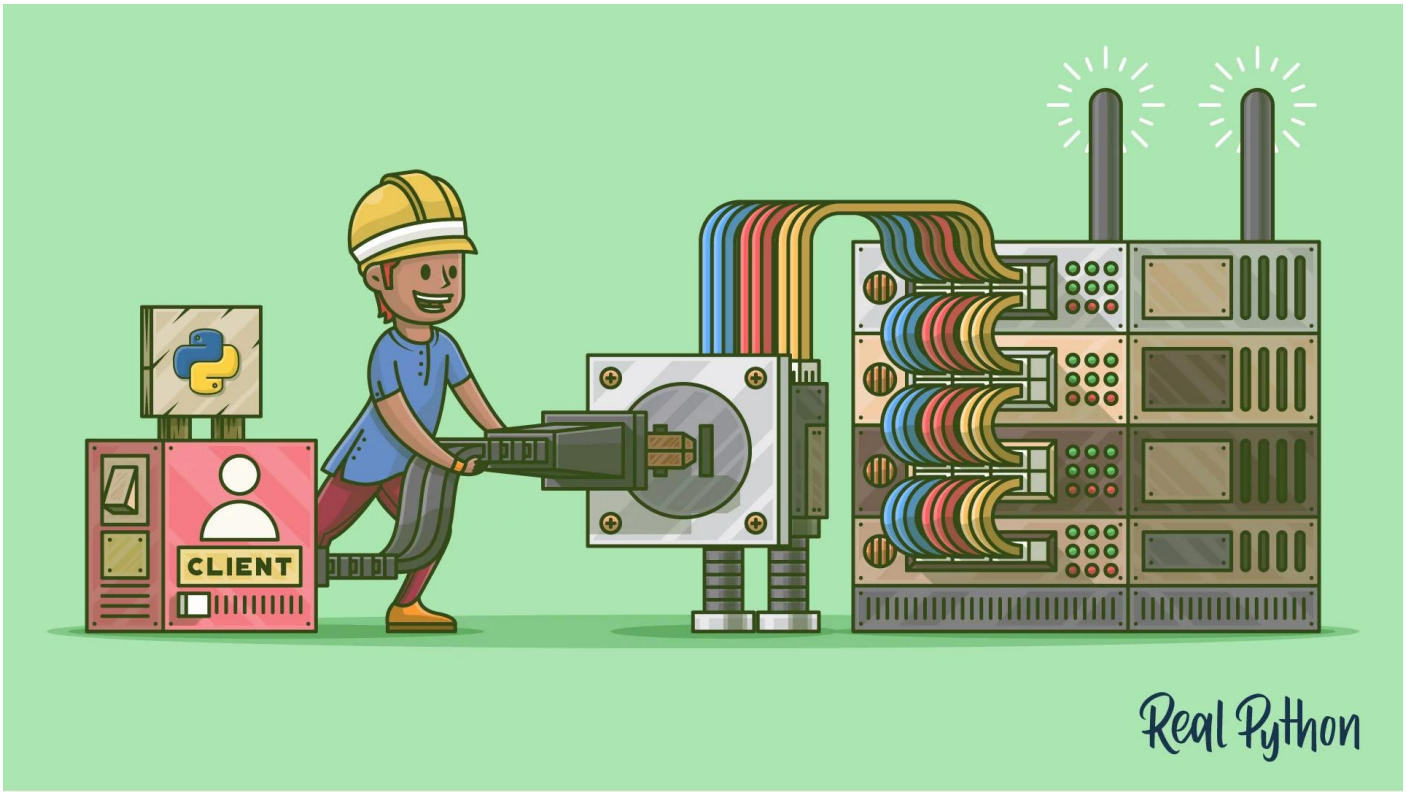
# آزمایشگاه شبکه‌های کامپیوتری ۱

نیم سال اول ۹۹-۹۸

## برنامه‌نویسی سوکت در زبان پایتون نسخه‌ی ۳




## هدف آزمایش

در این دستورکار قصد داریم برنامه نویسی با سوکت در پایتون را مرور کنیم. امروزه پایتون یکی از کاربردی‌ترین زبان‌های برنامه‌نویسی است و در زمینه‌های متفاوتی می‌توان از آن استفاده کرد. در کتاب مرجع درس نیز از ویرایش ششم به بعد از زبان برنامه‌نویسی پایتون استفاده شده است.

## مقدمه

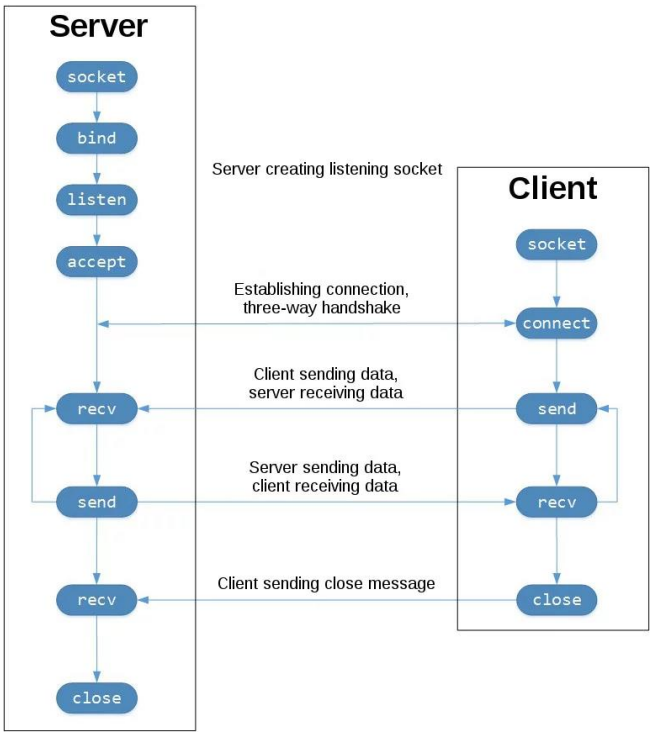
در این دستورکار قصد داریم یک سرور و کلاینت با استفاده از زبان پایتون پیاده‌سازی کنیم. در این برنامه کلاینت پس از متصل شدن به سرور، متن پیش فرض را به سرور ارسال می‌کند. سرور پس از دریافت متن، آن را دوباره به کلاینت ارسال می‌کند. کلاینت با دریافت دوباره متن، متن دریافت شده را چاپ کرده و اتصال را قطع می‌کند.

## فعالیت‌های قبل آزمایش

- مرور پروتکل‌های لایه ارتباط
- آشنایی با یک زبان برنامه‌نویسی
- سیستم عامل لینوکس

## شرح آزمایش

برای پیاده‌سازی سرور و کلاینت Echo از پروتکل TCP برای ارتباط استفاده می‌کنیم. هدف این برنامه باز ارسال داده‌های کلاینت به همان کلاینت می‌باشد. این برنامه‌ها در گذشته برای بررسی صحت ارتباط‌های سطح بالا استفاده می‌شدند. برای یک ارتباط TCP چرخه‌ی حیات زیر را می‌توان در نظر گرفت:




### سرور Echo

کد سرور به شرح زیر است. هر بخش کد به تفکیک در ادامه توضیح داده می‌شود.

```
import socket

HOST = '127.0.0.1'  # Standard loopback interface address (localhost)
PORT = 1373         # Port to listen on (non-privileged ports are > 1023)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    conn, addr = s.accept()
    with conn:
        print('Connected by', addr)
        while True:
            data = conn.recv(1024)
            if not data:
                break
            conn.sendall(data)
```

در ابتدا یک سوکت ایجاد می‌کنیم. کد زیر برای ایجاد سوکت استفاده می‌شود.:

```
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

ورودی اول تابع socket نوع آدرس‌دهی و ورودی دوم نوع انتقال داده است. در این مثال با انتخاب AF_INET نوع آدرس دهی را از نوع IPv4 انتخاب میکنیم در صورتی که قصد داریم آدرس را به صورت IP نوع چهارم یا host name استفاده کنیم باید این نوع از آدرس دهی را به عنوان ورودی به تابع بدهیم. SOCK_STREAM برای ارتباط بر روی پروتکل TCP می باشد. در صورتی که قصد داشته باشیم پروتکل لایه انتقال را تغییر دهیم از این ورودی استفاده می‌کنیم.

پس از ایجاد سوکت آدرس مربوطه را به آن تخصیص می‌دهیم. در اینجا آدرس داده شده به سوکت آدرسی است که سوکت اطلاعات را از آن دریافت و از طریق آن ارسال می‌کند. آدرس شامل دوبخش host و port است. Host یک رشته است که مشخص می‌کند این عمل گوش دادن روی کدام کارت شبکه صورت بگیرد و قسمت port مشخص کننده‌ی port گوش دهنده است. مقدار Host با توجه به نوع آدرس مشخص شده می تواند به صورت IP نوع چهارم، host name یا رشته خالی باشد. در صورتی که رشته خالی به عنوان host ارسال شود، سرور از تمام کارت های شبکه خود برای ارتباط گیری استفاده می‌کند. در نهایت با مشخص شدن این اطلاعات می‌خواهیم که سوکت در حالت قبول تقاضا قرار بگیرد.

```
s.bind((HOST, PORT))
s.listen(1)
```

پس از مشخص شدن اطلاعات آدرس، سوکت باید منتظر برقرای ارتباط بماند. برنامه پس از برخورد با کد زیر در این همان جا متوقف می شود و منتظر دریافت درخواست ایجاد یک ارتباط توسط کلاینت می ماند. پس از دریافت درخواست برقراری ارتباط، سرور با فراخوانی کد زیر درخواست را تایید می کند. پس از تایید درخواست، آدرس درخواست دهنده و ارتباط ایجاد شده در دو متغیر addr و conn ذخیر می گردد.

```
conn, addr = s.accept()
```

حالا که ارتباط برقرار شده است، امکان انتقال اطلاعات برقرار می باشد. سرور هر بار به اندازه مشخصی از داده را می خواند. در این نمونه کد در هر بار مقدار 1024 بیت داده خوانده می شود. به همین دلیل عمل خواندن داده باید در یک حلقه بی انتها قرار داده شود تا این عمل تا جایی که دیگر داده ای وجود نداشته باشد ادامه یابد.

```
with conn:
  print('Connected by', addr)
  while True:
    data = conn.recv(1024)
    if not data:
      break
```

پس از اتمام دریافت داده، داده بدست آمده دوباره برای سرور ارسال می گردد. تابع sendAll تمام مقدار داده شده را برروی ارتباط برقرار شده ارسال می نماید.

```
conn.sendall(data)
```

## انجام دهید

۱. سرور Echo را مطابق با آنچه توضیح داده شد پیاده‌سازی نمایید.

۲. استفاده از مقدار

```
HOST = '0.0.0.0'
```

چه تغییری ایجاد می‌کند.

## کلاینت Echo به وسیله‌ی Telnet

پس از ایجاد سرور نیاز است که یک کلاینت نیز ایجاد کنیم تا بتوانیم با سرور ارتباط برقرار کنیم. یکی از راه‌های ساده تست سرور پیش از پیاده‌سازی کلاینت استفاده از دستور telnet در سیستم عامل لینوکس می‌باشد.. ساختار ساده این دستور به شکل زیر است:

```
telnet [ip] [port]
```

دستور telnet یک ارتباط TCP با آدرس IP و پورت داده شده برقرار می‌کند. شما می‌توانید هر آنچه که می‌خواهیم نوشته و با فشردن دکمه‌ی enter آن را به مقصدتان ارسال کنید.

## انجام دهید

۱. سروری را که در قسمت قبل پیاده‌سازی کرده‌اید با دستور telnet آزمایش نمایید.

## کلاینت Echo

```python
import socket

HOST = '127.0.0.1'  # The server's hostname or IP address
PORT = 1373         # The port used by the server

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall('سلام دنیا'.encode())
    data = s.recv(1024)
    message = data.decode()
    print(f'received {message!r}')
```

مانند سرور در این بخش نیز نیاز است ابتدا یک سوکت با ویژگی های یکسان با سوکت ساخته شده در سرور بسازیم:

```
socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

پس از ایجاد سوکت، آدرس سرور را به سوکت داده تا درخواست ارتباط به سرور ارسال گردد. کد در این خط باقی می ماند تا زمانی که درخواست ارتباط توسط سرور قبول، رد و یا منقضی گردد. در صورت قبول نشدن درخواست برنامه خطا داده و متوقف می گردد. حتما در نظر داشته باشید که در هنگام استفاده از این سوکت در نرم افزارها، خطاهای بوجود آمده به علت عدم برقراری ارتباط به برنامه اصلی شما صدمه ای نزند. در صورت تایید ارتباط، برنامه ادامه می یابد.

```
s.connect((HOST, PORT))
```

برای رسیدگی به خطاها در نظر داشته باشید که می‌توان از ساختار Try/Except در پایتون استفاده کرد:

```python
try:
    with open('file.log') as file:
        read_data = file.read()
except:
    print('Could not open file.log')
```

یا

```python
try:
    with open('file.log') as file:
        read_data = file.read()
except FileNotFoundError as fnf_error:
    print(fnf_error)
```

کد زیر رشته Hello world را به سرور ارسال می کند.

```
s.sendall('Hello, world'.encode())
```

پس از ارسال داده، مانند بخش سرور منتظر دریافت داده دریافتی می مانیم.

```
while True:
 data = conn.recv(1024)
 if not data:
  break
```

پس از پایان یافتن برنامه کلاینت ارتباط بین کلاینت و سرور و برنامه سرور نیز پایان می یابند.

## انجام دهید

۱. کلاینت Echo را مطابق با آنچه توضیح داده شد پیاده‌سازی نمایید.

۲. بر اساس کدهای فوق یک برنامه چت را پیاده‌سازی کنید. در این برنامه کلاینت پس از ایجاد ارتباط شروع به ارسال داده می‌کند. سرور با دریافت داده‌ها آن‌ها را چاپ می‌کند. کلاینت پس از ارسال کلمه end نوبت ارسال پیام را به سرور می‌دهد و منتظر دریافت داده‌ها می‌ماند. سرور به طور مشابه شروع به ارسال داده می‌کند و در انتها با ارسال کلمه end نوبت را جابجا می‌کند. این ارتباط تا جایی که یکی از برنامه ها متوقف شود ادامه می‌یابد.

پس از ایجاد برنامه، سعی کنید با برنامه سایر دانشجویان ارتباط برقرار کنید. برای فهمیدن چگونگی خواندن داده از صفحه کلید و سایر نیازمندی های مربوط به زبان پایتون از اینترنت استفاده کنید.

## ارتباط مبتنی بر UDP

زبان پایتون این امکان را می‌دهد که بتوانید با استفاده از پروتکل UDP به پراکنده کردن داده بپردازید. در مثال پیشرو یک کلاینت UDP برای ارسال پیام‌های همه‌پخشی ایجاد شده است. این کلاینت به آدرس 0.0.0.0 و پورت 44444 بسته شده و از آن برای ارسال و دریافت بسته‌های UDP استفاده می‌کند.

```
import socket
import time

server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
server.settimeout(0.2)
server.bind(("", 44444))
message = b"your very important message"
while True:
    server.sendto(message, ('<broadcast>', 37020))
    print("message sent!")
    time.sleep(1)
```

کلاینت دوم در ادامه به آدرس 0.0.0.0 و پورت 37020 بسته شده و داده‌ها را دریافت می‌کند.

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP
client.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)
client.bind(("", 37020))
while True:
    data, addr = client.recvfrom(1024)
    print("received message:",data,addr)
```

تفاوت اصلی میان ارتباط TCP و UDP در چگونگی ساخت سوکت و توابع ارسال و دریافت اطلاعات است. البته از درس به خاطر دارید که در ارتباط UDP یک اتصال شکل نمی‌گیرد و داده‌ها در قالب پیام به دست برنامه‌ی کاربردی می‌رسند. این پیام‌ها می‌توانند از ترتیب اولیه خود خارج شده باشند یا اینکه دچار خطا باشند.

**موفق باشید.**

*نسخه‌ی اولیه این دستورکار در بهار ۱۳۹۸ توسط سپهر صبور حاضر شده است. نسخه‌ی حاضر تنها شامل بهبودهایی اندک نسبت به نسخه‌ی اصلی می‌باشد.*