Концепция, описанная в данной главе, может быть использована для идентификации и обработки всех возможных исключений, которые могут возникать в процессе выполнения программы. Наличие блоков `try` и `except` позволит вам уберечь свою программу от аварийного завершения и должным образом обрабатывать все возникающие ошибки.

7.7. Упражнения

В большинстве упражнений из данной главы вам придется работать с файлами. В каких-то из них содержание файлов будет не важно, в других вам придется заранее создать и заполнить их при помощи любого текстового редактора. Будут в этой главе и упражнения на чтение конкретных данных, таких как список слов, имен или химических элементов. Эти наборы данных можно скачать с сайта автора по следующей ссылке: http://www.cpsc.ucalgary.ca/~bdstephe/PythonWorkbook.

Упражнение 149. Отображаем начало файла

(*Решено. 40 строк*)

В операционных системах на базе Unix обычно присутствует утилита с названием `head`. Она выводит первые десять строк содержимого файла, имя которого передается в качестве аргумента командной строки. Напишите программу на Python, имитирующую поведение этой утилиты. Если файла, указанного пользователем, не существует, или не задан аргумент командной строки, необходимо вывести соответствующее сообщение об ошибке.

Упражнение 150. Отображаем конец файла

(*Решено. 35 строк*)

Продолжая тему предыдущего упражнения, в тех же операционных системах на базе Unix обычно есть и утилита с названием `tail`, которая отображает последние десять строк содержимого файла, имя которого передается в качестве аргумента командной строки. Реализуйте программу, которая будет делать то же самое. Так же, как и в упражнении 149, в случае отсутствия файла, указанного пользователем, или аргумента командной строки вам нужно вывести соответствующее сообщение.

Данную задачу можно решить сразу несколькими способами. Например, можно все содержимое файла целиком загрузить в список и затем выбрать из него последние десять элементов. А можно дважды прочитать содержимое файла: первый раз, чтобы посчитать количество строк, а второй – чтобы отобразить последние десять из них. При этом оба пере-

численных подхода нежелательны, если речь идет о файлах достаточного объема. Существует решение, требующее единственного чтения файла и сохранения всех десяти строк за раз. В качестве дополнительного задания разработайте такой алгоритм.

Упражнение 151. Сцепляем файлы

(*Решено. 28 строк*)

Продолжаем тему операционных систем на базе Unix, в которых обычно также есть утилита с названием `cat`, что является сокращением от concatenate (сцепить). Эта утилита выводит на экран объединенное содержимое нескольких файлов, имена которых передаются ей в качестве аргументов командной строки. При этом файлы сцепляются в том порядке, в котором указаны в аргументах.

Напишите программу на Python, имитирующую работу этой утилиты. В процессе работы программа должна выдавать сообщения о том, какие файлы открыть не удается, и переходить к следующим файлам. Если программа была запущена без аргументов командной строки, на экран должно быть выведено соответствующее сообщение об ошибке.

Упражнение 152. Нумеруем строки в файле

(*23 строки*)

Напишите программу, которая будет считывать содержимое файла, добавлять к считанным строкам порядковый номер и сохранять их в таком виде в новом файле. Имя исходного файла необходимо запросить у пользователя, так же, как и имя целевого файла. Каждая строка в созданном файле должна начинаться с ее номера, двоеточия и пробела, после чего должен идти текст строки из исходного файла.

Упражнение 153. Самое длинное слово в файле

(*39 строк*)

В данном упражнении вы должны написать программу, которая будет находить самое длинное слово в файле. В качестве результата программа должна выводить на экран длину самого длинного слова и все слова такой длины. Для простоты принимайте за значимые буквы любые непробельные символы, включая цифры и знаки препинания.

Упражнение 154. Частота букв в файле

(*43 строки*)

Одна из техник декодирования простейших алгоритмов шифрования заключается в применении частотного анализа. Иными словами, вы просто

анализируете зашифрованный текст, подсчитывая частоту употребления всех букв. Затем можно использовать операции подстановки для замены наиболее популярных символов на часто используемые в языке буквы (в английском это, например, буквы E и T).

Напишите программу, которая будет способствовать дешифрации текста путем вывода на экран частоты появления разных букв. При этом пробелы, знаки препинания и цифры должны быть проигнорированы. Также не должен учитываться регистр, то есть символы `a` и `A` должны восприниматься как одна буква. Имя файла для анализа пользователь должен передавать программе посредством аргумента командной строки. Если программе не удастся открыть файл для анализа или аргументов командной строки будет слишком много, на экране должно быть отображено соответствующее сообщение об ошибке.

Упражнение 155. Частота слов в файле

(*37 строк*)

Разработайте программу, которая будет показывать слово (или слова), чаще остальных встречающееся в текстовом файле. Сначала пользователь должен ввести имя файла для обработки. После этого вы должны открыть файл и проанализировать его построчно, разделив при этом строки по словам и исключив из них пробелы и знаки препинания. Также при подсчете частоты появления слов в файле вам стоит игнорировать регистры.

Подсказка. Возможно, при решении этого задания вам поможет код, реализованный вами в упражнении 117.

Упражнение 156. Сумма чисел

(*Решено. 26 строк*)

Напишите программу, которая будет суммировать все числа, введенные пользователем, игнорируя при этом нечисловой ввод. Выводите на экран текущую сумму чисел после каждого очередного ввода. Ввод пользователем значения, не являющегося числовым, должен приводить к появлению соответствующего предупреждения, после чего пользователю должно быть предложено ввести следующее число. Выход из программы будет осуществляться путем пропуска ввода. Удостоверьтесь, что ваша программа корректно обрабатывает целочисленные значения и числа с плавающей запятой.

Подсказка. В данном упражнении вам придется поработать не с файлами, а с исключениями.

Упражнение 157. Буквенные и числовые оценки

(*106 строк*)

Напишите программу, выполняющую перевод из буквенных оценок в числовые и обратно. Программа должна позволять пользователю вводить несколько значений для перевода – по одному в каждой строке. Для начала предпримите попытку сконвертировать введенное пользователем значение из числового в буквенное. Если возникнет исключение, попробуйте выполнить обратное преобразование – из буквенного в числовое. Если и эта попытка окончится неудачей, выведите предупреждение о том, что введенное значение не является допустимым. Пусть ваша программа конвертирует оценки до тех пор, пока пользователь не оставит ввод пустым. При решении данного задания вам могут помочь наработки из упражнений 52 и 53.

Упражнение 158. Удаляем комментарии

(*Решено. 53 строки*)

Как вы знаете, в языке Python для создания комментариев в коде используется символ #. Комментарий начинается с этого символа и продолжается до конца строки – без возможности остановить его раньше.

В данном упражнении вам предстоит написать программу, которая будет удалять все комментарии из исходного файла с кодом на языке Python. Пройдите по всем строкам в файле на предмет поиска символа #. Обнаружив его, программа должна удалить все содержимое, начиная с этого символа и до конца строки. Для простоты не будем рассматривать ситуации, когда знак решетки встречается в середине строки. Сохраните новое содержимое в созданном файле. Имена файла источника и файла назначения должны быть запрошены у пользователя. Удостоверьтесь в том, что программа корректно обрабатывает возможные ошибки при работе с обоими файлами.

Упражнение 159. Случайный пароль из двух слов

(*Решено. 39 строк*)

Создание пароля посредством генерирования случайных символов может обернуться сложностью в запоминании полученной относительно надежной последовательности. Некоторые системы создания паролей рекомендуют сцеплять вместе два слова на английском языке, тем самым упрощая запоминание заветного ряда символов – правда, в ущерб его надежности.

Напишите программу, которая будет открывать файл со списком слов, случайным образом выбирать два из них и сцеплять вместе для получения итогового пароля. При создании пароля исходите из следующего требования: он должен состоять минимум из восьми символов и максимум из

десяти, а каждое из используемых слов должно быть длиной хотя бы в три буквы. Кроме того, сделайте заглавными первые буквы обоих слов, чтобы легко можно было понять, где заканчивается одно и начинается другое. По завершении процесса полученный пароль должен быть отображен на экране.

Упражнение 160. Странные слова

(*67 строк*)

Ученикам, желающим запомнить правила написания слов в английском языке, часто напоминают следующее рифмованное одностишие: «I before E except after C» (I перед E, если не после C). Это правило позволяет запомнить, в какой последовательности писать буквы I и E, идущие в слове одна за другой, а именно: буква I должна предшествовать букве E, если непосредственно перед ними не стоит буква C. Если стоит – порядок гласных будет обратным. Примеры слов, на которые действует это правило: believe, chief, fierce, friend, ceiling и receipt. Но есть и исключения из этого правила, и одним из них является слово weird (странный).

Напишите программу, которая будет построчно обрабатывать текстовый файл. В каждой строке может присутствовать много слов, а может и не быть ни одного. Слова, в которых буквы E и I не соседствуют друг с другом, обработке подвергать не следует. Если же такое соседство присутствует, необходимо проверить, соответствует ли написание анализируемого слова указанному выше правилу. Создайте и выведите на экран два списка. В первом должны располагаться слова, следующие правилу, а во втором – нарушающие его. При этом списки не должны содержать повторяющиеся слова. Также отобразите на экране длину каждого списка, чтобы пользователю было понятно, сколько слов в файле не отвечает правилу.

Упражнение 161. Что за химический элемент?

(*59 строк*)

Напишите программу, которая будет считывать файл, содержащий информацию о химических элементах, и сохранять ее в более подходящей для этого структуре данных. После этого пользователь должен ввести значение. Если введенное значение окажется целочисленным, программа должна вывести на экран обозначение и название химического элемента с введенным количеством протонов. При вводе пользователем строки необходимо отобразить количество протонов элемента с введенным пользователем обозначением или названием. Если введенное пользователем значение не соответствует ни одному из элементов в файле, необходимо вывести соответствующее сообщение об ошибке. Позвольте пользователю вводить значения до тех пор, пока он не оставит ввод пустым.

Упражнение 162. Книги без буквы E

(*Решено. 50 строк*)

Истории литературы известен случай написания романа объемом около 50 тыс. слов, в котором ни разу не была употреблена самая популярная в английском алфавите буква E. Название его – «Gadsby».

Напишите программу, которая будет считывать список слов из файла и собирать статистику о том, в каком проценте слов используется каждая буква алфавита. Выведите результат для всех 26 букв английского алфавита и отдельно отметьте букву, которая встречалась в словах наиболее редко. В вашей программе должны игнорироваться знаки препинания и регистр символов.

Примечание. Липограмма представляет собой литературный прием, состоящий в написании текста без использования одной из букв (или группы букв). Часто отвергнутой буквой является одна из распространенных гласных, хотя это условие и не обязательно. Например, в стихотворении Эдгара Аллана По «Ворон» («The Raven»), состоящем из более тысячи слов, ни разу не встречается буква Z, что делает его полноценной липограммой. Еще одним примером липограммы является роман «Исчезание» («La Disparition»). Во французской и английской версиях этого произведения общим объемом примерно в 300 страниц не употребляется буква E, за исключением фамилии автора.

Упражнение 163. Популярные детские имена

(*Решено. 54 строки*)

Набор данных, содержащий детские имена, состоит более чем из 200 файлов, в каждом из которых, помимо сотни имен, указано количество названных тем или иным именем детей. При этом файлы отсортированы по убыванию популярности имен. Для каждого года присутствует по два файла: в одном перечислены мужские имена, в другом – женские. Совокупный набор данных содержит информацию для всех лет, начиная с 1900-го и заканчивая 2012-м.

Напишите программу, которая будет считывать по одному все файлы из набора данных и выделять имена, которые были лидерами по частоте использования как минимум в одном году. На выходе должно получиться два списка: в одном из них будут присутствовать наиболее популярные имена для мальчиков, во втором – для девочек. При этом списки не должны содержать повторяющиеся имена.

Упражнение 164. Универсальные имена

(*56 строк*)

Некоторые имена, такие как Бен, Джонатан и Эндрю, подходят только для мальчиков, другие – Ребекка или Флора – только для девочек. Но есть

и универсальные имена наподобие Крис или Алекс, которые могут носить и мальчики, и девочки.

Напишите программу, которая будет выводить на экран имена, использованные для мальчиков и девочек в указанном пользователем году. Если в этом году универсальных имен не было, нужно известить об этом пользователя. Кроме того, если за указанный пользователем год не было данных по именам, выведите соответствующее сообщение об ошибке. Дополнительные детали по хранению имен в файлах – в упражнении 163.

Упражнение 165. Самые популярные имена за период

(*76 строк*)

Напишите программу, которая будет определять самые популярные имена для детей в выбранном пользователем периоде. Используйте базу данных из упражнения 163. Позвольте пользователю выбрать первый и последний год анализируемого диапазона. В результате программа должна вывести на экран мужское и женское имена, которые были чаще остальных даны детям в заданный период времени.

Упражнение 166. Имена без повторов

(*41 строка*)

Продолжаем использовать базу имен из упражнения 163. Проходя по файлам, выберите имена без дублирования отдельно для мальчиков и для девочек и выведите их на экран. Разумеется, повторяющихся имен в этих списках быть не должно.

Упражнение 167. Проверяем правильность написания

(*Решено. 58 строк*)

Автоматическая проверка орфографии не помешала бы многим из нас. В данном упражнении мы напишем простую программу, сверяющую слова из текстового файла со словарем. Неправильно написанными будем считать все слова, которых не нашлось в словаре.

Имя файла, в котором требуется выполнить орфографическую проверку, пользователь должен передать при помощи аргумента командной строки. В случае отсутствия аргумента должна выдаваться соответствующая ошибка. Сообщение об ошибке также должно появляться, если не удается открыть указанный пользователем файл. Используйте свои наработки из упражнения 117 при решении данной задачи, чтобы знаки препинания не мешали вам проводить орфографический анализ текста. Также вам следует игнорировать регистр символов при выполнении проверки.

Подсказка. Конечно, вы могли бы загрузить все слова из текста в список и анализировать их при помощи функции оператора `in`. Но эта операция будет выполняться довольно долго. Гораздо быстрее в Python выполняется проверка на наличие определенного ключа в словаре или значения в наборе. Если вы остановитесь на варианте со словарем, ключами должны стать слова, а значениями – ноль или любое другое число, поскольку их мы в данном случае использовать не будем.

Упражнение 168. Повторяющиеся слова

(*61 строка*)

Проверка орфографии – лишь составная часть расширенного текстового анализа на предмет наличия ошибок. Одной из самых распространенных ошибок в текстах является повторение слов. Например, автор может по ошибке дважды подряд написать одно слово, как в следующем примере:

```
At least one value must be entered
entered in order to compute the average.
```

Некоторые текстовые процессоры умеют распознавать такой вид ошибок при выполнении текстового анализа.

В данном упражнении вам предстоит написать программу для определения наличия дублей слов в тексте. При нахождении повтора на экран должен выводиться номер строки и дублирующееся слово. Удостоверьтесь, что программа корректно обрабатывает случаи, когда повторяющиеся слова находятся на разных строках, как в предыдущем примере. Имя файла для анализа должно быть передано программе в качестве единственного аргумента командной строки. При отсутствии аргумента или невозможности открыть указанный файл на экране должно появляться соответствующее сообщение об ошибке.

Упражнение 169. Редактирование текста в файле

(*Решено. 52 строки*)

Перед публикацией текста или документа обычно принято удалять или изменять в них служебную информацию.

В данном упражнении вам необходимо написать программу, которая будет заменять все служебные слова в тексте на символы звездочек (по количеству символов в словах). Вы должны осуществлять регистрозависимый поиск служебных слов в тексте, даже если эти слова входят в состав других слов. Список служебных слов должен храниться в отдельном файле. Сохраните отредактированную версию исходного файла в новом файле. Имена исходного файла, файла со служебными словами и нового файла должны быть введены пользователем.

Примечание. Вам может пригодиться метод `replace` для работы со строками при решении этого задания. Информацию о работе данного метода можно найти в интернете.

В качестве дополнительного задания расширьте свою программу таким образом, чтобы она выполняла замену служебных слов вне зависимости от того, какой регистр символов используется в тексте. Например, если в списке служебных слов будет присутствовать слово `exam`, то все следующие варианты слов должны быть заменены звездочками: exam, Exam, ExaM и EXAM.

Упражнение 170. Пропущенные комментарии

(*Решено. 48 строк*)

При написании функций хорошей практикой считается предварение ее блоком комментариев с описанием назначения функции, ее входных параметров и возвращаемого значения. Но некоторые разработчики просто не пишут комментарии к своим функциям. Другие честно собираются написать их когда-нибудь в будущем, но руки так и не доходят.

Напишите программу, которая будет проходить по файлу с исходным кодом на Python и искать функции, не снабженные блоком комментариев. Можно принять за аксиому, что строка, начинающаяся со слова `def`, следом за которым идет пробел, будет считаться началом функции. И если функция документирована, предшествующая строчка должна начинаться со знака #. Перечислите названия всех функций, не снабженных комментариями, вместе с именем файла и номером строки, с которой начинается объявление функции.

Одно или несколько имен файлов с кодом на языке Python пользователь должен передать в функцию в качестве аргументов командной строки. Для файлов, которые не существуют или не могут быть открыты, должны выдаваться соответствующие предупреждения, после чего должна быть продолжена обработка остальных файлов.

Упражнение 171. Строки фиксированной длины

(*45 строк*)

Ширина окна терминала обычно составляет 80 символов, хотя есть более широкие и узкие терминалы. Это затрудняет отображение текстов, разбитых на абзацы. Бывает, что строки в исходном файле оказываются слишком длинными и автоматически переносятся, тем самым затрудняя чтение, или слишком короткими, что приводит к недостаточному заполнению свободного места на экране.

Напишите программу, которая будет открывать файл и выводить его на экран с постоянной длиной строк. Если в исходном файле строка ока-

зывается слишком длинной, «лишние» слова должны быть перенесены на следующую строку, а если слишком короткой, слова со следующей строки должны перекочевать в конец текущей до полного ее заполнения. Например, следующий отрывок из «Алисы в Стране чудес»:

```
Alice was
beginning to get very tired of sitting by her
sister
on the bank, and of having nothing to do: once
or twice she had peeped into the book her sister
was reading, but it had
no
pictures or conversations in it, "and what is
the use of a book," thought Alice, "without
pictures or conversations?"
```

в отформатированном виде с длиной строки в 50 символов будет выглядеть так:

```
Alice was beginning to get very tired of sitting
by her sister on the bank, and of having nothing
to do: once or twice she had peeped into the book
her sister was reading, but it had no pictures or
conversations in it, "and what is the use of a
book," thought Alice, "without pictures or
conversations?"
```

Убедитесь, что ваша программа корректно обрабатывает тексты, в которых присутствуют абзацы. Идентифицировать конец одного абзаца и начало другого можно по наличию пустой строки в тексте после удаления символа конца строки.

Подсказка. Используйте константу для задания максимальной ширины строк. Это позволит вам легко адаптировать программу под любые окна.

Упражнение 172. Слова с шестью гласными в ряд

(*56 строк*)

Существует как минимум одно слово в английском языке, содержащее все гласные буквы в том порядке, в котором они расположены в алфавите, а именно A, E, I, O, U и Y. Напишите программу, которая будет просматривать текстовый файл на предмет поиска и отображения таких слов. Имя исходного файла должно быть запрошено у пользователя. Если имя файла окажется неверным или возникнут иные ошибки при чтении файла, выведите соответствующее сообщение об ошибке.