

# コンピュータリテラシ発展 ～Pythonを学ぶ～

## 第9回：Excel作業の前工程・後工程の自動化

( [shimizu@info.shonan-it.ac.jp](mailto:shimizu@info.shonan-it.ac.jp) )

# 今回の授業内容

## 今回の授業内容

- 
- 8
- 
- CSV
-

## 前回の課題解説

## 前回の課題解説

- 
- 

## 解答例

[https://colab.research.google.com/drive/1il56vahBZGQGQwkpblwzJ92\\_Eodw6xGf?  
usp=sharing](https://colab.research.google.com/drive/1il56vahBZGQGQwkpblwzJ92_Eodw6xGf?usp=sharing)

## 第8回の授業内容の訂正

## 第8回の授業内容の訂正

P.13の内容「Colabで実行するので「¥」ではなく「/(スラッシュ)」で表示されます」

訂正

- Windows                      /(スラッシュ)                      ¥(円マーク)
-

# 文字列操作



# 文字列操作

- Python
  - 
  - 正規表現
- - <https://www.tohoho-web.com/ex/regexp.html>

# 文字列操作

- 
- Google Drive `text_search`
- `text_search` `file.txt`
- 

東京タワーの郵便番号は105-0011で、東京スカイツリーの郵便番号は131-0045です。

# 文字列操作

- 

```
import os
# 作業場所に「text_search」フォルダを作成する
os.makedirs('/content/drive/MyDrive/???/text_search', exist_ok=True)

# 「text_search」フォルダに「file.txt」ファイルを作成して文章を入力する
with open('/content/drive/MyDrive/???/text_search/file.txt', 'x') as f:
    f.write('東京タワーの郵便番号は105-0011で、東京スカイツリーの郵便番号は131-0045です。')
```

# 文字列操作

- - in演算子
    - 
    - 
    -

**True**

**False**

# 文字列操作

- 
- `find()` メソッド
  - 
  -

0

位置						
	0	1	2	3	4	...
文字	東	京	タ	ワ	ー	...

# 文字列操作

- 
- **in**演算子
- **find()** メソッド

```
with open('/content/drive/MyDrive/???.txt', encoding='UTF-8') as f:  
    text = f.read()  
  
if 'タワー' in text:  
    fd = text.find('タワー')  
    print('タワーという文字列が' + str(fd + 1) + '字目に含まれています')  
else:  
    print('タワーという文字列は含まれていません')
```

正規表現を使って文字列を検索

# 正規表現を使って文字列を検索

- " "



## メタ文字の種類と意味

表現	意味
.	1
*	0 0 =
{N}	N
^	
\$	
A B	A B 1

## メタ文字の種類と意味

表現	意味
[X]	[] 1
[X-Y]	[] - 1
[^X]	[] X 1
\d	[0-9]
\D	
\w	

# 正規表現を使って文字列を検索

- 

- 

- 

数字 3 桁-数字 4 桁

- 

```
\d\d\d-\d\d\d\d
```

# 正規表現を使って文字列を検索

- Python
  - `re`
    - regular expression
    - 
    -

# 正規表現を使って文字列を検索

- Python

- re

- findall()

- 

- 

-

# 正規表現を使って文字列を検索

- Python

- re

- findall()

- raw

- 

- \

raw

# 正規表現を使って文字列を検索

- Python

- re

- findall()

```
import re # reモジュールをインポート

# 検索対象のファイルのパスを指定
file_path = '/content/drive/MyDrive/???.txt'

# ファイルをUTF-8エンコーディングで読み込みモードで開く
with open(file_path, encoding='UTF-8') as f:
    # ファイルの内容を読み取り、正規表現を使って郵便番号のリストを抽出
    # 正規表現 r'\d\d\d-\d\d\d\d' は 'xxx-xxxx' 形式の郵便番号にマッチする
    postal_code_list = re.findall(r'\d\d\d-\d\d\d\d', f.read())

# 抽出された郵便番号のリストを表示
print(postal_code_list)
```

# 正規表現を使って文字列を検索

- Python
  - `re`
    - `compile()`
      - 
      -



# 正規表現を使って文字列を検索

- Python

- `re`

- `compile()`

```
import re

file_path = '/content/drive/MyDrive/???/text_search/file.txt'

# 郵便番号を表す正規表現パターンをオブジェクト化
# 正規表現 r'\d\d\d-\d\d\d\d' は 'xxx-xxxx' 形式の郵便番号にマッチする
postal_code_regex = re.compile(r'\d\d\d-\d\d\d\d')

with open(file_path, encoding='UTF-8') as f:
    # ファイルの内容を読み取り、正規表現を使って郵便番号のリストを抽出
    postal_code_list = postal_code_regex.findall(f.read())

print(postal_code_list)
```

## 正規表現を使って文字列の位置を調べる

- 
- `search()`
  - 
  - 
  -

# 正規表現を使って文字列の位置を調べる

- 
- `search()`

```
import re

file_path = '/content/drive/MyDrive/???/text_search/file.txt'

with open(file_path, encoding='UTF-8') as f:
    # ファイルの内容を読み取り、正規表現を使って最初にマッチする郵便番号を検索
    # 正規表現 r'\d\d\d-\d\d\d\d' は 'xxx-xxxx' 形式の郵便番号にマッチする
    print(re.search(r'\d\d\d-\d\d\d\d', f.read()))
```

# 正規表現を使って文字列の位置を調べる

- 
- `search()`
  - - `group()` :
    - `start()` :
    - `end()`
    - `span()` :

# 正規表現を使って文字列の位置を調べる

- 
- `search()`

```
file_path = '/content/drive/MyDrive/???/text_search/file.txt'
postal_code_regex = re.compile(r'\d\d\d-\d\d\d\d')

with open(file_path, encoding='UTF-8') as f:
    # ファイルの内容を読み取り、正規表現を使って最初にマッチする郵便番号を検索
    postal_code_match = postal_code_regex.search(f.read())

    if postal_code_match: # マッチする郵便番号が見つかった場合
        print(postal_code_match) # マッチした結果のMatchオブジェクトを表示
        print(postal_code_match.group()) # マッチした文字列自体を表示
        print(postal_code_match.start()) # マッチした文字列の開始位置を表示
        print(postal_code_match.end()) # マッチした文字列の終了位置を表示
        print(postal_code_match.span()) # マッチした文字列の開始位置と終了位置のタプルを表示
```

正規表現を使って文字列の位置を調べる

# 文字列の置換

- 
- `sub()`
  - 
  - 
  -

# 文字列の置換

- 
- `sub()`
- 東京 Tokyo

```
import re

file_path = '/content/drive/MyDrive/???/text_search/file.txt'

with open(file_path, encoding='UTF-8') as f:
    # ファイルの内容をすべて読み取り、正規表現を使って文字列を置換
    # '東京' という文字列を 'Tokyo' に置換する
    text_mod = re.sub('東京', 'Tokyo', f.read())

    # 置換後のテキストを表示
    print(text_mod)
```



# CSVデータの処理

# CSVの出力

- CSV
  - Comma-Separated Values
  - ,
- CSV
  - python CSV csv
  -

# CSVの出力

- CSV
- CSV
- `writer()`
- `delimiter`
- `writerow()`

CSV

`writer()`

`delimiter`

`,`

CSV

# CSVの出力

- CSV

## sample.csv

```
import os, csv # os, csvモジュールをインポート

# CSVファイルを保存するためのフォルダのパスを指定
path = '/content/drive/MyDrive/???/CSV/'

# 指定されたパスにフォルダを作成. 既に存在する場合はエラーを出さずに無視する
os.makedirs(path, exist_ok=True)

# 'sample.csv'ファイルを新規作成モード ('x') で開き, 改行コードを指定してファイルオブジェクトを取得
with open(path + 'sample.csv', 'x', newline='') as f:
    # csv.writerオブジェクトを作成し, カンマ区切り (delimiter=',') に設定
    w = csv.writer(f, delimiter=',')
    # 1行目のデータ ['1', '2', '3'] をCSVファイルに書き込む
    w.writerow(['1', '2', '3'])
    # 2行目のデータ ['4', '5', '6'] をCSVファイルに書き込む
    w.writerow(['4', '5', '6'])
```

# CSVの出力

- CSV

# CSVの加工

- CSV
- 

- pandas

CSV

pandas

# CSVの加工

- pandas CSV read\_csv()
- CSV DataFrame
- DataFrame 0  
df.loc[0]=['10','20','30']
- to\_csv() CSV
- index=False header=None

# CSVの加工

- pandas CSV

```
import pandas as pd # pandasモジュールをインポート

path = '/content/drive/MyDrive/???/CSV/'

# 'sample.csv'ファイルを読み込み、データフレームに格納
# header=None はCSVファイルにヘッダー行がないことを示す
df = pd.read_csv(path + 'sample.csv', header=None)
# データフレームの最初の行（インデックス0）を新しいデータ ['10', '20', '30'] で置き換える
df.loc[0] = ['10', '20', '30']
# 変更されたデータフレームを新しいCSVファイル 'sample2.csv' に保存
# index=False は行番号を保存しないことを示す
# header=None はヘッダー行を保存しないことを示す
df.to_csv(path + 'sample2.csv', index=False, header=None)
```



# CSVの加工

- pandas
- pandas
- 
- 
- <https://pandas.pydata.org/>
- <https://qiita.com/tags/pandas>

Qiita

# 課題

## 課題

- Moodle SCfCL-9th-prac.ipynb Colab
- File > Download > Download .ipynb .ipynb
- .ipynbファイル Moodle
- 6月20日(木) 20時まで