In [1]:

```
import re
```

In [ ]:

```
# найдем идентификаторы пользователей в тексте
msg = 'В розыгрыше победили: id1234563, id4653, id46!'
```

In [ ]:

```
re.findall(r'id\d+', msg)
```

In [ ]:

```
# ПРАКТИКА. Найдем хэштеги в твите
tweet = 'когда #эпидемия, то обязательно #оставайсядома'
```

In [ ]:

```

```

In [ ]:

```
# посмотрим внимательнее, как работают quantifiers
text = """Что такое регулярные выражения и как их использовать?
Говоря простым языком, регулярное выражение — это последовательность символов, используемая
их поддерживает множество языков общего назначения: Python, Perl, R.
Так что изучение регулярных выражений рано или поздно пригодится."""
```

In [ ]:

```
re.findall(r'\w+', text)
```

In [ ]:

```
re.findall(r'\w*', text)
```

In [ ]:

```
re.findall(r'\w?', text)
```

In [ ]:

```
# необходимо из строки с условными датами вытащить их
registration = 'Date of start: 4-12. Date of registration: 20-11'
```

In [ ]:

```
re.findall(r'\d{1,2}-\d{1,2}', registration)
```

In [ ]:

```
# вытащим номера телефонов и текста
phone_numbers = 'Мария: 8-943-342-23-32 Александр: 8-323-432-23-67'
```

In [ ]:

```
phone_pattern = r'\d-\d{3}-\d{3}-\d{2}-\d{2}'
re.findall(phone_pattern, phone_numbers)
```

In [ ]:

```
# ПРАКТИКА. Свалидируем дату
date = '1st september 2019 17:25'
```

In [ ]:

```

```

In [ ]:

```
# ПРАКТИКА. Посчитаем количество лайков и репостов по всем сообщениям
messages = ['Опять дождь! Лайков: 5, Репостов: 4', 'Крутой был концерт! Лайков: 28, Репосто
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```
# посчитаем количество восклицательных предложений в твите
tweet_2 = 'Какое замечательное место! Обязательно вернусть сюда снова. Всем советую!!!'
```

In [ ]:

```
re.split(r'!+', tweet_2)
# re.split(r'!+ ', tweet_2)
# len(re.split(r'!+\s', tweet_2))
```

In [ ]:

```

```

In [ ]:

```
# search ищет по всей строке, а match только в начале
text = """Что такое регулярные выражения и как их использовать?
Говоря простым языком, регулярное выражение — это последовательность символов, используемая
их поддерживает множество языков общего назначения: Python, Perl, R.
Так что изучение регулярных выражений рано или поздно пригодится."""

result = re.match(r'Python', text)
result
# print('Нашел' if result else 'Не нашел')
```

In [ ]:

```
re.match(r'Что', text)
```

In [ ]:

```
res = re.search(r'Python', text)
# res.start(), res.end()
# res.group()
```

In [ ]:

```

```

In [ ]:

```
# определим находится ли домен в зоне com
domain = 'www.site.com'
```

In [ ]:

```
re.findall('www.+com', domain)
```

In [ ]:

```
# поищем только в начале и конце строки
history_comment = '20 век был более опасным, чем 19 век'
```

In [ ]:

```
re.findall(r'\d{1,2}\s\век', history_comment)
```

In [ ]:

```
re.findall(r'^\d{1,2}\s\век', history_comment)
```

In [ ]:

```
re.findall(r'\d{1,2}\s\век$', history_comment)
```

In [ ]:

```
# экранирование. Разделим текст по точкам
text = """Что такое регулярные выражения и как их использовать?
Говоря простым языком, регулярное выражение — это последовательность символов, используемая
их поддерживает множество языков общего назначения: Python, Perl, R.
Так что изучение регулярных выражений рано или поздно пригодится."""
```

In [ ]:

```
re.split(r'\.\s', text)
```

In [ ]:

```
# ИЛИ
market_search = 'Что лучше: ноутбук или компьютер?'
re.findall('ноутбук|компьютер', market_search)
```

In [ ]:

```
# [] - найдем слово во всем склонениях
text = """Что такое регулярные выражения и как их использовать?
Говоря простым языком, регулярное выражение — это последовательность символов, используемая
их поддерживает множество языков общего назначения: Python, Perl, R.
Так что изучение регулярных выражений рано или поздно пригодится."""
pattern = 'регулярн[а-я]+ выражен[а-я]+'
re.findall(pattern, text)
```

In [ ]:

```
websites = 'www.101.com  www.google.com'
```

In [ ]:

```
re.findall(r'www[^0-9]+com', websites)
```

In [ ]:

```
# ПРАКТИКА. Разделим текст по запятой и по точкам
tweet_23 = 'Какое замечательное место! Обязательно вернусть сюда снова. Всем советую!!!'
```

In [ ]:

```

```

In [ ]:

```
# ПРАКТИКА. Напишите регулярку, которая отберет только валидный пароль по критериям:
# - содержит латинские символы в верхнем и нижнем регистре;
# - содержит числа;
# - содержит символы из *, #, $, %, !, &, .;
# - от 8 до 20 символов в длину;
passwords = ['Apple34!rose', 'My87hou#4$', 'abc123']
```

In [ ]:

```
regex_pass = r''
for password in passwords:
    if re.match(regex_pass, password):
        print('The password {} is a valid password'.format(password))
    else:
        print('The password {} is invalid'.format(password))
```

In [ ]:

```
# произведем замену на regex
re.sub(r'регулярн[а-я]+ выражен[а-я]+', 'regex', text)
```

In [ ]:

```

```

In [2]:

```
# скобочные группы
# выделим группы символов из посадочного талона, где:
# - первые 2 символа - это авиакомпания;
# - 4 следующие символа - номер рейса;
# - 3 следующих символа - аэропорт отправления;
# - 3 следующих - аэропорт прибытия;
# - оставшиеся - дата рейса
# Все буквы всегда в верхнем регистре
```

In [3]:

```
flight = 'Boarding pass: LA4214 AER-CDB 06NOV'
```

In [4]:

```
flight = 'Boarding pass: LA4214 AER-CDB 06NOV'
regex_flight = r'([A-Z]{2})(\d{4})\s([A-Z]{3})-([A-Z]{3})\s(\d{2}[A-Z]{3})'
```

In [5]:

```
flight_result = re.findall(regex_flight, flight)
```

In [6]:

```
print(flight_result)
```

```
[('LA', '4214', 'AER', 'CDB', '06NOV')]
```

In [ ]:

```
flight_match_result = re.search(regex_flight, flight)
print(flight_match_result.group(2))
```

In [ ]:

```
# квантификатор внутри группы и после нее  - это совершенно разные вещи!
numbers = 'Мои счастливые числа 777 и 5534'
re.findall(r'\d+', numbers)
```

In [ ]:

```
re.findall(r'(\d)+', numbers)
# так происходит, потому что эти цифры последовательно повторяются 1 и более раз
```

In [ ]:

```
# переведем даты к другому формату при помощи групп
date = '08/30/1991'
re.sub(r'(\d\d)/(\d\d)/(\d{4})', r'\2.\1.\3', date)
```

In [ ]:

```
# удалим повторяющиеся числа
numbers = '44444 6666 8888'
```

In [ ]:

```
re.sub(r'(\d)+', r'\1', numbers)
```

In [ ]:

```

```

In [ ]:

```
#non-captureing groups
# попробуем достать оценочные слова, которые стоят перед "фильм, кино"
review = 'Мне очень понравился этот фильм! Понравилось еще расположение кинотеатра, очень у
regex_negative = r'(понравился|понравилось).+(?:фильм|кино).+'
negative_matches = re.findall(regex_negative, review)
negative_matches
```

In [ ]:

```
# lookaround. Достанем имена только загруженных файлов
report = 'файл 11.txt загружен, файл 22.txt загружен, файл 33.txt ошибка'
find_downloaded = r'\d+\.txt(?=\sзагружен)'
files_error = r'\d+\.txt(?!\sзагружен)'
re.findall(find_downloaded, report)
re.findall(files_error, report)
```

In [ ]:

```
# ПРАКТИКА. Получить только рублевые цены
prices = 'RUB4.44, RUB10.88, EUR0.99, RUB99.99'
```