**Задание: Работа с регулярными выражениями в Python**

Описание:

Изучите и научитесь использовать регулярные выражения в Python для обработки строк. Ваша задача — написать несколько скриптов, которые выполняют различные текстовые операции с использованием регулярных выражений.

**Шаги выполнения задания:**

1. Установка и импорт модуля re:

* Убедитесь, что в вашем окружении установлен Python.
* Импортируйте модуль re, который предоставляет поддержку регулярных выражений в Python.

1. Задачи для выполнения:

* Написать скрипт для проверки, соответствует ли строка шаблону адреса электронной почты.
* Извлечь все числовые значения из строки.
* Проверить, содержит ли строка только цифры.
* Разбить длинную строку на части по пробелам и/или запятым.
* Написать скрип для проверки, соответствует ли строка шаблону номера телефона +7999-999-99-99

**Теоретические сведения**

С помощью regex можно искать как вполне конкретные выражения (например, слово «век» — последовательность букв «в», «е» и «к»), так и что-то более общее (например, любую букву или цифру).

Для обозначения второй категории существуют специальные символы. Вот некоторые из них:

| **Символ** | **Что означает** | **Пример использования шаблона** | **Пример вывода** |
| --- | --- | --- | --- |
| . | Любой символ, кроме новой строки (\n) | H.llo, .orld  20.. год | Hello, world;  Hallo, 2orld  2022 год,  2010 год |
| […] | Любой символ из указанных в скобках. Символы можно задавать как перечислением, так и указывая диапазон через дефис | [abc123]  [A-Z]  [A-Za-z0-9]  [А-ЯЁа-яё] | а; 1  B; T  A; s; 1  А; ё |
| [^…] | Любой символ, кроме указанных в скобках | [^A-Za-z] | з, 4 |
| ^ | Начало строки | ^Добрый день, | 0 |
| $ | Конец строки | До свидания!$ | 0 |
| | | Логическое ИЛИ. Регулярное выражение будет искать один из нескольких вариантов | [0-9]|[IVXLCDM] — регулярное выражение будет находить совпадение, если цифра является либо арабской, либо римской | 5; V |
| \ | Экранирование. Помогает регулярным выражениям ориентироваться, является ли следующий за \ символ обычным или специальным | \A\d\w\Z — экранирование превращает буквы алфавита в спецсимволы.  \[\.\] — экранирование превращает спецсимволы в обычные | 0 |

**Важное замечание 1.** Регулярные выражения зависимы от регистра, то есть «А» и «а» при поиске будут считаться разными символами.

**Важное замечание 2.** Буквы «Ё» и «ё» не входят в диапазон «А — Я» и «а — я». Так что, задавая русский алфавит, их нужно выписывать отдельно.

На экранировании остановимся подробнее. По умолчанию символы .^$\*+? {}[]\|() являются спецсимволами — то есть они выполняют определённые функции. Чтобы сделать спецсимволы обычными, их нужно экранировать \.

Таким образом, . будет обозначать любой символ, а \. — знак точки. Чтобы написать обратный слеш, его тоже нужно экранировать, то есть в регулярных выражениях он будет выглядеть так: \\.

Обратная ситуация с некоторыми алфавитными символами. По умолчанию они считаются просто буквами, но при экранировании начинают играть роль спецсимволов.

| **Символ** | **Что означает** |
| --- | --- |
| \d | Любая цифра. То же самое, что [0-9] |
| \D | Любой символ, кроме цифры. То же самое, что [^0-9] |
| \w | Любая буква, цифра и нижнее подчёркивание |
| \W | Любой символ, кроме буквы, цифры и нижнего подчёркивания |
| \s | Любой пробельный символ (пробел, новая строка, табуляция, возврат каретки и тому подобное) |
| \S | Любой символ, кроме пробельного |
| \A | Начало строки. То же самое, что ^ |
| \Z | Конец строки. То же самое, что $ |
| \b | Начало или конец слова |
| \B | Середина слова |
| \n, \t\, \r | Стандартные строковые обозначения: новая строка, табуляция, возврат каретки |

**Важное замечание.** \**A**, \Z, \b и \B указывают не на конкретный символ, а на положение других символов относительно друг друга. Можно сказать, что они указывают на пространство между символами.

Например, регулярное выражение \**b**[А-ЯЁаяё]\**b** будет искать только те буквы, которые отделены друг от друга пробелами или знаками препинания.

Часто при записи регулярного выражения какая-то часть шаблона должна повторяться определённое количество раз. Число вхождений в синтаксисе regex задают с помощью квантификаторов. Они всегда помещаются после той части шаблона, которую нужно повторить.

| **Символ** | **Что означает** | **Примеры шаблона** | **Примеры вывода** |
| --- | --- | --- | --- |
| {} | Указывает количество вхождений, можно задавать единичным числом или диапазоном | \d{4} — цифра, четыре подряд  \d{1,4} — цифра, от одного до четырёх раз подряд  \d{2,} — цифра, от двух раз подряд  \d{,4} — цифра, от 0 до 4 раз подряд | 1243, 1876   1, 12, 176, 1589    22, 456, 988888   5, 15, 987, 1234 |
| ? | От нуля до одного вхождения. То же самое, что {0,1} | \d? | 0 |
| \* | От нуля вхождений. То же самое, что {0,} | \d\* | 0 |
| + | От одного вхождения. То же самое, что {1,} | \d+ |  |

**Регулярные выражения в Python: модуль re и Match-объекты**

Чтобы работать с регулярными выражениями в Python, необходимо импортировать модуль **re**:

Import re

Это даёт доступ к нескольким функциям. Вот их краткое описание.

| **Функция** | **Что делает** | **Если находит совпадение** | **Если не находит совпадение** |
| --- | --- | --- | --- |
| re.match (pattern, string) | Ищет pattern в начале строки string | Возвращает Match-объект | Возвращает None |
| re.search (pattern, string) | Ищет pattern по всей строке string | Возвращает Match-объект с первым совпадением, остальные не находит | Возвращает None |
| re.finditer (pattern, string) | Ищет pattern по всей строке string | Возвращает итератор, содержащий Match-объекты для каждого найденного совпадения | Возвращает пустой итератор |
| re.findall (pattern, string) | Ищет pattern по всей строке string | Возвращает список со всеми найденными совпадениями | Возвращает None |
| re.split (pattern, string, [maxsplit=0]) | Разделяет строку string по подстрокам, соответствующим pattern | Возвращает список строк, на которые разделила исходную строку | Возвращает список строк, единственный элемент которого — неразделённая исходная строка |
| re.sub (pattern, repl, string) | Заменяет в строке string все pattern на repl | Возвращает строку в изменённом виде | Возвращает строку в исходном виде |
| re.compile (pattern) | Собирает регулярное выражение в объект для будущего использования в других re-функциях | Ничего не ищет, всегда возвращает Pattern-объект | 0 |

С дополнительным материалом можете ознакомиться по ссылке

<https://skillbox.ru/media/code/regulyarnye-vyrazheniya-v-python-sintaksis-poleznye-funktsii-i-zadachi/>

Тренажёр регулярных выражений https://regex101.com/