Розділ 7

Файли

7.1 Постійність даних

Ми вже опрацювали написання програм та передання команд *центральному процесору* за допомогою виконання умов, функцій та ітерацій. Також був висвітлений процес створення та використання структур даних в *основній пам'яті*. Центральний процесор і пам'ять – це місце, де працює і запускається програмне забезпечення. Саме тут відбувається процес «мислення».

Однак, якщо пригадати з попередніх розділів про архітектуру апаратного забезпечення, то після вимкнення живлення все, що зберігається в центральному процесорі чи основній пам'яті, видаляється. Тому до цього моменту наші програми були лише прохідними веселими вправами для вивчення Python.

У цьому розділі розпочнемо роботу з *постійною пам'яттю* (або файлами). Постійна пам'ять не видаляється при вимкненні живлення. Або, як у варіанті з флеш-накопичувачем, дані, які ми записуємо з наших програм, можуть бути видалені з однієї системи і перенесені в іншу.

Переважно зосередимося на зчитуванні та запису текстових файлів, наприклад, тих, які ми створюємо у текстовому редакторі. Пізніше розберемося, як працювати з файлами баз даних, які є двійковими файлами, розробленими для зчитування та запису за допомогою програмного забезпечення для роботи з базами даних.

7.2 Відкриття файлів

Щоб прочитати чи записати файл (наприклад, на жорсткому диску), його потрібно спочатку *відкрити*. Під час відкриття файлу відбувається взаємодія з операційною системою, яка знає, де зберігаються дані кожного файлу. У процесі відкриття файлу ви надсилаєте операційній системі запит знайти файл за його назвою і перевірити, чи він існує. У наведеному прикладі відкриваємо файл *mbox.txt*, який має бути у тій самій папці, в якій ви знаходитесь під час запуску Python. Цей файл можна завантажити з www.py4e.com/code3/mbox.txt

```
>>> fhand = open('mbox.txt')
>>> print(fhand)
<_io.TextIOWrapper name='mbox.txt' mode='r' encoding='cp1252'>
```

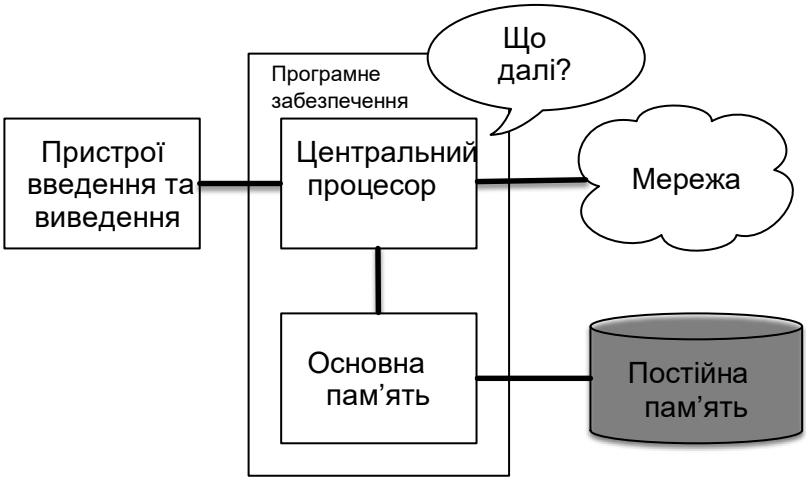

Рисунок 7.1: Постійна пам'ять

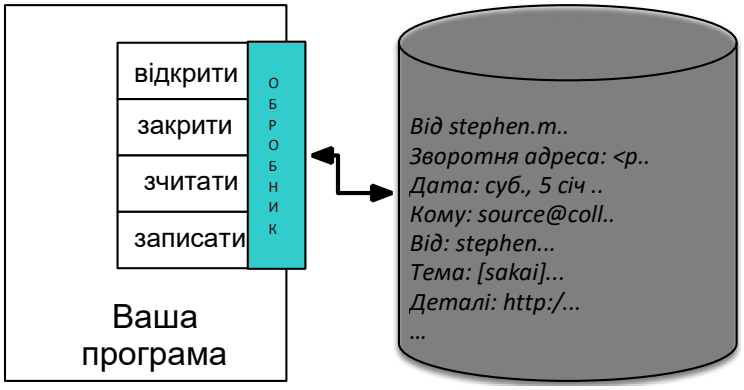

Рисунок 7.2: Обробник файлу

Якщо процес відкриття пройшов успішно, операційна система повертає *обробник файлу*. Обробник файлу – це не самі дані, що містяться у файлі, а «обробник», який використовується для зчитування даних. Якщо вказаний файл існує, і у вас є відповідні дозволи на його зчитування, вам буде надано обробник.

Якщо файлу не існує, команда відкриття поверне помилку, і ви не отримаєте обробник для доступу до вмісту:

```
>>> fhand = open('stuff.txt')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: 'stuff.txt'
```

Пізніше скористаємося try і except, щоб коректніше вирішити ситуацію, у якій намагаємося відкрити файл, якого не існує.

7.3 Текстові файли та рядки

Текстовий файл можна розглядати як послідовність рядків, подібно до того, як рядок у Python можна розглядати як послідовність символів. Наведемо приклад текстового файлу, який відображає листування різних користувачів у команді розробників проєкту з відкритим вихідним кодом:

```
Від stephen.marquard@uct.ac.za суб., 5 січня 09:14:16 2008
Зворотня адреса: <postmaster@collab.sakaiproject.org>
Дата: суб., 5 січня 2008 09:12:18 -0500
Кому: source@collab.sakaiproject.org
Від: stephen.marquard@uct.ac.za
Тема: [sakai] svn commit: r39772 - content/branches/
Деталі: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772 ...
```

Весь файл взаємодії з поштою доступний за посиланням: www.py4e.com/code3/mbox.txt

Скорочена версія цього файлу: www.py4e.com/code3/mbox-short.txt

Ці файли мають стандартний формат для файлів, що містять кілька повідомлень електронної пошти. Рядки, що починаються з «Від», відокремлюють повідомлення, а рядки, що починаються з «Від:», є частиною повідомлень. Докладніше про формат mbox описано у статті https://en.wikipedia.org/wiki/Mbox.

Для поділу файлу на рядки існує спеціальний символ, який позначає «кінець рядка» – *символ переходу на новий рядок*.

У Python *символ переходу на новий рядок* у рядкових константах позначається символом оберненої скісної риски n. Незважаючи на те, що це схоже на два символи, насправді це один символ. Якщо подивитися на змінну, ввівши «stuff» у інтерпретаторі, він покаже \n у рядку, але коли вивести рядок за допомогою print, побачимо, що його розбито на два рядки символом переходу на новий рядок.

```
>>> stuff = 'Привіт\nсвіте!'
>>> stuff
'Привіт\nсвіте!'
>>> print(stuff)
Привіт
Світе!
>>> stuff = 'X\nY'
>>> print(stuff)
X
Y
>>> len(stuff)
3
```

Також видно, що довжина рядка X\nY дорівнює трьом символам, оскільки символ переходу на новий рядок – це один символ. Отже, розглядаючи на рядки у файлі, необхідно уявити, що в кінці

кожного рядка є окремий невидимий символ, який називається символ переходу на новий рядок, що позначає кінець рядка.

Таким чином, символ переходу на новий рядок розділяє символи у файлі на рядки.

7.4 Зчитування файлів

Попри те, що *обробник файлу* не містить даних файлу, досить легко побудувати цикл for, який прочитає і підрахує кожен рядок у файлі:

```
fhand = open('mbox-short.txt')
count = 0
for line in fhand:
    count = count + 1
print('Кількість рядків:', count)

# Код: https://www.py4e.com/code3/open.py
```

Обробник файлу можна використати як послідовність у нашому циклі for. Цей цикл for просто підраховує кількість рядків у файлі і виводить її на екран. Приблизний переклад циклу for українською мовою такий: «для кожного рядка у файлі, представленого обробником, додайте одиницю до змінної count».

Функція open не зчитує весь файл, тому що файл може бути досить великим і містити багато гігабайтів даних. Інструкція open займає однакову кількість часу незалежно від розміру файлу. Цикл for фактично робить зчитування даних з файлу.

Коли файл зчитується за допомогою циклу for, Python розбиває дані у файлі на окремі рядки за допомогою символу переходу на новий рядок. Python зчитує кожен рядок через символ переходу і включає цей символ як останній символ у змінну line для кожної ітерації циклу for.

Оскільки цикл for зчитує дані по одному рядку за раз, він може ефективно зчитувати і рахувати рядки у дуже великих файлах, не витрачаючи при цьому основну пам'ять для зберігання даних. Наведена програма може порахувати рядки у файлі будь-якого розміру, використовуючи мало пам'яті, оскільки кожен рядок зчитується, рахується, а потім очищається.

Якщо ви знаєте, що файл відносно невеликий порівняно з розміром вашої основної пам'яті, можете зчитати весь файл в один рядок за допомогою методу read у обробнику файлу.

```
>>> fhand = open('mbox-short.txt')
>>> inp = fhand.read()
>>> print(len(inp))
94626
>>> print(inp[:20])
Від stephen.marquar
```

У цьому прикладі весь вміст (усі 94626 символів) файлу mbox-short.txt зчитується безпосередньо у змінну inp. Використовується зріз рядків, щоб вивести перші 20 символів даних рядка, що зберігаються у inp.

Коли файл зчитується у такий спосіб, усі символи, включно з усіма рядками і символами переходу на новий рядок, стають одним великим рядком у змінній inp. Доцільно зберігати результати зчитування у змінній, оскільки кожен виклик read вичерпує ресурс:

```
>>> fhand = open('mbox-short.txt')
>>> print(len(fhand.read()))
94626
>>> print(len(fhand.read()))
0
```

Пам'ятайте, що цю форму функції open слід використовувати лише в тому випадку, якщо дані файлу легко поміщаються в оперативній пам'яті вашого комп'ютера. Якщо файл занадто великий, щоб поміститися в оперативній пам'яті, слід написати програму, яка зчитуватиме файл частинами, використовуючи цикл for або while.

7.5 Пошук у файлі

Під час пошуку даних у файлі дуже поширеним шаблоном є зчитування файлу, ігнорування більшості рядків і обробка лише тих рядків, які відповідають певній умові. Можна поєднати шаблон зчитування файлу з методами рядків для побудови простих механізмів пошуку.

Наприклад, якщо потрібно зчитати файл і вивести лише ті рядки, що починаються з префіксу «Від:», можна скористатися методом рядка startswith, щоб виділити лише ті рядки, що нам потрібно:

```
fhand = open('mbox-short.txt')
for line in fhand:
    if line.startswith('Від:'):
        print(line)
```

 # Код: https://www.py4e.com/code3/search1.py

Після запуску цієї програми отримаємо такий результат:

```
Від: stephen.marquard@uct.ac.za

Від: louis@media.berkeley.edu

Від: zqian@umich.edu

Від: rjlowe@iupui.edu
...
```

Результат виглядає чудово, оскільки ми бачимо лише ті рядки, які починаються з «Від:», але чому з'явилися зайві порожні рядки? Це пов'язано з невидимим *символом переходу на новий рядок*. Кожен ряд закінчується символом переходу на новий рядок, тому інструкція print виводить рядок у змінній *line*, який містить символ переходу, а потім print додає *ще один* символ переходу, що створює ефект подвійного інтервалу, який ми і бачимо.

Можна скористатися зрізом рядка, щоб вивести всі символи, крім останнього, але простіше застосувати метод *rstrip*, який видаляє пробіли з правого боку рядка:

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if line.startswith('Від:'):
        print(line)
```

 # Код: https://www.py4e.com/code3/search2.py

Після запуску цієї програми отримаємо такий результат:

```
Від: stephen.marquard@uct.ac.za
Від: louis@media.berkeley.edu
Від: zqian@umich.edu
Від: rjlowe@iupui.edu
Від: zqian@umich.edu
Від: rjlowe@iupui.edu
Від: cwen@iupui.edu
...
```

З підвищенням складності ваших програм обробки файлів, ви ймовірно забажаєте структурувати цикли пошуку за допомогою continue. Основна ідея циклу пошуку полягає у тому, щоб знайти «потрібні» рядки і пропустити «непотрібні». Після того, як буде знайдено потрібний рядок, з ним буде виконано певні дії.

Можна структурувати цикл так, щоб пропустити непотрібні рядки у такий спосіб:

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    # Пропускаємо ‘непотрібні рядки’
    if not line.startswith('Від:'):
        continue
    # Опрацьовуємо ‘потрібний рядок
    print(line)
```

 # Код: https://www.py4e.com/code3/search3.py

На виході програма видає той самий результат. В українській мові непотрібні рядки – це ті, що не починаються з «Від:», які ми пропускаємо за допомогою continue. Для «потрібних» рядів (тобто тих, що починаються з «Від:») ми виконуємо обробку.

За допомогою методу рядка find можна відтворити пошук у текстовому редакторі, який знаходить рядки, де рядок, що потрібно знайти, міститься у будь-якому місці рядка файлу. Оскільки метод find шукає появу рядка у іншому рядку і повертає або позицію рядка, або -1, якщо рядок не знайдено, ми можемо написати наступний цикл, щоб показати рядки, які містять підрядок «@uct.ac.za» (тобто, вони належать університету Кейптауна у Південній Африці):

```
fhand = open('mbox-short.txt')
for line in fhand:
    line = line.rstrip()
    if line.find('@uct.ac.za') == -1: continue
    print(line)
```

 # Код: https://www.py4e.com/code3/search4.py

В результаті отримуємо:

```
Від stephen.marquard@uct.ac.za суб., 5 січня 09:14:16 2008
X-Authentication-Warning: встановіть відправника stephen.marquard@uct.ac.za за допомогою -f
Від: stephen.marquard@uct.ac.za
Автор: stephen.marquard@uct.ac.za
Від david.horwitz@uct.ac.za пт., 4 січня 07:02:32 2008
X-Authentication-Warning: встановіть відправника david.horwitz@uct.ac.za за допомогою -f
Від: david.horwitz@uct.ac.za
Автор: david.horwitz@uct.ac.za
...
```

Тут також використовується скорочена форма інструкції if, де ми розміщуємо continue на тому самому рядку, що й if. Ця скорочена форма if працює так само, як і у випадку, коли continue знаходиться на наступному рядку з відступом.

7.6 Можливість користувачеві вводити ім’я файлу

Незручно щоразу, коли необхідно обробити інший файл, редагувати код Python. Доцільніше було б просити користувача вводити рядок імені файлу при кожному запуску програми, щоб він міг використовувати нашу програму для різних файлів, не змінюючи код Python.

Це досить просто зробити, зчитавши ім'я файлу від користувача, використовуючи функцію input, як показано нижче:

```
fname = input('Введіть ім’я файлу: ')
fhand = open(fname)
count = 0
for line in fhand:
    if line.startswith('Тема:'):
    count = count + 1
print('Всього було', count, 'рядків теми у файлі', fname)
```

 # Код: https://www.py4e.com/code3/search6.py

Ми зчитуємо ім'я файлу від користувача, поміщаємо його у змінну з ім'ям fname і відкриваємо цей файл. Тепер можна запускати програму неодноразово на різних файлах.

```
python search6.py
Введіть ім’я файлу: mbox.txt
Всього було 1797 рядів теми у файлі mbox.txt

python search6.py
Введіть ім’я файлу: mbox-short.txt
Всього було 27 рядів теми у файлі mbox-short.txt
```

Перш ніж перейти до наступного пункту, перегляньте наведену вище програму і запитайте себе: «Що може піти не так?» або «Що може зробити наш любий користувач, що змусить нашу чудову невеличку програму ганебно завершити роботу з помилкою, виставивши нас не дуже крутими в очах користувачів?».

7.7 Застосування try, except та open

Я ж казав, не підглядати. Даю останній шанс.

Що, якщо наш користувач введе не ім'я файлу, а щось інше?

```
python search6.py
Введіть ім’я файлу: missing.txt
Traceback (most recent call last):
  File "search6.py", line 2, in <module>
    fhand = open(fname)
FileNotFoundError: [Errno 2] No such file or directory:: 'missing.txt'

python search6.py
Введіть ім’я файлу: ля ля ля бе бе бе
Traceback (most recent call last):
  File "search6.py", line 2, in <module>
  fhand = open(fname)
FileNotFoundError: [Errno 2] No such file or directory: 'ля ля ля бе бе бе'
```

Не смійтеся. Користувачі зрештою зроблять все можливе, щоб зламати ваші програми, помилково чи навмисно. Насправді, важлива частина будь-якої команди розробників програмного забезпечення – це людина або група, яка називається *забезпеченням якості* (або скорочено QA), чия робота полягає у здійсненні найбожевільніших спроб зламати програмне забезпечення, яке створив програміст.

Команда QA відповідає за пошук недоліків у програмах до того, як ми передамо програму кінцевим користувачам, які, можливо, купують програмне забезпечення або платять нам за написання програмного забезпечення. Отже, команда QA – це найкращі друзі програміста.

Отже, тепер, коли ми бачимо недоліки в програмі, то можемо вправно виправити їх, використовуючи структуру try/except. Припустимо, що виклик open може завершитися з помилкою, і додамо код відновлення, як показано на прикладі нижче:

```
fname = input('Введіть ім’я файлу: ')
try:
fhand = open(fname)
except:
       print('Неможливо відкрити файл:', fname)
       exit()
count = 0
for line in fhand:
       if line.startswith('Тема:'):
              count = count + 1
print('Всього було', count, 'рядів теми у файлі', fname)

# Код: https://www.py4e.com/code3/search7.py
```

Функція exit завершує роботу програми. Це функція, яку ми викликаємо і яка нічого не повертає. Тепер, коли наш користувач (або команда QA) вводить дурниці або неправильні імена файлів, ми «ловимо» їх і з легкістю відновлюємо:

```
python search7.py
Введіть ім’я файлу: mbox.txt
Всього було 1797 рядів теми у файлі mbox.txt

python search7.py
Введіть ім’я файлу: ля ля ля бе бе бе
Неможливо відкрити файл: ля ля ля бе бе бе
```

Захист виклику функції open є гарним прикладом правильного використання try та except у програмі мовою Python. Фахівці використовують термін «пітонівський», коли ми робимо щось «пітонівським способом». Можна сказати, що наведений вище приклад – це пітонівський спосіб відкриття файлу.

Коли станете більш досвідченими у Python, зможете посперечатися з іншими програмістами Python, щоб вирішити, яке з двох еквівалентних рішень проблеми є «більш пітонівським». Мета зробити програму «більш пітонівською» відображає ідею того, що програмування – це частково інженерія, а частково мистецтво. Ми не завжди зацікавлені в тому, щоб просто щось працювало, ми також хочемо, щоб наше рішення було ефективним і щоб його високо оцінили наші колеги.

7.8 Запис файлів

Щоб записати файл, потрібно відкрити його з режимом «w» у як другий параметр:

```
>>> fout = open('output.txt', 'w')
>>> print(fout)
<_io.TextIOWrapper name='output.txt' mode='w' encoding='cp1252'>
```

Якщо файл вже існує, то під час його відкриття у режимі запису видаляються старі дані і починається все з чистого аркуша, тож будьте обережні! Якщо файлу не існує, буде створено новий.

За допомогою методу write об'єкта обробника файлу дані записуються у файл, повертаючи кількість записаних символів. За замовчуванням використовується текстовий режим написання (і зчитування) рядків.

```
>>> line1 = "Це тин,\n"
>>> fout.write(line1)
24
```

Знов-таки, об'єкт файлу відстежує, де знаходиться курсор, тож якщо ви знову викликаєте write, він додасть нові дані в кінець.

Потрібно переконатися, що завершення рядків під час запису файлу контролюються за допомогою явної вставки символу переходу на новий рядок. Інструкція print автоматично додає символ переходу на новий рядок, однак метод write цього не робить.

```
>>> line2 = 'емблема нашого краю.\n'
>>> fout.write(line2)
24
```

Після завершення запису необхідно закрити файл, щоб переконатися, що останній біт даних фізично записано на диск, і вони не будуть втрачені у разі вимкнення живлення.

```
>>> fout.close()
```

Можна також закривати файли, які відкриваються для зчитування, але ми можна трохи розслабитись і не так уважно стежити за всім, якщо відкриваємо лише кілька файлів, оскільки Python гарантує, що всі відкриті файли будуть закриті після завершення роботи програми. Коли записуємо у файли, потрібно явно закривати їх, щоб не залишати жодних шансів для випадкових дій.

7.9 Налагодження програми

Під час зчитування і запису файлів ви можете зіштовхнутися з проблемами пробілів. Ці помилки важко налагодити, адже пробіли, табуляції та переноси рядків зазвичай невидимі:

```
>>> s = '1 2\t 3\n 4'
>>> print(s)
1 2 3
 4
```

На допомогу може прийти вбудована функція repr. Вона приймає будь-який об'єкт як аргумент і повертає його подання у вигляді рядка. Для рядків вона відображає пробіли з послідовністю обернених скісних рисок:

```
>>> print(repr(s))
'1 2\t 3\n 4'
```

Це може допомогти у налагодженні програми.

Ще одна проблема, з якою можна зіткнутися – різні системи використовують різні символи для позначення кінця рядка. Деякі системи використовують символ переходу на новий рядок, який позначається \n. Інші – символ повернення каретки, який позначається \r. Деякі використовують обидва символи. Якщо ви переносите файли між різними системами, ці невідповідності можуть ускладнити роботу.

Для більшості систем існують програми для конвертації з одного формату на інший. Ви можете знайти їх (і почитати більше про це) за посиланням https://www.wikipedia.org/wiki/Newline. Або, звісно, можете написати програму самостійно.

7.10 Словник

catch (ловити) Запобігати завершенню програми через виняток за допомогою операторів try та except.

newline (символ переходу на новий рядок) Спеціальний символ, який використовується у файлах і рядках для позначення кінця рядка.

Pythonic (пітонівський) Метод, який ефективно працює в Python. «Використання try and except – це пітонівський спосіб відновлення відсутніх файлів».

Quality Assurance (Забезпечення Якості (QA)) Особа або команда, яка займається забезпеченням загальної якості програмного продукту. QA часто беруть участь у тестуванні продукту та виявленні проблем до того, як продукт буде випущено.

text file (текстовий файл) Послідовність символів, що зберігається в постійному сховищі, наприклад, на жорсткому диску.

7.11 Вправи

Вправа 1: Напишіть програму, яка зчитує файл і виводить його вміст (рядок за рядком) у верхньому регістрі. Програма має виконуватись таким чином:

```
python shout.py
Введіть ім’я файлу: mbox-short.txt
ВІД STEPHEN.MARQUARD@UCT.AC.ZA СУБ., 5 СІЧНЯ 09:14:16 2008
ЗВОРОТНА АДРЕСА: <POSTMASTER@COLLAB.SAKAIPROJECT.ORG>
ОТРИМАНО: ВІД MURDER (MAIL.UMICH.EDU [141.211.14.90])
        ДО FRANKENSTEIN.MAIL.UMICH.EDU (CYRUS V2.3.8) WITH LMTPA;
        СУБ., 05 СІЧНЯ 2008 09:14:16 -0500
```

Завантажити файл можна за посиланням www.py4e.com/code3/mbox-short.txt

Вправа 2: Напишіть програму, яка запитує ім'я файлу, а потім зчитує його і шукає рядки такої форми:

X-DSPAM-Confidence: 0.8475

Коли ви зустрінете рядок, який починається з «X-DSPAM-Confidence:», розділіть його на частини, щоб отримати число з рухомою крапкою. Порахуйте ці рядки, а потім обчисліть загальну суму значень достовірності спаму з них. Коли дійдете до кінця файлу, виведіть середнє значення достовірності спаму.

Введіть ім’я файлу: mbox.txt
Середній рівень довіри до спаму: 0.894128046745

Введіть ім’я файлу: mbox-short.txt
Середній рівень довіри до спаму: 0.750718518519

Перевірте ваш файл на файлах *mbox.txt* та *mbox-short.txt*.

Вправа 3:

Іноді, коли програмістам стає нудно або вони хочуть трохи розважитися, вони додають до своєї програми нешкідливу приховану. Змініть програму, яка запитує у користувача ім'я файлу, таким чином, щоб вона виводила кумедне повідомлення, коли користувач вводить саме таке ім'я файлу «ля ля ля бе бе бе». Програма повинна нормально працювати з усіма іншими файлами, як наявними, так тими, яких не існує. Нижче наведено приклад запуску програми:

python egg.py
Введіть ім’я файлу: mbox.txt
Всього було 1797 рядів теми у файлі mbox.txt

python egg.py
Введіть ім’я файлу: missing.tyxt
Неможливо відкрити файл: missing.tyxt

python egg.py
Введіть ім’я файлу: ля ля ля бе бе бе
ЛЯ ЛЯ ЛЯ БЕ БЕ БЕ ДЛЯ ТЕБЕ – Тебе розіграли!;)

Ми не заохочуємо додавати прихованки у ваші програми; це просто вправа.