Занятие 16

Регулярные выражения

Декораторы

Что напечатают эти операторы?

```
import re
st = 'but bot bit bite bottle rabbit bat batman'
print(re.findall(r"b.t", st))                          # 1
print(re.findall(r"b\w*t", st))                        # 2
print(re.findall(r"...", st))                          # 3
print(re.findall(r"..t", st))                          # 4
print(re.findall(r"\bb\b", st))                        # 5
print(re.findall(r"b.*", st))                          # 6
print(re.findall(r"\b\w*b[aioeu]t\w* \w*\b", st))      # 7
print(re.findall(r"\w*b[aioeu]t\w* \w*", st))          # 8
print(re.findall(r"\w*b[aioeu]t\w* \w* \w*", st))      # 9
print(re.findall(r"\w*b[aioeu]t\w* \w* \w* \w*", st))  # 10
```

Задача 15-1

Создайте рекурсивную функцию, которая получает как аргумент dct словарь, который может содержать словари, которые могут содержать словари и т.д. и как аргумент x значение ключа.

Например:

dct = {1:1, 2:2, 3:{2:22, 3:{1:111, 2:222, 3:{0:1111, 1:2222, 2:3333}}, 1:11,}, 6:22}

x = 1

Функция должна составить список, состоящий только из значений словаря с ключем x.

Например:

[1, 111, 2222, 11]

Задача 15-2

Напишите функцию, которая принимает строку символов, и печатает все содержащиеся в ней номера автомашин по следующему правилу:

LDDDLL78 или LDDDLL178,

где L – буквы, совпадающие по начертанию в русском и латинском алфавите, D – цифры от 0 до 9.

Например, A123BC78 или X666XX178

Задача 15-3

Напишите функцию, которая находит в строке все телефонные номера, которые удовлетворяют следующим шаблонам:

+7(812)DDD-DDDD, +7(812)DDD-DD-DD, +7(921)DDD-DDDD, +7(921)DDD-DD-DD

где D любая цифра

RE

Regular expressions

re – модуль Питона (import re)

- Который позволяет использовать все богатство регулярных выражений по поиску, замене, выборкам текстов
- Используется не только в Питоне
- Почему недостаточно большого количества функций по работе со строками: поиск, замена, вставка, удаление подстрок?
- Есть классы задач по обработке символьной информации:
  - Проверить текст на соответствие шаблону, например (адрес электронной почты), поиск по шаблону и др.

Регулярное выражение

- Регулярное выражение — это строка, задающая шаблон поиска подстрок в тексте.
- Одному шаблону может соответствовать много разных строк.
- Регулярное выражение состоит из обычных символов и специальных командных последовательностей.
- Например, \d задаёт любую цифру, а \d+ — задает любую последовательность из одной или более цифр., замены или разделения

Примеры

Регулярное выражение	Значение
ITMO	В точности “ITMO”
\d{5}	5 цифр подряд, например: ‘12345’ или ‘88888’
\d\d/\d\d/\d\d\d\d	Дата в формате ДД/ММ/ГГГГ, например 12/04/1961 или 35/78/9876
\b\w{3}\b	Слово точно из трех букв
[+-]?\d+	Целое число со знаком или без знака. Хотя бы одна цифра должна быть

Немного метасимволов

\d – любая цифра

\w – любая буква

\b – начало и конец слова

+ - один или больше элементов

* - ноль или больше элементов

? – ноль или один элемент

Что найдет следующее выражение?

```
import re
string = “0abracadabra1”
regex = r”.a.” # a если regex = r”\da\w”    regex = r”a\d”
re.findall(regex, string)
```

Примеры

Выражение	Примеры
c.t	cat, cut, c#t, c{t
c[aui]t	cat, cut, cit
c[a-c]t	cat, cbt, cct
c[^aui]t	cbt, cct, c2t, и др.
c[^a-c]t	cdt, cet, c%t

Знак ^ в квадратных скобках означает КРОМЕ

Давайте потренируемся.
string = “cat cet cit cot cut c#t c{t c2t”

Пробуйте разные regex из левого столбика на этой строке

Наиболее употребительные метасимволы

Use	To match any character
[set]	Один из этих символов
[^set]	Ни один из этих символов
[a-z]	Любой из символов от a до z
[^a-z]	Ни один из символов от a до z
.	Любой из символов кроме \n (новая строка)
\char	Экранирование спецсимвола

Некоторые полезные символы

Метасимвол	Диапазон	Описание
\d	[0-9]	любая цифра
\D	[^0-9]	любой нецифровой символ
\w	[0-9a-zA-Za-яA-ЯёЁ_]	любой алфавитно-цифровой символ и символ _
\W	[^0-9a-zA-Za-яA-ЯёЁ_]	любой символ, отличный от алфавитно-цифрового и _
\s	[\f\n\r\t\v]	любой пробельный символ
\S	[^ \f\n\r\t\v]	любой непробельный символ

Установление соответствия в случае интервала-диапазона

\d{1,3}	От 1 до 3 цифр
\w{2}	Ровно две буквы
\d{2,}	Две и больше цифры
\w{,5}	Не больше 5 букв

re.sub()

```
import re
text = 'Java самый популярный язык программирования в 2023 году.’
res = re.sub(r'Java', r'Python', text)
print(res)
```

re.subn()

```
import re
text = 'Java самый популярный язык Java самый популярный язык’
res, n = re.subn(r'Java', r'Python', text)
print(res, n)
```

Что напечатает?

```
import re
 text = 'fizz.123.buzz.456.fizzbuzz'
 res1 = re.sub(r'\d+', r'#', text)
 res2 = re.sub(r'[a-z]+', r'(*)', text)
 print(res1)
 print(res2)
```

Замена с помощью функции

```
import re
def fullname(x):
    s = x.group() # x специальный объект, который содержит значение, индекс начала
                  # и конца
    print(x.group(), x.start(), x.end())
    match s:
        case 'Коля': return 'Николай'    # Коля 12 16
        case 'Миша': return 'Михаил'   # Миша 26 30
    #if s == 'Коля' : return 'Николай'
    #elif s == 'Миша' : return 'Михаил'


text = 'Здравствуй, Коля! Привет, Миша!'
print(re.sub(r"\b\w{4}\b", fullname, text))
# Здравствуй, Николай! Привет, Михаил!
```

#Напишите программу, которая в тексте заменит LED на Пулково, MSQ на Минск, SVO на Шереметьево, SVX на Кольцово (добавьте любимые коды)

Использование скобок в шаблоне

```
# например, для замены местами слов
import re
text = 'first second'
print(re.sub(r"(first) (second)", r'\2 \1', text))
#second first
print(re.sub(r"(first) (second)", r'\1 \1 \2 \2', text))
```

Введите строку «Фамилия Имя Отчество», подставив свое имя.

Получите строку «Имя Отчество Фамилия»

Некоторые полезные функции

Функция	Её смысл
re.search(pattern, string)	Найти в строке string первую строчку, подходящую под шаблон pattern
re.fullmatch(pattern, string)	Проверить, подходит ли строка string под шаблон pattern
re.split(pattern, string, maxsplit=0)	Аналог str.split(), только разделение происходит по подстрокам, подходящим под шаблон pattern
re.findall(pattern, string)	Найти в строке string все непересекающиеся шаблоны pattern
re.finditer(pattern, string)	Итератор по всем непересекающимся шаблонам pattern в строке string (выдаются match-объекты);
re.sub(pattern, repl, string, count=0)	Заменить в строке string все непересекающиеся шаблоны pattern на repl

Ограничение числа замен, флаг re.I

```
import re

text = 'Программы на Java транслируются в байт-код java, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.'

res = re.sub(r'Java', r'Python', text, count=2, flags=re.I)

print(res)


# Программы на Python транслируются в байт-код Python, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.
```

Выбор части текста, используя скобки в шаблоне

```
import re
text = '123 first 234 second'
print(re.findall(r"\d+ (\w+)", text))     # →     ['first', 'second’]
#Если в шаблоне несколько скобок, то выдается список кортежей:
import re
text = 'Миша:123 Коля:234 Сеня:345'
res = re.findall(r"(\w+):(\d+)", text)
print(res)     #     →          [('Миша', '123'), ('Коля', '234'), ('Сеня', '345’)]
```

Напишите программу, которая из текста, содержащего сайты www.????.ru или .by или .com, выбирает только имя сайта, например, из www.itmo.ru выбирает только itmo

```
 re.findall(r"www.(\w+).(?:com|ru|by)", "www.itmo.ru www.epam.com")
```

re.finditer()

```
import re
text = 'abracadabra'
res = re.finditer(r"a", text)
for i in res:
    print(i.group(), i.start(), i.end()) # Что, начиная с какого индекса, заканчивая
```

А если так?

```
res = re.finditer(r"[^a]", text)
res = re.finditer(r"[ab]", text)
```

Задание

Давайте напишем программу, которая заменит все буквы ‘a’ в слове abracadabra на большие буквы ‘A’ + номер этой буквы ‘a’ в слове, т.е. на выходе должно получиться:

A1brA2cA3dA4brA5

Подсказка, используйте re.sub

Задание

Напишите функцию, которая из текста выбирает слова, в которых хотя бы одна буква повторяется два или более раз,

Т.е. “Питон” не подходит, “ванна” подходит, “силос” подходит, “кино” не подходит, “Java” подходит.

Подсказка, выберите все слова с помощью findall.

Потом используйте collections.Counter

re.split()

```
import re
text = "1     + 2222 * 3 -    7 / 2"
print(re.split(r"\s+", text))
# ['1', '+', '2222', '*', '3', '-', '7', '/', '2']
```

А как выбрать только числа?

Задание

Напишите функцию, которая разбивает текст по символам точка, запятая, точка с запятой, восклицательный и вопросительный знак.

re.compile()

Если шаблон часто используется, то его можно скомпилировать и запомнить под каким-то именем и использовать, не переписывая шаблон с риском ошибки

```
import re
numbers = re.compile(r'\d+’)

Способ 1.
print(re.findall(numbers, 'Я живу в доме 5 в квартире 7’))   #['5', '7']

Способ 2.
print(numbers.findall('Прочитайте абзац 8 на странице 356’)) #['8', '356']
```

Совместное использование строк r и f

```
import re
x = 5
re.findall(fr"{x}", '112233445566’) #    →    [‘5’, ‘5’]
# А так? re.findall(fr"{str(x)*2}", '112233445566')
# А что напечатают следующие строки???

 res = '|'.join(str(i) for i in range(x))          # чему равно res?
 re.findall(fr"{res}", '112233445566')
```

Выводы

1. RE – очень мощное средство
2. Необходимо очень осторожно использовать в реальных задачах
3. Очень много функциональных возможностей, которые необходимо тщательно изучить и проверить, как они реально работают.
4. Если предстоят задачи по анализу текста, то скорее всего следует освоить более плотно.
5. Если задачи не очень сложные, то обычно достаточно функций Python для работы со строками.
6. Используются во многих языках: Perl, Java, Javascript, Ruby, и др.

Декораторы

Декораторы

- Иногда нам нужно модифицировать существующую функцию, не меняя при этом ее исходный код.
- Например, оценить время выполнение функции или занести какую-то информацию о выполнении функции в файл или что-то другое

- **Декоратор — это функция, которая принимает другую функцию, расширяет ее поведение, не изменяя ее явно, и возвращает новую функцию.**

Для чего использовать декораторы?

- Ведение протокола операции (журналирование)
- Обеспечение контроля за доступом и аутентификацией
- Функции хронометража
- Ограничение частоты вызова API
- Кеширование и др.

Функции

Некоторые важные особенности функций:

- функции являются объектами первого класса, то есть функции можно передавать и использовать в качестве аргументов
- функции можно определять внутри других функций
- вложенные функции могут использовать локальные переменные родительских функций
- функции могут возвращать другие функции в качестве результата

Пример

```
def func(f):
        return f

def hello():
        print(‘Привет!!!’)
hello()
func(hello)()
```

Функции могут быть вложенными

Python допускает определение функций внутри других функций.

Такие функции называются вложенным функциями (nested function) или внутренними функциями (inner function)

```
def speak(text):
        def whisper(t):                    # определяем новую функцию
                return t.lower()           # переводит в нижний регистр
        res = whisper(text)                # вызываем новую функцию
        return res                         # возвращаем ее результат как результат speak
        #   return  whisper(text)          # а можно в одну строчку
print(speak('Hello, World'))
```

Всякий раз, когда вы вызываете функцию speak, она определяет новую функцию whisper и после этого ее вызывает

Вложенная функция

Внимание! Вложенная функция whisper не существует за пределами функции speak

```
def speak(text):
   def whisper(t):
         return t.lower() +  '...'
  return  whisper(text)
# Попытка вызвать функцию whisper
>>> whisper('Hello, World')
NameError: name 'whisper' is not defined
```

Вопрос! Как получить доступ к вложенной функции whisper за пределами функции speak?

Возврат функции в качестве значения

Функции являются объектами – и вы можете вернуть вложенную функцию в качестве значения.

```
def speak():
   def whisper(t):
        return t.lower() +  '...'
   return  whisper             # мы не вызываем ее, а возвращаем!!!

wr = speak()             # Получаем из функции speak объект функции whisper

print(wr('Hello, World’))          # Вызываем функцию с одним аргументом
```

Функции могут захватывать локальные состояния

Перепишем функцию speek следующим образом

```
def speak(text):
   def whisper():
        print(text.lower() +  '...')
   return  whisper

foo = speak('Hello World')
bar = speak('Wellcome')
foo()  → hello, world...
bar()  → wellcome…
```

Внутренние функции получают доступ к родительскому параметру text, определенному в родительской функции.

Такой доступ называется лексическим замыканием или для краткости замыканием.

Простейший декоратор

```
def null_decorator(func):
    return func   # Возвращает саму функцию


def hello():
    print('Hello world!')


hello = null_decorator(hello) # декорируем функцию hello
hello()
```

Пример декоратора

```
def sample_decorator(func): # определяем декоратор
        def wrapper():
                print('Начало функции’)
                func() # мы обертываем функцию func, не вмешиваясь  в ее работу
                print('Конец функции’)
        return wrapper
def say():
        print('Привет Мир!’)

say = sample_decorator(say)              # декорируем функцию
say()                                    # вызываем декорированную функцию

# При вызове функции sample_decorator(say) с переданной в качестве аргумента функцией say() возвращается вложенная функция wrapper() в качестве результата.
# Определите свою функцию и декорируйте ее
```

Декоратор, который переводит результат в верхний регистр

```
def uppercase_decorator(func):
        def wrapper():
                original_result = func()
                modified_result = original_result.upper()
                return modified_result
        return wrapper
# @uppercase_decorator
def h():
    return 'Hello’
# h = uppercase_decorator(h)
print(h())
```

Задания

Создайте декоратор, который переводит все слова к следующему виду, первая буква в верхнем регистре, а остальные в нижнем.

Создайте декоратор, который меняет порядок слов на противоположный.

Создайте декоратор, который сохраняет порядок слов, но порядок слов в каждом слове меняет на противоположный.

Задача 16-1

Напишите функцию, которая использует RE.

На вход подается строка, например, Институт точной механики оптики

Функция создает сокращение ИТМО (из первых букв слов, которые переведены в заглавные).

Есть много разных способов это сделать, например использовать re.sub, а потом re.split.

Задача 16-2

Вводится двухзначное число, например: 45.

Напишите такой шаблон, чтобы функция re.findall находила только те положительные целые числа, которые не больше, чем это введенное число.

Т.е. для 45 она находила бы 0, 1, 12, 45, но не 46, 100, 1000

Задача 16-3

Создайте декоратор, который переводит результат функций в нижний регистр.