برنامه نویسی سوکت با پایتون

در پایتون، برنامه نویسی سوکت (Socket Programming) با استفاده از کتابخانه‌ی استاندارد سوکت (socket) امکان‌پذیر است. در برنامه نویسی سوکت به طور معمول نیاز به برنامه نویسی در دو سمت سرور (Server) و کلاینت (Client) داریم، هر چند امکان برقراری ارتباطات غیر کلاینت-سروری نیز وجود دارد. اما ما در اینجا ارتباطات سرور-کلاینتی را پوشش می‌دهیم، لذا ابتدا کدهای سمت سرور و سپس سمت کلاینت را برنامه نویسی می‌کنیم.

سمت سرور

قبل از هر چیز کتابخانه‌ی سوکت را در ابتدای کد خود وارد می‌کنیم و در ادامه یک شی از سوکت می‌سازیم که ما نام آن را server_socket تعیین کرده‌ایم.

```
3  import socket
4  def server_program():
5      host = '127.0.0.1'
6      port = 5000  # initiate port no above 1024
7      print (host)
8
9      server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # get instance
```

همان‌طور که می‌بینید در هنگام تعریف شی، باید دو آرگومان نیز وارد کنیم که آرگومان اول نوع IP را مشخص می‌کند. در اینجا چون ما قصد استفاده از IPV4 را داریم، گزینه‌ی AF_INET را وارد کردیم. اما اگر قصد استفاده از IPV6 داشتیم، باید AF_INET6 را انتخاب می‌کردیم. آرگومان دوم نیز نوع پروتکل را معین می‌کند که در اینجا چون ما از پروتکل TCP استفاده می‌کنیم، گزینه‌ی SOCK_STREAM را انتخاب کرده‌ایم. حال که سوکت خود را ایجاد کردیم، باید آن را قابل دسترسی کنیم که برای این کار از متد ()bind استفاده می‌کنیم. در متد ()bind باید دو آرگومان که اولی IP و دومی port است را وارد کنیم. به این صورت:

```
11      # look closely. The bind() function takes tuple as argument
12      server_socket.bind((host, port))  # bind host address and port together
```

در این مثال، IPاستاندارد local host که همان ’127.0.0.1‘ است را وارد کردیم، اما می‌توانیم یک رشته‌ی خالی (" ") نیز استفاده کنیم که در آن صورت، سرور تمامی آدرس‌های IP را می‌پذیرد. ضمن این‌که می‌توانیم به جای IP و رشته‌ی خالی، از نام هاست (hostname) نیز استفاده کنیم، اما ممکن است برنامه رفتار عادی از خود

نشان ندهد، لذا بهتر است به جای نام هاست از IP استفاده کنیم. برای شماره پورت نیز می‌توانیم از پورت‌های ۱ تا ۶۵۰۰۰ استفاده کنیم، اما بهتر است از پورت ۱ تا ۱۰۲۴ که پورت‌های رزرو شده هستند، استفاده نکنیم. پس از در دسترس قرار دادن سوکت، باید آن را در حالت شنود یا گوش دادن قرار دهیم تا بتواند درخواست اتصال کلاینت‌ها را متوجه شود. برای این کار از متد listen() استفاده می‌کنیم.

```
14      # configure how many client the server can listen simultaneously
15      server_socket.listen(2)
```

همان‌طور که مشاهده می‌کنید، ما مقدار ۲ را به متد listen() داده‌ایم که این عدد تعداد درخواست همزمان اتصال کلاینت‌ها را تعیین می‌کند و از ۰ تا ۵ قابل مقداردهی است.

اکنون لازم است سرور خود را منتظر اتصال یک کلاینت کنیم برای همین از متد accept() استفاده می‌کنیم. به این صورت که ابتدا دو متغیر با نام دلخواه، که ما conn و addr را برگزیدیم، تعریف کرده و سپس سوکت ایجاد شده را accept() می‌کنیم.

```
17      conn, address = server_socket.accept()  # accept new connection
```

در قطعه کد بالا، مشخصات کلاینت متصل شده در conn ذخیره شده و در addr یک تاپل (Tuple) که شامل IP و port کلاینت متصل شده است، ذخیره می‌شود.

برای نمایش برقراری صحیح اتصال به کاربر، با استفاده از دستور print یک پیغام مناسب به همراه مقدار متغیر addr را چاپ می‌کنیم.

```
19          print("Connection from: " + str(address))
```

حال یک حلقه ی true while مطابق زیر در نظر می گیریم تا سرور و کلاینت به یکدیگر پیام ارسال کنند.

```
21      while True:
22          # receive data stream. it won't accept data packet greater than 1024 bytes
23          data = conn.recv(1024).decode()
24          if not data:
25              # if data is not received break
26              break
27          print("from connected user: " + str(data))
28          data = input(' -> ')
29          conn.send(data.encode())  # send data to the client
```

در پایان نیز کلاینت متصل را با استفاده از متد close() می‌بندیم.

```
31          conn.close()  # close the connection
```

سمت کلاینت

برای برنامه نویسی سمت کلاینت نیز همانند سمت سرور، ابتدا کتابخانه سوکت را داخل کد خود وارد می‌کنیم. سپس یک شی با نام دلخواه از سوکت می‌سازیم توضیحات لازم در مورد آرگومان‌های داخل متد socket() را پیش‌تر در بخش کدهای سمت سرور آورده‌ایم .به این صورت:

```
3  import socket
4
5  def client_program():
6      host = '127.0.0.1'  # as both code is running on same pc
7      port = 5000  # socket server port number
8      client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  # instantiate
```

اکنون باید سوکت ایجاد شده‌ی خود را به سرور متصل کنیم که برای این کار از متد connect() استفاده می‌کنیم. برای اتصال به سرور باید IP و port داده شده در سرور را در متد connect() وارد کنیم. به این صورت:

```
10     client_socket.connect((host, port))  # connect to the server
```

حال یک حلقه ی while مطابق زیر در نظر می گیریم تا سرور و کلاینت به یکدیگر پیام ارسال کنند. پایان ارسال و دریافت پیام ها ارسال کلمه ی "bye" توسط کلاینت است.

```
12     message = input(" -> ")  # take input
13     while message.lower().strip() != 'bye':
14         client_socket.send(message.encode())  # send message
15         data = client_socket.recv(1024).decode()  # receive response
16
17         print('Received from server: ' + data)  # show in terminal
18
19         message = input(" -> ")  # again take input
```

سپس سوکت خود را با متد close() می‌بندیم .

```
21     client_socket.close()  # close the connection
```