

An iceberg floating in a deep blue ocean under a clear blue sky. The visible tip of the iceberg is small and jagged, while the much larger, submerged part is visible below the water line. The text is overlaid on the submerged part of the iceberg.

# Регулярные выражения

Начало

# Найти все слова, начинающиеся на “при” и заканчивающиеся на “тся” или “ться”

```
1. def filter_words(words):
2.     results = []
3.     for word in words:
4.         word = word.lower()
5.         if word.startswith('при') and \
6.             (word.endswith('тся') or word.endswith('ться')):
7.             results.append(word)
8.     return results
```

# Найти все слова с приставкой “при” и суффиксами “тся” или “ться”

```
1. import re
2.
3. def filter_words_regex(words):
4.     results = []
5.     pattern = '^при.*ть?ся$'
6.     for word in words:
7.         if re.match(pattern, word, re.IGNORECASE):
8.             results.append(word)
9.     return results
```

# Проверить корректность email

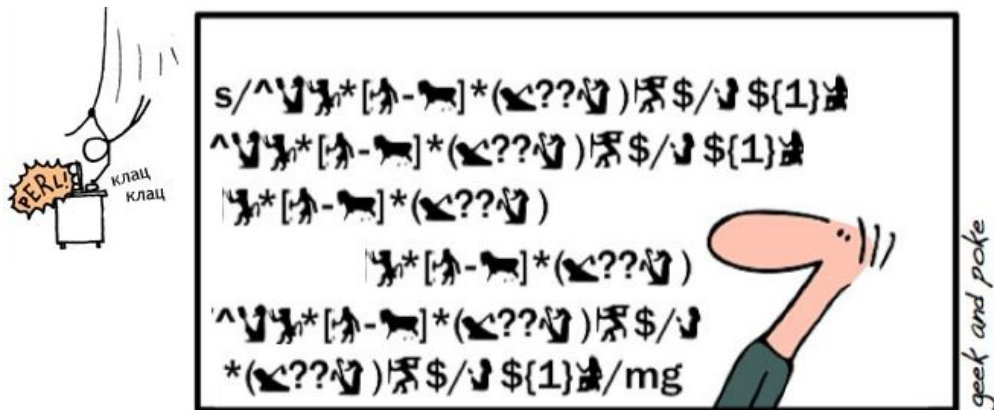
kiton1994@gmail.com

- Есть собака @
- Строка перед @ состоит из строчных и заглавных букв латинского алфавита, цифр, символов “\_”, “.”, “+”, “-” и не может быть пустой
- Строка после @ (домен) состоит из строчных и заглавных букв латинского алфавита, цифр, символа “-” и не может быть пустой
- После домена идет точка
- После точки идет строка, состоящая из строчных и заглавных букв латинского алфавита, цифр, символа “-”, которая не может быть пустой

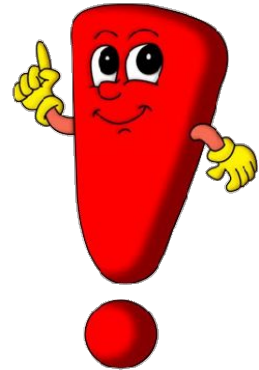
(^[a-zA-Z0-9\_+.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\$)

# Зачем нужны регулярные выражения?

- Сравнение с шаблоном
- Извлечение шаблонных фрагментов информации
- Замена
- Портируемость
- Быстро и лаконично



Если можно не пользоваться  
регулярками -  
не пользуйтесь ими!



# Как использовать регулярки

```
>>> import re
>>> text = 'Кот пришёл к коту и спросил кот: «Бойкот, котелок или скотч?»'
>>> pattern = 'кот.'
>>> result = re.search(pattern, text)
>>> print(result)
>>> print(result.group())
<_sre.SRE_Match object; span=(13, 17), match='коту'>
коту
```

# Как использовать регулярки

**`re.search(pattern, string)`** – возвращает первое вхождение подстроки, которая подходит под регулярное выражение

**`re.findall(pattern, string)`** – возвращает все вхождения подходящих строк

**`re.match(pattern, string)*`** – возвращает True, если строка совпадает с шаблоном и False, если не совпадает



## КИТАЙСКИЙ

是展现一种态



Пририсуйте к любому иероглифу крышу, и получится пагода.

# Язык регулярок

## КОРЕЙСКИЙ

원인으로



Много кружочков и овалов, можно дорисовать котиков.

# Точка “.”

Один любой символ

```
>>> text = 'кот! коты коту котами'
>>> pattern = 'кот.'
>>> result = re.search(pattern, text)
>>> print(result.group())
кот!
```

# Вопросительный знак “?”

Ноль или ровно одно вхождение **предыдущего** символа

```
>>> text = 'кот! коты коту котами'
```

```
>>> pattern = 'кот.?'
```

```
>>> result = re.findall(pattern, text)
```

```
>>> print(result)
```

```
['кот!', 'коты', 'коту', 'кота']
```

## Звездочка

Ноль или сколько угодно вхождений **предыдущего** символа ( $\geq 0$ )

```
>>> text = 'abb abba abbbbba aaaab'
```

```
>>> pattern = 'ab*a'
```

```
>>> result = re.findall(pattern, text)
```

```
>>> print(result)
```

```
['abba', 'abbbbba', 'aa', 'aa']
```

## Плюс “+”

Одно или сколько угодно вхождений **предыдущего** символа (> 0)

```
>>> text = 'aaaaa abba abaa'
```

```
>>> pattern = 'ab+a'
```

```
>>> result = re.findall(pattern, text)
```

```
>>> print(result)
```

```
['abba', 'aba']
```

# Крышка “^”

Начало строки. То есть строка должна начинаться с шаблона

```
>>> import re
>>> words = ['кот', 'мрот', 'скот', 'крот']
>>> pattern = '^к.+от'
>>> for word in words:
...     if re.match(pattern, word):
...         print(word)
```

крот

# Доллар "\$"

Конец строки. То есть строка должна оканчиваться шаблоном

```
>>> import re
>>> words = ['коты', 'мрот', 'от', 'скот', 'кота', 'живот']
>>> pattern = '.+от$'
>>> for word in words:
...     if re.match(pattern, word):
...         print(word)
мрот
скот
живот
```

## В квадратной скобке “[abc]”

Здесь должно быть что-то из символов в скобках: “a” или “b” или “c”

```
>>> import re
>>> text = 'кот мрот скот крот'
>>> pattern = '[кmp]от'
>>> result = re.findall(pattern, text)
>>> print(result)
['кот', 'рот', 'кот', 'рот']
```



# Диапазон в квадратных скобках “[a-я0-9]”

Внутри скобок можно указывать диапазоны:

- a-я - все строчные буквы русского алфавита
- A-Z - все заглавные буквы английского алфавита
- 0-9 - все цифры

Внутри квадратных скобок большинство специальных символов не действуют: . обозначает точку, ? — вопросительный знак. Вне квадратных скобок, чтобы получить точку или, например, плюс, специальные символы надо экранировать с помощью \ (\. обозначает точку, \+ обозначает плюс).

## Диапазон в квадратных скобках “[a-я0-9]”

```
>>> import re
>>> text = 'Я-родился-30февраля-1930года-в-15:45'
>>> pattern = '[А-Я0-9]+'
>>> result = re.findall(pattern, text)
>>> print(result)

['Я', '30', '1930', '15', '45']
```

# Крышка внутри квадратных скобок “[^abc]”

Все, кроме любого из символов в скобках

```
>>> import re
>>> text = 'кот мрот скот крот'
>>> pattern = '[^мр]от'
>>> result = re.findall(pattern, text)
>>> print(result)
```

```
['кот', 'кот']
```

# Помощник `"\d"`. Настоящее название - **символьный класс**

Любая цифра, аналогично [0-9]

```
>>> import re
>>> text = 'Я-родился-30февраля-1930года-в-15:45'
>>> pattern = '[^а-я]\d'
>>> result = re.findall(pattern, text)
>>> print(result)

['-3', '-1', '93', '-1', ':4']
```

# Помощник **"\D"**

Любой символ, кроме цифр (эквивалентно `[^0-9]`)

```
>>> import re
>>> text = 'Яродился30февраля1930года'
>>> pattern = '\D+'
>>> result = re.findall(pattern, text)
>>> print(result)
['Яродился', 'февраля', 'года']
```

# Помощник `"\w"`

Буквы, цифры, `"_"`

```
>>> import re
>>> text = '«Бойкот, котелок или скотч?» 18 февраля __main__'
>>> pattern = '\w+'
>>> result = re.findall(pattern, text)
>>> print(result)
```

```
['Бойкот', 'котелок', 'или', 'скотч', '18', 'февраля', '__main__']
```

# Помощник **"\W"**

Все, кроме букв, цифр и знака **"\_"**

```
>>> import re
>>> text = 'иван, петр, маша!надо.было.быть'
>>> pattern = '\W+'
>>> result = re.findall(pattern, text)
>>> print(result.count(','))
```

0

# Помощники “\s” и “\S”

Все пробелоподобные символы, аналогично [ \t\n\r]. Любой непобелоподобный символ

```
>>> import re
>>> text = "Кот пришёл к коту и спросил коты:
... «Бойкот, котелок или скотч?»"
>>> pattern = '\S+'
>>> result = re.findall(pattern, text)
>>> print(result)
```

```
['Кот', 'пришёл', 'к', 'коту', 'и', 'спросил', 'кота:', '«Бойкот,', 'котелок', 'или', 'скотч?»']
```





**KEEP  
CALM  
AND  
WRITE  
<CODE>**

[http://  
ejudge.sapunovnik.ru  
/tasks/14](http://ejudge.sapunovnik.ru/tasks/14)