16 Python基礎

OSインターフェイス

- `os` モジュールを利用する

```
import os
```

- osモジュール内には、数多くのメソッドが用意されている

> https://docs.python.org/ja/3/library/os.html# （本家）
>
> https://zenn.dev/robes/articles/ae1f1dbcb0c5aa （以下URLより）

関数	内容
os.getcwd()	現在の作業フォルダのpathを取得する
os.getenv(key)	環境変数keyが存在する場合はその値を取得する
os.chdir(path)	作業フォルダをpathに変更する
os.mkdir(path)	フォルダ(ディレクトリ)をpathに作成する
os.makedirs(path)	深い階層のディレクトリまで再帰的に作成する
os.rmdir(path)	フォルダを削除する。フォルダの中身は空でないと削除できない
os.path.exists(path)	ファイルが存在するか確認する
os.path.isdir(path)	ディレクトリが存在するか確認する
os.path.dirname(path)	フォルダ名だけを取り出す
os.path.basename(path)	ファイル名だけを取り出す
os.path.join(path1,path2,path3)	パスを結合する
os.listdir(path)	指定したディレクトリの中のファイルやフォルダのリストを返す
os.remove(path)	指定したファイルを削除する
os.rename(src, dst)	ファイルやディレクトリの名前を変更する
os.stat(path)	ファイルやディレクトリの属性を取得する
os.path.split(path)	パスをディレクトリとファイル名に分割する
os.path.abspath(path)	指定したパスの絶対パスを返す
os.path.isfile(path)	指定したパスがファイルかどうかを確認する
os.path.getsize(path)	指定したファイルのサイズを取得する
os.environ	環境変数を辞書として取得する
os.system(command)	シェルコマンドを実行する

    - 一部加筆

※ 注意事項

- 使用しているOSにより、若干動作が異なるので注意が必要です。
    - 吉村は基本Linux/Mac上でテストしているため、挙動が異なっている場合があります。

- 明らかに挙動が異なると分かっている場合はWindows10/11上でテストを行っていますが、疑問点があれば聞いてください。

練習

以下のプログラムを作成してください

1. `os` モジュールを使用して、現在の作業ディレクトリを取得し、それを表示するプログラムを作成してください。（ren1_1.py）
   【実行結果】

```
$ python ren1_1.py
現在の作業ディレクトリは: /home/yoshimura/NT_Python基礎/src/16
```

```
>python ren1_1.py
現在の作業ディレクトリは: Z:\NT_Python基礎\src\16
```

2. `os` モジュールを使用して、新しいディレクトリを作成するプログラムを作成してください。
   ディレクトリ名はユーザーからの入力とします。(ren1_2.py)

> https://note.nkmk.me/python-os-mkdir-makedirs/

- エラー処理はどうするのが良いだろうか？　正しく処理を組み込んでください

【実行結果】

```
$ python ren1_2.py
作成するディレクトリ名を入力してください: temp/test1
ディレクトリ 'temp/test1' を作成しました。

$ tree ./
./
├── ren1_1.py
├── ren1_2.py
└── temp
    └── test1

2 directories, 2 files
```

```
>python ren1_2.py
作成するディレクトリ名を入力してください: test/test1
ディレクトリ 'test/test1' を作成しました。

>tree ./
フォルダー パスの一覧:  ボリューム VBOX_yoshimura
ボリューム シリアル番号は 000000A6 0001:0303 です
Z:\NT_PYTHON基礎\SRC\16
├─temp
│  └─test1
└─test
    └─test1
```

3. `os` モジュールを使用して、指定された環境変数の値を取得し、表示するプログラムを作成してください。
   環境変数の名前はユーザーからの入力とします。(ren1_3.py)
   【実行結果】

```
$ python ren1_3.py
取得する環境変数の名前を入力してください: path
環境変数 'path' は存在しません。
$ python ren1_3.py
取得する環境変数の名前を入力してください: PATH
PATH =
/home/yoshimura/.local/bin:/home/yoshimura/bin:/home/yoshimura/.modular/pkg/packages.modular.com_mojo/bin:/home/yoshimura/.nvm/versions/node/v16.20.0/bin:/usr/local/cuda/bin:/home/yoshimura/.rbenv/shims:/home/yoshimura/.rbenv/bin:/home/yoshimura/anaconda...
```

```
>python ren1_3.py
取得する環境変数の名前を入力してください: path
path =
C:\Users\satos\anaconda3\envs\py39;C:\Users\satos\anaconda3\envs\py39\Library\mingw-w64\bin;C:\Users\satos\anaconda3\envs\py39\Library\usr\bin;C:\Users\satos\anaconda3\envs\py39\Library\bin;C:\Users\satos\anacon...

>python ren1_3.py
取得する環境変数の名前を入力してください: PATH
PATH =
C:\Users\satos\anaconda3\envs\py39;C:\Users\satos\anaconda3\envs\py39\Library\mingw-w64\bin;C:\Users\satos\anaconda3\envs\py39\Library\usr\bin;C:\Users\satos\anaconda3\envs\py39\Library\bin;C:\Users\satos\anacon...
```

   - Linuxでは大文字・小文字を厳密に区別しているが、Windowsでは区別していないことが判る

4. `os` モジュールを使用して、指定されたファイルのサイズをバイト単位で取得し、表示するプログラムを作成してください。
   ファイルパスはユーザーからの入力とします。(ren1_4.py)
   【実行結果】

```
$ python ren1_4.py
サイズを取得するファイルのパスを入力してください: ren1_1.py
ファイル 'ren1_1.py' のサイズは 111 バイトです。

$ python ren1_4.py
サイズを取得するファイルのパスを入力してください: hoge.py
ファイル 'hoge.py' が見つかりません。
```

5. `os` モジュールを使用して、指定されたディレクトリ内のファイルとディレクトリの一覧を取得し、表示するプログラムを作成してください。
   ディレクトリのパスはユーザーからの入力とします。(ren1_5.py)
   【実行結果】

```
$ python ren1_5.py
ファイルとディレクトリを一覧表示するディレクトリのパスを入力してください: ./
ディレクトリ './' 内の項目:
 - ren1_3.py
 - temp
 - ren1_4_try.py
 - ren1_2.py
 - ren1_1.py
 - test
 - ren1_4.py
 - .mypy_cache
 - ren1_5.py

$ python ren1_5_try.py
ファイルとディレクトリを一覧表示するディレクトリのパスを入力してください: aaa
ディレクトリ 'aaa' が見つかりません。
```

ワイルドカードとコマンドライン引数

- `glob` モジュール、 `sys` モジュール（ `sys.argv`)の使い方を確認してください

練習

1. コマンドライン引数として名前を受け取り、それを表示するプログラムを作成してください(ren2_1.py)
   【実行結果】

```
$ python ren2_1.py
使用方法: python ren2_1.py <名前>

$ python ren2_1.py 吉村
こんにちは、吉村さん！
```

2. コマンドライン引数として2つの数値を受け取り、その合計を計算して表示するプログラムを作成してください(ren2_2.py)
   【実行結果】

```
$ python ren2_2.py
使用方法: python ren2_2.py <数値1> <数値2>

$ python ren2_2.py 100 50
100.0 + 50.0 = 150.0
```

3. コマンドライン引数としてファイルパスを受け取り、そのファイルの内容を表示するプログラムを作成してください(ren2_3.py)
   【実行結果】

```
$ python ren2_3.py
使用方法: python ren2_3.py <ファイルパス>

$ python ren2_3.py ren1_1.py
import os

current_directory = os.getcwd()
print(f"現在の作業ディレクトリは: {current_directory}")
```

4. コマンドライン引数としてディレクトリのパスを受け取り、そのディレクトリ内のファイル一覧を表示するプログラムを作成してください(ren2_4.py)
   【実行結果】

```
$ python ren2_4.py
使用方法: python ren2_4.py <ディレクトリパス>

$ python ren2_4.py temp
ディレクトリ 'temp' 内のファイル一覧:
 - test1
```

5. コマンドライン引数として数値のリストを受け取り、その平均を計算して表示するプログラムを作成してください(ren2_5.py)
【実行結果】

```
$ python ren2_5.py
使用方法: python ren2_5.py <数値1> <数値2> ...

$ python ren2_5.py 10 20 30 40
数値リストの平均: 25.0
```

確認

- 以下のプログラムは、一見正しく動いているのだが、OSにより挙動が異なる

【argv_1.py】

```
import sys

argv = sys.argv

print(argv)
```

【実行結果】Windows（MS-DOSコマンドプロンプト）

```
>python argv_1.py a b *.py
['argv_1.py', 'a', 'b', '*.py']
```

【実行結果】Linux（bash）

```
$ python argv_1.py a b *.py
['argv_1.py', 'a', 'b', 'argv_1.py', 'argv_2.py', 'ren1_1.py', 'ren1_2.py', 'ren1_3.py',
'ren1_4.py', 'ren1_4_try.py', 'ren1_5.py', 'ren1_5_try.py']
```

- なぜこのようなことが起きるのだろうか？

練習

- Windows上でも、Linux版と同じ結果になるように自前で実装してください。(argv_2.py)
    - 考え方（アルゴリズム）案
        - `sys.argv` の値を1つずつ確認する。
            - 文字 `*` もしくは `?` を含んでいたら、 `glob.glob()` を使用し、ファイルのリストを取得し、リストに格納
            - 含んでいない場合は、そのままの値をリストに格納

【実行結果】

```
>python argv_2.py abc test/*
['argv_2.py', 'abc', 'test\\test1']

>python argv_2.py argv_?.py xyz
['argv_2.py', 'argv_1.py', 'argv_2.py', 'xyz']
```

やや規模の大きいプログラムを作成してみよう1

コマンドラインオプションの処理

- 例えば、`python`自体のコマンドラインオプションとして以下のようなものがある。

```
$ python -V
Python 3.9.15

$ python --version
Python 3.9.15
```

- `mkdir`コマンドのコマンドラインオプションを確認してみる。

```
$ mkdir --help
使用法: mkdir [OPTION]... DIRECTORY...
ディレクトリが存在しない場合に、ディレクトリを作成します。

長いオプションで必須となっている引数は短いオプションでも必須です。
  -m, --mode=MODE   set file mode (as in chmod), not a=rwx - umask
  -p, --parents     no error if existing, make parent directories as needed
  -v, --verbose     print a message for each created directory
  -Z, --context=CTX 作成した各ディレクトリに SELinux セキュリティコンテキスト CTX
                    を設定する
  -Z                    作成した各ディレクトリに SELinux セキュリティコンテキストを
                        デフォルトタイプに設定する
      --context[=CTX]   -Z と同様だが、 CTX が指定された場合は、
                        SELinux や SMACK のセキュリティコンテキストを CTX に設定する
      --help     この使い方を表示して終了する
      --version  バージョン情報を表示して終了する

GNU coreutils のオンラインヘルプ: <https://www.gnu.org/software/coreutils/>
翻訳に関するバグは <https://translationproject.org/team/ja.html> に連絡してください。
詳細な文書 <https://www.gnu.org/software/coreutils/mkdir>
(ローカルでは info '(coreutils) mkdir invocation' で参照可能)。
```

- 通常コマンドには、長いオプションと短いオプションが存在することが多い。
  - pythonには`getopt`モジュール（C言語風というか古くから存在する形式)、`argparse`モジュールが用意されている。
  - どのように使用するのか確認してください。

> getopt https://docs.python.org/ja/3.9/library/getopt.html
>
> argparse https://docs.python.org/ja/3.9/library/argparse.html#module-argparse

- `argparse`の使い方を理解しよう

> 公式チュートリアル https://docs.python.org/ja/3.9/howto/argparse.html

- Linuxコマンドに存在する `wc` について調査してください
  - 何をするコマンドなのか？
  - どのようなことができるのか？
  - 何が結果として出力されるのか？
  - 具体的な使い方は？

> https://eng-entrance.com/linux-command-wc

目的

- 最終的には `wc` コマンドと同じ動作をする、wc.pyを作成する。
- まずは、基本的な動作を行うプログラムを作成する。
- コマンドオプションは、最初は無しとし、今後付加する。
  - オプションは実装しない
  - ディレクトリは対応しない
  - 複数のファイルには対応する
  - 指定されたファイルの、行数・単語数・文字数を出力する

【実行例】

```
$ cat -n argv_1.py
     1  import sys
     2
     3  argv = sys.argv
     4
     5  print(argv)
```

```
$ wc argv_1.py
 5  6 41 argv_1.py
```

```
$ wc *.py
   5    6   41 argv_1.py
  13   34  237 argv_2.py
   4    7  111 ren1_1.py
   5   11  213 ren1_2.py
```

```
  8   21  290 ren1_3.py
  8   20  343 ren1_4.py
  8   20  338 ren1_4_try.py
 10   25  424 ren1_5.py
 10   25  415 ren1_5_try.py
  8   17  194 ren2_1.py
 10   30  258 ren2_2.py
 13   32  397 ren2_3.py
 14   35  490 ren2_4.py
  8   24  268 ren2_5.py
124  307 4019 合計
```

練習

1. 上記の仕様に基づいた `wc.py` を作成してみよう。
2. getoptもしくはargparseを使用して、コマンドラインオプションを理解するようにしてみよう。
    - モジュールの利用方法について個別に十分テストした上で、組み込んでください。

```
-v
-h
-c
-m
-l
-w
```

        - 上記のオプションを1つずつ追加してみよう