

# ІНФОРМАТИКА ТА ПРОГРАМУВАННЯ

---

## Тема 21. 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
<b>Рядок звичайних символів</b>	Відповідає такому ж рядку	<b>сон</b> відповідає рядку <b>сон</b>
.	Відповідає будь-якому одному звичайному символу, окрім символу переходу на новий рядок <code>\n</code>	<b>.он</b> відповідає рядкам <b>сон, тон</b> тощо
<b>\d</b>	Відповідає будь-якій цифрі від 0 до 9 ( <code>\D</code> є запереченням <code>\d</code> , тобто, відповідає будь-якому символу, що не є цифрою)	<b>\d\d</b> відповідає <b>25, 74</b> тощо
<b>\w</b>	Відповідає будь-якому символу, що є літерою або цифрою ( <code>\W</code> є запереченням <code>\w</code> )	<b>\won</b> відповідає рядкам <b>сон, тон, 2он</b> тощо
<b>\s</b>	Відповідає символу пропуску, включаючи табуляцію, але не включаючи перехід на новий рядок <code>\n</code> ( <code>\S</code> є запереченням <code>\s</code> )	<b>\s1</b> відповідає рядку <b>A</b> <b>1</b>

# 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 рядка та шаблону.

Сим-вол	Опис	Приклад
<b>^</b>	Відповідає початку рядка	<b>^сон</b> відповідає рядкам <b>сон, сонг, сонм, сон у літню ніч</b> тощо
<b>\$</b>	Відповідає закінченню рядка	<b>сон\$</b> відповідає рядкам <b>сон, фасон, клаксон</b> тощо
<b>\b</b>	Відповідає границі слова, тобто, початку або завершенню слова (\B є запереченням \b)	<b>\bson\b</b> відповідає рядкам <b>сон, сон у літню ніч, дивний сон</b> 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 *	<b>A\d*</b> відповідає рядкам <b>A</b> , <b>A2</b> , <b>A5897</b> тощо
+	Одне або більше повторень шаблону	<b>A\d+</b> відповідає рядкам <b>A2</b> , <b>A5897</b> тощо
?	Нуль або одне повторення шаблону	<b>A\d?</b> відповідає рядкам <b>A</b> , <b>A2</b> , <b>A5</b> тощо
{N}	N повторень шаблону, де N – натуральне число	<b>A\d{2}</b> відповідає рядкам <b>A25</b> , <b>A58</b> тощо
{N,M}	Від N до M повторень шаблону, де N, M – натуральні числа	<b>A\d{3,4}</b> відповідає рядкам <b>A725</b> , <b>A5897</b> 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  $a\backslash w^*b$  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  $a\backslash w^*?b$  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 рисою |.
- 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.
- 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
<code>re.search(pattern, string, flags=0)</code> або <code>regex.search(string)</code>	Знайти перший підрядок, що відповідає шаблону, у рядку <code>string</code> . Повертає об'єкт класу <code>match</code> або <code>None</code> , якщо не знайдено.
<code>re.match(pattern, string, flags=0)</code> або <code>regex.match(string)</code>	Перевіряє, чи відповідає початок рядка <code>string</code> шаблону. Повертає об'єкт класу <code>match</code> або <code>None</code> , якщо не відповідає.
<code>re.findall(pattern, string, flags=0)</code> або <code>regex.findall(string)</code>	Повертає список усіх підрядків рядка <code>string</code> , що відповідають шаблону.
<code>re.finditer(pattern, string, flags=0)</code> або <code>regex.finditer(string)</code>	Повертає ітератор, який пробігає послідовність усіх підрядків рядка <code>string</code> , що відповідають шаблону. Кожен елемент послідовності – це об'єкт класу <code>match</code> .

# 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
<code>mt.group()</code>	<code>group</code> без параметрів повертає підрядок, що відповідає шаблону.
<code>mt.start()</code>	Повертає початкову позицію у рядку підрядка, що відповідає шаблону.
<code>mt.end()</code>	Повертає кінцеву позицію у рядку підрядка, що відповідає шаблону.
<code>mt.span()</code>	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`.

# 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-яґєії]\*\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.
- 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[A-ЯҐЄІІ][A-ЯҐЄІІa-яґєії]\*)'**

**P\_STUD\_PASS = r'([A-ЯҐЄІІ|A-Z]{2}\d{8})'**

**P\_PHONE = r'(\+380\s{1}\d{2})\s{1}\d{3}\s{1}\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 уууу-мм-дд**

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

- 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 будемо 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, - токенизатором.
- 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-ЯҐЄІіа-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

**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.4

- 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 токена та ігнорує пропуски а також коментарі.
- Кожний токен представлений як іменованний кортеж

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

- де type – тип токена ('NAME', 'EQ' тощо), value – підрядок, що є значенням токена.

# 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
  - `minx` – мінімальна координата точки (центра кола) по `x` або `y`
  - `maxx` – 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. Інформатика та програмування. Курс на основі Python. Матеріали лекцій. – К., Основа, 2017
2. Wesley J. Chun - Core Python Programming - 2001
3. David Beazley - Python Cookbook, 3rd edition – 2013
4. Mark Pilgrim. - Dive into Python, Version 5.4 - 2004
5. Mark Lutz - Programming Python. 4<sup>th</sup> Edition - 2011
6. Felix Lopez, Victor Romero - Mastering Python Regular Expressions – 2014
7. Прохоренко Н.А. - Python 3 и PyQt. Разработка приложений – 2012
8. Hellmann D. - The Python Standard Library by Example – 2011
9. <http://www.python-course.eu>