

2023年7月5日

Mikawa Lab. B3ゼミ

ゼロから学ぶ
Pythonプログラミング

第5章 文字列

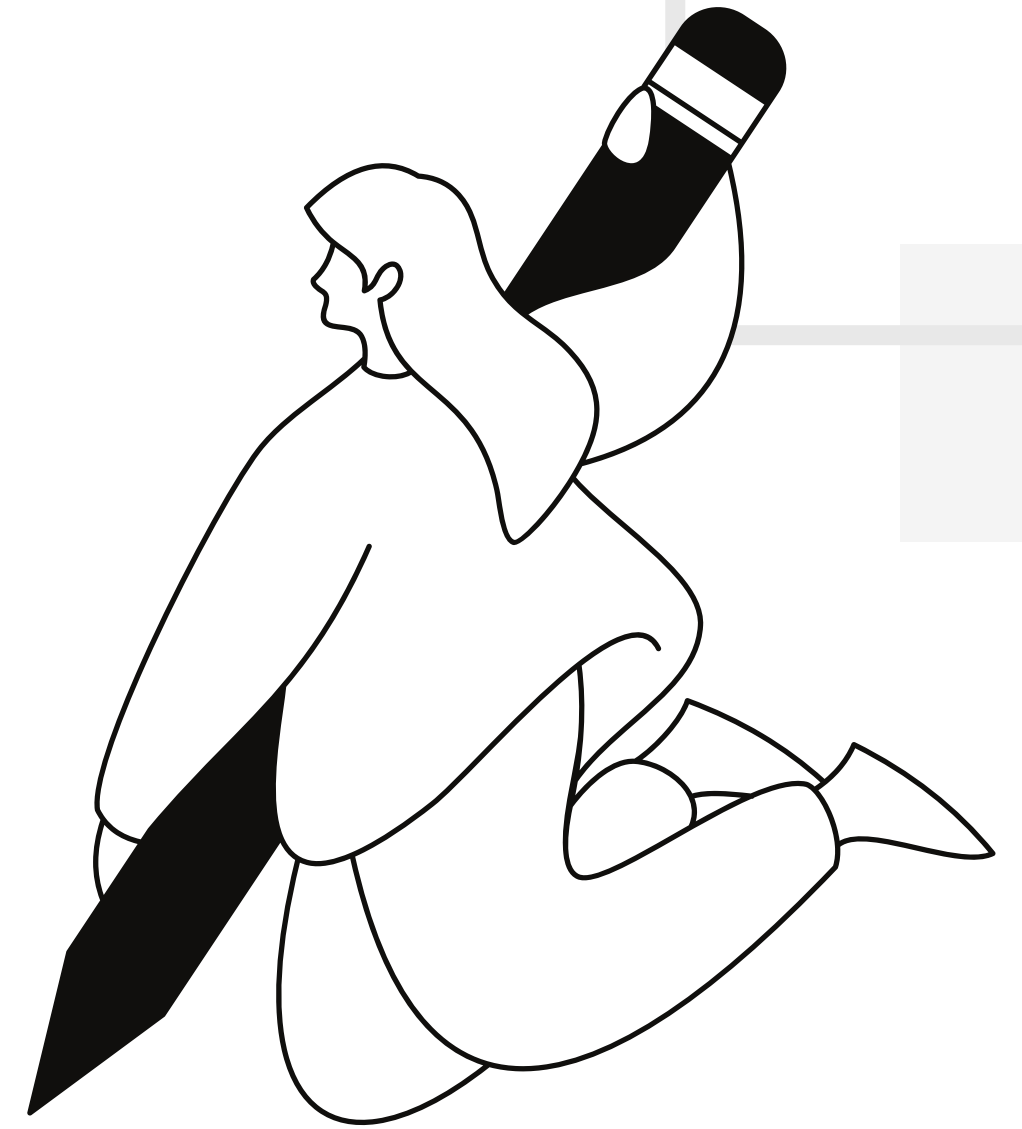
担当：村井希優

01. 文字列と文字コード

02. 辞書

03. 正規表現

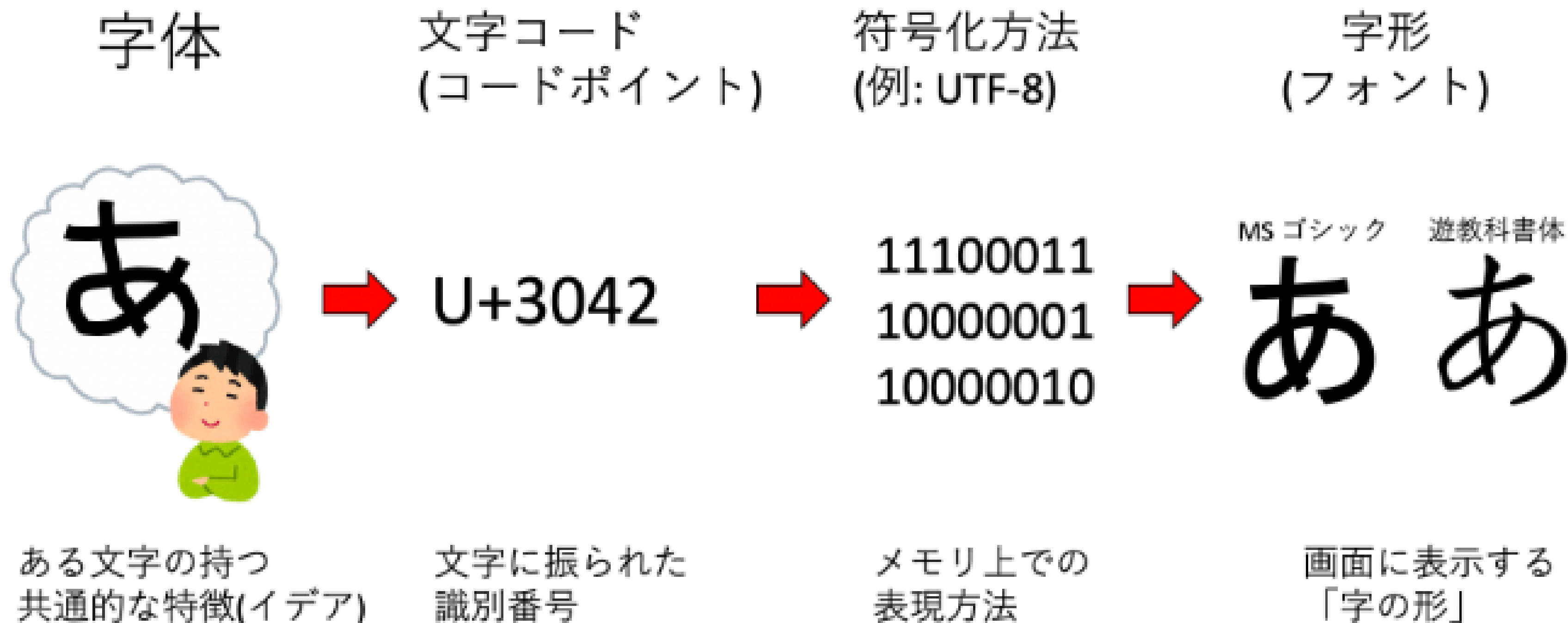
04. ワードクラウド



01

文 字 列 と 文 字 コ ー ド

文字とは何か



「表1」という文字列を.....

シフトJISで表す

95:10010101 } 表
5c:01011100 }
31:00110001 } 1

2バイト目にASCIIが
表れる

5c:01011100 } ¥

EUC-JPで表す

c9:11001001 } 表
bd:10111101 }
31:00110001 } 1

2バイト目にASCIIが
現れない
(最上位ビットが1になる)

JISコードで表す

1b:00011011 }
24:00100100 } 漢字スタート
42:01000010 }
49:01001001 } 表
3d:00111101 }
1b:00011011 }
28:00101000 } ASCIIスタート
42:01000010 }
31:00110001 } 1

最上位ビットが必ず0

UTF-8における3バイト表現

連続する1がバイト数表現

1110AAAA

10BBBBCC

10CCDDDD

データ部分

2バイト目以降の予約ビット

「あ」 U+3042

11100011 E3

10000001 81

10000010 82

「あ」 U+3042のビット表現

3 0011

0 0000

4 0100

2 0010

02

辞書

辞書型の使い方

辞書: キーと値を結び付けるデータ構造

{ **キー**1: **値**1, **キー**2: **値**2, **キー**3: **値**3 }

d["**キー**"] = 要素


```
[9] d = {"Apple": 158, "Banana": 198, "Orange": 100}
```

d["Apple"] = 158



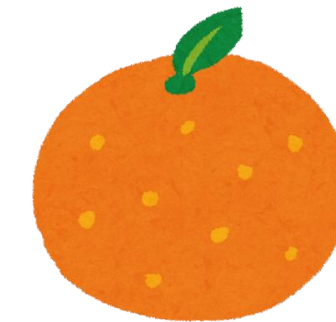
158

d["Banana"] = 198



198

d["orange"] = 100



100

```
[1] d = {}
```

辞書型は中括弧で初期化

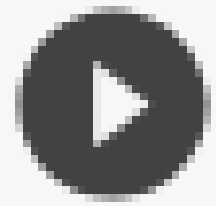
```
[4] d["Apple"] = 158
```

「文字列」と「整数」を結び付け

```
▶ print(d["Apple"])
```

Print(d["キー"])
でデータを取得

```
☞ 158
```



```
d[1] = "one"  
d[(1, 2)] = (2, 4)
```

- ✳ キーや要素はどんなものでも指定できる
- ✳ 同じ辞書に異なる種類のキーや要素が共存できる

```
[3] for k in d:  
      print(k)
```

```
Apple  
Banana  
Orange  
1  
(1, 2)
```

✿ print(k)でキーを取得

```
[4] for k in d:  
      print(d[k])
```

158

198

100

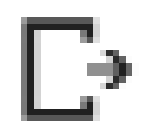
one

(2, 4)

✿ print(d[k])で要素を取得



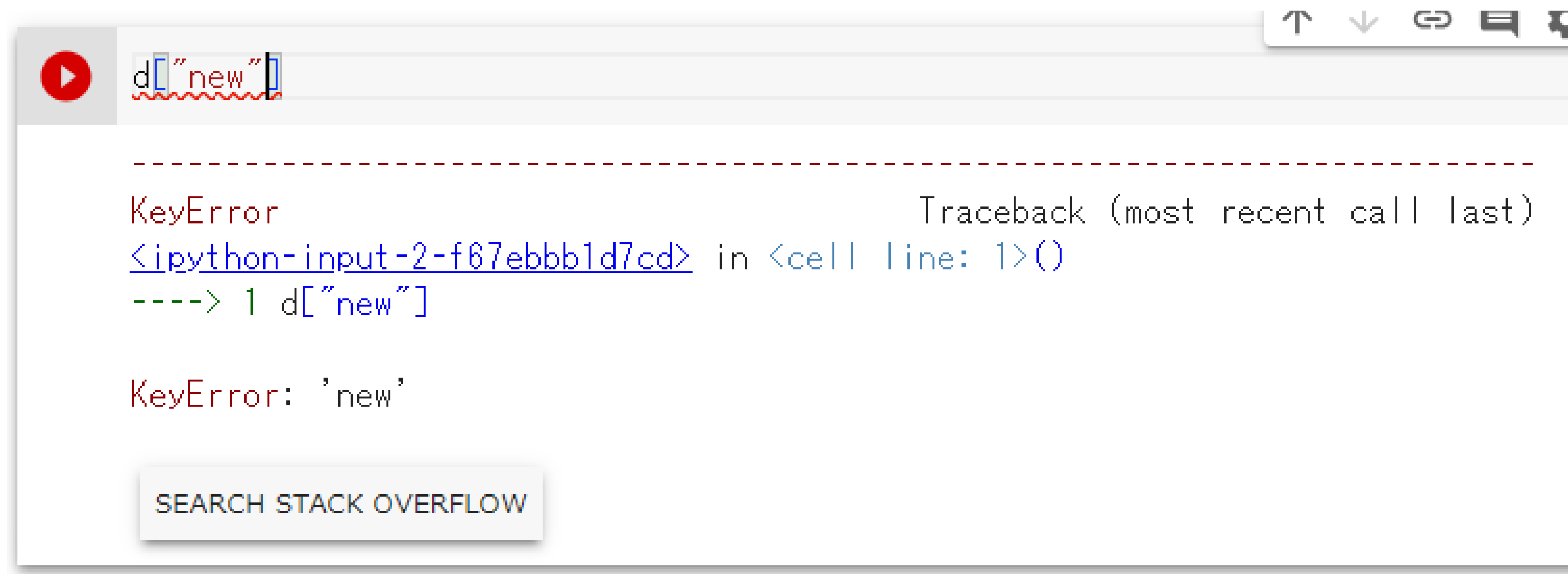
```
for k,v in d.items():  
    print(k,v)
```



```
Apple 158  
Banana 198  
Orange 100  
1 one  
(1, 2) (2, 4)
```

✿ print(k,v)で要素を取得

存在しないキーを入力すると...



The screenshot shows a Jupyter Notebook interface. At the top, there is a toolbar with icons for undo, redo, run, and other actions. Below the toolbar is a code editor with a red squiggly line under the text `d["new"]`, indicating a syntax or runtime error. The output area below the code editor displays a traceback for a `KeyError` exception. The traceback shows the error occurred in a cell at line 1, column 1, with the code `d["new"]`. The error message is `KeyError: 'new'`. At the bottom of the output area, there is a button labeled "SEARCH STACK OVERFLOW".

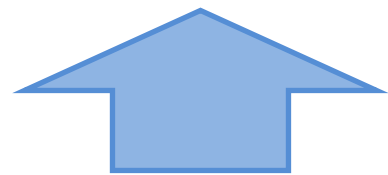
```
-----  
KeyError                                Traceback (most recent call last)  
<ipython-input-2-f67ebbb1d7cd> in <cell line: 1>()  
----> 1 d["new"]  
  
KeyError: 'new'
```

SEARCH STACK OVERFLOW

「存在しないキー」を指定したときに、デフォの値があると便利

```
[4] from collections import defaultdict
```

```
[5] d = defaultdict(int)
```



デフォの値として「0」を持つような辞書を作る

文字や単語の出現頻度のカウント



```
s = "すももももももものうち"  
for c in list(s):  
    d[c] += 1  
for k, v in d.items():  
    print(k, v)
```

```
す 1  
も 8  
の 1  
う 1  
ち 1
```

03

正 規 表 現

正規表現とは？

文字列の集まりを一つの形式
で表すための特別な書き方

text = ‘ たった一《ひと》つの真実 《しんじつ》 見抜《みぬ》く
見《み》た目《め》は子供《こども》、頭脳《ずのう》は大人
《おとな》、その名《な》は、名探偵《めいたんてい》コナン。 ’

出典: 青山剛昌『名探偵コナン』

text = ' たった一

見《み》た目

《おとな》、その

《》内のルビを
消したい

“見抜《みぬ》く

《ずのう》は大人

《ってい》コナン。’

正規表現

```
[10] text = 'たった一《ひと》つの真実《しんめい》は子供《め》は子供'
```

```
▶ in_bracket = False
for s in list(text):
    if in_bracket:
        if s == '》':
            in_bracket = False
            continue
    if s == '《':
        in_bracket = True
        continue
    print(s, end='')
print()
```

- ・文字を一文字ずつ処理
- ・《》に囲まれた状態なのかを判断
 - 囲まれた状態ならスキップ
 - 囲まれてなければ表示

たった一つの真実見抜く見た目は子供、頭脳は大人、その名は、名探偵コナン。



これを正規表現
で実行すると...

- . どんな文字にもマッチ
- * 直前のパターンが0回以上の繰り返し
- ? 直前のパターンが0回か1回の繰り返し
- + 直前のパターンが1回以上の繰り返し

. *bc

bc

xxxbc

xxxbcyyy

. ?bc

bc

xxxbc

xxxbcyyy

. +bc

bc

xxxbc

xxxbcyyy

正規表現

```
[14] import re  
      print(re.sub(r' ', '', text))
```

たった一つの真実見抜く見た目は子供、頭脳は大人、その名は、名探偵コナン。

コマンドの意味

re.sub (r' 正規表現 , 置き換える文字列 , 元の文字列)

↑
今回は空白にしたいから
“ ” だけ

- . どんな文字にもマッチ
- * 直前のパターンが0回以上の繰り返し
- ? 直前のパターンが0回か1回の繰り返し
- + 直前のパターンが1回以上の繰り返し

正規表現

このようなものを正規表現という

```
[14] import re  
      print(re.sub(r'《.*?》', '', text))
```

たった一つの真実見抜く見た目は子供、頭脳は大人、その名は、名探偵コナン。

《 . * ? 》

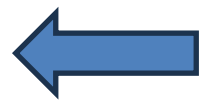
<<で始まり何か文字が0以上あるよう
な文字列で最短マッチで>>で終わる

```
[22] text = 'hanamogera'
```



```
m = re.search(r'moge', text)  
m.span()
```

(4, 8)



4以上8未満

```
[25] s, e = m.span()  
text[s:e]
```

'moge'

0123456789
hanamogera

正規表現

```
[22] text = 'hanamogera'
```



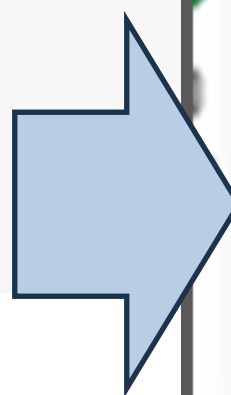
```
m = re.search(r'moge', text)  
m.span()
```

```
(4, 8)
```

同じこと!!

```
[25] s, e = m.span()  
text[s:e]
```

```
'moge'
```

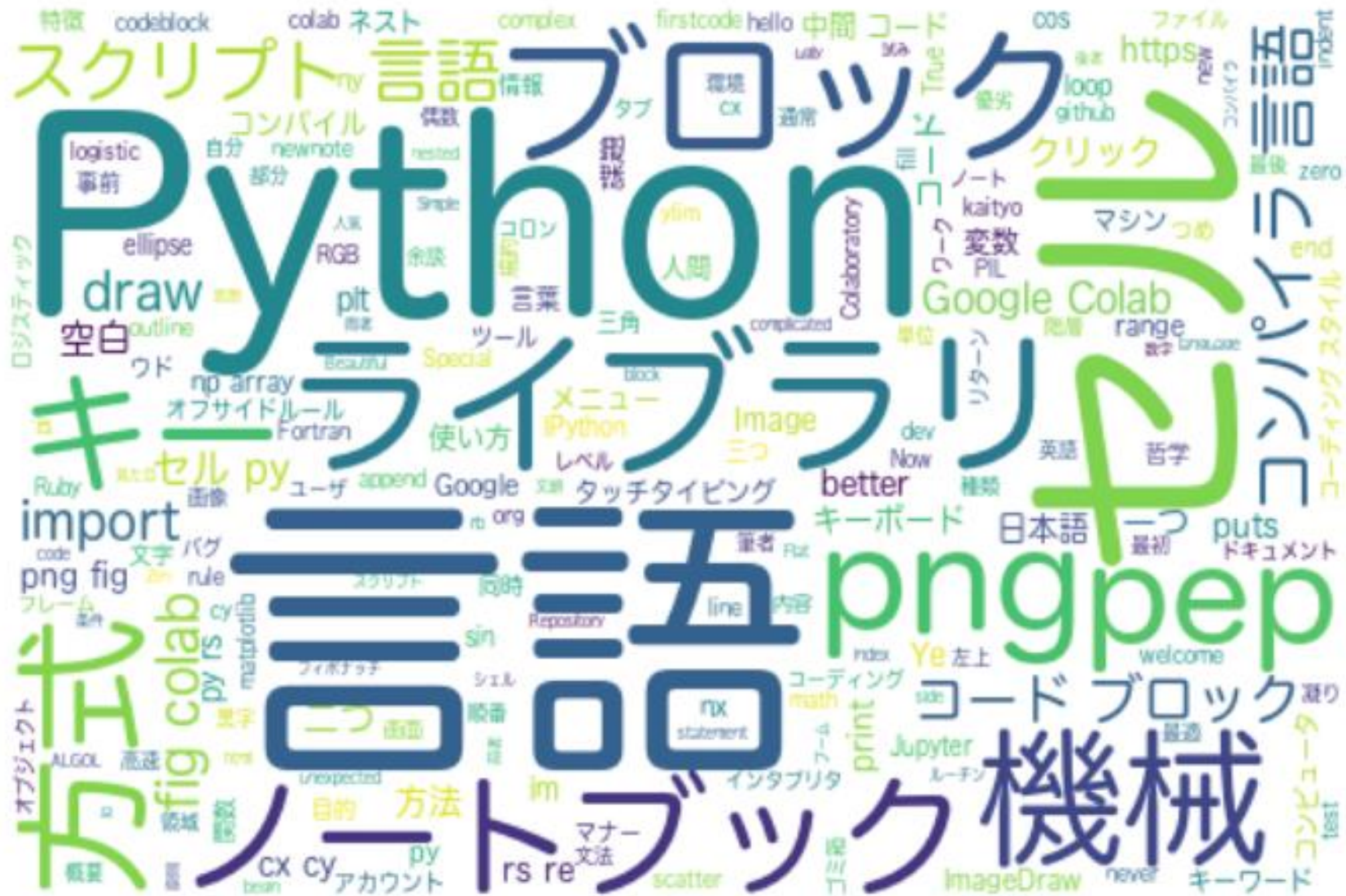


```
m.group()
```

```
'moge'
```

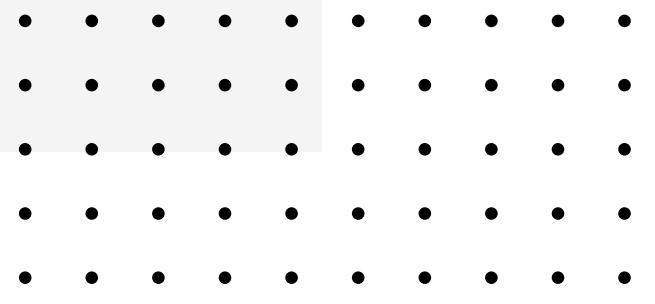
04

ワードクラウド

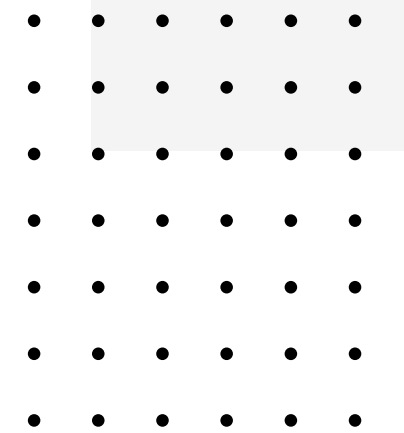


ワードクラウドの例

ワードクラウド



演習



課題1

形態素解析

与えられた文章を意味を持つ言葉の
最小単位(形態素)に分解すること

形態素解析ライブラリ「MeCab」をインストールし、形態素解析を行う。

```
Import MeCab  
m = MeCab.Tagger()  
print(m.parse(" 品詞分解したい文 "))
```

「すももももももものうち」を品詞分解してみる

形態素解析成功



```
m = MeCab.Tagger()
print(m.parse("すももももももものうち"))
```

すもも	名詞,一般,*,*,*,*,すもも,スモモ,スモモ
も	助詞,係助詞,*,*,*,*,も,モ,モ
もも	名詞,一般,*,*,*,*,もも,モモ,モモ
も	助詞,係助詞,*,*,*,*,も,モ,モ
もも	名詞,一般,*,*,*,*,もも,モモ,モモ
の	助詞,連体化,*,*,*,*,の,ノ,ノ
うち	名詞,非自立,副詞可能,*,*,*,うち,ウチ,ウチ
EOS	

課題2: タイトルを出力

必要なライブラリをインポート



Webからデータを取得する関数load_from_urlを実装



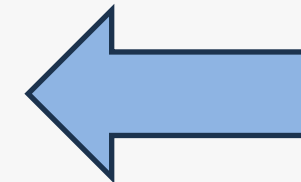
実装したものを実行



タイトルが出力される

- 「山月記」(中島敦)
https://www.aozora.gr.jp/cards/000119/files/624_ruby_5668.zip
- 「学問のすすめ」(福沢 諭吉)
https://www.aozora.gr.jp/cards/000296/files/47061_ruby_28378.zip
- 「走れメロス」(太宰治)
https://www.aozora.gr.jp/cards/000035/files/1567_ruby_4948.zip
- 「吾輩は猫である」(夏目 漱石)
https://www.aozora.gr.jp/cards/000148/files/789_ruby_5639.zip

```
[6] from collections import defaultdict
import re
import io
import urllib.request
from zipfile import ZipFile
```



必要なライブラリ

```
[9] def load_from_url(url):
    data = urllib.request.urlopen(url).read()
    zipdata = ZipFile(io.BytesIO(data))
    filename = zipdata.namelist()[0]
    text = zipdata.read(filename).decode("shift-jis")
    text = re.sub(r'[\.\*?]', '', text)
    text = re.sub(r'《.\*?》', '', text)
    return text
```

```
[10] URL = "https://www.aozora.gr.jp/cards/000119/files/624_ruby_5668.zip"
text = load_from_url(URL)
text.split()[0]
```

‘山月記’

MeCabを利用して文中に出現する名詞トップ10を抽出してみる

```
[11] def show_top10(text):
    m = MeCab.Tagger()
    node = m.parseToNode(text)
    dic = defaultdict(int)
    while node:
        a = node.feature.split(",")
        key = node.surface
        if a[0] == u"名詞" and a[1] == u"一般" and key != "":
            dic[key] += 1
        node = node.next
    for k, v in sorted(dic.items(), key=lambda x: -x[1])[0:10]:
        print(k + ":" + str(v))
```

```
[12] URL = "https://www.aozora.gr.jp/cards/000119/files/624\_ruby\_5668.zip"  
text = load_from_url(URL)  
show_top10(text)
```

自分:24

声:22

徴:19

袁:19

人間:18

虎:16

叢:15

詩:9

曾:7

姿:7



課題3

ワードクラウド
を作ってみよう

① 必要なライブラリをインポート

```
[14] import IPython  
      from wordcloud import WordCloud
```

- ② 一般名詞だけを空白文字列を区切り文字としてつないだ文字列を返す関数、get_wordsを実装

```
[15] def get_words(text):  
    w = ""  
    m = MeCab.Tagger()  
    node = m.parseToNode(text)  
    while node:  
        a = node.feature.split(",")  
        if a[0] == u"名詞" and a[1] == u"一般":  
            w += node.surface + "  
        node = node.next  
    return w
```

③ これを実行

```
▶ URL = "https://www.aozora.gr.jp/cards/000119/files/624\_ruby\_5668.zip"
text = load_from_url(URL)
words = get_words(text)
fpath = '/usr/share/fonts/opentype/ipafont-gothic/ipagp.ttf'
wc = WordCloud(background_color="white", width=480, height=320, font_path=fpath)
wc.generate(words)
wc.to_file("wc.png")
IPython.display.Image("wc.png")
```

④ このようなものが出来ていれば成功

