

ІНФОРМАТИКА ТА ПРОГРАМУВАННЯ

Тема 12. Файли

Файли

- Файл (від англійського file - папка) – це одиниця збереження інформації на зовнішньому пристрої комп'ютера.
- З точки зору програми, **файли** – це послідовності, як правило, однотипних елементів, у яких в кожний момент часу визначено деякий елемент, що вважається поточним.
- Файли можна поділити на **послідовні** та **файли прямого доступу**.
- У послідовних файлах дані обробляються послідовно від першого до останнього елемента.
- У файлах прямого доступу є можливість отримати доступ до довільного елемента файлу.
- Елементи послідовних файлів та файлів прямого доступу ще називають записами.
- Окремим видом файлів є текстові файли.
- **Текстовий файл** складається з символів, які утворюють рядки змінної довжини.
- Рядки відділяються між собою символом(ами) кінця рядка.

Носій типу файл

- Нехай A, B – послідовності елементів деякого типу T ; \wedge - маркер, що не належить T .
- Тоді носієм типу файл з елементів типу T є множина $M_f = \{(A, \wedge, B)\}$
- Тобто, кожний окремий файл є трійкою $\{(A, \wedge, B)\}$.
- Маркер позначає поточне місце у файлі, знаходиться перед елементом, що буде оброблятися.
- Для текстових файлів зазначені послідовності A, B – це послідовності символів.

Файли у Python

- У Python більш розвинутими є засоби роботи саме з текстовими файлами.
- Нетекстові файли називають ще **бінарними** та, як правило, обробляють за допомогою спеціальних модулів.
- На відміну від раніше розглянутих типів, ми не будемо виділяти окремо операції, відношення та інструкції для файлів.
- Переважна більшість дій над файлами у Python є інструкціями.

Основні дії над файлами

- Основні дії над файлами у Python – це відкриття файлу, закриття файлу, читання з файлу та запис у файл.
- Відкриття файлу позначається так:

f = open(fname, mode)

- де `fname` – ім'я файлу з точки зору операційної системи, `mode` – режим роботи з файлом. `fname` та `mode` – це рядки.

Основні дії над файлами.2

- Рядок `mode` повинен набувати одного з наступних значень:
 - 'r' - відкриття існуючого текстового файлу для читання
 - 'w' - створення нового текстового файлу для подальшого запису
 - 'a' - відкриття існуючого текстового файлу для додавання даних у кінець файлу
 - 'rb' - відкриття існуючого бінарного файлу для читання
 - 'wb' - створення нового бінарного файлу для подальшого запису
 - 'ab' - відкриття існуючого бінарного файлу для додавання даних у кінець файлу
 - 'r+' - відкриття існуючого текстового файлу для читання або запису
 - 'r+b' - відкриття існуючого бінарного файлу для читання або запису
- Закриття раніше відкритого файлу позначається

f.close()

Читання з файлу

- Для читання всього вмісту файлу використовують

cnt = f.read()

- Для читання n байтів від положення маркера -

sl = f.read(n)

- Для текстових файлів також можна вказати читання одного або всіх рядків такого файлу:

line = f.readline()

- прочитати один рядок текстового файлу.

lineslist = f.readlines()

- прочитати всі рядки текстового файлу у список рядків.

Читання з файлу.2

- При читанні окремих рядків чи всіх рядків текстового файлу в кінці кожного рядка вставляється символ переходу на новий рядок ('\\n').
- Для читання рядків файлу `f` можна також застосувати цикл

for line in f:

P

- де *P* – інструкція.
- Якщо читання з файлу здійснюється по окремих рядках, то повернення `read()` або `readline()` порожнього рядка означає досягнення кінця файлу.

Запис у файл

- Для запису інформації у файл використовують

f.write(s)

- При цьому, для запису у текстовий файл s повинно бути рядком, а для запису у бінарний файл – рядком байтів.
- Для запису у файл списку рядків треба писати

f.writelines(lineslist)

- де lineslist – список, що складається з рядків.
- write() та writelines() не додають до рядків символ переходу на новий рядок ('\n').
- Тому у випадку, якщо рядки файлу формуються у програмі, цей символ треба додавати до рядків самостійно.
- Ще одним способом запису у текстовий файл є використання print з ключовим параметром file.

print(s, file = f)

Приклад

- Стиснути текстовий файл шляхом видалення зайвих пропусків між словами та на початку і в кінці рядків.

Додаткові дії над файлами

- До додаткових дій віднесемо обчислення поточної позиції маркера у файлі

f.tell()

- та встановлення позиції маркера у файлі

f.seek(n)

- Ці дії визначені для файлів, які обробляють як файли прямого доступу.

Запис та читання з текстового файлу нетекстових значень

- Якщо потрібно записати у текстовий файл нетекстові значення, ці значення треба спочатку перетворити у рядки.
- Таке перетворення досягається застосуванням функції `format` або `str`.
- Ще один варіант – використати `print` замість `write`.
- Після читання з файлу рядків, що містять нетекстові значення, треба конвертувати ці рядки у відповідні значення.

Приклад

- Обчислити добуток двох поліномів степені та коефіцієнти яких записані у текстових файлах. Поліном, що є результатом, також записати у файл (версія 1).
- Будемо вважати, що у кожному рядку файлу записано спочатку степінь а потім – коефіцієнт при цій степені через один або декілька пропусків.

Запуск програми з командного рядка та передача їй параметрів

- Ми вже знаємо, що програму у Python можна запустити з командного рядка.
- При цьому є можливість передати у програму параметри, вказані у рядку запуску.
- Ці параметри вказують через пропуск(и).
- Загальний синтаксис має вигляд:

```
python programfile param1 ... paramn
```

- де *programfile* – файл з текстом програми, *param*₁, ... , *param*_n – параметри.
- Для доступу до параметрів з програми треба імпортувати модуль `sys` та використати параметри, які зберігаються у змінній `sys.argv`. `sys.argv` – це список.
- У `sys.argv[0]` зберігається ім'я файлу програми (*programfile*).
- Починаючи з `sys.argv[1]`, у списку наводяться передані параметри.

Приклад

- Обчислити добуток двох поліномів степені та коефіцієнти яких записані у текстових файлах. Поліном, що є результатом, також записати у файл (версія 2).

Обробка нетекстових файлів. pickle

- Для обробки нетекстових файлів у Python існує спеціальний модуль pickle (у перекладі з англійської – консервування, маринування).
- Цей модуль містить функції для збереження у файлі та відновлення практично будь-якої змінної з програми у Python.



Обробка нетекстових файлів. pickle.2

- Відкриття та закриття файлу виконується так, як вже було розглянуто. Тобто за допомогою open (значення mode – ‘rb’ або ‘wb’) та close.
- Для використання pickle спочатку треба імпортувати модуль

import pickle

- Для збереження значення змінної x у відкритому для запису файлі f треба написати

pickle.dump(x, f)

- Для читання значення змінної x з відкритого для читання файлу f треба написати

x = pickle.load(f)

- Як ми бачимо, працювати з pickle дуже просто.

Приклад

- Скласти програму, яка реалізує простий телефонний довідник з функціями створення довідника, додавання запису до довідника, пошуку номера телефону за прізвищем а також заміни номеру телефону новим номером. (версія 1)
- Довідник будемо зберігати у словнику з ключами – прізвищами знайомих та зі значеннями – номерами телефонів.

Обробка нетекстових файлів. *shelve*

- Для роботи з нетекстовими файлами, які мають вигляд словника, зручно використовувати ще один модуль Python – *shelve* (у перекладі з англійської – ставити на полицю).
- *Shelve* створює окрім основного декілька додаткових службових файлів, що використовуються для організації доступу до даних.



Обробка нетекстових файлів. `shelve.2`

- Для використання `shelve` спочатку треба імпортувати модуль

```
import shelve
```

- Для того, щоб почати роботу з файлом у `shelve`, треба написати

```
x = shelve.open(filename)
```

- У кінці роботи закрити файл як звичайно

```
x.close()
```

- Між відкриттям та закриттям до файлу можна звертатись як до словника.
- Тобто, читати значення за ключем, змінювати значення за ключем тощо.

Приклад

- Скласти програму, яка реалізує простий телефонний довідник з функціями створення довідника, додавання запису до довідника, пошуку номера телефону за прізвищем а також заміни номеру телефону новим номером. (версія 2)

Резюме

- Ми розглянули:
 1. Файли, носій типу файл, текстові та нетекстові файли.
 2. Відкриття та закриття файлів.
 3. Читання з файлу та запис у файл.
 4. Додаткові дії для файлів.
 5. Запис та читання з текстового файлу нетекстових значень.
 6. Параметри при запуску програми з командного рядка.
 7. Обробку нетекстових файлів: `pickle` та `shelve`.

Де прочитати

1. A Byte of Python (Russian) Версія 2.01 Swaroop С Н
(Translated by Vladimir Smolyar),
<http://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.01.pdf>
2. Марк Лутц, Изучаем Python, 4-е издание, 2010,
Символ-Плюс
3. Python 3.4.3 documentation
4. Бублик В.В., Личман В.В., Обвінцев О.В..
Інформатика та програмування. Електронний
конспект лекцій, 2003 р.,
5. http://www.python-course.eu/python3_file_management.php