応用プログラミング
ex4

本日のレシピ

1. telnet コマンド
2. ブロッキングIO
3. スレッド thread.py
4. httpThreadServer.py

telnet

- TCP/IPプロトコルでパケットの読み書きをするコマンド
  - telnet でクライアントになってサーバとデータのやり取りする

HTTPリクエスト（httpクライアント）の例：

```
(aprog) a00007@s01cd0542-161:~/apro$ telnet localhost 50007
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
GET / HTTP/1.1

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>

Connection closed by foreign host.
```

※ 類似のコマンドで nc という高機能版もあります。

本日のレシピ

1. telnet コマンド
2. ブロッキングIO
3. スレッド thread.py
4. httpThreadServer.py

ブロッキング？

httpServer.py

```
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, True)  # ex2.1-2 メモを参照
    s.bind((HOST, port))
    while True:
        s.listen(1)
        conn, addr = s.accept()          # クライアントの接続を待つ = 実行がブロックされる
        with conn:
            print("Connected by,", addr)
            data = b""
            while True:
                data += conn.recv(1024)  # クライアントからのデータ受信を待つ = 実行がブロックされる
                if b"\r\n\r\n" in data:
                    break
            print("Received:", data.decode(), end="")  # 改行しない
```

ブロッキングI/O

- ソケットなどの入出力処理中は、処理が終わるまで待つ（実行をブロックする）
- 処理手続きがわかりやすいが、複数の処理を同時に処理できない

httpサーバなので同時に処理したい、どうする？

1. スレッドを使って複数の処理を同時に処理する（並行処理・マルチタスク）
2. ブロッキングしない（ノンブロッキングI/O・イベントループ）→ ここでは取り扱わない

本日のレシピ

1. telnet コマンド
2. ブロッキングIO
3. スレッド thread.py
4. httpThreadServer.py

スレッド（Thread）

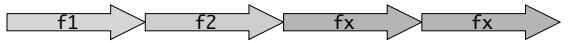
- 複数の処理を同時並行的に実行する仕組み（CPUの複数コアでも並列実行）

```
from time import sleep
import threading

Repeat = 5

def f1():
    for i in range(Repeat):
        print("f1", i)
        sleep(0.5)

def f2():
    for i in range(Repeat):
        print("f2", i)
        sleep(0.4)

def fx(n):
    for i in range(Repeat):
        print("fx", n, i)
        sleep(n)
                :
            （続く）
```

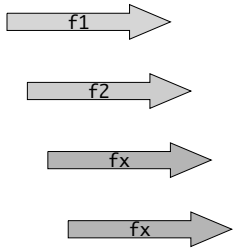
```
                （続き）
                :
print("### start function call")
f1()
f2()
fx(1)
fx(2)
print("### end function call")

thread1 = threading.Thread(target=f1)
thread2 = threading.Thread(target=f2)
thread1x = threading.Thread(target=fx, args=(1,))
thread2x = threading.Thread(target=fx, args=(2,))

print("### start threads")
thread1.start()
thread2.start()
thread1x.start()
thread2x.start()
print("### end threads???")

thread1.join()
thread2.join()
thread1x.join()
thread2x.join()
print("### end threads!!!")
```


thread.py

通常の関数呼び出し：



スレッド実行：



スレッド作成：
Thread()
target=関数名
args=引数タプル

スレッド実行：
Thread.start()

スレッド終了を待機：
Thread.join()

```
print("### start function call")
f1()
f2()
fx(1)
fx(2)
print("### end function call")

thread1 = threading.Thread(target=f1)
thread2 = threading.Thread(target=f2)
thread1x = threading.Thread(target=fx, args=(1,))
thread2x = threading.Thread(target=fx, args=(2,))

print("### start threads")
thread1.start()
thread2.start()
thread1x.start()
thread2x.start()
print("### end threads???")

thread1.join()
thread2.join()
thread1x.join()
thread2x.join()
print("### end threads!!!")
```

本日のレシピ

1. telnet コマンド
2. ブロッキングIO
3. スレッド thread.py
4. httpThreadServer.py

httpThreadServer.py

```
def httpResponse(connection, address):
    with connection:
        print("Connected by,", address)
           # : (省略)
        connection.close()

if __name__ == "__main__":
    if argc > 2:
        print("Usage:", args[0], "[port]")
        exit()
    elif argc > 1:
        port = int(args[1])

    HOST = ""  # サーバホスト名（'' とすると実行マシン上の接続可能な全てのホスト名）
    print("port=", port, ", file=", file)

    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, True)  # ex2.1-2 メモを参照
        s.bind((HOST, port))
        while True:
            s.listen(1)
            conn, addr = s.accept()
            #httpResponse(conn, addr)  # 関数 httpResponse() は単純にこの部分を関数化しただけ
            http_thread = threading.Thread(target=httpResponse, args=(conn, addr))
            http_thread.start()
```


httpThreadServer.py を起動してtelnetで通信