# 応用プログラミング ex4 演習課題

## 1. thread.py へ新しいスレッド実行を追加

1. /roes/sample/sano/apro/thread.py をホームディレクトリ以下の適当な場所にコピーしなさい。
2. thread.py を実行し複数スレッドによる並行実行の様子を確認しなさい。
3. 2つの引数 n と r をもつ関数 fxx(n, r) を定義しなさい。fxx(n, r) は、“fxx” という関数名と引数 n, r の値を繰り返し表示するような関数である。ただし、n は1回の表示後に実行を停止(sleep)する秒数、r は表示を繰り返す回数である。また、関数名と n, r の表示には、何回目の表示であるかも合わせて表示すること。
4. thread.py を修正し、関数 fxx() をスレッドとして実行するプログラム thread2.py を作成しなさい。

### 実行例：fxx(0.3, 5) として呼び出し

```
(aprog) t190900@s01cd0542-161:~/apro$ python thread2.py
### start function call
f1 0
f1 1
f1 2
f2 0
f2 1
f2 2
fx 1 0
fx 1 1
fx 1 2
fxx 0.3 5 0
fxx 0.3 5 1
fxx 0.3 5 2
fxx 0.3 5 3
fxx 0.3 5 4
### end function call
### start threads
f1 0
f2 0
fx 1 0
fxx 0.3 5 0
### end threads???
fxx 0.3 5 1
f2 1
f1 1
fxx 0.3 5 2
f2 2
fxx 0.3 5 3
f1 2
fx 1 1
fxx 0.3 5 4
fx 1 2
### end threads!!!
```

## 2. コマンドラインオプションの追加 httpThreadServer2.py

1. /roes/sample/sano/apro/ 以下の httpThreadServer.py と server.html（ex3 と同じものです） をホームディレクトリ以下の適当な場所にコピーしなさい。
2. httpThreadServer.py を適当なコマンドライン引数と共に実行し、Firefox などの HTTPクライアントアプリケーション(Webブラウザ)から httpThreadServer.py にアクセスしてその動作を確認しなさい。
3. httpThreadServer.py を修正し、2つ目のコマンドライン引数として任意のファイル名を指定することで、指定されたファイルの内容を返信できる(httpServer2.py と同様の) httpThreadServer2.py を作成しなさい。
4. ただし、httpThreadServer2.py は、2つの引数の両方、または、ファイル名のみを省略できるものとする。また、ポート番号が省略された場合は 50007、ファイル名が省略された場合は、server.html が指定されたものとして実行されること。
5. httpThreadServer2.py がコマンドライン引数によって異なるデータを送信できることを Webブラウザからアクセスして確認しなさい。

**実行例1：引数指定なし**

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpThreadServer2.py
port= 50007 , file= server.html
Connected by, ('127.0.0.1', 64596)
Received: GET / HTTP/1.1
Upgrade-Insecure-Requests: 1
 (中略)
Connection: keep-alive

Sent: HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

<html>
  <head><meta charset="UTF-8"></head>
  <body>
  これは server.html です
  </body>
</html>
```

**実行例2：ポート番号のみ指定**

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpThreadServer2.py 5000
port= 5000 , file= server.html
Connected by, ('127.0.0.1', 64596)
Received: GET / HTTP/1.1
Upgrade-Insecure-Requests: 1
 (中略)
Connection: keep-alive

Sent: HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

<html>
  <head><meta charset="UTF-8"></head>
  <body>
```

```
  これは server.html です
  </body>
</html>
```

**実行例3：ポート番号とファイル名 foo.html を指定**

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpThreadServer2.py 5000
foo.html
port= 5000 , file= foo.html
Connected by, ('127.0.0.1', 64596)
Received: GET / HTTP/1.1
Upgrade-Insecure-Requests: 1
 (中略)
Connection: keep-alive

Sent: HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

<html>
  <head><meta charset="UTF-8"></head>
  <body>
  これは foo.html です
  </body>
</html>
```

**実行例4：ポート番号と存在しないファイル名 bar.html を指定**

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpThreadServer2.py 5000
bar.html
port= 5000 , file= bar.html
Connected by, ('127.0.0.1', 64596)
Received: GET / HTTP/1.1
Upgrade-Insecure-Requests: 1
 (中略)
Connection: keep-alive

Sent: HTTP/1.1 404 Not Found
Content-Type: text/html; charset=utf-8

<html><body>bar.html is not found</html></body>
```

## 3. GET 以外のリクエストに対してエラーメッセージを返信する httpThreadServer3.py

1. httpThreadServer.py を修正し、GET 以外のコマンドを受け取った場合は、

   ```
   HTTP/1.1 501 Not Implemented
   ```

   という HTTPレスポンスを返信するHTTPサーバプログラム httpThreadServer3.py を作成しなさい。

## 4. リクエストするファイル名をURLで指定する httpThreadServer4.py

1. Webブラウザなどのhttpクライアントから、次のようなURL(URI)

   ```
   http://localhost:50007/foo.html
   ```

   で指定される HTTPリクエスト(クライアントからサーバへ送信される GET コマンド)は、

   ```
   GET /foo.html HTTP/1.1
   ```

   のような形式でサーバに送られる。

2. すなわち、

   ```
   http://ホスト名:ポート番号/ファイル名
   ```

   としてURL に任意のファイル名を与えることでWebブラウザが要求するファイル名をサーバに伝えることができる。

3. Webブラウザからファイル名を含んだ URL によって HTTPリクエストを受け取ると、指定されたファイルの内容を返信する HTTPサーバ httpThreadServer4.py を作成しなさい。
4. ただし、指定されたファイル名が存在しない場合は、httpThreadServer.py などと同様に *404 Not Found* エラーを返信するものとする。
5. **httpThreadServer4.py の動作が確認できたらTA予約を行いTAの指示に従いなさい。**

**実行例1： http://localhost:50007/**

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpThreadServer4.py
port= 50007 , file= server.html
Connected by, ('127.0.0.1', 64938)
Received: GET / HTTP/1.1
Host: localhost:50007
Upgrade-Insecure-Requests: 1
 (中略)
Connection: keep-alive

Sent: HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

<html>
  <body>
  これは server.html です
  </body>
</html>
```

**実行例2：http://localhost:50007/foo.html**

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpThreadServer4.py
```

```
port= 50007 , file= server.html
Connected by, ('127.0.0.1', 65042)
Received: GET /foo.html HTTP/1.1
Host: localhost:50007
Upgrade-Insecure-Requests: 1
 (中略)
Connection: keep-alive

Sent: HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

<html>
  <body>
  これは foo.html です
  </body>
</html>
```

**実行例3：http://localhost:50007/bar.html**

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpThreadServer4.py
port= 50007 , file= server.html
Connected by, ('127.0.0.1', 65058)
Received: GET /bar.html HTTP/1.1
Host: localhost:50007
Upgrade-Insecure-Requests: 1
 (中略)
Connection: keep-alive

Sent: HTTP/1.1 404 Not Found
Content-Type: text/html; charset=utf-8

<html><body>bar.html is not found</html></body>
```

文字列型の操作については テキスト p.167 以降を参照。文字列を分割する split() や置換・削除する replace() などが使えると思います。

From:
https://www-slab.math.ryukoku.ac.jp/ - **www-slab.math**

Permanent link:
**https://www-slab.math.ryukoku.ac.jp/lecture/apro/2019/ex4**

Last update: **2019/12/10 10:07**