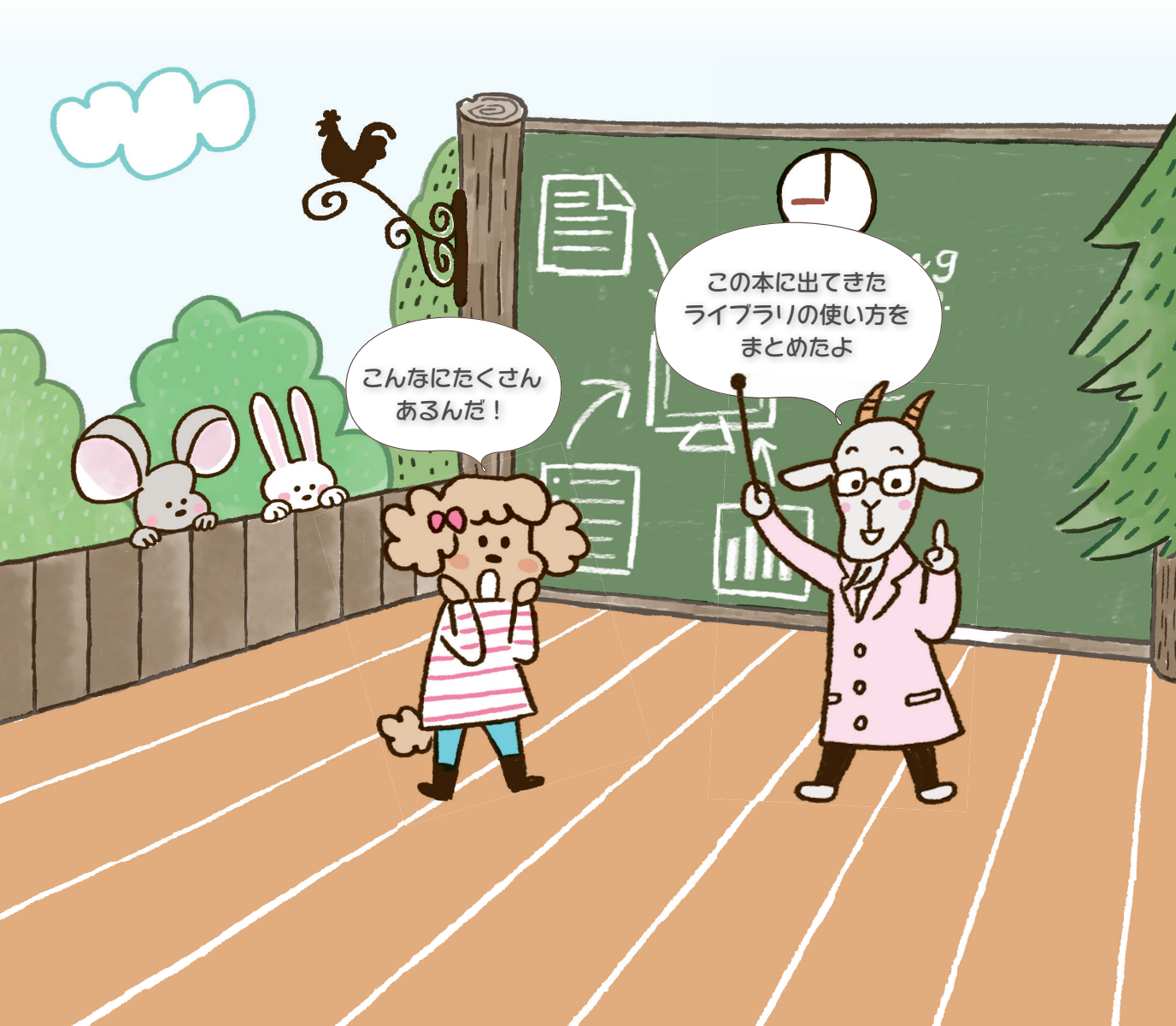
特典PDF
ライブラリ簡易マニュアル



requests

インターネット上のページ情報や画像ファイルなどを取得できるライブラリ

インストール方法 *1
`pip install requests`
importする
`import requests`

Webページを取得する
`response = requests.get(URL)`
ページを文字化けしないようにする
`response.encoding = response.apparent_encoding`
取得したページの文字データを取得する
`response.text`
取得したページのバイナリーデータを取得する
`response.content`
取得したページのURLを取得する
`response.url`
取得したページのステータスコードを取得する
`response.status_code`
取得したページのレスポンスヘッダーを取得する
`response.headers`

*1　インストール方法はWindowsの場合を掲載しています。

Beautiful Soup

HTMLを解析してスクレイピングできるライブラリ

インストール方法
`pip install beautifulsoup4`
importする
`from bs4 import BeautifulSoup`

HTMLを解析する
`soup = BeautifulSoup(html.content, "html.parser")`
タグを探して要素を取り出す
`soup.find("タグ名")`
すべてのタグを探して要素をリストで取り出す
`soup.find_all("タグ名")`
idで探して要素を取り出す
`soup.find(id="id名")`
classで探して要素を取り出す
`soup.find(class_="class名")`
要素の属性の値を取り出す
`要素.get("属性名")`

urllib

URLにアクセスしたり、URLの処理を行えるライブラリ

インストール不要（標準ライブラリ）

importする

```
import urllib
```

相対URLを絶対URLに変換する

```
parse.urljoin(ベースURL, 調べるURL)
```

pathlib

ファイルシステムのパスを扱えるライブラリ

インストール不要（標準ライブラリ）
importする
`from pathlib import Path`

フォルダを作る
`パス = Path("フォルダ名")` `パス.mkdir(exist_ok=True)`
フォルダ内のファイルにアクセスするパスを作る
`パス.joinpath("ファイル名")`
ファイルやフォルダがあるかどうかを取得する
`パス.exists()`
パスがファイルかどうかを取得する
`パス.is_file()`
パスがフォルダかどうかを取得する
`パス.is_dir()`
「相対パス」を「絶対パス」に変換する
`パス.resolve()`
カレントフォルダを取得する
`Path.cwd()`

time

時間を扱えるライブラリ

インストール不要（標準ライブラリ）
importする
`import time`

現在の時刻を取得する
`time.ctime()`
指定した秒数一時停止する
`time.sleep(秒数)`

pandas（パンダス）

表データの読み書きや集計、分析を行えるライブラリ

インストール方法
`pip install pandas`

importする
`import pandas as pd`

CSVファイルを読み込む
`df = pd.read_csv("ファイル名.csv")`
CSVファイルを読み込む（ヘッダーがなく、1行目からデータを読む）
`df = pd.read_csv("ファイル名.csv", header=None)`
CSVファイルを読み込む（3行目にヘッダーがあるデータを読む）
`df = pd.read_csv("ファイル名.csv", header=2)`
CSVファイルを読み込む（1列目をインデックスとして読む）
`df = pd.read_csv("ファイル名.csv", index_col=0)`
CSVファイルを読み込む（Shift JISコードのデータを読む）
`df = pd.read_csv("ファイル名.csv", encoding="Shift_JIS")`
ZIP圧縮されたCSVファイルを読み込む（ZIP圧縮されたまま読める）
`df = pd.read_csv("ファイル名.zip")`

CSVファイルに出力する
`df.to_csv("ファイル名.csv")`
CSVファイルに出力する（インデックスを削除）
`df.to_csv("ファイル.csv", index=False)`
CSVファイルに出力する（ヘッダーを削除）
`df.to_csv("ファイル.csv", header=False)`

CSVファイルに出力する（インデックスとヘッダーを削除）

```
df.to_csv("ファイル.csv", index=False, header=False)
```

DataFrameを新しく作る（空のデータ）

```
df = pd.DataFrame()
```

DataFrameを新しく作る（2列、2行のデータ）

```
df = pd.DataFrame({"列1":["行1","行2"], "列2":["行1","行2"]})
```

DataFrameのカラム（列名）を取得する

```
df.columns.values
```

DataFrameのインデックスを取得する

```
df.index.values
```

1列のデータを取得する

```
df["列名"]
```

複数列のデータを取得する

```
df[["列名1","列名2"]]
```

1行のデータを取得する

```
df.loc[行番号]
```

複数行のデータを取得する

```
df.loc[[行番号1,行番号2]]
```

1つのデータを取得する

```
df.loc[行番号]["列名"]
```

1行追加する

```
df.loc[2] = ["行3A","行3B"]
```

1列追加する

```
df["列C"] = ["行1C","行2C","行3C"]
```

指定した列を削除する

```
df = df.drop("列名",axis=1)
```

指定した行を削除する
df = df.drop(行番号,axis=0)

条件に合うデータを抽出する（90以上のデータを抽出）
df = df[df["列名"] >= 90]
集計（最大値、最小値、平均値、中央値、合計など）した結果を取得する
df["列名"].max()
df["列名"].min()
df["列名"].mean()
df["列名"].median()
df["列名"].sum()

データをソートする（昇順：小さい値ほど前へ）
df = df.sort_values("列名")
データをソートする（降順：大きい値ほど前へ）
df = df.sort_values("列名",ascending=False)
列同士の計算結果を新しい列に入れる
df["新しい列"] = df["列1"] - df["列2"]
行と列を入れ替える
df.T
データをPythonのリスト化する
df.values

matplotlib

いろいろな種類のグラフを描画できるライブラリ

インストール方法
pip install matplotlib pip install japanize_matplotlib
importする
import matplotlib.pyplot as plt import japanize_matplotlib

折れ線グラフを作る
data.plot()
棒グラフを作る
data.plot.bar()
棒グラフを表示する画面サイズを指定する
data.plot.bar(figsize=(幅インチ, 高さインチ))
棒グラフ（横向き）を作る
data.plot.barh()
積み上げ棒グラフを作る
data.plot.bar(stacked=True)
積み上げ棒グラフ（横向き）を作る
data.plot.barh(stacked=True)
円グラフを作る
data.plot.pie()
円グラフを作る（ラベルの表示位置を変える）
data.plot.pie(labeldistance=距離)
作ったグラフを表示する
plt.show()

作ったグラフを画像ファイル出力する

```
plt.savefig("ファイル名.png")
```

タイトルを表示する

```
plt.title("タイトル")
```

凡例の位置を指定する

```
plt.legend(loc="指定文字列")
```

表示位置	指定文字列	数値
ベストな位置	best	0
右上	upper right	1
左上	upper left	2
左下	lower left	3
右下	lower right	4
右	right	5
中央左	center left	6
中央右	center right	7
中央下	lower center	8
中央上	upper center	9
ど真ん中	center	10

横軸の目盛りを置き換える

```
plt.xticks(置き換える目盛りのリスト, 目盛りのリスト)
```

縦軸の目盛りを置き換える

```
plt.yticks(置き換える目盛りのリスト, 目盛りのリスト)
```

横軸の最大値、最小値を指定する

```
plt.xlim([最小値,最大値])
```

縦軸の最大値、最小値を指定する

```
plt.ylim([最小値,最大値])
```

openpyxl

Excelの読み書きができるライブラリ

インストール方法
`pip install openpyxl xlrd xlwt`

importする
`import openpyxl`

Excelファイルに出力する
`df.to_excel("ファイル名.xlsx")`

Excelファイルに出力する（インデックスを削除）
`df.to_excel("ファイル名.xlsx", index=False)`

Excelファイルに出力する（シート名を指定）
`df.to_excel("ファイル名.xlsx", sheet_name="シート名")`

複数のシートを1つのExcelファイルに出力する

`with pd.ExcelWriter("ファイル名.xlsx") as writer:`
`  df1.to_excel(writer, sheet_name="シート名1")`
`  df2.to_excel(writer, sheet_name="シート名2")`

Excelファイルを読み込む
`df = pd.read_excel("ファイル名.xlsx")`

Excelファイルを読み込む（複数シートから）
`df = pd.read_excel("ファイル名.xlsx", sheet_name="シート名")`

folium

データを地図上に表示できるライブラリ（Web地図のLeaflet.jsを使用）

インストール方法
`pip install folium`
importする
`import folium`

指定した地点の地図を作る
`m = folium.Map([緯度, 経度], zoom_start=ズーム倍率)`
マーカーを追加する
`folium.Marker([緯度, 経度]).add_to(m)`
マーカーを追加する（ツールチップ付き）
`folium.Marker([緯度, 経度], tooltip="文字列").add_to(m)`
地図を表示するHTMLファイルを書き出す
`m.save("ファイル名.html")`