**آموزش سوکت نویسی با پایتون Socket Programming در پایتون**

از زمان اختراع تلفن توسط گراهام بل، این وسیله یک ارتباط دهنده‌ی مهم بین انسان‌ها بوده است؛ چرا که می‌توانند با استفاده از آن از راه دور و نزدیک با یکدیگر ارتباط برقرار کنند. سوکت نیز در شبکه‌، نقشی همانند تلفن ایفا می‌کند و برقراری ارتباط بین یک یا چند رایانه را در شبکه‌های رایانه‌ای مقدور می‌سازد. از این رو، یادگیری برنامه نویسی سوکت اهمیت بسیاری دارد. در پایتون نیز با استفاده از کتابخانه‌ی سوکت، امکان برنامه نویسی سوکت برای توسعه‌دهندگان فراهم شده است. در این مقاله، شما با **برنامه نویسی سوکت در پایتون** آشنا خواهید شد، در ادامه مقاله سوکت پروگرمینگ در پایتون همراه ما باشید.

**سوکت چیست؟**

کانال ارتباطی برنامه که اطلاعات برنامه از طریق آن ارسال و دریافت می‌شود، سوکت نام دارد. در ادامه با چند مورد از اصطلاحات شبکه به صورت خلاصه آشنا می‌شویم.

**آدرس** **ip**

هر رایانه و ماشین متصل به شبکه، دارای یک آدرس ip منحصر به فرد است. از طریق آدرس ip یک ماشین یا رایانه، ماشین‌های دیگر در شبکه می‌توانند به آن متصل شوند.

آدرس IP دارای دو نوع IPv4 و IPv6 است. IPv4 یا IP نسخه‌ی 4، از چهار عدد ده دهی تشکیل شده که از 0 تا 255 مقداردهی و با نقطه از هم جدا می‌شوند. مثل 127.0.0.1 که IP استاندارد لوکال ‌هاست است. اما ارقام این نوع IP روزی محدود خواهد شد، به همین جهت IPv6 ساخته شده است.

IPv6 یا IP نسخه‌ی 6، نوع جدیدتری از IP است که شامل اعداد و حروف استاندارد هگزا دسیمال است و بخش‌های آن با کاراکتر “:” از هم جدا شده‌اند. این نوع IP از فضای آدرس‌دهی 128 بیتی استفاده می‌کند که می‌تواند 2 به توان 128 آدرس بی‌همتا را برای ما ایجاد کند.

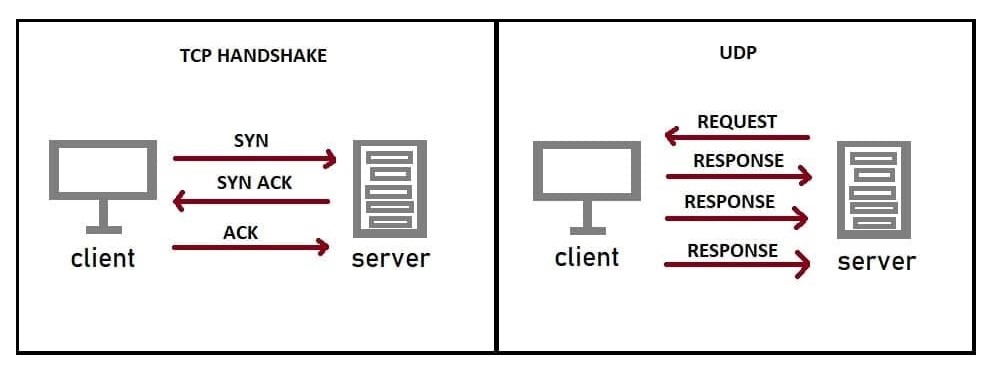
**پورت**

هر IP دارای 65535 پورت (port) است که امکان اتصال ماشین به همین تعداد برنامه و ماشین دیگر را می‌دهد. پورت‌ها شامل دو گروه رزرو شده و رزرو نشده هستند که از پورت‌های 1 تا 1024 متعلق به گروه رزرو شده و مابقی پورت‌ها رزرو نشده هستند و از این دسته می‌توان برای ارتباطات مورد نیاز استفاده کرد.

**پروتکل**

قراداد یا پروتکل (Protocol)، قوانین و قراردادهایی هستند که تعیین می‌کنند، برنامه‌ها و ماشین‌های مختلف چگونه با هم ارتباط برقرار کنند. در واقع، پروتکل‌های شبکه، ماشین‌ها را قادر می‌سازند تا از طریق یک زبان مشترک با یکدیگر صحبت کرده و تبادل اطلاعات کنند.

شبکه دارای چند نوع پروتکل است که از بین آن‌ها TCP و UDP بیش‌ترین مورد استفاده را دارند.



**پروتکل** **TCP**

پروتکل TCP مخفف (Transmissin Control Protocol) به معنای پروتکل کنترل انتقال است و در جایی استفاده می‌شود که لازم باشد انتقال داده‌ها با امنیت بالا، به ترتیب و بی‌عیب و نقص انجام گیرد. در این پروتکل خطاها بررسی شده و اگر داده‌ای دریافت نشده باشد، دوباره ارسال می‌گردد. مانند دانلود فایل در اینترنت.

**پروتکل** **UDP**

پروتکل UDP مخفف (User Datagram Protocol) به معنای پروتکل بسته‌داده‌ی کاربر است. این نوع پروتکل برای کاربردهایی استفاده می‌شود که نیاز به اتصال ندارند و داده‌های از دست رفته، دوباره ارسال نمی‌شوند. مانند پخش فایل تصویری در اینترنت.

اکنون که با برخی اصطلاحات شبکه آشنا شدید، به اصل مطلب می‌پردازیم. سوکت ترکیبی از آدرس IP و port است، که جهت ارسال و دریافت اطلاعات، کانالی در شبکه ایجاد می‌کند و داده‌ها از طریق آن در شبکه رد و بدل می‌شوند.

در مثال تلفن، سوکت نقش گوشی تلفن را دارد که به وسیله‌ی آن دو طرف با هم ارتباط برقرار می‌کنند. کد کشور و شهر، نقش آدرس IP و شماره‌ی منزل نقش port را دارد که با آن‌ها محل دقیق برای ارتباط تعیین می‌شود. اکنون برای برقرار ماندن تماس، لازم است هر دو طرف با یک زبان با یکدیگر صحبت کنند، که زبان مشترک نقش پروتکل را دارد.

**انواع** **سوکت‌ها**

سوکت‌ها دارای انواع مختلفی هستند اما دو نوع از سوکت‌ها بیش‌ترین استفاده را دارند که بر اساس پروتکل‌های TCP و UDP دسته‌بندی شده‌اند. در ادامه به معرفی مختصر آن‌ها می‌پردازیم.

**سوکت‌های** **استریم** (**STREAM**)  
اساس کار این سوکت‌ها بر پایه‌ی پروتکل TCP طراحی شده و با آن کار می‌کنند. در این سوکت‌ها باید قبل از جابه‌جایی اطلاعات، یک اتصال امن و قدرتمند ایجاد شده تا داده‌ها با نظم و دقت، ارسال و دریافت شوند. در ارتباطات با پروتکل‌های HTTP ،FTP و SMTP از این نوع سوکت‌ها استفاده می‌شود.

**سوکت‌های** **دیتاگرام** (**DATAGRAM**)  
این نوع سوکت‌ها براساس پروتکل UDP کار می‌کند و هیچ اتصالی از قبل برای جابه‌جایی داده‌ها لازم ندارد. در این سوکت‌ها، انتقال صحیح و کامل داده‌ها اهمیت ندارد و هیچ وقت رسیدن داده به مقصد بررسی نمی‌شود. مهم‌ترین دلیل استفاده از این سوکت‌ها، سرعت بالای انتقال داده‌هاست و بیش‌تر در انتقال صوت و تصویر استفاده می‌شود.

**برنامه نویسی سوکت با پایتون**  
برنامه نویسی سوکت در پایتون (Socket Programming) با استفاده از کتابخانه‌ی استاندارد سوکت (socket) امکان‌پذیر است. در این قسمت با زبانی نسبتا ساده و ذکر یک مثال، این کتابخانه را به شما معرفی می‌کنیم.  
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. ما در این آموزش، 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 tcpSocket تعیین کرده‌ایم. اگر با مباحث شی و شی‌گرایی آشنایی ندارید

.

import socket

tcpSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

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:

tcpSocket.bind(('127.0.0.1', 5230))

در این مثال، IP استاندارد لوکال‌ هاست 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 1 تا 65000 استفاده کنیم، اما بهتر است از پورت 1 تا 1024 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.

tcpSocket.listen(5)

همان‌طور که مشاهده می‌کنید، ما مقدار 5 را به متد ()listen داده‌ایم که این عدد تعداد درخواست همزمان اتصال کلاینت‌ها را تعیین می‌کند و از 0 تا 5 قابل مقداردهی است.

در ادامه برای آن که سرور ما تا بی‌نهایت پاسخگو باشد، یک حلقه‌ی تکرار while را در حالت True نوشته و مابقی کد خود را در آن می‌نویسیم. 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 client و addr را برگزیدیم) تعریف کرده و سپس سوکت ایجاد شده را ()accept می‌کنیم.

while True:

client, addr = tcpSocket.accept()

در قطعه کد بالا، مشخصات کلاینت متصل شده در client ذخیره شده و در addr یک تاپل (Tuple) که شامل IP و port کلاینت متصل شده است، ذخیره می‌شود. می‌توان به جای متغیر addr یک تاپل که شامل دو متغیر IP و port می‌شود را نیز جایگزین کرد که در هر صورت به یک جواب خواهیم رسید. به این صورت:

while True:

client, (ip, port) = tcpSocket.accept()

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

while True:

client, addr = tcpSocket.accept()

print(“Received connection from”, addr)

حال لازم است پیغامی به کلاینتی که متصل به سرور شده است، ارسال کنیم. برای این کار از متد ()send استفاده می‌کنیم. به این صورت:

while True:

client, addr = tcpSocket.accept()

print(“Received connection from”, addr)

client.send("thank you for connecting")

به احتمال زیاد پیغام ارسالی ما به کلاینت با خطا مواجه می‌شود، چرا که متد ()send قابلیت ارسال رشته را ندارد و باید آن را به صورت بایت ارسال کنیم. برای این کار حرف b را قبل از رشته‌ی خود می‌گذاریم. با این شیوه پیغام بدون خطا برای کلاینت متصل، ارسال می‌شود. به این شکل:

client.send(b"thank you for connecting")

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

while True:

client, addr = tcpSocket.accept()

print(“Received connection from”, addr)

client.send("thank you for connecting")

client.close()

در زیر می‌توانید، کد تکمیل شده‌ی قسمت سرور ما را ببینید.

import socket

ip = '127.0.0.1'

port = 5230

tcpSocket = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

tcpSocket.bind((ip , port))

tcpSocket.listen(5)

while True:

client, addr = tcpSocket.accept()

print("Got connection from", addr)

client.send(b"thank you for connecting")

client.close()

همان‌طور که مشاهده می‌کنید، برای اینکه کد ما تمیزتر شود و تغییرات بعدی راحت‌تر انجام گیرد، شماره‌ی IP و port را ابتدا در دو متغیر به همین نام‌ها قرار داده و متغیرهایمان را در متد ()bind جایگزین کرده‌ایم.  
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:

import socket

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

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:

s.connect(("127.0.0.1" , 5230))

در ادامه برای دریافت پیامی که از سمت سرور ارسال می‌شود، یک متغیر به نام data تعریف کرده و با استفاده از متد ()recv پیام سرور را دریافت می‌کنیم. متد ()recv یک مقدار می‌گیرد که مربوط به اندازه‌ی پیامی می‌شود، که می‌خواهیم دریافت کنیم. ما در اینجا مقدار 1024 را به صورت زیر وارد می‌کنیم.

data = s.recv(1024)

حال متغیر data را با دستور print، چاپ می‌کنیم و سپس سوکت خود را با متد ()close می‌بندیم. به این صورت:

data = s.recv(1024)

print(data)

s.close()

کد سمت کلاینت ما نیز به همین سادگی نوشته شد. کد کامل شده‌ی آن را در زیر می‌بینید:

import socket

ip = "127.0.0.1"

port = 5230

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

s.connect((ip , port))

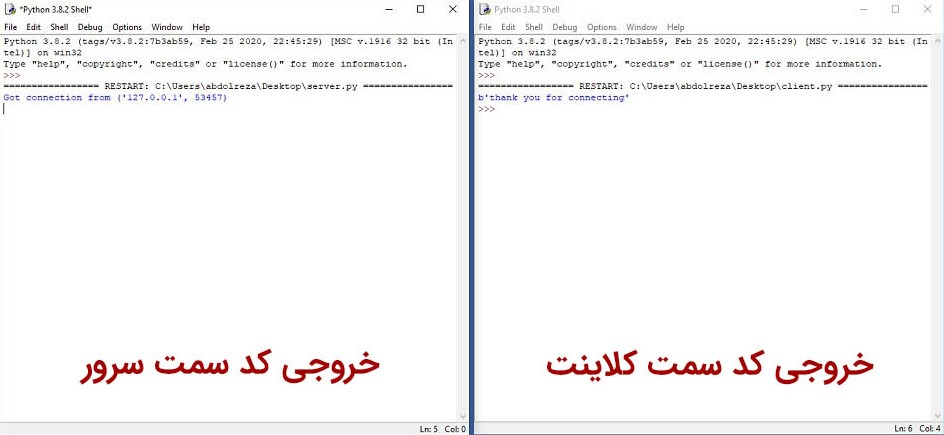
data = s.recv(1024)

print(data)

s.close()

همان طور که ملاحظه می‌کنید، همانند کد سمت سرور، در اینجا نیز ما برای تمیزی و خوانایی کد، شماره IP و port خود را در متغیرهایی به همین نام اختصاص داده‌ایم و متغیرها را در متد ()connect جایگزین کرده‌ایم.

**اجرای کد سمت سرور و کلاینت**  
کدها را جداگانه اجرا می‌کنیم و اگر تمامی مراحل را به درستی انجام دهید خروجی هر دو کد شبیه به عکس زیر خواهد بود.



ما در مقاله سوکت پروگرمینگ در پایتون، نحوه‌ی برنامه‌نویسی سوکت در پایتون، با سوکت‌ نوع استریم و پروتکل TCP را به سادگی توضیح دادیم. اگر علاقه‌مند به این مبحث هستید، می‌توانید به صورت حرفه‌ای برنامه نویسی سوکت در پایتون را ادامه دهید.

**جمع بندی**

در مقاله سوکت پروگرمینگ در پایتون ابتدا شما با آدرس IP، پورت، پروتکل و انواع پروتکل از جمله TCP و UDP آشنا شدید. سپس به سراغ سوکت رفتیم، آن را با تلفن مقایسه کرده و انواع سوکت‌ها نظیر سوکت STREAM و سوکت Datagram را معرفی کردیم. در ادامه به برنامه نویسی سوکت در پایتون پرداختیم و با نوشتن یک برنامه‌ی ساده‌ طبق پروتکل TCP، نحوه‌ی برنامه نویسی سوکت با کتابخانه‌ی socket پایتون را برای شما توضیح دادیم. برنامه نویسی سوکت یکی از کاربردی‌ترین مباحث در شبکه است و در بازار کار تقاضای قابل توجهی برای آن وجود دارد. لذا پیشنهاد می‌کنیم به صورت جدی به یادگیری این مباحث بپردازید.