Python Socket通信

Socket—低水準ネットワークインターフェース

●Socket通信とは

Socketとは、データの送受信を行うためのインターフェースです。そのsocketを使用した通信のことをソケット通信と呼びます。ソケット通信には、ネットワーク上の通信(INET)とPC内の(UNIX)があります。

●ネットワークプログラミングAPI

ネットワークプログラミングAPIは、ソケット以外にもいくつかあります。

種類	説明
SOCKET	TCP/UDPポートをそのまま実装した基本的な低レベルのAPI
CORBA	オブジェクト指向プログラミングシステム
SOAP	Webサービスを用いたXMLベースのメッセージ交換システム
MPI	並列処理を目的としたメッセージ交換システム

●Socke通信

クライアントとサーバが相互に接続し、データの送受信を行うクライアントサーバーモデルです。ほとんどのインターネットアプリケーションはこのモデルになります。これは、サーバは常時待ち受け状態(パッシブオープン)で起動し、クライアントは必要な時にアクセスしてくる(アクティブオープン)というモデルです。

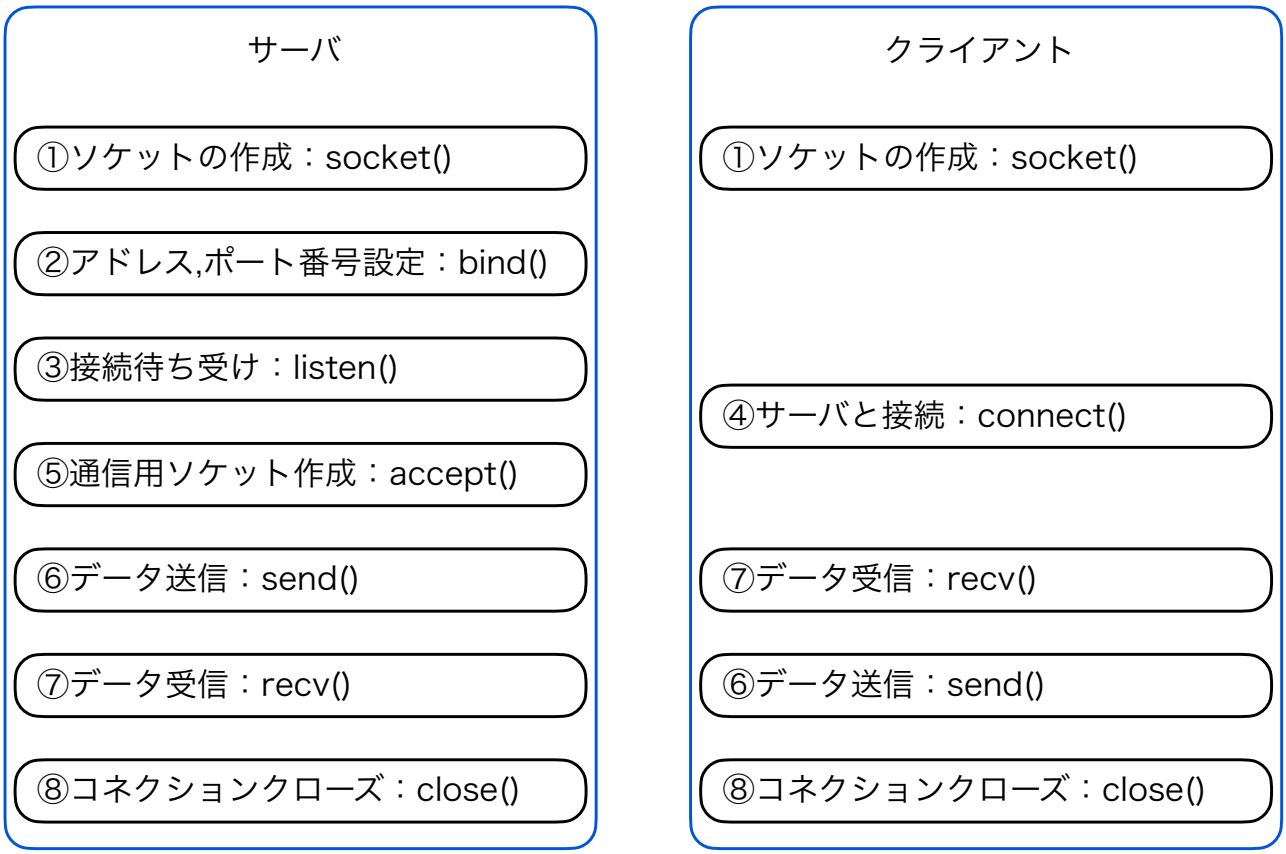
端末	モード
サーバ	パッシブオープン
クライアント	アクティブオープン

●ソケットインターフェース

・ソケットはアプリケーションで作成される
・ソケットを特定のポートに接続する
・TCP、UDPコネクションを利用してデータ通信を行える
・アプリケーションは複数のソケットを利用できる
・ソケットは複数のアプリケーションから利用される
・ソケットによる接続が確立されると、ソケットの書き込みはファイルの書き込みと同じI/O処理で行える。

●socket通信の流れ

クライアントサーバモデルのソケット通信の流れを示します。



サーバとクライアントで少し手順が違います。

●サーバ

サーバは①でソケットを作成し、アドレスやポート番号を設定します。そのソケットで接続の待ち受け（③）を行います。受動的に接続を受け入れるためパッシブオープンと言います。
クライアントから接続要求があった場合、⑤で新しく通信用のソケットを作成し、そのソケットで通信を行います。

●クライアント

クライアントでは、ソケットを作成し、サーバーと接続を行います。能動的に接続を行うのでアクティブオープンと言います。

データの送受信が終わったら必ずクローズを行ってください。クローズはサーバ側からでもクライアント側からでもどちらからもできます。

サーバ（単方向通信：送信1回で切断）

まずは、localhostでサーバとクライアントを作成し、クライアントからの接続要求後、サーバからクライアントにメッセージを送信し、サーバ側から接続を解除するアプリを作成しましょう。
クライアントはメッセージを受け取り、そのメッセージを表示します。
サーバから構築してみましょう。

1	import socket	
2	server = socket.soket(socket. AF_INET, socket. SOCK_STREAM)	①ソケット作成
3	server.bind((“”,50000))	②設定
4	server.listen()	③待ち受け
5	client,addr = server.accept()	⑤通信用ソケット
6	romantic.sendall(“サーバからメッセージを送信”.encode(“UTF-8”))	⑥送信
7	client.close()	⑧クローズ
8	server.close()	⑧クローズ

●import

socketはpythonに標準ライブラリとして準備されています。使用するには、サーバーアプリ作成時、クライアントアプリ作成時でもまず「socket」のimportが必要になります。

```
import socket
```

●①ソケットの作成：socket.soket()

ソケットはトランスポート以下の通信のプロトコルセットを指定します。第一引数でインターネット層のプロトコルを、第二引数でトランスポート層のプロトコルを指定します。

```
socket.socket(family=AF_INET,type=SOCK_STRAM,
                                    port=0,fileno=None
```

引数の「family」ではアドレスファミリを指定します。アドレスファミリはソケット通信で使用されるアドレス構造体のフォーマットです。

アドレスファミリ	内容
AF_UNIX	同一マシン上のプロセス間で使用するアドレスファミリ
AF_INET	インターネットプロトコルIPv4を使用するアドレスファミリ
AF_INET6	インターネットプロトコルIPv6を使用するアドレスファミリ
AF_BLUETOOTH	Bluetoothアドレスを使用するアドレスファミリ
AF_CAN	Controller Area Network。自動車や工業機器の通信に使われる
AF_PACKET	ﾊﾟｹｯﾄﾚﾍﾞﾙのinterface。ネットワークディバイスを直接操作可能
AF_RDS	高パフォーマンスかつ低遅延通信プロトコル

引数の「type」ではソケットタイプを指定します。ソケットタイプとは通信方法です。TCPやUDPなどを指定します。

ソケットタイプ	内容
SOCK_STREAM	TCP接続
SOCK_DGRAM	UDP接続
SOCK_RAM	低レベルプロトコルに直接アクセス。ICMP(pingなど)
SOCK_RDM	信頼性のある双方向通信（あまり使われていない）
SOCK_SEQPACKET	信頼性のある双方向通信。順序に従って送信(ストリーミングなど)

引数の「port」では、プロトコル番号を指定しますが、通常は設定しないので省略して構いません。「TCP=6」「UDP=17」です。
引数の「fileno」も通常は設定しないので省略して構いません。ここにはファイルディスクプリンタ（OSがリソースを管理するためにつけた識別子）を指定します。別のアプリのsocketを使用する場合などに使用します。

2	server = socket.soket(socket. AF_INET, socket. SOCK_STREAM)
	#IPv4でTCPプロトコルのソケットを作成。第三、第四引数省略

●②アドレスとポート番号の設定：socket.bind()

socket.bind(address)

addressはタプルで渡します。タプルの内容は①で選択したアドレスファミリによって変わってきます。下記表を参考にしてください。

アドレスファミリ	addressフォーマット
AF_UNIX	
AF_INET	(host,port)
AF_INET6	(host,port,flowinfo,scope_id)
AF_BLUETOOTH	
AF_CAN	(interface,)
AF_PACKET	(i name,port[,pkttype[,hatype[,addr]]])

サーバーを建てる場合、アドレスは「””」空にします。

3	server.bind((“”,50000))	

●③接続待受：listen()

作ったソケットを「listen()」で待受状態にします。

socket.listen([backlog])

「listen()」の引数でサーバーに対する接続数の上限を指定できます。省略した場合はシステムにより適当な値が設定されます。

4	server.listen()	待受開始

●⑤クライアントからの接続要求の受け入れ：accept()

「accept()」はクライアントからの接続要求があった場合に、その要求を受け入れて新しい通信用ソケットを作成します。

conn,address=server.accept()

戻り値の「conn」は新しいソケットです。「address」は、クライアントのアドレスとポート番号です。新しいソケットの中には情報が含まれていますので、「print」で確認してください。

5	client,addr = server.accept()	
	#client:<fd,family,type,porto,laddr,raddr>	print(client)
	#addr:クライアントのアドレスとポート番号	print(addr)

●⑥データ送信：sendall()

socket.sendall(bytes[,flags])

「sendall」は、引数に指定されている「bytes」の全てのデータを送信します。正常終了の場合は「None」を返し、エラーが発生した場合は例外が発生します。
引数はバイトオブジェクトなので、メッセージを送る際は変換する必要があります。

6	romantic.sendall(“サーバからメッセージを送信”.encode(“UTF-8”))	

「flags」を指定すると、エラー発生時にキューに入れられたエラーをソケットのエラーキューから取り出せるようになります。

●⑧コネクションクローズ：close()

通信終了後、「close()」で待受を解除します

socket.close()

7	client.close()	
8	server.close()	

サーバ（双方向通信：送信1回で切断）

1	import socket	
2	client = socket.soket(socket. AF_INET, socket. SOCK_STREAM)	①ソケット作成
3	client.connect((“localhost”,50000))	④接続
4	data = client.recv(4096)	⑦受信
5	print(data.decode(“UTF-8”))	
7	#client.close() 今回はサーバからクローズするので必要ない	

●④サーバと接続：connect()

socket.connect(address)

「connect」は「address」で指定したリモートソケットに接続します。「address」のフォーマットはアドレスファミリによって異なります。4ページ参照。

3	client.connect((“localhost”,50000))	

●⑦データ受信：recv()

socket.recv(bufsize[,flags])

「recv」はデータを受信します。引数に指定されている「bufsize」で指定した量だけデータを受信できます。戻り値はバイトオブジェクトですので、元に戻す必要があります。

	data = client.recv(4096)	
	print(data.decode(“UTF-8”))	

クライアント