Прикладне програмування

ТЕМА 1. РЕГУЛЯРНІ ВИРАЗИ

Регулярні вирази

Регулярні вирази є інструментом обробки текстової інформації.

Інструментом доволі потужним, який дозволяє без написання великої кількості програмного коду розв’язувати складні задачі.

Використання регулярних виразів для обробки текстів полягає у описі та застосуванні шаблонів, що допомагають знаходити або замінювати частини рядків.

Для опису цих шаблонів існує мова зі спеціальним синтаксисом.

Регулярні вирази реалізовані у ряді мов програмування.

Ми розглянемо їх реалізацію у Python.

Регулярні вирази.2

Шаблон регулярного виразу – це рядок, що містить звичайні символи та так звані метасимволи.

Метасимвол може відповідати багатьом звичайним символам, вказувати місце застосування шаблону у рядку або кількість повторень шаблону.

Метасимволи ще називають спеціальними символами та позначають звичайними символами або escape-послідовностями (використовуючи символ обернена коса риска ‘\’).

Шаблон **відповідає** (match) рядку або його підрядку, якщо застосуванням шаблону до рядка можна отримати відповідний підрядок.

Спеціальні символи, що відповідають багатьом звичайним символам

Основні спеціальні символи, що відповідають багатьом звичайним символам

Символ	Опис	Приклад
Рядок звичайних символів	Відповідає такому ж рядку	**сон** відповідає рядку **сон**
.	Відповідає будь-якому одному звичайному символу, окрім символу переходу на новий рядок \n	**.он** відповідає рядкам **сон**, **тон** тощо
\d	Відповідає будь-якій цифрі від 0 до 9 (\D є запереченням \d, тобто, відповідає будь-якому символу, що не є цифрою)	**\d\d** відповідає **25**, **74** тощо
\w	Відповідає будь-якому символу, що є літерою або цифрою (\W є запереченням \w)	**\wон** відповідає рядкам **сон**, **тон**, **2он** тощо
\s	Відповідає символу пропуску, включаючи табуляцію, але не включаючи перехід на новий рядок \n (\S є запереченням \s)	**A\s1** відповідає рядку **A 1**

Спеціальні символи, що відповідають місцю застосування шаблону

Спеціальні символи, що відповідають місцю застосування шаблону не відповідають кожному символу рядка, але позначають, звідки треба починати або закінчувати порівняння радка та шаблону.

Сим-вол	Опис	Приклад
^	Відповідає початку рядка	**^сон** відповідає рядкам **сон**, **сонг**, **сонм**, **сон у літню ніч** тощо
$	Відповідає закінченню рядка	**сон$** відповідає рядкам **сон**, **фасон**, **клаксон** тощо
\b	Відповідає границі слова, тобто, початку або завершенню слова (\B є запереченням \b)	**\bсон\b** відповідає рядкам **сон**, **сон у літню ніч**, **дивний сон** тощо

Спеціальні символи, що задають кількість повторень шаблону

Ці спеціальні символи задають кількість повторень шаблону, що йде перед символом. Також кількість повторень може бути задана натуральними числами у фігурних дужках { }.

Символ або позначення	Опис	Приклад
*	Нуль або більше повторень шаблону, що йде перед символом *	**A\d*** відповідає рядкам **A**, **A2**, **A5897** тощо
+	Одне або більше повторень шаблону	**A\d+** відповідає рядкам **A2**, **A5897** тощо
?	Нуль або одне повторення шаблону	**A\d?** відповідає рядкам **A**, **A2**, **A5** тощо
{N}	N повторень шаблону, де N – натуральне число	**A\d{2}** відповідає рядкам **A25**, **A58** тощо
{N,M}	Від N до M повторень шаблону, де N, M – натуральні числа	**A\d{3,4}** відповідає рядкам **A725**, **A5897** тощо

«Жадібність» символів повторення та її обмеження

Символи повторення *, +, ? є «жадібними» у тому розумінні, що вони намагаються захопити відповідний підрядок найбільшої можливої довжини.

Наприклад шаблон **a\w*b** при застосуванні до рядка **ababab** відповідає тільки всьому рядку.

Така поведінка часто є небажаною.

Для того, щоб шаблон «захоплював» якнайменше символів рядка, треба після символу повторення ставити знак питання *?, +?, ??.

Тоді шаблон **a\w*?b** при застосуванні до рядка **ababab** відповідає трьом підрядкам **ab**.

Завдання множин символів

Для завдання множини символів у певній позиції шаблону, використовують квадратні дужки [ ].

У дужках просто перераховують символи, що відповідають шаблону у даній позиції, або задають діапазон символів між c_1, c_2, вказуючи c_1-c_2.

Інколи кажуть, що квадратні дужки задають класи символів, хоча ці класи не мають відношення до об’єктно-орієнтованого програмування.

Наприклад, [aeiou] відповідає довільній англійській голосній літері, а шаблон **c[aeiou]t** відповідає рядкам **cat**, **cut** тощо.

Так само, [0-9] відповідає довільній цифрі, а шаблон **B[0-9]B** відповідає рядкам **B1B**, **B2B** тощо.

Для того, щоб визначити заперечення наявності певної множини символів у даній позиції, у квадратних дужках перед символами та/або діапазоном вказують символ ^.

Наприклад, [^aeiou] відповідає будь-якому символу, що не є англійською голосною літерою, а шаблон **a[^aeiou]e** відповідає рядкам **ace**, **abe**, **a1e**, **a e** тощо.

Вибір з декількох варіантів

Регулярний вираз може задавати вибір одного з декількох варіантів. Варіанти розділяються вертикальною рискою |.

Наприклад, **__|\d{2}** відповідає двом підкресленням або двом цифрам: __, **01**, **25** тощо.

При застосуванні шаблону Python перевіряє варіанти зліва направо.

Використання звичайних символів, які позначаються як метасимволи

Для використання у шаблоні метасимволів як звичайних символів, перед метасимволом вказують обернену косу риску \.

Те ж стосується символів, які є елементами синтаксису регулярних виразів, наприклад, квадратних та круглих дужок.

Наприклад, \. означає «символ крапка», а шаблон **\d+\.** відповідає будь-якій послідовності цифр, за якою слідує крапка: **239.**, **7.** тощо.

Завдання шаблонів у Python

У Python шаблони регулярних виразів – це рядки-константи.

Але у рядках-константах обернена коса риска використовується для позначання escape-послідовностей.

Тобто, щоб позначити \, треба писати \\.

Наприклад, **‘\\b\\d+\\b’**.

Щоб уникнути дублювання обернених косих рисок у шаблоні, використовують так звані «необроблені» рядки (raw strings).

Необроблені рядки у Python позначають символом r перед початком рядка-константи.

Наприклад, **r‘\b\d+\b’**

Знаходження всіх входжень шаблону у рядок

Модуль re містить декілька функцій та класів для обробки регулярних виразів.

Модуль імпортується командою

import re

Найпростіший варіант – знайти всі підрядки, що відповідають заданому шаблону.

Це можна зробити за допомогою функції

re.findall(patt, s)

- де patt – шаблон, s – рядок, до якого застосовується шаблон.

Функція повертає список підрядків, що відповідають шаблону.

Приклад

У текстовому файлі міститься текст, що включає інформацію про роки, коли відбулися певні події.

Побудувати впорядкований список років з тексту разом з номерами рядків файлу, де вказано той чи інший рік (версія 1).

Будемо вважати, що рік задається 3 або 4 цифрами. Тоді отримаємо такий шаблон:

```
P_YEAR = r'\b\d{3,4}'
```

Повернення списку дат здійснюється за допомогою функції findall.

Компіляція регулярних виразів

Для більш швидкої обробки рядка за шаблоном регулярного виразу, цей вираз варто скомпілювати.

Компіляція виконується функцією compile:

re.compile(patt, flags = 0)

- де patt – шаблон, flags – так звані «прапорці», що визначають режими компіляції.

Функція compile повертає відкомпільований шаблон як об’єкт класу regex, який далі використовується для обробки тексту.

Деякі значення параметру flags показано у таблиці нижче:

Компіляція регулярних виразів.2

Прапорець	Опис
re.I або re.IGNORECASE	Не зважати на регістр літер
re.S або re.DOTALL	Символ крапка (.) відповідає будь-якому символу, включаючи символ нового рядка \n
re.X або re.VERBOSE	Дає можливість вставити коментарі, що починаються символом #, у кінці кожного рядка багаторядкового шаблону

Якщо потрібно з’єднати декілька прапорців, їх об’єднують операцією |, наприклад,

re.DOTALL | re.VERBOSE

Функції модуля re та методи класу regex для пошуку відповідних підрядків

Для більшості функцій модуля re існує опис як у вигляді функції, так і у вигляді метода класу regex.

Якщо вказують функцію, то шаблон задають у вигляді рядка.

Якщо ж застосовують метод класу regex до об’єкту цього класу, то використовують вже відкомпільований шаблон.

У таблиці нижче наведено основні функції та методи для пошуку або перевірки відповідності шаблону (rgx – це відкомпільований шаблон, pattern, - шаблон у вигляді рядка).

Функції модуля re та методи класу regex для пошуку відповідних підрядків.2

Функція (метод)	Опис
re.search(pattern, string, flags=0) **або** **rgx.search(string)**	Знайти перший підрядок, що відповідає шаблону, у рядку string. Повертає об’єкт класу match або None, якщо не знайдено.
re.match(pattern, string, flags=0) **або** **rgx.match(string)**	Перевіряє, чи відповідає початок рядка string шаблону. Повертає об’єкт класу match або None, якщо не відповідає.
re.findall(pattern, string, flags=0) **або** **rgx.findall(string)**	Повертає список усіх підрядків рядка string, що відповідають шаблону.
re.finditer(pattern, string, flags=0) **або** **rgx.finditer(string)**	Повертає ітератор, який пробігає послідовність усіх підрядків рядка string, що відповідають шаблону. Кожен елемент послідовності – це об’єкт класу match.

Клас match

Клас match дозволяє визначити характеристики підрядка, який відповідає шаблону.

Об’єкти класу match є результатами функцій (методів) search, match, finditer.

Клас match містить такі основні методи (mt – об’єкт класу match):

Метод	Опис
mt.group()	group без параметрів повертає підрядок, що відповідає шаблону.
mt.start()	Повертає початкову позицію у рядку підрядка, що відповідає шаблону.
mt.end()	Повертає кінцеву позицію у рядку підрядка, що відповідає шаблону.
mt.span()	Повертає кортеж, складений з початкової та кінцевої позиції у рядку підрядка, що відповідає шаблону.

Приклад

У текстовому файлі міститься текст, що включає інформацію про роки, коли відбулися певні події.

Побудувати впорядкований список років з тексту разом з номерами рядків файлу, де вказано той чи інший рік.

Окрім номерів рядків вказати також позицію дати у рядку та позицію дати відносно початку файлу (версія 2).

Повернення дат здійснюється за допомогою функції finditer, аналіз характеристик, - за допомогою методів класу match.

Дата та час

Python має потужні інструменти для роботи з датою та часом

Вони зібрані у 2 стандартних модулях:

- time
- datetime

У модулі time розглянемо 2 функції

time() – повертає поточний час як дійсне число у секундах від epoch, тобто 1970, 00:00:00 (UTC)

sleep(seconds) – призупинити виконання програми на задану кількість секунд (дійсне число)

У модулі datetime розглянемо 2 класи:

- datetime
- timedelta

Клас datetime

Конструктор класу приймає такі головні параметри

d = datetime.datetime(year, month, day, hour=0,

minute=0, second=0, microsecond=0)

Розглянемо також методи

datetime.now() – повертає поточну дату та час

d.weekday() – повертає день тижня від 0 (понеділок) до 6

d.timestamp() – повертає дійсне число у секундах від epoch до d

Клас datetime.2

d.year – рік

d.month – місяць між 1 та 12

d.day – день між 1 та кількістю днів у заданому місяці заданого року

d.hour – година у range(24)

d.minute – хвилина у range(60)

d.second – секунда у range(60)

d.microsecond – мікросекунда у range(1000000)

Клас datetime.3

classmethod datetime.strptime(date_string, format) – повертає об’єкт класу datetime за рядком дати відповідно формату format

d.strftime(format) – повертає рядок, що є представленням дати відповідно формату format

Клас datetime.4

Деякі елементи формату

Елемент формату	Значення
%d	день місяця.
%m	місяць
%y	рік без сторіччя
%Y	рік зі сторіччям
%H	година
%M	хвилина
%S	секунда
%f	мікросекунда.

Клас timedelta

Конструктор класу приймає такі головні параметри

td = datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0)

Над об’єктами класу timedelta можна виконувати арифметичні операції а також комбінувати їх у виразах разом з об’єктами класу datetime.

Наприклад,

d2 = d1 + td

td // datetime.timedelta(hours=1)

Приклад: мінімальна та максмальна дата

У текстовому файлі, окрім іншого тексту, містяться дати.

Знайти мінімальну та максимальні дату, кількість днів між ними а також день тижня максимальної дати

Приклад: інформація про студентів (версія 1)

Установа збирає дані про студентів різних навчальних закладів.

Дані надаються у текстових файлах.

Для кожного студента вказують прізвище, ім’я, по-батькові, номер студентського квитка, дату народження, телефон.

Треба побудувати словник з ключами – номерами студентських квитків та даними – інформацією про студентів.

Також перевірити правильність наданої інформації.

Очікуваний формат даних, наприклад:

Іваненко Іван Іванович АБ23445645 14.02.1998 +380 (50) 234 4567

Приклад: інформація про студентів (версія 1).2

Використаємо регулярні вирази для аналізу тексту.

Шаблони для прізвища (ім’я, по-батькові), номера студентського квитка, номеру телефону та дати мають вигляд:

```
P_NAME = r'\b[A-ЯҐЄІЇ][A-ЯҐЄІЇa-яґєіїʼ]*\b'

P_STUD_PASS = r'[A-ЯҐЄІЇ|A-Z]{2}\d{8}'

P_PHONE = r'\+380\s\(\d{2}\)\s\d{3}\s\d{4}'

P_DATE = r'\d{2}\.\d{2}\.\d{4}'
```

Виділяти підрядки будемо за допомогою функції findall.

Якщо формат даних неправильний, ініціюється виключення ValueError.

Групування. Підгрупи

Окремі частини шаблону можна об’єднувати у підгрупи.

Це дає можливість аналізувати частини тексту, що відповідає шаблону.

Для групування слід взяти частину шаблону, що задає підгрупу, у круглі дужки ( ).

Наприклад шаблон дати у форматі dd.mm.yyyy з виділенням підгруп днів, місяців та років виглядає так: **(\d{2})\.(\d{2})\.(\d{4})**

Круглі дужки також застосовують просто для об’єднання частин шаблону при визначенні варіантів.

Щоб об’єднати частину шаблону без виділення підгрупи, треба після відкриваючої круглої дужки вказати знак питання та двокрапку (?:)

Групування. Підгрупи.2

Для того, щоб повернути виділену підгрупу, застосовують метод group класу match, вказуючи номер підгрупи в якості параметру, або метод groups того ж класу.

mt.group(n) повертає підрядок, що є підгрупою з номером n підрядка, який відповідає шаблону.

Нумерація підгруп починається з 1.

Наприклад, для описаного шаблону дати mt.group(1) поверне день, mt.group(2), - місяць, а mt.group(3), - рік.

mt.groups() повертає кортеж з усіма підрядками-підгрупами.

Приклад: інформація про студентів (версія 2)

Установа збирає дані про студентів різних навчальних закладів.

Дані надаються у текстових файлах.

Для кожного студента вказують прізвище, ім’я, по-батькові, номер студентського квитка, дату народження, телефон.

Треба побудувати словник з ключами – номерами студентських квитків та даними – інформацією про студентів.

Також перевірити правильність наданої інформації.

Почали надходити дані ще з декількох навчальних закладів.

При цьому в них локальні бази даних повертають дату в форматі не тільки dd.mm.yyyy, але й yyyy-mm-dd або yyyy/mm/dd.

Також день та/або місяць може містити одну цифру, а не 2.

Приклад: інформація про студентів (версія 2).2

Очікуваний формат даних, наприклад:

Іваненко Іван Іванович АБ23445645 14.02.1998 +380 (50) 234 4567

або

Іваненко Іван Іванович АБ23445645 1998-02-14 +380 (50) 234 4567

або

Іваненко Іван Іванович АБ23445645 1998/2/14 +380 (50) 234 4567

Приклад: інформація про студентів (версія 2)

Шаблони для прізвища (ім’я, по-батькові), номера студентського квитка та номеру телефону залишаються майже незмінними: в них тільки виділено підгрупи.

```
P_NAME = r'(\b[А-ЯҐЄІЇ][А-ЯҐЄІЇа-яґєії]*)'

P_STUD_PASS = r'([А-ЯҐЄІЇ|A-Z]{2}\d{8})'

P_PHONE = r'(\+380\s\(\d{2}\)\s\d{3}\s\d{4})'
```

Оновлений шаблон дати має вигляд:

```
P_DATE = r'''(\d{1,2}\.\d{1,2}\.\d{4} # дата розділена крапками dd.mm.yyyy
        |\d{4}-\d{1,2}-\d{1,2}  # дата розділена мінусами yyyy-mm-dd
        |\d{1,2}/\d{1,2}/\d{4}) # дата розділена косими рисками mm/dd/yyyy
        '''
```

Об’єднаємо всі шаблони в один та виконаємо їх компіляцію з прапорцем re.VERBOSE.

Виділяти підрядки будемо за допомогою функції search, а аналізувати, - використовуючи метод groups.

Функції модуля re та методи класу regex для зміни рядків

Модуль re також надає засоби для зміни рядка у відповідності з шаблоном регулярного виразу.

Функція (метод)	Опис
re.sub(pattern, repl, string, count=0, flags=0) **або** **rgx.sub(repl, string, count=0)**	Замінити у рядку string всі підрядки, що відповідають шаблону, на repl. Параметр count вказує кількість замін, якщо він не дорівнює 0. repl може бути рядком або функцією, що має 1 аргумент – об'єкт класу match та повертає змінений підрядок.
re.split(pattern, string, maxsplit=0, flags=0) **або** **rgx.split(string, maxsplit =0)**	Поділити рядок string на підрядки, використовуючи в якості рядків-розділювачів підрядки, що відповідають шаблону. Повертає список підрядків після поділу. maxsplit вказує максимальну кількість утворених після поділу підрядків (якщо не дорівнює 0).

Функції модуля re та методи класу regex для зміни рядків.2

sub та split схожі на відповідні методи для рядків replace та split.

Але функції з модуля re є набагато більш потужними та гнучкими.

Наприклад,

```
re.split(r'''[ !?.,+\-:;"'()]+''', st)
```

розіб’є рядок st на слова, вилучивши всі символи-розділювачі.

Приклад: інформація про студентів (версія 3)

Установа збирає дані про студентів різних навчальних закладів. Дані надаються у текстових файлах. Для кожного студента вказують прізвище, ім’я, по-батькові, номер студентського квитка, дату народження, телефон. Треба побудувати словник з ключами – номерами студентських квитків та даними – інформацією про студентів. Також перевірити правильність наданої інформації. Почали надходити дані ще з декількох навчальних закладів. При цьому в них локальні бази даних повертають дату в форматі не тільки dd.mm.yyyy, але й yyyy-mm-dd або yyyy/mm/dd. Також день та/або місяць може містити одну цифру, а не 2.

Також у багатьох записах номер телефону зазначений без вказання + або зовсім без коду країни.

Всі телефони необхідно привести до єдиного формату.

Окрім цього, часті зупинки через неправильні дані роблять роботу неефективною.

Потрібно не зупинятися після неправильних даних, а записувати їх в окремий файл.

Нарешті, потрібно побудований словник з відфільтрованими даними записати у файл.

Приклад: інформація про студентів (версія 3).2

Очікуваний формат даних, наприклад:

Іваненко Іван Іванович АБ23445645 14.02.1998 380 50 234-45-67

або

Іваненко Іван Іванович АБ23445645 1998-02-14 050 234 4567

Шаблони для прізвища (ім’я, по-батькові), номера студентського квитка та дати залишаються незмінними.

Оновлений шаблон номера телефону має вигляд:

```
P_PHONE = r'''(\+380\s\(\d{2}\)\s\d{3}\s\d{4}   # номер у 'канонічному вигляді'
          |\D?380\D{0,2}\d{2}\D{0,2}\d{3}\D?\d{2}\D?\d{2} # номер з кодом країни
          |\b0\d{2}\D?\d{3}\D?\d{2}\D?\d{2})  # номер без коду країни
```

Приклад: інформація про студентів (версія 3).3

Знову об’єднаємо всі шаблони в один та виконаємо їх компіляцію з прапорцем re.VERBOSE.

Виділяти підрядки будемо за допомогою функції search, а аналізувати, - використовуючи метод groups.

Застосуємо функцію sub для видалення зайвих символів з телефонного номера.

Результат будем записувати у файл pickle, а неправильні дані, - у текстовий файл.

Іменовані підгрупи

Підгрупам у шаблонах можна надавати імена.

Для того, щоб задати іменовану підгрупу, треба у круглих дужках написати знак питання, P та ім’я у кутових дужках (?P<name>)

Наприклад, у шаблоні дати можемо виділити іменовані підгрупи для днів, місяців та років:

(?P<day>\d{2})\.(?P<month>\d{2})\.(?P<year>\d{4})

Якщо шаблон містить іменовані підгрупи, то у методі group об’єкту класу match ми можемо отримати підрядок, що відповідає цій підгрупі, звертаючись до неї за ім’ям:

mt.group(‘name’)

Наприклад, для шаблону дат mt.group(‘day’) поверне підрядок, що визначає день дати.

Іменовані підгрупи.2

Також можна отримати словник з інформацією про всі іменовані підгрупи (ім’я підгрупи – ключ у словнику, а відповідний підрядок, - значення)

mt.groupdict()

або ім’я останньої виділеної підгрупи

mt.lastgroup

Ім’я останньої виділеної підгрупи особливо корисне, коли шаблон містить варіанти і потрібно дізнатись, якому варіанту відповідає виділений підрядок.

Приклад: Обробка конфігураційних файлів

Конфігураційний файл складається з рядків, що задають значення певних параметрів програм.

Кожний рядок має вигляд <ім’я> = <значення>.

У більшості великих систем є конфігураційні файли, оскільки вони дозволяють унести зміни до програми без її перепрограмування.

Звичайно, програма повинна очікувати на такі зміни та аналізувати конфігураційний файл.

Необхідно описати модуль, що аналізує конфігураційний файл та повертає словник з параметрами, у якому імена є ключами, а значення, - значеннями.

Обробка конфігураційних файлів. Шаблони та глобальні змінні

Використаємо регулярні вирази для аналізу конфігураційного файлу та виділення його елементів.

Такі елементи ще називають токенами (символами), а програму, що їх виділяє, - токенізатором.

У нашому випадку токенами будуть:

- Ім’я (NAME)
- Знак «дорівнює» (EQ)
- Ціле число (NUM_INT)
- Дійсне число (NUM_FLOAT)
- Рядок (STRING)
- Пропуск (WS)
- Кінець рядка (EOL)
- Коментар (COMMENT)
- Інше (OTHER)

Обробка конфігураційних файлів. Шаблони та глобальні змінні.2

Синтаксис конфігураційного файлу доволі простий.

Як вже було зазначено, один рядок файлу виглядає наступним чином:

<ім’я> = <значення>

Між ім’ям та значенням може стояти довільна кількість пропусків, а після значення – кінець рядка.

Також будемо допускати порожні рядки та коментарі у кінці рядків у стилі Python (починаються символом #).

Значення можуть бути цілими або дійсними числами, а також рядками.

Числа позначаються звичайним чином, а рядки беруться у апострофи або подвійні лапки з обох боків.

Для перевірки синтаксичної правильності конфігураційного файлу достатньо проаналізувати пару токенів: поточний токен та наступний токен.

Якщо ця пара належить множині допустимих пар токенів, то все нормально. Якщо ж ні, - маємо помилку синтаксису.

Обробка конфігураційних файлів. Шаблони та глобальні змінні.3

Задамо шаблони для токенів

```
P_NAME = r'(?P<NAME>[A-Za-zA-ЯҐЄІЇа-яґєії_]\w*)'# шаблон для імені

P_EQ = r'(?P<EQ>=)'                    # шаблон для знаку 'дорівнює'

P_NUM_INT = r'(?P<NUM_INT>[+-]?\d+)'   # шаблон для цілого числа

P_NUM_FLOAT = r'(?P<NUM_FLOAT>[+-]?\d+\.\d+)'   # шаблон для дійсного числа

P_STRING = r'''(?P<STRING>(?:'[^']*')|(?:"[^"]*"))''' # шаблон для рядка
```

Обробка конфігураційних файлів. Шаблони та глобальні змінні.4

```
P_WS = r'(?P<WS>[ \t]+)'                # шаблон для пропусків
P_COMMENT = r'(?P<COMMENT>#.*)'         # шаблон для коментарів
P_EOL = r'(?P<EOL>[\n])'                # шаблон для кінця рядка файлу
P_OTHER = r'(?P<OTHER>.+)'              # шаблон для інших (помилкових символів)
```

Об’єднаємо їх у спільний шаблон аналізу конфігураційного файлу як варіанти

```
P_CONFIG = '|'.join([P_NAME, P_EQ, P_NUM_FLOAT, P_NUM_INT, P_STRING, P_WS, P_COMMENT, P_EOL, P_OTHER])
```

Обробка конфігураційних файлів. Шаблони та глобальні змінні.5

Також визначимо словник допустимих пар токенів

```
VALID_PAIRS = {'NAME': {'EQ'},
               'EQ': {'NUM_INT', 'NUM_FLOAT', 'STRING'},
               'NUM_INT': {'EOL'},
               'NUM_FLOAT': {'EOL'},
               'STRING': {'EOL'},
               'EOL': {'NAME', 'EOL'},
               'OTHER': set()}
```

та множину токенів, які треба пропустити

```
SKIP = {'WS', 'COMMENT'}
```

Обробка конфігураційних файлів. Генератор токенів

Генератор токенів – це генератор-функція, яка за рядком, що складається з багатьох рядків файлу, повертає послідовність токенів.

Генератор токенів використовує шаблон для виділення наступного токену та ігнорує пропуски а також коментарі.

Кожний токен представлений як іменований кортеж

```
Token = namedtuple('Token', ['type', 'value'])
```

- де type – тип токену (‘NAME’, ‘EQ’ тощо), value – підрядок, що є значенням токену.

Обробка конфігураційних файлів. Клас ConfigDict

Клас ConfigDict читає весь вміст конфігураційного файлу та використовує генератор токенів для побудови словника, що складається з імен та значень, які містяться у конфігураційному файлі.

При створенні об’єкту даного класу може бути переданий словник значень параметрів за угодою.

Цей словник модифікується даними з конфігураційного файлу.

Головний метод класу – getconfig, який і повертає словник.

Клас також містить внутрішній метод _check_syntax, який здійснює перевірку синтаксичної правильності та ініціює виключення SyntaxError, якщо у конфігураційному файлі є синтаксична помилка.

Використання конфігураційного файлу. Зображення точок та кіл

У темі «Метакласи та метапрограмування» ми розглядали приклад абстрактного класу Drawer та його нащадку TurtleDraw, який використовує графічну бібліотеку turtle для зображення та переміщення точок та кіл.

Нехай нам потрібно зобразити випадковим чином задану кількість точок та кіл.

При цьому, точки та кола повинні зображуватись своїми кольорами.

Використаємо конфігураційний файл, у якому визначимо такі параметри:

- points_num - кількість точок
- circles_num - кількість кіл
- minxy – мінімальна координата точки (центра кола) по x або y
- maxxy – максимальна координата точки (центра кола) по x або y
- maxr - максимальний радіус кола
- color - колір точок

Для розв’язання задачі застосуємо наш клас ConfigDict.

При створенні об’єкту класу ConfigDict, окрім імені файлу, передамо словник параметрів за угодою, що містить зокрема, ще один параметр – color2 – колір кіл.

Зауваження щодо використання регулярних виразів

Регулярні вирази є потужним засобом аналізу та обробки текстів.

Але, як будь-яка потужна зброя, вони є також небезпечними.

Небезпека використання регулярних виразів полягає у тому, що для нетривіального шаблону важко передбачити роботу компілятора регулярних виразів для всіх можливих рядків.

Щоб зменшити ймовірність помилок треба перевіряти роботу з регулярним виразом на багатьох різних прикладах рядків, застосовувати виведення проміжних результатів (підрядків, що відповідають шаблону), аналізувати позиції виділених підрядків, коментувати складні шаблони.

Слідування цим нескладним правилам допоможе великою мірою уникнути ризиків застосування регулярних виразів.

А переваги їх застосування, звичайно, залишаться.

Резюме

Ми розглянули:

1. Шаблони регулярних виразів.
2. Спеціальні символи
3. Визначення множин символів. Вибір варіантів
4. Компіляція регулярних виразів
5. Функції модуля re та методи класу regex
6. Клас match
7. Групування. Підгрупи
8. Іменовані підгрупи

Де прочитати

1. Обвінцев О.В. Об’єктно-орієнтоване програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. Wesley J. Chun - Core Python Programming – 2001
3. Уэсли Дж. Чан. - Python: создание приложений. Библиотека профессионала, 3-е изд. Пер.с англ. - М. : ООО "И.Д. Вильяме", 2015. - 816 с.
4. David Beazley - Python Cookbook, 3rd edition – 2013
5. Mark Pilgrim. - Dive into Python, Version 5.4 - 2004
6. Mark Lutz - Programming Python. 4th Edition - 2011
7. Felix Lopez, Victor Romero - Mastering Python Regular Expressions – 2014
8. Прохоренок Н.А. - Python 3 и PyQt. Разработка приложений – 2012
9. Hellmann D. - The Python Standard Library by Example – 2011
10. http://www.python-course.eu