16\_string), яка повертає True, якщо рядки дорівнюють собі, і False — якщо ні.

def is\_equal\_string(utf8\_string, utf16\_string):

new\_utf8 = utf8\_string.decode()

new\_utf16 = utf16\_string.decode('utf-16')

if new\_utf8 == new\_utf16:

return True

else:

return False

#10

Дані про користувачів краще зберігати у форматі бінарних файлів. Тому вам необхідно створити функцію, яка буде записувати логін та пароль користувача у файл.

Реалізуйте функцію save\_credentials\_users(path, users\_info), яка зберігає інформацію про користувачів з паролями в бінарний файл.

Де:

* path – шлях до файлу.
* users\_info - словник типу {'andry':'uyro18890D', 'steve':'oppjM13LL9e'}, де ключ — логін (username) користувача, а значення — його пароль (password).

Вимоги:

* Кожен рядок файлу повинен мати такий вигляд username:password. Такий формат запису використовують при [Базовій аутентифікації](https://en.wikipedia.org/wiki/Basic_access_authentication).
* Відкрийте файл для запису та збережіть ключ та значення зі словника users\_info у вигляді окремого рядка username: password для кожного елемента словника users\_info

*def* save\_credentials\_users(*path*):  # users\_info

    users\_info = {'andry': 'uyro18890D',

                  'steve': 'oppjM13LL9e'}  # username:password

    my\_list = []

    for key, value in users\_info.items():

        my\_list.append("{}:{}".format(key, value))

    print(my\_list)

    with open(*path*, 'wb') as fh:

        for item in my\_list:

            item += "\n"

            item = bytes(item, 'utf-8')

            fh.write(item)

save\_credentials\_users(*r*"credentials\_users.txt")

#11

**ЧИТАННЯ БІНАРНИХ ФАЙЛІВ У PYTHON**

**ЗАВДАННЯ**

Реалізуйте функцію get\_credentials\_users(path), яка повертає список рядків із бінарного файлу, створеного в попередньому завданню, де:

* path – шлях до файлу.

Формат файлу:

andry:uyro18890D

steve:oppjM13LL9e

Відкрийте файл для читання, використовуючи with та режим rb. Сформуйте список рядків із файлу та поверніть його з функції get\_credentials\_users у наступному форматі:

['andry:uyro18890D', 'steve:oppjM13LL9e']

Вимоги:

* Використовуйте менеджер контексту для читання з файлу

def get\_credentials\_users(path):

list = []

with open(path, 'rb') as fh:

while True:

item = fh.readline()

if len(item) > 0:

item = item.decode()

item = item.removesuffix('\n')

list.append(item)

if not item:

break

return list

#12

### ЗАВДАННЯ

Функція get\_credentials\_users із попереднього завдання повертає нам список рядків username:password:

['andry:uyro18890D', 'steve:oppjM13LL9e']

Реалізуйте функцію encode\_data\_to\_base64(data), яка приймає у параметрі data зазначений список, виконує кодування кожної пари username:password у формат Base64 та повертає список із закодованими парами username:password у вигляді:

['YW5kcnk6dXlybzE4ODkwRA==', 'c3RldmU6b3Bwak0xM0xMOWU=']

import base64

data = ['andry:uyro18890D', 'steve:oppjM13LL9e']

*def* encode\_data\_to\_base64(*data*):

    list = []

    for item in *data*:

        message\_bytes = item.encode("utf-8")

        base64\_bytes = base64.b64encode(message\_bytes)

        base64\_message = base64\_bytes.decode("utf-8")

        list.append(base64\_message)

    return list

encode\_data\_to\_base64(data)

#13

Реалізуйте функцію create\_archive(path, file\_name, employee\_residence)

Де:

* path — шлях до файлу
* file\_name — ім'я файлу
* employee\_residence — словник, у якому ключ — ім'я користувача, а значення — країна проживання. Вигляд: {'Michael': 'Canada', 'John':'USA', 'Liza': 'Australia'}

Функція повинна працювати так:

1. Створювати бінарний файл file\_name за шляхом path
2. Зберігати дані словника employee\_residence у файл, де кожен новий рядок — це ключ значення через пробіл як "Michael Canada"
3. Архівувати теку по шляху path за допомогою shutil
4. Ім'я архіву має бути 'backup\_folder.zip'
5. Функція має повернути рядок шляху до архіву 'backup\_folder.zip'

Вимоги:

* запишіть вміст словника employee\_residence у бінарний файл з ім'ям file\_name у теку path за допомогою оператора with.
* використовуйте символ /, щоб розділити шлях для path та file\_name
* вигляд рядка файлу — Michael Canada, в кінці кожного рядка додається перенесення рядка '\n'.
* при збереженні кожен рядок файлу кодується методом encode
* при записі рядків використовуємо лише метод write
* архів має бути у форматі zip з ім'ям 'backup\_folder', створений за допомогою make\_archive.

import base64

import shutil

def create\_backup(path, file\_name, employee\_residence):

my\_list = []

for key, value in employee\_residence.items():

my\_list.append("{} {}".format(key, value))

with open(path+'/'+file\_name, 'wb') as fh:

for item in my\_list:

item += "\n"

#item = bytes(item, 'utf-8')

message\_bytes = item.encode("utf-8")

#base64\_bytes = base64.b64encode(message\_bytes)

#base64\_message = base64\_bytes.decode("utf-8")

fh.write(message\_bytes)

archive\_name = shutil.make\_archive('backup\_folder', 'zip', path)

return archive\_name

#14

Створіть функціонал для розпакування архіву.

Зробіть import пакету shutil

Створіть функцію unpack(archive\_path, path\_to\_unpack), яка викликатиме метод пакета shutil unpack\_archive та розпаковуватиме архів archive\_path у місце path\_to\_unpack.

Функція нічого не повертає.

import shutil

def unpack(archive\_path, path\_to\_unpack):

shutil.unpack\_archive(archive\_path, path\_to\_unpack)