Python でのファイル操作

■ファイルの読み込み

ファイルからデータを読み込むには「open」関数を使います。読み込みはオプションに「r」（read）を指定。

以下は、テキストファイルを読み込む基本的なコード。

```
#1
with open('sample.txt', 'r') as file:
    data = file.read()
    print(data)
```

■ファイルへの書き込み

ファイルにデータを書き込む場合も「open」関数を使い、オプションには「w」（write）を指定。「w」指定は**上書き**される。「write」関数で書き込む

```
#2
with open('sample.txt', 'w') as file:
    file.write("Hello, World!")
```

「w」指定で、もしファイルが存在しない場合には、**新しくファイルが作成**されます。

・追記モード　a

open 関数のオプション指定を「a」にすれば、既存のファイルの末尾に新しい内容が追記される

■ファイルの編集

ファイル編集では、まずファイルを読み込み、必要な編集を加えた後、ファイルに書き戻す。

```
#3
with open('sample.txt', 'r') as file:
    data = file.read()
    data = data.replace("Hello", "Hi")

with open('sample.txt', 'w') as file:
    file.write(data)
```

・文字列の置き換え　replace

#3 のコードには「replace」がある。

replace は、**文字列の内容を変更**する。文字列だけを操作するもので、ファイル操作には関係しない。

文法：

```
replace(“古い文字列”,  ”新しい文字列”)
```

■ファイルを閉じる

ファイルの操作後には「close」でファイルを閉じることが重要。

ファイルを閉じることで、ファイルとの接続を切断して、システムのリソースを解放します。

文法：

```
file = open('sample.txt', 'r')
      # ファイル操作...
file.close()
```

◆コンテキストマネージャー(‘with’ステートメント)

Python では、’with’ステートメントを使用してファイル操作をすることが推奨されています。これにより、ファイル操作が終了した時点で自動的にファイルが閉じられる。ファイルを手動で閉じ忘れるリスクが避けられる。

このコードでは、’with’ブロックの中でファイル操作を行い、ブロックを抜けると自動的にファイルが閉じられる。

文法：

```
with open('example.txt', 'r') as file:
    # ファイル操作...
# ここで自動的にファイルが閉じられる
```

・ほかの’with’ステートメントの用法

例外処理、コードの可読性の上昇、データベース接続を開く、スレッドのロック管理、一時的な状態の設定、リソースのライフサイクル管理など