Socket Programming

عمید اسدالهی مجد

دانشگاه مازندران
بهار ۹۹

چکیده

این برنامه از دو بخش کلاینت (کاربر) و سرور تشکیل شده است.
وظیفه بخش سرور دریافت اطلاعات و ارائه پاسخ مناسب متناسب با ورودی است و همچنین بخش کاربر هم از سرویس هایی که سرور ارائه میدهد استفاده می‌کند.
در این برنامه کاربر عبارتی ریاضی را به عنوان درخواست به سرور ارسال می‌نماید و سپس سرور پاسخ عبارت ریاضی درخواستی را با کاربر ارسال می‌کند. کاربر با فرستادن عبارت "done"
می‌تواند به ارتباط ایجاد شده خاتمه دهد اما سرور پس از خاتمه ارتباط، در انتظار ارتباط (درخواست) جدید باقی می‌ماند.
همچنین آدرس سوکت سرور localhost یا 127.0.0.1 در پورت 8080 می‌باشد که به معنای محلی بودن ارتباط است و با متغیر server_address تعریف شده است.
در تمام این ارتباطات داده‌ها برای ارسال به نوع بایت تبدیل شده (encode) و در هنگام دریافت از نوع بایت به رشته تبدیل شدند. (decode)
برای کار کردن با سوکت ها در پایتون از کتابخوانه socket که به صورت پیشفرض در پایتون موجود است، استفاده شد.

* لازم به ذکر است در این برنامه به دلیل کوتاهی درخواست و جواب از حلقه ای برای خواندن مرحله به مرحله بافر در نظر گرفته نشد. در غیر این صورت به سادگی با یک حلقه با شرط خاتمه صفر شدن اندازه بافر و در هر مرحله اضافه کردن داده‌های دریافتی از بافر به متغییر اصلی درنظر گرفته شده در برنامه قابل پیاده‌سازی است.

```
while True:
        if  len(buffer) == 0:
                break
        data += buffer.decode()
```

سرور

ابتدا یک سوکت جدید ایجاد میکنیم. این سوکت از نـوع TCP بـا IPv4 بـوده و بـا نـوع ارتبـاطی Stream یـا جریـانی می‌باشد. نوع TCP/IPv4 با ورودی AF_INET و نوع ارتباط با ورودی SOCK_STREAM مشخص می‌شود.

```
server_address = ('127.0.0.1', 8080)
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(server_address)
```

سپس با کد زیر شروع گوش دادن سرور را اعلام میکنیم. مقدار ۱ برای تعداد ارتباطاتی است که سرور نگهداری می‌کند و بعد از آن را دور می‌اندازد.

```
server_socket.listen(1)
```

با شروع حلقه بی نهایت که شرط خاتمه آن بسته شدن سوکت می‌باشد به دلیل انتظار سرور برای ارتباطات جدید است.
سپس ارتباط جدید را قبول می‌کنیم :

```
client_socket, client_address = server_socket.accept()
```

در ادامه با حلقه بی نهایت بعدی تا خاتمه ارتباط توسط کاربر صبر کرده و در هر مرحله عبارت ریاضی درخواستی را دریافت و جواب آن ارسال می‌شود.
تابع eval در پایتون برای اعتبار بخشیدن به یک رشته استفاده می‌شود که در این برنامه درواقع با اعتبار بخشیدن به رشته ریاضی جواب آن برگردانده میشود.
همچنین اندازه بافر ورودی ۱۵۰ بایت تعریف شد.

```
while True:
        in_data = client_socket.recv(150)
        in_data = in_data.decode('utf-8')
        if in_data:
            try:
                answer = str(eval(in_data))
            except:
                answer = 'error!'

            client_socket.sendall(answer.encode('utf-8'))
        else:
            break
```

در انتهای ارتباط سوکت مربوط به کاربر بسته شده و خاتمه می‌یابد، اما سرور منتظر ارتباط جدید می‌ماند.

```
client_socket.close()
```

کلاینت (کاربر)

مشابه سوکت سرور، سوکتی برای کاربر از جنس TCP/IPv4 و STREAM می‌سازیم و سپس ارتباط جدیدی با سرور برقرار می‌کنیم.

```
server_address = ('127.0.0.1', 8080)
client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(server_address)
```

در ادامه متغیر request برای ساختن درخواست مورد استفاده قرار می‌گیرد و حلقه بی نهایت به صورت مداوم به گرفتن ورودی از کاربر، ارسال آن به سرور و سپس نمایش جواب ارسالی از سمت سرور می‌پردازد. شرط خاتمه این حلقه دریافت ورودی "done" از کاربر است.
همچنین بافر درنظر گرفته شده برای جواب دریافتی ۱۰۰ بایت است.

```
while True:
    request = input('\nWhat is your math question? [sample: 5*8+4] ')
    if request == 'done':
        break
    client_socket.sendall(request.encode('utf-8'))

    in_data = client_socket.recv(100)
    in_data = in_data.decode('utf-8')
    print(f'The answer of {request} is {in_data}.')
```

در انتها پس از خروج از حلقه بی نهایت بالا سوکت کاربر بسته شده و خاتمه می یابد.

```
client_socket.close()
```

در انتها نتیجه به صورت زیر خواهد بود :

```
amid manjaro ~/amid_projects/network python socket_server.py
started: <socket.socket fd=3, family=AddressFamily.AF_INET, type=SocketKind.SOCK_STREAM, proto=0, laddr=('127.0.0.1', 8080)>

waiting for request ...
connection accepted from: ('127.0.0.1', 39952)

input data: 8 * 4
Answered 8 * 4

input data: 7 ** 12 + 31284
Answered 7 ** 12 + 31284

input data: er;lkasdk
Answered er;lkasdk

input data:
No data recived from ('127.0.0.1', 39952)
connection closed

waiting for request ...
```

```
amid manjaro ~/amid_projects/network python socket_client.py
connecting to ('127.0.0.1', 8080)

What is your math question? [sample: 5*8+4] 8 * 4
The answer of 8 * 4 is 32.

What is your math question? [sample: 5*8+4] 7 ** 12 + 31284
The answer of 7 ** 12 + 31284 is 13841318485.

What is your math question? [sample: 5*8+4] er;lkasdk
The answer of er;lkasdk is error!.

What is your math question? [sample: 5*8+4] done
connection closed
```