# Python Essential

## Регулярні вирази (Regex)

## План заняття

1. Що таке регулярні вирази?
2. Регулярні вирази у Python
3. Основні інструменти для роботи з RegEx
4. Основні метасимволи в RegEx
5. Використання в Python

# Після уроку обов'язково


Повторіть цей урок у форматі відео на ITVDN.com

Доступ можна отримати через керівництво вашого навчального центру


Перевірте, як Ви засвоїли цей матеріал на TestProvider.com

Тема

# Регулярні вирази (Regex)

## Регулярні вирази (Regex)

**Регулярні вирази** (англ. *regular expressions*) — формальна мова, що використовується в комп'ютерних програмах, які працюють із текстом, для пошуку та здійснення маніпуляцій з підрядками в тексті, заснована на використанні метасимволів (символів-джокерів, англ. wildcard characters).

Для пошуку використовується рядок-зразок (англ. *pattern*, українською його часто називають «шаблоном», «маскою»), що складається з символів і метасимволів та задає правило пошуку. Для маніпуляцій із текстом додатково задається рядок заміни, який також може містити спеціальні символи.


```
[^ ]*?@[^ ]*?\.[^ ]*
```

# Регулярні вирази (Regex)

## Початок роботи з Regex в Python

Для того, щоб приступити до роботи з регулярними виразами в Python, необхідно імпортувати модуль **re** у свій проект за допомогою наступної команди:

```
import re
```


```
[^ ]*?@[^ ]*?\.[^ ]*
```

## Основні функції модуля re

**re.match(*pattern, string*)** – знаходить входження фрагменту **на початку рядка:**

```
import re

string = "Test1 Test2 Test3 Test4 Test5"

result = re.match(r"Test", string)
print(result)

# метод group() повертає фрагмент рядку, в якому було виявлено співпадіння
print(result.group(0))

# поле string поверне рядок, який передавали для пошуку
print(result.string)

# Результат - None, оскільки фрагмент знаходиться не на початку
result1 = re.match(r"Test2", string)
print(result1)
```


## Основні методи модуля re

**re.match(*pattern, string*)** – знаходить входження фрагменту **на початку рядка:**

```
import re

string = "Test1 Test2 Test3 Test4 Test5"

result = re.match(r"Test", string)

# метод span() поверне кортеж, який містить початкову та кінцеву позиції шуканого фрагмента
print(result.span())

# Позиція початку шаблону в рядку
print(result.start())

# Позиція кінця шаблону в рядку
print(result.end())

print(result, result.span(), result.string, result.group(), sep='\n')
```


## Основні функції модуля re

**re.search(*pattern, string*)** – схожа на функцію match(), але здійснює пошук заданого шаблону по всьому рядку, повертає тільки перше знайдене співпадіння:

```
import re

string = "Test1 Test2 Test3 Test4 Test3"

result = re.search(r"Test3", string)
print(result.group())
print(result.group(0))

# IndexError: no such group
# print(result.group(1))
```


# Регулярні вирази (Regex)

## Основні функції модуля re

**re.findall(*pattern, string*)** – призначена для пошуку за заданим шаблоном та повертає **всі знайдені значення**:

```
import re

string = "Test1 Test2 Test3 Test4 Test5"

result = re.findall(r"Test", string)

print("Список усіх знайдених співпадінь:", result)
print("Кількість усіх співпадінь =", len(result))
```


## Основні функції модуля re

**re.finditer(*pattern, string*)** – використовується для пошуку всіх співпадінь у шаблоні, які не пересікаються.

- повертає ітератор з об’єктами Match.
- функція finditer() повертає ітератор навіть у тому випадку, коли співпадіння не знайдено.
- функція finditer відмінно підходить для обробки тих команд, виведення яких відображається стовпцями.

```
import re
string = 'Test1 Test2 Test3 Test4 Test5'
pattern = "Test"
for match in re.finditer(pattern, string):
    s = "Found '{group}' at {begin}:{end}".format(
        group=match.group(), begin=match.start(),
        end=match.end())
    # Виводимо кожний знайдений результат:
 print(s)
```


## Основні функції модуля re

**re.sub(*pattern, repl, string*) –** призначена для пошуку за заданим шаблоном та заміни на вказаний підрядок

```
import re

string = "test1 test2 test3 test4 test5"

result = re.sub(r"t", "T", string)
print(result)

# Якщо шукане не знайшли, рядок залишається незмінним
result = re.sub(r"I", "A", string)
print(result)
```


## Основні функції модуля re

**re.split(*pattern, string, [maxsplit=0]*)** – призначена для ділення рядку за заданим шаблоном на таку кількість поділів, на яку це можливо:

```
import re

string = "test1 test2 test3 test4 test5"

result = re.split(r"t", string)
print(result)

# maxsplit – кількість поділів
result = re.split(r"t", string, maxsplit=5)
print(result)

# maxsplit – кількість поділів
result = re.split(r"t", string, maxsplit=10)
print(result)
```


# Регулярні вирази (Regex)

## Основні функції модуля re

**re.compile(*pattern, repl, string*)** – можливість зібрати регулярний вираз в об'єкт, який в свою чергу можна використовувати для пошуку. Призначений для пошуку за заданим шаблоном і дозволяє уникнути переписування одного і того ж коду (виразу):

```
import re

string = "Test1 Test2 Test3 Test4 Test5"
pattern = re.compile("T")
result1 = pattern.findall(string)
print(result1)
# Якщо шукане не знайшли, результат — пустий список
result2 = pattern.findall(string.lower())
print(result2)
```


# Регулярні вирази (Regex)

## Основні прапори модуля re

| Короткий синтаксис | Повний синтаксис | Призначення |
|---|---|---|
| re.A | re.ASCII | Повертає співпадіння лише по ASCII-символам замість усієї таблиці Unicode. |
| re.I | re.IGNORECASE | Ігнорує регістр символів. |
| re.M | re.MULTILINE | Використовується спільно з метасимволами ^ та $. У першому випадку повертає співпадіння на початку кожного нового рядка \n, у другому – наприкінці \n. |
| re.S | re.DOTALL | Змушує метасимвол . повертати співпадіння за абсолютно всіма символами, включаючи \n. Без цього прапору крапка . відповідає будь-якому символу, крім \n. |
| re.X | re.VERBOSE | Дозволяє коментарі в Regex-виразах. |
| re.L | re.LOCALE | Враховує регіональні налаштування при використанні \w, \W, \b, \B, \s і \S. Використовується **тільки** при роботі з байтовими рядками, несумісний з re.ASCII. |

# Регулярні вирази (Regex)

## Основні метасимволи в Regex

| Повний синтаксис | Призначення | Повний синтаксис | Призначення |
|---|---|---|---|
| . | Один будь-який символ, крім нового рядка \n | \b | Повертає співпадіння, якщо слово починається або закінчується потрібною послідовністю символів |
| ? | 0 або 1 входження шаблону зліва | \B | Повертає співпадіння, якщо певні символи є у рядку, але не на початку чи не наприкінці слова |
| + | 1 і більше входжень шаблону зліва | [..] | Один із символів у дужках |
| * | 0 і більше входжень шаблону зліва | [^..] | Будь-який символ, крім тих, що у дужках |
| \w | Будь-яка цифра чи літера | \ | Екранування спеціальних символів (\. означає крапку, або \+ - знак «плюс») |
| \W | Все, окрім літери або цифри | ^ | Початок рядку |
| \d | Будь-яка цифра [0-9] | $ | Кінець рядку |
| \D | Все, окрім цифри | {n,m} | Від n до m входжень ({,m} — від 0 до m) |
| \s | Будь-який символ пробілу | a\|b | Відповідає a або b |
| \S | Будь-який непробільний символ | () | Групує вираз та повертає знайдений текст |
| \A | Чи починається рядок з певної послідовності символів | \t, \n, \r | Символ табуляції, нового рядку та повернення каретки відповідно |
| \Z | Чи закінчується рядок потрібною послідовністю символів | | |

# Дивіться наші уроки у відео форматі

## ITVDN.com


Перегляньте цей урок у відео форматі на освітньому порталі ITVDN.com для закріплення пройденого матеріалу.

Усі курси записані сертифікованими тренерами, які працюють у навчальному центрі CyberBionic Systematics.

# Перевірка знань

## TestProvider.com


TestProvider – це online-сервіс перевірки знань з інформаційних технологій. З його допомогою Ви можете оцінити Ваш рівень та виявити слабкі місця. Він буде корисним як у процесі вивчення технології, так і для загальної оцінки знань IT-спеціаліста.

Після кожного уроку проходьте тестування для перевірки знань на TestProvider.com

Успішне проходження фінального тестування дозволить Вам отримати відповідний Сертифікат.

# Q&A

# Дякую за увагу!