# Библиотеки Python. Часть 4. Морфология

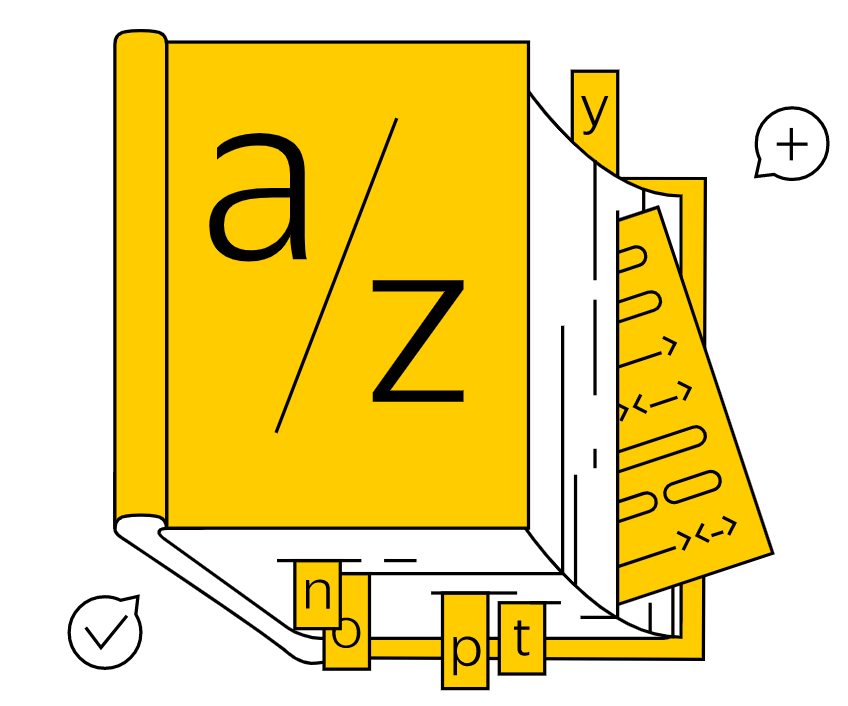
1. [Библиотека pymorphy2](https://lyceum.yandex.ru/courses/468/groups/3752/lessons/2503/materials/6263#1)
2. [Морфологический анализ](https://lyceum.yandex.ru/courses/468/groups/3752/lessons/2503/materials/6263#2)
3. [Работа с тегами](https://lyceum.yandex.ru/courses/468/groups/3752/lessons/2503/materials/6263#3)
4. [Постановка слов в начальную форму](https://lyceum.yandex.ru/courses/468/groups/3752/lessons/2503/materials/6263#4)
5. [Согласование с числительными](https://lyceum.yandex.ru/courses/468/groups/3752/lessons/2503/materials/6263#5)
6. [Итоги](https://lyceum.yandex.ru/courses/468/groups/3752/lessons/2503/materials/6263#6)

## Аннотация

*Мы уже говорили, что в Python имеется множество библиотек, реализующих абстракции для различных предметных областей. Поэтому мы занимаемся в основном верхним «этажом» логики. Сегодня речь пойдет о библиотеке, позволяющей работать с текстами.*

## Библиотека pymorphy2

Вы уже знаете стандартные строковые функции и пользовались ими. Мы умеем работать со строками посимвольно и знаем, как представляется текстовая информация. Давайте перейдем на уровень выше. Строка и текст в общем случае состоят не из набора букв, а из слов, и иногда нужно работать именно со словами, а не просто с последовательностями байтов.



Возьмем для примера склонение существительных с числительными. Например, на форуме надо написать, что в теме «21 комментарий», но «24 комментария». То же самое нужно делать и для других слов: например, «новость», «пользователь». Иногда на сайтах обходят эту проблему и вставляют машинное «комментариев: 21».

Давайте посмотрим, какие средства работы со словами есть в Python, и познакомимся с библиотекой pymorphy2 (морфология).

Если эта библиотека отсутствует в вашем Python, ее надо установить с помощью утилиты pip.

pip install pymorphy2

Словари распространяются отдельными пакетами. Для русского языка используется pymorphy2-dicts-ru.

Они обновляются время от времени, для обновления используйте команду

pip install -U pymorphy2-dicts-ru

## Морфологический анализ

В pymorphy2 для морфологического анализа слов есть класс MorphAnalyzer.

**Морфологический анализ**

Морфологический анализ — определение характеристик слова на основе того, как оно пишется. При морфологическом анализе не учитываются соседние слова.

Для любого слова библиотека делает несколько предположений, что оно может означать, и обозначает свою уверенность в этом предположении.

Метод MorphAnalyzer.parse() возвращает один или несколько объектов типа Parse с информацией о том, как слово может быть разобрано.

import pymorphy2

morph = pymorphy2.MorphAnalyzer()

morph.parse('Ваня')

[Parse(word='ваня', tag=OpencorporaTag('NOUN,anim,masc,Name sing,nomn'),

normal\_form='ваня', score=1.0,

methods\_stack=((<DictionaryAnalyzer>, 'ваня', 407, 0),))]

В данном случае предположение одно с уверенностью score=1.0. Итак, мы имеем дело с существительным NOUN, именем собственным, одушевленным, мужского рода.

Конечно, предположений может быть несколько:

import pymorphy2

import pprint

morph = pymorphy2.MorphAnalyzer()

res = morph.parse('пила')

print(len(res))

pprint.pprint(res)

4

[Parse(word='пила', tag=OpencorporaTag('NOUN,inan,femn sing,nomn'),

normal\_form='пила', score=0.428571,

methods\_stack=((<DictionaryAnalyzer>, 'пила', 55, 0),)),

Parse(word='пила', tag=OpencorporaTag('VERB,impf,tran femn,sing,past,indc'),

normal\_form='пить', score=0.285714,

methods\_stack=((<DictionaryAnalyzer>, 'пила', 444, 8),)),

Parse(word='пила', tag=OpencorporaTag('NOUN,anim,masc,Name sing,gent'),

normal\_form='пил', score=0.142857,

methods\_stack=((<DictionaryAnalyzer>, 'пила', 1124, 1),)),

Parse(word='пила', tag=OpencorporaTag('NOUN,anim,masc,Name sing,accs'),

normal\_form='пил', score=0.142857,

methods\_stack=((<DictionaryAnalyzer>, 'пила', 1124, 3),))]

У каждого разбора есть тег:

p = morph.parse('пила')[1]

print(p)

Parse(word='пила', tag=OpencorporaTag('VERB,impf,tran femn,sing,past,indc'),

normal\_form='пить', score=0.285714,

methods\_stack=((<DictionaryAnalyzer>, 'пила', 444, 8),))

**Тег**

Тег — набор граммем (грамматических признаков), характеризующих данное слово.

Например, тег 'VERB,impf, tran femn, sing,past,indc' означает, что слово — глагол (VERB) несовершенного вида (impf), переходный (tran), женского рода (femn), единственного числа (sing), прошедшего времени (past), изъявительного наклонения (indc).

Полный список граммем (грамматических единиц) можно посмотреть [тут](http://opencorpora.org/dict.php?act=gram) и [тут](https://pymorphy2.readthedocs.io/en/latest/user/grammemes.html#grammeme-docs).

## Работа с тегами

Чтобы проверить, есть ли в данном теге отдельная граммема (или все граммемы из указанного множества), используйте оператор in:

'NOUN' in p.tag # => False

Пример:

import pymorphy2

morph = pymorphy2.MorphAnalyzer()

res = morph.parse('питона')[1]

# В нулевом предположении будет родительный падеж - он чаще встречается

# проверка на глагол

print("VERB" in res.tag)

# проверка на винительный падеж

print("accs" in res.tag)

# cуществительное в винительном падеже?

print({"NOUN", 'accs'} in res.tag)

False

True

True

Кроме того, у каждого тега есть атрибуты, через которые можно получить часть речи, число и другие характеристики:

**Например, для глагола бывают теги:**

p.tag.POS # часть речи

p.tag.animacy # одушевленность

p.tag.aspect # вид: совершенный или несовершенный

p.tag.gender # род (мужской, женский, средний)

p.tag.involvement # включенность говорящего в действие

p.tag.mood # наклонение (повелительное, изъявительное)

p.tag.number # число (единственное, множественное)

p.tag.person # лицо (1, 2, 3)

p.tag.tense # время (настоящее, прошедшее, будущее)

p.tag.transitivity # переходность (переходный, непереходный)

p.tag.voice # залог (действительный, страдательный)

Если мы попытаемся получить доступ к граммеме, которой нет для слова данной части речи, получим None.

import pymorphy2

morph = pymorphy2.MorphAnalyzer()

res = morph.parse('писал')[0]

print(res.tag.mood)

print(res.tag.tense)

# попытка доступа к граммеме, которой нет для этой части речи

print(res.tag.case)

indc

past

None

Итак, мы можем разбирать большие тексты на части и узнавать информацию о словах: например, искать глаголы, подсчитывать имена и т. д.

Для изменения слов (к примеру, склонения существительных) можно использовать метод inflect:

word = morph.parse('случай')[0]

print(word.inflect({'gent'}))

Parse(word='случая', tag=OpencorporaTag('NOUN,inan,masc sing,gent'),

normal\_form='случай', score=1.0,

methods\_stack=((<DictionaryAnalyzer>, 'случая', 175, 1),))

И во множественном числе:

print(word.inflect({'gent', 'plur'}))

Parse(word='случаев', tag=OpencorporaTag('NOUN,inan,masc plur,gent'),

normal\_form='случай', score=1.0,

methods\_stack=((<DictionaryAnalyzer>, 'случаев', 175, 7),))

Метод inflect работает и с другими частями речи — например, глаголами:

word = morph.parse('программировать')[0]

print(word.inflect({'VERB', 'impf', 'masc', 'sing', 'past', 'indc'}))

Parse(word='программировал', tag=OpencorporaTag('VERB,impf,tran masc,sing,past,indc'),

normal\_form='программировать', score=1.0,

methods\_stack=((<DictionaryAnalyzer>, 'программировал', 168, 7),))

Вернемся к параметру score.

morph.parse('пила')

# => [Parse(word='пила', tag=OpencorporaTag('NOUN,inan,femn sing,nomn'),

# normal\_form='пила', score=0.428571,

# methods\_stack=((<DictionaryAnalyzer>, 'пила', 55, 0),)),

# => Parse(word='пила', tag=OpencorporaTag('VERB,impf,tran femn,sing,past,indc'),

# normal\_form='пить', score=0.285714,

# methods\_stack=((<DictionaryAnalyzer>, 'пила', 444, 8),)),

# => Parse(word='пила', tag=OpencorporaTag('NOUN,anim,masc,Name sing,gent'),

# normal\_form='пил', score=0.142857,

# methods\_stack=((<DictionaryAnalyzer>, 'пила', 1124, 1),)),

# => Parse(word='пила', tag=OpencorporaTag('NOUN,anim,masc,Name sing,accs'),

# normal\_form='пил', score=0.142857,

# methods\_stack=((<DictionaryAnalyzer>, 'пила', 1124, 3),))]

Мы видим, что предложенные четыре варианта разбора имеют параметр score, который говорит нам о том, какой вариант предпочтительнее. Pymorphy2 использует статистические методы и ориентируется на данные проекта [OpenCorpora](http://opencorpora.org/" \t "_blank), чтобы вычислить значение параметра score. Мы не будем останавливаться на этом подробно. Интересующиеся могут прочитать о внутренней кухне на странице документации по Pymorphy2. Скажем только, что эти вычисления не всегда точны.

Разборы сортируются по убыванию score, поэтому почти везде в примерах берется первый вариант разбора из возможных (например, morph.parse('пила')[0]).

## Постановка слов в начальную форму

Нормальную (начальную) форму слова можно получить через атрибуты Parse.normal\_form и Parse.normalized. Например, для глаголов в нем будет храниться инфинитив. Таким образом, можно привести любую форму глагола к единому виду.

Но что считается за нормальную форму у других слов? Например, возьмем слово «изучающим». Иногда мы захотим нормализовать его в «изучать», иногда — в «изучающий», иногда — в «изучающая».

Посмотрим на примере, что сделает pymorphy2:

morph.parse('изучающим')[0].normal\_form

# => 'изучать'

Pymorphy2 сейчас использует алгоритм нахождения нормальной формы, который работает наиболее быстро (берется первая форма в лексеме) — именно поэтому, например, все причастия сейчас нормализуются в инфинитивы. Это можно считать деталью реализации.

Если требуется нормализовать слова иначе, можно воспользоваться методом Parse.inflect():

morph.parse('изучающим')[0].inflect({'sing', 'nomn'}).word

# => 'изучающий'

## Согласование с числительными

Помимо разбора слов, библиотека может их изменять — например, сопоставлять с числами. Давайте посмотрим, как можно решить задачу с подсчетом и выводом количества комментариев на форуме:

comment = morph.parse('комментарий')[0]

comment.make\_agree\_with\_number(1).word # => 'комментарий'

comment.make\_agree\_with\_number(2).word # => 'комментария'

comment.make\_agree\_with\_number(7).word # => 'комментариев'

**Важно!**

Интересно, что библиотека пытается работать даже со словами, которых не знает, обращаясь к оракулу:

word = morph.parse('Мегатрон')[0]

word.make\_agree\_with\_number(7).word # => 'мегатронов'

word.make\_agree\_with\_number(2).word # => 'мегатрона'

## Итоги

Мы выяснили, что работа со словами в Python довольно проста. Сторонние библиотеки позволяют упростить работу с морфологией языков.

Мы не рассматривали средства и библиотеки для извлечения знаний и фактов из текстов на естественных языках. Например, по тексту новости (цитируется портал lenta.ru):

Россиянка Дарья Виролайнен досрочно стала обладательницей Большого Хрустального глобуса, вручаемого победительнице общего зачета Кубка Международного союза биатлонистов (IBU). Об этом сообщается на *[OpenCorpora](http://www.biathlonworld.com/news/detail/alexander-loginov-wins-ibu-cup-pursuit" \t "_blank)* IBU.

На всех этапах Виролайнен набрала 684 очка, ее соотечественница Анна Никулина, идущая второй, — 526 баллов. На последнем этапе Кубка IBU в Эстонии россиянка не выступит: она вызвана в основной состав сборной России на этап Кубка мира. Тем не менее, Никулина не сумеет догнать ее.

Виролайнен стала обладательницей и Малого Хрустального глобуса в зачете гонок преследования.

На чемпионате мира по биатлону, прошедшем с 8 по 19 февраля в австрийском Хохфильцене, спортсменка была в составе сборной России, однако не провела ни одной гонки.

Можно узнать, что речь идет о конкретном человеке, странах и датах, что тут упоминается вид спорта и т. д. Со временем вы усвоите и эти возможности библиотек Python.