# Практическое задание №1. Осень 2022

# Введение

Регулярные выражения - мощный инструмент для обработки текстовых данных, включая тексты на естественных языках. Регулярные выражения используются в различных задачах, таких как предварительная обработка данных, системы интеллектуального анализа информации на основе правил, сопоставление с образцом, разработка текстовых функций, валидация входных данных, извлечение данных из интернета, и т. д.

# Постановка задачи

Требуется составить регулярные выражения, для решения следующих независимых подзадач:

- проверка корректности пароля;
- проверка корректности web цвета;
- токенизация математического выражения;
- проверка корректности даты;

## 1. Проверка корректности пароля

В рамках этой подзадачи требуется разработать регулярное выражение, которым возможно проверить, может ли являться входная строка (целиком) корректным паролем.

**Ограничения на пароли:**

- пароль должен содержать только латинские символы, цифры и специальные символы ^$%@#&*!?
- пароль должен состоять из не менее чем восьми символов
- пароль должен содержать по крайней мере один латинский символ в верхнем регистре
- пароль должен содержать по крайней мере один латинский символ в нижнем регистре
- пароль должен содержать по крайней мере одну цифру
- пароль должен содержать по крайней мере два различных специальных символа
- пароль не должен содержать двух одинаковых символов подряд

**Примеры корректных паролей:**

- rtG3FG!Tr^e
- aA1!*!1Aa
- oF^a1D@y5e6
- enroi#$rkdeR#$092uwedchf34tguv394h

**Примеры некорректных паролей:**

- пароль
- password
- qwerty
- lOngPa$$W0Rd

## 2. Проверка корректности web цвета

В рамках этой подзадачи требуется разработать регулярное выражение, которым возможно проверить, может ли являться входная строка (целиком) корректной записью цвета в одном из трёх web форматов:

- **rgb:** `rgb(r, g, b)`, где «r, g, b» - это комбинация из трёх целых чисел (от 0 до 255) или трёх процентных значений (от 0% до 100%), перечисленных через запятую.
- **hex** (шестнадцатеричный код цвета, `#rrggbb`) – это шестизначное представление цвета в RGB пространстве. Первые две цифры (rr) – представляют собой красное значение, следующие две – зелёное значение (gg), а последние – синее значение (bb). Перед значениями каналов предшествует символ #. Также допускается сокращённый вид записи – по одной цифре – #rgb
- **hsl** (тон, насыщенность и светлота, `hsl(h, s, l)`) - записывается похожим на rgb формат образом. Тон – целое число в диапазоне от 0 до 360, насыщенность и светлота - целочисленные процентные значения.

**Примеры корректных цветов:**

- #21f48D
- #888
- rgb(255, 255,255)
- rgb(10%, 20%, 0%)
- hsl(200,100%,50%)
- hsl(0, 0%, 0%)

**Примеры некорректных цветов:**

- #2345
- ffffff
- rgb(257, 50, 10)
- hsl(20, 10, 0.5)
- hsl(34%, 20%, 50%)

# 3. Токенизация математического выражения

Целью данной подзадачи является создание регулярного выражения, способного разбить строку, содержащую математическое выражение, на токены (элементарные части) и определить тип этих токенов. Математическое выражение может состоять из следующих элементов:

- **переменная (тип variable)** – строка из латинских букв, цифр и символа нижнего подчёркивания (_), начинающаяся не с цифры: a, var123, some_var_name;
- **число (тип number)** – строка, являющаяся целым или вещественным числом в общей форме без знака (в качестве разделителя целой и дробной части – точка): 42, 123456789, 23.567, 0.6734537
- **константа (тип constant)** – строка из списка: pi, e, sqrt2, ln2, ln10
- **функция (тип function)** – строка из списка: sin, cos, tg, ctg, tan, cot, sinh, cosh, th, cth, tanh, coth, ln, lg, log, exp, sqrt, cbrt, abs, sign
- **операция (тип operator)** – строка из списка ^, *, /, -, +
- **круглые скобки (тип `left_parenthesis` и `right_parenthesis`)**

Токены выражения могут отделяться друг от друга произвольным количеством пробелов (возможно, нулевым). Выделять пробельные символы в токены не нужно. Названия переменных не могут совпадать с именами функций и констант.

**Примеры выражений и ожидаемых токенов:**

- **выражение:** `"sin(x) + cos(y) * 2.5"`
  **токены:**
  ```
  {"type": "function", "span": [0, 3]},
  {"type": "left_parenthesis", "span": [3, 4]},
  {"type": "variable", "span": [4, 5]},
  {"type": "right_parenthesis", "span": [5, 6]},
  {"type": "operator", "span": [7, 8]},
  {"type": "function", "span": [9, 12]},
  {"type": "left_parenthesis", "span": [12, 13]},
  {"type": "variable", "span": [13, 14]},
  {"type": "right_parenthesis", "span": [14, 15]},
  {"type": "operator", "span": [16, 17]},
  {"type": "number", "span": [18, 21]}
  ```
- **выражение:** `"pi    +        usO5NlMvU"`
  **токены:**
  ```
  {"type": "constant", "span": [0, 2]},
  {"type": "operator", "span": [6, 7]},
  {"type": "variable", "span": [15, 24]}
  ```
- **выражение:** `"(         63393394.98 /8505      )"`
  **токены:**
  ```
  {"type": "left_parenthesis", "span": [0, 1]},
  {"type": "number", "span": [10, 21]},
  {"type": "operator", "span": [22, 23]},
  {"type": "number", "span": [23, 27]},
  {"type": "right_parenthesis", "span": [33, 34]}
  ```

Для токенизации выражения будет использоваться метод `finditer`. Для получения типа и границ токенов из каждого совпадения (match) будут извлекаться `match.lastgroup` и `match.span()` соответственно. Пример:

```
for match in regexp.finditer(string):
    print(f'type: {match.lastgroup}, span: {match.span()}')
```

# 4. Проверка корректности даты

Требуется разработать регулярное выражение, способное определить, является ли входная строка (целиком) датой в одном из нескольких форматов. Допускаются следующие форматы даты:

- **день.месяц.год** (14.09.2022, 5.02.1995, 01.4.2012)
- **день/месяц/год** (14/09/2022, 5/02/1995, 01/4/2012)
- **день-месяц-год** (14-09-2022, 5-02-1995, 01-4-2012)
- **год.месяц.день** (2022.09.14, 1995.02.5, 2012.4.01)
- **год/месяц/день** (2022/09/14, 1995/02/5, 2012/4/01)
- **год-месяц-день** (2022-09-14, 1995-02-5, 2012-4-01)
- **день месяц_rus год** (14 сентября 2022, 5 февраля 1995, 01 апреля 2012)
- **Месяц_eng день, год** (September 14, 2022, February 5, 1995, April 01, 2012)
- **Мес_eng день, год** (Sep 14, 2022, Feb 5, 1995, Apr 01, 2012)
- **год, Месяц_eng день** (2022, September 14, 1995, February 5, 2012, April 01)
- **год, Мес_eng день** (2022, Sep 14, 1995, Feb 5, 2012, Apr 01)

**Примеры корректных дат:**

- 20 января 1806
- 1924, July 25
- 26/09/1635
- 3.1.1506

**Примеры некорректных дат:**

- 25.08-1002
- декабря 19, 1838
- 8.20.1973
- Jun 7, -1563

**Примечание:** год должен быть неотрицательным.

# Решение задачи

## Теоретические аспекты

- docs.python - Документация на библиотеку регулярных выражений в Python3
- Habr - Регулярные выражения в Python. От простого к сложному:
- regex101 - Тестирование и отладка регулярных выражений с возможностью выбора языка программирования:
- towardsdatascience - Применение регулярных выражений для NLP:

## Тестирование

На личной странице (2022.tpc.ispras.ru/submissions/regexp) находится статистика со всеми результатами в т.ч. результатами последнего тестирования (дата, метрики качества).

На странице 2022.tpc.ispras.ru/results доступны результаты всех участников. Таблица обновляется раз в неделю.

## Загрузка решения

Загружаемый файл должен представлять собой zip архив с любым именем. Архив должен обязательно содержать:

- Решение в файле solution.py. В файле должны содержаться следующие строки, содержащие регулярные выражения:
- Регулярное выражение для проверки пароля на корректность (`PASSWORD_REGEXP`)
- Регулярное выражение для проверки цвета (`COLOR_REGEXP`)
- Регулярное выражение для токенизации выражения (`EXPRESSION_REGEXP`)
- Регулярное выражение для проверки дат (`DATES_REGEXP`)

Описание найденных регулярных выражений в файле description.txt. Пожалуйста, напишите подробное описание, как были найдены регулярные выражения. Это описание будет выложено вместе с решением после завершения курса.

Каждое регулярное выражение должно являться строкой, записанной по правилам python regexp. В противном случае система проверки выдаст ошибку.

**Пример решения, возвращающего пустые результаты для всех подзадач:**

```
PASSWORD_REGEXP = r''
COLOR_REGEXP = r''
EXPRESSION_REGEXP = r''
DATES_REGEXP = r''
```

## Ограничения

- Каждую неделю можно послать не более 10 решений.
- Внимание! Итоговое тестирование будет проводиться на последнем загруженном решении.
- Размер загружаемого архива не должен превышать 15Мб.
- Время тестирования каждого регулярного выражения не должно превышать 2 секунд на тексте из 1000 символов.
- На проверяющей машине доступно 16 Гб оперативной памяти.

## Оценка качества

Для оценки задания используется усредненная $F_1$ мера по каждой из подзадач. Для подзадач валидации используется $F_1$ мера для задачи бинарной классификации

$$P = \frac{tp}{tp + fp}, \qquad R = \frac{tp}{tp + fn}, \qquad F_1 = \frac{2PR}{P + R};$$

Для оценки остальных подзадач используется micro-averaged $F_{1^1}$ мера точного совпадения границ искомых подстрок:

$$P = \frac{|correct|}{|predicted|}, \qquad R = \frac{|correct|}{|expected|}, \qquad F_1 = \frac{2PR}{P + R};$$

При проверке результатов валидации строки, в случае превышения ограничения по времени, считается, что ответ противоположен правильному.