応用プログラミング ex3 演習課題

1. 実行と動作確認 httpServer.py

1. /roes/sample/sano/apro/ 以下の httpServer.py と server.html 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. httpServer.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 httpServer.py にアクセスしてその動作を確認しなさい。
3. 自分の計算機の 50007番ポートで起動した httpServer.py に対して、その計算機上の Webブラウザから接続するための URL は、たとえば次のようなものである。

```
http://localhost:50007/
```

```
http://127.0.0.1:50007/
```

2. コマンドラインオプションの追加 httpServer2.py

1. httpServer.py は、(省略可能な)1つのコマンドライン引数によってサーバのポート番号を指定することができる。
2. httpServer.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 を作成しなさい。
3. ただしhttpServer2.py は、2つの引数の両方、または、2つめの引数であるファイル名を省略できるものとする。また、1つめの引数であるポート番号が省略された場合は 50007、2つめの引数である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。
4. httpServer2.py の動作を確認するために server.html の他に幾つかの HTML ファイルを作成し、コマンドラインオプションで返信データを変更できることを確認しなさい。（呼び出されるファイルは、httpServer.py と同じディレクトリにあることが仮定されています。）
5. server.html は Webブラウザが表示できる HTML というマークアップ言語で書かれているHTML ファイルの書き方やフォーマットは各自で調べなさい。

実行例1：ポート番号のみ指定

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpServer2.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2：ポート番号とファイル名 foo.html を指定

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpServer2.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3：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 httpServer2.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 httpServer3.py

1. httpServer.py では□HTTPリクエストが *“GET ”* で始まる場合にのみ□HTTPプロトコルに沿った返信を行っている(httpServer.py 40行目)。

   ```
   if data.startswith(b"GET "):
   ```

2. **httpServer2.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 httpServer3.py を作成しなさい。

3. GET 以外のコマンドをWebブラウザ上のURL入力で確認するのは難しいため、作成した httpServer3.py の動作は以下のようなコマンドを端末(ターミナル)から実行することで確認することができます。このコマンドでは、自分自身(localhost)の50007番ポートに PUT コマンドを送信しています。

```
$ echo -en "PUT /server.html HTTP/1.1\r\n\r\n" | nc localhost
50007
```

4. **httpServer3.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。**

startswith() メソッドについては テキスト p.175 を参照。

4. リクエストするファイル名をURLで指定する httpServer4.py

1. httpServer.py などの出力から確認できるようにWebブラウザから

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

   のような URL での HTTPリクエスト(クライアントからサーバへ送信されるコマンド)は、

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

   のような形式となる。

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 URL によってファイル名を指定した HTTPリクエストを受け取ると、指定されたファイル名の内容を返信する HTTPサーバ httpServer4.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、httpServer.py などと同様に *404 Not Found* エラーを返信するものとする。

実行例1： **http://localhost:50007/**

```
(aprog) t190900@s01cd0542-161:~/apro$ python httpServer4.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 httpServer4.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 httpServer4.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() などが使えます。