

Python: Читання/запис файлів

Прикладна аналітика при розробці IT

Ігор Мірошніченко

КНУ імені Тараса Шевченка, ФІТ

Прикладна аналітика при розробці IT



ЗМІСТ

- Читання та запис файлів



ЧИТАННЯ ТА ЗАПИС ФАЙЛІВ

ЗАПИС ДАНИХ

До цього часу більшість програм, які ми писали, просто зберігали всю інформацію **в пам'яті**, тобто в змінних або всередині самої програми.

Недоліком цього є те, що як тільки програма завершує роботу, все, що ви ввели, все, що ви робили з цією програмою, втрачається.

Використовуючи файли, ви можете зберігати інформацію довгостроково, і **введення/виведення файлів** (англ. *file I/O*) в контексті програмування - це написання коду, який може читати з файлів, тобто завантажувати інформацію з них, або записувати до них, тобто зберігати інформацію у самих файлах.



ЗАПИС ДАНИХ

Для початку пропоную розглянути знайому структуру даних, яку ми бачили раніше - `list`.

Створимо програму `names.py`, яка буде зберігати імена у списку, а потім виводити їх на екран:

Terminal

```
1 code names.py
```

```
1 name = input("Як Вас звати? ")
2 print(f'Привіт, {name}!')
```



ЗАПИС ДАНИХ

Припустімо, що ми хочемо додати підтримку збереження декількох імен, наприклад трьох. Для цього ми можемо використати список. Для цього необхідно створити пустий список `names` і додавати (`append`) до нього імена, які вводить користувач. Вивід імен відсортуємо за алфавітом:

```
1 names = []
2
3 for _ in range(3):
4     names.append(input("Як Вас звати? "))
5
6 for name in sorted(names):
7     print(f'Привіт, {name}!')
```

```
1 Як Вас звати? Гаррі
2 Як Вас звати? Рон
3 Як Вас звати? Герміона
4 Привіт, Гаррі!
5 Привіт, Герміона!
6 Привіт, Рон!
```

Звичайно, якщо я запусну цю програму ще раз, всі імена пропадуть. Було б непогано мати можливість якось зберігати цю інформацію. І саме тут з'являється ввід-вивід файлів, і саме тут з'являються файли.



ЗАПИС ДАНИХ

Давайте перепишемо нашу програму так, щоб вона зберігала імена у файлі `names.txt`.

Для цього нам необхідно відкрити файл, використовуючи функцію `open()` - ця функція приймає два аргументи: ім'я файлу і режим відкриття.

Режим відкриття може бути:

- `r` (англ. *read*) - читання, це режим за замовчуванням.
- `w` (англ. *write*) - запис, цей режим перезаписує файл.
- `a` (англ. *append*) - дописування, цей режим додає дані до файлу.

Якщо файл не існує, то він буде створений. Давайте перепишемо нашу програму з використанням функції `open()`:

```
1 name = input("Як Вас звати? ")
2
3 file = open('names.txt', 'w')
4 file.write(name)
5 file.close()
```

Прикладна аналітика при розробці ІТ



ЗАПИС ДАНИХ

Запустимо цю програму і перевіримо, чи вона працює:

Terminal

```
1 python names.py
2 Як Вас звати? Гаппі
3 code names.txt
```

Відкриємо створений файл:

names.txt

```
1 Гаппі
```

Все працює! Тепер давайте виконаємо цю програму ще раз, але цього разу введемо ім'я Рон:

Terminal

```
1 python names.py
2 Як Вас звати? Рон
3 code names.txt
```

Відкриємо створений файл:

names.txt

```
1 Рон
```

Як бачимо, файл перезаписався, і тепер в ньому знаходиться тільки ім'я Рон.

ЗАПИС ДАНИХ

Якщо ми хочемо додати ім'я до файлу, а не перезаписати його, то використовувайте режим **a**.

Видаліть файл **names.txt** і давайте перепишемо нашу програму так, щоб вона дописувала імена до файлу **names.txt**:

Terminal

```
1 rm names.txt
2 remove names.txt? y
```

```
1 name = input("Як Вас звати? ")
2
3 file = open('names.txt', 'a')
4 file.write(name)
5 file.close()
```

Запустимо цю програму і перевіримо, чи вона працює:

Terminal

```
1 python names.py
2 Як Вас звати? Герміона
3 code names.txt
```

names.txt

```
1 Герміона
```

ЗАПИС ДАНИХ

Запустимо програму ще раз і спробуємо додати ім'я **Гаррі** та **Рон**:

Terminal
1 <code>python</code> names.py 2 <code>Як</code> Вас звати? Гаррі 3 <code>python</code> names.py 4 <code>Як</code> Вас звати? Рон 5 <code>code</code> names.txt
names.txt
1 <code>ГерміонаГарріРон</code>

Зовсім не той результат, який ми очікували.

Імена записалися в один рядок. Це тому, що функція `write` не додає символ переносу рядка (`\n`) після запису імені.



ЗАПИС ДАНИХ

Щоб це виправити, ми можемо додати символ переносу рядка після запису імені:

Terminal

```
1 rm names.txt
2 remove names.txt? y

1 name = input("Як Вас звати? ")
2
3 file = open('names.txt', 'a')
4 file.write(name + '\n')
5 file.close()
```

Запустимо цю програму:

Terminal

```
1 python names.py
2 Як Вас звати? Герміона
3 python names.py
4 Як Вас звати? Гаррі
5 python names.py
6 Як Вас звати? Рон
7 code names.txt
```

names.txt

```
1 Герміона
2 Гаррі
3 Рон
```



Примітка

Документація до функції `open`: <https://docs.python.org/3/library/functions.html#open>



КОНТЕКСТНИЙ МЕНЕДЖЕР

Під час написання коду дуже легко забути закрити файли і це може стати проблемою. Тому ми можемо піти більш безпечним шляхом і використовувати контекстний менеджер.

Контекстний менеджер - це спеціальна конструкція мови Python, яка дозволяє виконувати певні дії до входу в блок коду і після виходу з блоку коду.

Для використання контекстного менеджера використовується ключове слово **with**. Давайте перепишемо нашу програму з використанням контекстного менеджера:

```
1 name = input("Як Вас звати? ")
2
3 with open('names.txt', 'a', encoding="utf8") as file:
4     file.write(name + '\n')
```

Такий підхід не змінює функціональність програми, але є більш пітоничним.



ЧИТАННЯ ДАНИХ

Для читання, у функції `open` використовується режим `r`.

Давайте створимо програму `names_read.py`, яка буде читати імена з файлу `names.txt` і виводити їх на екран:

Terminal

```
1 code names_read.py
```

Для читання використаємо метод `readlines`, яка повертає список рядків, які містяться у файлі.

Цей метод повертає список, тому ми можемо використати цикл `for` для виведення імен на екран. Також слід врахувати, що метод `readlines` повертає список, в якому кожен рядок містить символ переносу рядка (`\n`).

Щоб цього уникнути, ми можемо використати метод `rstrip`, який видаляє символ переносу рядка з кінця рядка:

```
1 with open('names.txt', 'r', encoding="utf8") as file:
2     lines = file.readlines()
3
4 for line in lines:
5     print(f'Привіт, {line.rstrip()}!')
```

```
1 Привіт, Герміона!
2 Привіт, Гаррі!
3 Привіт, Рон!
```

ЧИТАННЯ ДАНИХ

Але у попередньому прикладі ми двічі проходимося по всьому файлу: спочатку ми читаємо його у список, а потім виводимо список на екран.

Це не є найкращим рішенням, оскільки ми можемо витратити багато пам'яті, якщо файл дуже великий.

Тому ми можемо використати цикл `for` безпосередньо для читання файлу:

```
1 with open('names.txt', 'r', encoding="utf8") as file:
2     for line in file:
3         print(f'Привіт, {line.rstrip()}!')
```

```
1 Привіт, Герміона!
2 Привіт, Гаррі!
3 Привіт, Рон!
```

ЧИТАННЯ ДАНИХ

Тепер трошки ускладнимо задачу.

Припустимо, що ми хочемо виводити привітання у алфавітному порядку. Для цього нам необхідно відсортувати список імен.

Для цього ми можемо використати функцію `sorted()`, яка повертає відсортований список:

```
1 names = []
2
3 with open('names.txt', 'r', encoding="utf8") as file:
4     for line in file:
5         names.append(line.rstrip())
6
7 for name in sorted(names):
8     print(f'Привіт, {name}!')
```

```
1 Привіт, Гаррі!
2 Привіт, Герміона!
3 Привіт, Рон!
```



ЧИТАННЯ ДАНИХ

Ми можемо зробити цю програму більш компактною. Для цього ми можемо відсортувати сам файл:

```
1 with open('names.txt', 'r', encoding="utf8") as file:
2     for line in sorted(file):
3         print(f'Привіт, {line.rstrip()}!!')
```

```
1 Привіт, Гаррі!
2 Привіт, Герміона!
3 Привіт, Рон!
```

Для зворотного сортування ми можемо використати параметр `reverse` функції `sorted`:

```
1 with open('names.txt', 'r', encoding="utf8") as file:
2     for line in sorted(file, reverse=True):
3         print(f'Привіт, {line.rstrip()}!!')
```

```
1 Привіт, Рон!
2 Привіт, Герміона!
3 Привіт, Гаррі!
```

Примітка

Документація до функції `sorted`: <https://docs.python.org/3/library/functions.html#sorted>

ФАЙЛИ CSV

Файли **csv** (англ. *comma-separated values*, значення, розділені комами) - це файли, які містять дані у вигляді таблиці, де значення розділені комами.

Давайте створимо файл `students.csv`:

Terminal

```
1 code students.csv
```

Запишемо у нього імена і додамо гуртожиток:

students.csv

```
1 Гаррі,Гріфіндор  
2 Герміона,Гріфіндор  
3 Рон,Гріфіндор  
4 Драко,Слизерин
```

ФАЙЛИ CSV

Тепер давайте створимо програму `students.py`, яка буде читати цей файл.

Terminal

```
1 code students.py
```

Ми можемо використати метод `split` для розділення рядка на частини. Давайте перепишемо нашу програму з використанням методу `split`:

```
1 with open('students.csv', 'r', encoding="utf8") as file:
2     for line in file:
3         row = line.rstrip().split(',')
4         print(f'{row[0]} живе в гуртожитку {row[1]}')
```

```
1 Гаррі живе в гуртожитку Гріфіндор
2 Герміона живе в гуртожитку Гріфіндор
3 Рон живе в гуртожитку Гріфіндор
4 Драко живе в гуртожитку Слизерин
```

ФАЙЛИ CSV

Коли у вас є змінна, яка є списком, наприклад `row`, вам не обов'язково переносити всі ці змінні у окремий список.

Ви можете розпакувати всю послідовність одразу.

Іншими словами, якщо ви знаєте, що функція типу `split` повертає список, який містить два елементи, ви можете розпакувати цей список у дві змінні:

```
1 with open('students.csv', 'r', encoding="utf8") as file:
2     for line in file:
3         name, house = line.rstrip().split(',')
4         print(f'{name} живе в гуртожитку {house}')
```

```
1 Гаррі живе в гуртожитку Гріфіндор
2 Герміона живе в гуртожитку Гріфіндор
3 Рон живе в гуртожитку Гріфіндор
4 Драко живе в гуртожитку Слизерин
```

ФАЙЛИ CSV

Уявімо, що нам треба відсортувати цей список даних.

Для цього я можу використати функцію `sorted()` і вказати, що я хочу сортувати за другим елементом списку:

```
1 students_lst = []
2
3 with open('students.csv', 'r', encoding="utf8") as file:
4     for line in file:
5         name, house = line.rstrip().split(',')
6         students_lst.append(f'{name} живе в гуртожитку {house}')
7
8 for student in sorted(students_lst):
9     print(student)
```

```
1 Драко живе в гуртожитку Слизерин
2 Гаррі живе в гуртожитку Гріфіндор
3 Герміона живе в гуртожитку Гріфіндор
4 Рон живе в гуртожитку Гріфіндор
```

ФАЙЛИ CSV

З технічної точки зору, це працює, але це не є найкращим рішенням, оскільки дані сортуються по цілому реченню.

Ми можемо вирішити таку задачу за допомогою **словників**.

Для цього нам необхідно створити пустий словник `student_dict` і додавати до нього інформацію про студентів:

```
1 students_lst = []
2
3 with open('students.csv', 'r', encoding="utf8") as file:
4     for line in file:
5         name, house = line.rstrip().split(',')
6         student_dict = {}
7         student_dict['name'] = name
8         student_dict['house'] = house
9         students_lst.append(student_dict)
10
11 for student in students_lst:
12     print(f'{student["name"]} живе в гуртожитку {student["house"]}')
```

ФАЙЛИ CSV

Ми можемо скоротити код шляхом присвоєння значень словнику одразу:

```
1 students_lst = []
2
3 with open('students.csv', 'r', encoding="utf8") as file:
4     for line in file:
5         name, house = line.rstrip().split(',')
6         student_dict = {'name': name, 'house': house}
7         students_lst.append(student_dict)
8
9 for student in students_lst:
10     print(f'{student["name"]} живе в гуртожитку {student["house"]})')
```

ФАЙЛИ CSV

Але результат все ще не відсортований.

Функція `sorted` приймає параметр `key`, який вказує, за яким ключем сортувати.

Для цього ми можемо використати функцію `get_name`, яка повертає ім'я студента і використаємо її як параметр `key`:

```
1 students_lst = []
2
3 with open('students.csv', 'r', encoding="utf8") as file:
4     for line in file:
5         name, house = line.rstrip().split(',')
6         student_dict = {'name': name, 'house': house}
7         students_lst.append(student_dict)
8
9 def get_name(student):
10     return student['name']
11
12 for student in sorted(students_lst, key=get_name):
13     print(f'{student["name"]} живе в гуртожитку {student["house"]}')
```

```
1 Драко живе в гуртожитку Слизерин
2 Гаррі живе в гуртожитку Гріфіндор
3 Герміона живе в гуртожитку Гріфіндор
4 Рон живе в гуртожитку Гріфіндор
```

ФАЙЛИ CSV

Якщо ж я захочу відсортувати за гуртожитком у зворотному порядку, то я можу використати функцію `get_house` і додати параметр `reverse=True` у функцію `sorted`:

```
1 students_lst = []
2
3 with open('students.csv', 'r', encoding="utf8") as file:
4     for line in file:
5         name, house = line.rstrip().split(',')
6         student_dict = {'name': name, 'house': house}
7         students_lst.append(student_dict)
8
9 def get_house(student):
10     return student['house']
11
12 for student in sorted(students_lst, key=get_house, reverse=True):
13     print(f'{student["name"]} живе в гуртожитку {student["house"]}')
```

```
1 Драко живе в гуртожитку Слизерин
2 Гаррі живе в гуртожитку Грифіндор
3 Герміона живе в гуртожитку Грифіндор
4 Рон живе в гуртожитку Грифіндор
```



Попередження

АНОНІМНІ ФУНКЦІЇ

У попередньому прикладі ми використовували функції `get_name` і `get_house`, які одразу використовуємо і більше ніколи до них не повертаємось.

Ми можемо спростити цей код і використати **анонімні функції** (англ. *lambda functions*), які дозволяють нам визначити функцію в одному рядку:

```
1 students_lst = []
2
3 with open('students.csv', 'r', encoding="utf8") as file:
4     for line in file:
5         name, house = line.rstrip().split(',')
6         student_dict = {'name': name, 'house': house}
7         students_lst.append(student_dict)
8
9 for student in sorted(students_lst, key=lambda student: student['name']):
10     print(f'{student["name"]} живе в гуртожитку {student["house"]}')

```

```
1 Драко живе в гуртожитку Слизерин
2 Гаррі живе в гуртожитку Гріфіндор
3 Герміона живе в гуртожитку Гріфіндор
4 Рон живе в гуртожитку Гріфіндор

```



ПАКЕТ **csv**

Читання csv-файлів

Давайте змінимо файл `students.csv` і замінимо гуртожитки на будинки де вони вирости:

students.csv

```
1 Гаррі,Тисова, 4
2 Рон,Нора
3 Драко,Маєток Мелфойв
```

Тепер давайте виведемо ці дані на екран:

```
1 students_lst = []
2
3 with open('students.csv', 'r', encoding="utf8") as file:
4     for line in file:
5         name, home = line.rstrip().split(',')
6         student_dict = {'name': name, 'home': home}
7         students_lst.append(student_dict)
8
9 for student in sorted(students_lst, key=lambda student: student['home']):
10     print(f'{student["name"]} з {student["home"]}')

```

```
1 ValueError: too many values to unpack (expected 2)
```

У нас виникла помилка. Це тому, що у нас є рядок, який містить дві коми, а ми спробували розпакувати його у дві змінні. Для вирішення цієї проблеми ми можемо використати в якості роздільника якийсь менш популярний символ, наприклад `|`:

students.csv

```
1 Гаррі|Тисова, 4
2 Рон|Нора
3 Драко|Маєток Мелфойв
```

ПАКЕТ csv

Інший варіант - це помістити значення у лапки:

students.csv

```
1 Гаррі,"Тисова, 4"
2 Рон,Нора
3 Драко,"Маєток Мелфоїв"
```

В будь-якому випадку необхідно буде змінювати код і продумувати логіку читання файлу. І це стає дуже незручним і складним, якщо у вас є багато різних файлів, які містять дані у різних форматах. Тому для роботи з csv-файлами використовують спеціальний пакет **csv**. Давайте перепишемо нашу програму з використанням пакету **csv**:

Варіант 1

Варіант 2

```
1 import csv
2
3 students_lst = []
4
5 with open('students.csv', 'r', encoding="utf8") as file:
6     reader = csv.reader(file)
7     for row in reader:
8         student_dict = {'name': row[0], 'home': row[1]}
9         students_lst.append(student_dict)
10
11 for student in sorted(students_lst, key=lambda student: student['home']):
12     print(f'{student["name"]} з {student["home"]}')

```

```
1 Драко з Маєток Мелфоїв
2 Рон з Нора
3 Гаррі з Тисова, 4
```

ПАКЕТ csv

Часто у табличних файлах перший рядок = за назву змінних. Давайте додамо `name` та `home` у `students.csv`:

students.csv

```
1 name,home
2 Гаррі,"Тисова, 4"
3 Рон,Нора
4 Драко,Маєток Мелфойв
```

В таких випадках ми можемо використати функцію `DictReader`, яка повертає словник, а не список:

```
1 import csv
2
3 students_lst = []
4
5 with open('students.csv', encoding="utf8") as file:
6     reader = csv.DictReader(file)
7     for row in reader:
8         students_lst.append({'name': row['name'], 'home': row['home']}) # або students_lst.append(row)
9
10 for student in sorted(students_lst, key=lambda student: student['home']):
11     print(f'{student["name"]} з {student["home"]}')

```

```
1 Драко з Маєток Мелфойв
2 Рон з Нора
3 Гаррі з Тисова, 4

```

Такий підхід є більш стійким до змін у файлі: хтось міг змінити порядок стовпчиків, але програма все одно буде працювати.



Примітка

Документація до пакету `csv`: <https://docs.python.org/3/library/csv.html>

ПАКЕТ `csv`

Запис csv-файлів

Припустимо, що ми створюємо програму, яка буде записувати дані про студентів у файл `students.csv`. Залишимо у файлі `students.csv` наступні дані:

students.csv

```
1 name,home
```

Давайте перепишемо програму `students.py`, яка буде записувати дані у файл `students.csv`:

```
1 import csv
2
3 name = input('Як Вас звати? ') # Гаррі
4 home = input('Де Ви живете? ') # Тисова, 4
5
6 with open('students.csv', 'a', encoding="utf8") as file:
7     writer = csv.writer(file)
8     writer.writerow([name, home])
```

Запустимо цю програму:

Terminal

```
1 python students.py
2 Як Вас звати? Гаррі
3 Де Ви живете? Тисова, 4
```

Відкриємо файл `students.csv`:

students.csv

```
1 name,home
2 Гаррі,"Тисова, 4"
```

Як бачите, Python автоматично взяв рядок з комою у лапки щоб уникнути помилки.

Прикладна аналітика при розробці IT



ПАКЕТ csv

Запис csv-файлів

Існує ще один спосіб реалізувати програму `students.py` не турбуючись про порядок змінних у списку. Для цього ми можемо використати функцію `DictWriter`, яка дозволяє записувати дані у файл у вигляді словника:

```
1 import csv
2
3 name = input('Як Вас звати? ') # Драко
4 home = input('Де Ви живете? ') # Маєток Мелфоїв
5
6 with open('students.csv', 'a', encoding="utf8") as file:
7     writer = csv.DictWriter(file, fieldnames=['name', 'home'])
8     writer.writerow({'name': name, 'home': home})
```

Запустимо цю програму:

Terminal

```
1 python students.py
2 Як Вас звати? Драко
3 Де Ви живете? Маєток Мелфоїв
```

Відкриємо файл `students.csv`:

students.csv

```
1 name,home
2 Гаррі,"Тисова, 4"
3 Драко,Маєток Мелфоїв
```

БІНАРНІ ФАЙЛИ

Бінарні файли - це файл, який складається лише з нулів та одиниць і дозволяє зберігати будь-які дані: зображення, відео, звук, текст, тощо.

В Python є популярна бібліотека під назвою `pillow`, яка дозволяє працювати з зображеннями, застосовувати фільтри, як в Instagram, створювати анімації, тощо.

Примітка

Документація до пакету `PIL`: <https://pillow.readthedocs.io>

Давайте створимо анімоване GIF-зображення. Сьогодні такі файли зустрічаються скрізь у вигляді мемів, анімацій, наклейок тощо. Анімоване GIF-зображення – це графічний файл, який містить кілька зображень всередині, а комп'ютер показує їх одне за одним, створюючи ефект анімації.

БІНАРНІ ФАЙЛИ

Почнемо з двох статичних зображень:

Terminal

```
1 code costume1.gif  
2 code costume2.gif
```



(a) *costume1.gif*



(b) *costume2.gif*

Рисунок 1: Статичні зображення



Примітка

Ці коти походять з мови програмування MIT під назвою Scratch.

Посилання на зображення ви можете знайти у репозиторії: <https://github.com/Aranaur/py4ds/tree/main/img/python>

БІНАРНІ ФАЙЛИ

Тепер створимо файл `costume.py`, який буде об'єднувати ці два зображення у анімацію:

Terminal

```
1 code costume.py
```

Для цього нам необхідно використати функцію `Image.open`, яка дозволяє відкрити зображення, а потім використати метод `save`, який дозволяє зберегти зображення у форматі GIF:

```
1 import sys
2 from PIL import Image
3
4 images_lst = []
5
6 for arg in sys.argv[1:]:
7     image = Image.open(arg)
8     images_lst.append(image)
9
10 images_lst[0].save(
11     'costume.gif',
12     save_all=True,
13     append_images=images_lst[1:],
14     duration=200,
15     loop=0)
```

БІНАРНІ ФАЙЛИ

Згустимо програму:

Terminal

```
1 python costume.py costume1.gif costume2.gif
```

Відкриємо файл *costume.gif*:

Terminal

```
1 code costume.gif
```



Рисунок 2: costume.gif

ДЯКУЮ ЗА УВАГУ!

📖 Матеріали курсу

✉ ihor.miroshnichenko@kneu.ua

📍 Data Mirosh

🌐 [@ihormiroshnichenko](#)

🐙 [@aranaur](#)

🏠 aranaur.rbind.io

