

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI
Khoa Điện tử



GIÁO TRÌNH THỰC HÀNH
LẬP TRÌNH MẠNG VÀ TRUYỀN THÔNG
(FE6072)

Nhóm biên soạn: ThS. Nguyễn Văn Cường
TS. Tống Văn Luyện

Hà Nội – 2023

MỤC LỤC

MỤC LỤC	2
DANH MỤC HÌNH VẼ	3
CÁC THUẬT NGỮ VIẾT TẮT.....	4
MỞ ĐẦU	5
LẬP TRÌNH SOCKET VỚI UDP VÀ TCP	6
LẬP TRÌNH MẠNG VỚI WEB VÀ HTTP	12
LẬP TRÌNH MẠNG VỚI HỆ THỐNG PHÂN GIẢI TÊN MIỀN	16
LẬP TRÌNH MẠNG VỚI THƯ ĐIỆN TỬ	20
LẬP TRÌNH MẠNG VỚI TRUYỀN NHẬN TẬP TIN	26
PHỤ LỤC	30

DANH MỤC HÌNH VẼ

Hình 1: Tập tin cài đặt PyCharm và thư viện.	30
Hình 2: Tập tin sau khi giải nén	30
Hình 3: Màn hình sau khi khởi động PyCharm.	31
Hình 4: Chọn Customize và Import Settings.	32
Hình 5: Chọn tập tin để nhập thông tin cài đặt vào PyCharm	32
Hình 6: Thông tin cài đặt được nhập vào PyCharm.	33
Hình 7: Nhập thông tin cài đặt và khởi động lại PyCharm.	33
Hình 8: Màn hình sau khi khởi động PyCharm.	35
Hình 9: Tạo dự án mới với tên là pythonProject.	35
Hình 10: Chọn cấu hình trình thông dịch có sẵn.	36
Hình 11: Chọn môi trường Conda và trình thông dịch.	36
Hình 12: Chọn đường dẫn đến trình thông dịch NetworkProgramming.	37
Hình 13: Chọn xong đường dẫn đến trình thông dịch NetworkProgramming.	37
Hình 14: Tạo dự án pythonProject với trình thông dịch NetworkProgramming.	38
Hình 15: Tạo tập tin .py để lập trình ngôn ngữ Python.	38
Hình 16: Đặt tên tin .py cho dự án.	39
Hình 17: Chạy tập tin chương trình đã tạo.	39
Hình 18: Kết quả trả về sau khi thực thi chương trình.	40
Hình 19: Giao diện của PyCharm sau khi khởi động.	40
Hình 20: Chọn đường dẫn đến dự án có sẵn.	41
Hình 21: Chạy dự án đã được mở.	41
Hình 22: Thêm dự án mới vào cửa sổ làm việc hiện tại.	42
Hình 23: Chọn dự án cần thêm.	43
Hình 24: Chạy dự án đã được thêm vào.	43
Hình 25: Đặt breakpoint để debug chương trình.	45
Hình 26: Hộp thoại Breakpoint.	46
Hình 27: Chọn chế độ Debug tại cửa sổ chương trình	46
Hình 28: Cửa sổ Debug	47

CÁC THUẬT NGỮ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
UDP	User Datagram Protocol	Giao thức dữ liệu người dùng
TCP	Tranmission Control Protocol	Giao thức kiểm soát đường truyền
HTTP	HyperText Transfer Protocol	Giao thức truyền siêu văn bản
DNS	Domain Name System	Hệ thống phân giải tên miền
SMTP	Simple Mail Transfer Protocol	Giao thức truyền thư đơn giản
POP	Post Office Protocol	Giao thức bưu điện
IMAP	Internet Message Access Protocol	Giao thức truy nhập thư Internet
FTP	File Transfer Protocol	Giao thức truyền tập tin

MỞ ĐẦU

Tài liệu này nhằm phục vụ sinh viên trong quá trình thực hành học phần **Lập trình mạng và truyền thông (FE6072)**. Tài liệu được chia thành 5 bài thực hành, mỗi bài gồm phần Mục tiêu, Gợi ý, Bài tập mẫu và Bài luyện tập. Phần Mục tiêu chỉ ra mục tiêu của bài thực hành. Phần Gợi ý đưa ra một số đặc điểm và hàm (phương thức) điển hình được sử dụng trong bài. Phần Bài tập mẫu giới thiệu các bài toán điển hình tương ứng với từng chủ đề kèm với chương trình mẫu. Phần Bài luyện tập giúp sinh viên rèn luyện tư duy lập trình nói chung cũng như kỹ năng lập trình mạng nói riêng để áp dụng vào bài toán ứng dụng trong thực tế.

Để thực hiện được các bài thực hành trong tài liệu này, máy tính có thể sử dụng PyCharm IDE cùng với công cụ quản lý môi trường Anaconda. Các công cụ được hướng dẫn cài đặt trong phần Phụ lục A.

BÀI THỰC HÀNH SỐ 1

LẬP TRÌNH SOCKET VỚI UDP VÀ TCP

Mục tiêu

Bài thực hành này giúp sinh viên có thể:

- Sử dụng thành thạo công cụ lập trình PyCharm IDE và trau dồi kỹ năng Debug trong lập trình nói chung
- Viết được chương trình minh họa quá trình truyền nhận dữ liệu giữa máy chủ và máy khách dựa trên lập trình Socket
- Nắm bắt được cách hoạt động của các giao thức UDP và TCP
- Vận dụng thành thạo kỹ năng lập trình Socket để giải quyết các bài toán liên quan đến ứng dụng mạng trong thực tế

Một số đặc điểm và hàm (phương thức) điển hình được sử dụng trong bài

Cách nhập thư viện	
<pre>import socket c1 = socket.socket()</pre>	<ul style="list-style-type: none">- Nhập thư viện vào chương trình- Tạo đối tượng socket
<pre>from socket import socket c2 = socket()</pre>	
<pre>from socket import * c3 = socket()</pre>	
Phương thức của đối tượng socket	
<pre>import socket maxBytes = 64 sock = socket.socket() sock.getsockname() sock.getpeername()</pre>	<ul style="list-style-type: none">- <i>maxBytes</i>: kích thước gửi nhận tối đa- <i>getsockname()</i>: trả về địa chỉ của socket hiện tại- <i>getpeername()</i>: trả về địa chỉ của socket được kết nối với chính nó
Lập trình socket với UDP	
<pre>sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) sock.bind() message, address = sock.recvfrom(maxBytes) sock.sendto(message, address)</pre>	<ul style="list-style-type: none">- <i>AF_INET</i>: địa chỉ IPv4- <i>SOCK_DGRAM</i>: sử dụng UDP- <i>sock.bind()</i>: gán socket tới địa chỉ IP và cổng dịch vụ- <i>recvfrom()</i>: nhận dữ liệu- <i>sendto()</i>: gửi dữ liệu đến địa chỉ chỉ định
Lập trình socket với TCP	
<pre>sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) sock.bind() sock.listen() sock.connect() connectionSocket, address = sock.accept()</pre>	<ul style="list-style-type: none">- <i>SOCK_STREAM</i>: sử dụng TCP- <i>sock.listen(n)</i>: lắng nghe n kết nối đến- <i>sock.connect()</i>: kết nối socket tới địa chỉ chỉ định- <i>sock.accept()</i>: chấp nhận kết nối- <i>send()</i>: gửi dữ liệu- <i>sendall()</i>: gửi tất cả dữ liệu

<pre>message = connectionSocket.recv(maxBytes) connectionSocket.send(message) connectionSocket.sendall(message) connectionSocket.close()</pre>	<ul style="list-style-type: none"> - <i>close()</i>: Ngắt kết nối
Các hàm bổ sung	
<p>Cách mở tập tin 1</p> <pre>filename = "Hello.html" f = open(filename) outputdata = f.read() f.close()</pre>	<p>Cách mở tập tin 1</p> <pre>filename = "Hello.html" with open(filename) as f: outputdata = f.read()</pre>
<pre>from datetime import datetime sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) print(datetime.now()) sock.settimeout() sock.gettimeout() sock.setsockopt() sock.getsockopt() raise RuntimeError raise IOError</pre>	<ul style="list-style-type: none"> - <i>settimeout()</i>: đặt thời gian chờ - <i>gettimeout()</i>: trả về thời gian chờ - <i>setsockopt()</i>: đặt tùy chọn socket - <i>getsockopt()</i>: nhận tùy chọn socket

Bài tập mẫu

Bài 1. Viết chương trình Server và Client chạy trên localhost sử dụng giao thức truyền vận UDP để truyền dữ liệu cho nhau. Yêu cầu:

- Tạo và kết nối các socket với nhau
- Tại một thời điểm, mỗi socket gửi nhận tối đa 4096 bytes

Server:

- In ra thời gian và độ dài của **dữ liệu** nhận được từ Client
- Viết hoa **dữ liệu** văn bản trong tập tin và gửi lại Client

Client:

- Gửi **dữ liệu** cho Server
- In ra **dữ liệu** đã viết hoa nhận được từ Server

Dữ liệu là đoạn văn bản sau: “Lập trình mạng”.

Server	Client
<pre>import socket from datetime import datetime serverIP = "127.0.0.2" serverPort = 10000 maxBytes = 4096 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) sock.bind((serverIP, serverPort)) while True: message, clientAddress = sock.recvfrom(maxBytes) print("Thời điểm:", datetime.now(), "nhận được dữ liệu có độ dài:", len(message), "bytes") message = message.decode() modifiedMessage = message.upper() sock.sendto(modifiedMessage.encode(), clientAddress) pass</pre>	<pre>import socket serverIP = "127.0.0.2" serverPort = 10000 maxBytes = 4096 sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) data = 'Lập trình mạng' sock.sendto(data.encode(), (serverIP, serverPort)) modifiedMessage, serverAddress = sock.recvfrom(maxBytes) sock.close() modifiedMessage = modifiedMessage.decode() print(modifiedMessage)</pre>
Màn hình hiển thị của	
Server	Client
<p>Thời điểm: 2023-02-22 19:15:15.773131 nhận được dữ liệu có độ dài: 19 bytes</p>	<p>LẬP TRÌNH MẠNG</p>

Bài 2. Viết chương trình Server và Client chạy trên localhost sử dụng giao thức truyền vận TCP để truyền dữ liệu cho nhau. Yêu cầu:

- Tạo và kết nối các socket với nhau
- Tại một thời điểm, mỗi socket gửi nhận tối đa 4096 bytes

Server:

- In ra thời gian và độ dài của **dữ liệu** nhận được từ Client
- Viết hoa **dữ liệu** văn bản trong tập tin và gửi lại Client

Client:

- Gửi **dữ liệu** cho Server
- In ra **dữ liệu** đã viết hoa nhận được từ Server

Dữ liệu là đoạn văn bản sau: “Lập trình mạng”.

Server	Client
<pre> import socket from datetime import datetime serverIP = "127.0.0.2" serverPort = 10000 maxBytes = 4096 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) sock.bind((serverIP, serverPort)) sock.listen() while True: connectionSocket, address = sock.accept() message = connectionSocket.recv(maxBytes) print("Thời điểm:", datetime.now(), "nhận được dữ liệu có độ dài:", len(message), "bytes") message = message.decode() modifiedMessage = message.upper() connectionSocket.send(modifiedMessage.encode()) pass </pre>	<pre> import socket serverIP = "127.0.0.2" serverPort = 10000 maxBytes = 4096 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) sock.connect((serverIP, serverPort)) data = 'Lập trình mạng' sock.send(data.encode()) modifiedMessage = sock.recv(maxBytes) sock.close() modifiedMessage = modifiedMessage.decode() print(modifiedMessage </pre>
Màn hình hiển thị của	
Server	Client
Thời điểm: 2023-02-22 19:15:15.773131 nhận được dữ liệu có độ dài: 19 bytes	LẬP TRÌNH MẠNG

Bài luyện tập

Bài 3. Tạo Server và Client chạy trên localhost sử dụng giao thức truyền vận UDP. Yêu cầu đối với từng thiết bị kết nối mạng:

Server:

- In ra địa chỉ IP và Port number của socket Server, Client
- Nhận và in ra thời gian nhận yêu cầu của Client
- Gửi thông tin độ dài bản tin đã nhận được ở trên cho Client

Client:

- In ra địa chỉ IP và Port number của socket Server, Client
- Gửi thời gian hiện tại cho Server
- In ra độ dài bản tin đã nhận được từ Server

Bài 4. Tạo Server và Client chạy trên localhost sử dụng giao thức truyền vận UDP. Yêu cầu đối với từng thiết bị kết nối mạng:

Server:

- In ra địa chỉ IP và Port number của socket Server, Client
- Nhận bản tin từ Client
- Gửi thông tin độ dài bản tin đã nhận được ở trên cho Client với tỉ lệ phản hồi của Server là **60%**.

Client:

- In ra địa chỉ IP và Port number của socket Server, Client
- Gửi thời gian hiện tại cho Server
- Đặt thời gian timeout cho socket Client là 0,1s. Thời gian trễ để nhận gói tin là 2s.
- In ra độ dài bản tin đã nhận được từ Server nếu thời gian nhận gói tin nhỏ hơn thời gian trễ để nhận gói tin, nếu không thì in ra thông tin lỗi nhận gói tin.

Bài 5. Tạo Server và Client chạy trên localhost sử dụng giao thức truyền vận TCP. Yêu cầu đối với từng thiết bị kết nối mạng:

Server:

- In ra địa chỉ IP và Port number của socket Server, Client
- Nhận và in ra thời gian nhận yêu cầu của Client
- Gửi thông tin độ dài bản tin đã nhận được ở trên cho Client

Client:

- In ra địa chỉ IP và Port number của socket Server, Client
- Gửi thời gian hiện tại cho Server
- In ra độ dài bản tin đã nhận được từ Server

Bài 6. Tạo Server và Client chạy trên localhost sử dụng giao thức truyền vận TCP. Tạo tập tin văn bản .txt có chứa nội dung “Lập trình mạng” lưu với tên content.txt. Yêu cầu đối với từng thiết bị kết nối mạng:

Server:

- In ra địa chỉ IP và Port number của socket Server, Client
- Nhận và in ra thời gian nhận yêu cầu của Client
- Đọc dữ liệu trong tập tin content.txt và gửi nội dung trong tập tin này cho Client

Client:

- In ra địa chỉ IP và Port number của socket Server, Client
- Gửi thời gian hiện tại cho Server
- In ra nội dung trong tập tin content.txt mà đã nhận được từ Server

Bài 7. Tạo Server và Client sử dụng giao thức truyền vận TCP. Tạo tập tin văn bản .txt có chứa nội dung “Lập trình mạng” lưu với tên content.txt. Yêu cầu đối với từng thiết bị kết nối mạng:

Server:

- Chạy trên máy tính của các bạn xung quanh
- In ra địa chỉ IP và Port number của socket Server, Client
- Nhận và in ra thời gian nhận yêu cầu của Client
- Đọc dữ liệu trong tập tin content.txt và gửi nội dung trong tập tin này cho Client

Client:

- Kết với Server được tạo trên máy tính của các bạn xung quanh
- In ra địa chỉ IP và Port number của socket Server, Client
- Gửi thời gian hiện tại cho Server
- In ra nội dung trong tập tin content.txt đã nhận được từ Server

Bài 8. Tạo Server và Client sử dụng giao thức truyền vận TCP. Tạo hình ảnh chứa văn bản “Lập trình mạng” bằng phần mềm Paint lưu với tên image.png có độ phân giải 200x50 pixels. Yêu cầu đối với từng thiết bị kết nối mạng:

Server:

- Chạy trên máy tính của các bạn xung quanh
- In ra địa chỉ IP và Port number của socket Server, Client
- Nhận và in ra thời gian nhận yêu cầu của Client
- Đọc dữ liệu trong tập tin image.png và gửi hình ảnh này cho Client

Client:

- Kết với Server được tạo trên máy tính của các bạn xung quanh
- In ra địa chỉ IP và Port number của socket Server, Client
- Gửi thời gian hiện tại cho Server
- Hiện thị hình ảnh image.png đã nhận được từ Server

BÀI THỰC HÀNH SỐ 2

LẬP TRÌNH MẠNG VỚI WEB VÀ HTTP

Mục tiêu

Bài thực hành này giúp sinh viên có thể:

- Viết được chương trình minh họa cách hoạt động của giao thức HTTP cũng như viết chương trình cho máy chủ Web và máy khách dựa trên lập trình Socket
- Nắm bắt được cách hoạt động của giao thức HTTP
- Vận dụng thành thạo kỹ năng lập trình Socket cho máy chủ Web và máy khách để giải quyết các bài toán liên quan đến ứng dụng mạng trong thực tế

Một số đặc điểm và hàm (phương thức) điển hình được sử dụng trong bài

<pre>#http://127.0.0.2:6789/Hello.html requestLine = "GET /Hello.html HTTP/1.1\r\n" statusLine = "HTTP/1.1 200 OK\r\n\r\n" statusLine = "HTTP/1.1 404 Not Found\r\n\r\n" htmlParagraph = "<p>404 Not Found</p>"</pre>	<ul style="list-style-type: none">- <i>requestLine</i>: ví dụ dòng yêu cầu trong bản tin HTTP yêu cầu, bao gồm: Method, Path, HTTP version- <i>statusLine</i>: ví dụ dòng trạng thái phản hồi trong bản tin HTTP phản hồi, bao gồm: HTTP version, status code và reason phrase
---	---

Bài tập mẫu

Bài 1. Viết chương trình Web Server chạy trên localhost để xử lý yêu cầu HTTP của Client. Client hiển thị nội dung trong tập tin “Helloworld.html” chứa văn bản “Lập trình mạng”, và hiển thị thời gian nhận được văn bản và độ dài của văn bản. Socket của Server gửi nhận tối đa 4096 bytes tại một thời điểm. Nếu không có tập tin “Helloworld.html” trong bộ nhớ lưu trữ của Server thì Client hiển thị “Không tìm thấy dữ liệu trong bộ nhớ của Server”. Client là trình duyệt Web.

Server
<pre>import socket from datetime import datetime serverIP = "127.0.0.1" serverPort = 10000 maxBytes = 4096 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM) sock.bind((serverIP, serverPort)) sock.listen() while True: connectionSocket, address = sock.accept()</pre>

```

message = connectionSocket.recv(maxBytes)
try:
    filename = message.decode().split()[1]
    with open(filename[1:], mode='rt', encoding='utf-8') as f:
        data = f.read()
        pass

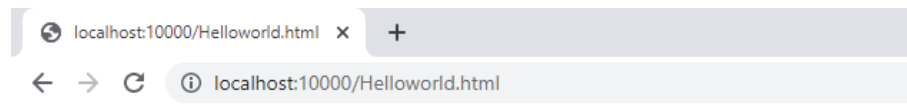
    connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n".encode())
    connectionSocket.send(data.encode())
    msg = f"<p>Thoi diem: {datetime.now()} nhan duoc van ban co do dai:
{len(data)} bytes</p>"
    connectionSocket.send(msg.encode())
    connectionSocket.send("\r\n".encode())
    connectionSocket.close()
    print("Đã gửi thành công")
except:
    connectionSocket.send("HTTP/1.1 404 Not Found\r\n\r\n".encode())
    data = "<p>Khong tim thay du lieu trong bo nho cua Server</p>"
    connectionSocket.send(data.encode())
    connectionSocket.close()
    pass
pass

```

Màn hình hiển thị của Server:
Đã gửi thành công

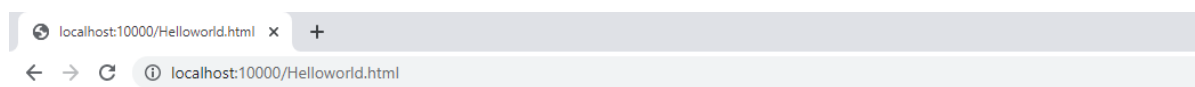
Màn hình hiển thị của Client:

Không có tập tin Helloworld.html:



Khong tim thay du lieu trong bo nho cua Server

Có tập tin Helloworld.html:



Lap trinh mang

Thoi diem: 2023-02-22 19:27:55.851867 nhan duoc van ban co do dai: 21 bytes

Bài luyện tập

Bài 2. Viết chương trình Client yêu cầu Server trả về kinh độ và vĩ độ của Quận Bắc Từ Liêm. Lập trình tại mức lớp ứng dụng, sử dụng thư viện request. Gợi ý: dùng công cụ tìm kiếm Nominatim với dữ liệu của OpenStreetMap có URL:

<https://nominatim.openstreetmap.org/search>.

Bài 3. Viết chương trình Client yêu cầu Server trả về kinh độ và vĩ độ của Quận Bắc Từ Liêm. Lập trình tại mức giao thức, sử dụng thư viện http. client. Gợi ý: dùng công cụ tìm kiếm Nominatim với dữ liệu của OpenStreetMap có URL:

<https://nominatim.openstreetmap.org/search>.

Bài 4. Viết chương trình Client yêu cầu Server trả về kinh độ và vĩ độ của Quận Bắc Từ Liêm. Lập trình tại mức socket, sử dụng thư viện socket. Gợi ý: dùng công cụ tìm kiếm Nominatim với dữ liệu của OpenStreetMap có URL:

<https://nominatim.openstreetmap.org/search>.

Bài 5. Viết chương trình Web Server để xử lý một yêu cầu HTTP tại một thời điểm. Chương trình này chấp nhận và phân tích yêu cầu HTTP -> trích xuất nội dung yêu cầu của Client từ bộ nhớ lưu trữ của Server -> gửi bản tin phản hồi HTTP bao gồm nội dung yêu cầu đến Client. Nếu nội dung yêu cầu không có trong bộ nhớ lưu trữ của Server, thì gửi bản tin phản hồi HTTP “404 Not Found” cho Client. Nội dung HTTP yêu cầu chứa trong tập tin Helloworld.html với nội dung “Hello world! Network programming”. Sử dụng trình duyệt Web (ví dụ: Chrome) làm Client để gửi yêu cầu HTTP, và in ra màn hình nội dung yêu cầu được phản hồi từ Server. (Socket của Web Server sử dụng địa chỉ localhost).

Bài 6. Viết chương trình Web Server để xử lý một yêu cầu HTTP tại một thời điểm. Chương trình này chấp nhận và phân tích yêu cầu HTTP -> trích xuất nội dung yêu cầu của Client từ bộ nhớ lưu trữ của Server -> gửi bản tin phản hồi HTTP bao gồm nội dung yêu cầu đến Client. Nếu nội dung yêu cầu không có trong bộ nhớ lưu trữ của Server, thì gửi bản tin phản hồi HTTP “404 Not Found” cho Client. Nội dung HTTP yêu cầu chứa trong tập tin Helloworld.html với nội dung “Hello world! Network programming”. Viết chương trình cho Client để gửi yêu cầu HTTP, và in ra màn hình nội dung yêu cầu được phản hồi từ Server. (Socket của Web Server sử dụng địa chỉ localhost).

Bài 7. Viết chương trình minh hoạ hoạt động của Proxy Server hay Web Cache. Cách hoạt động của Proxy Server như sau:

- B1. Client thiết lập kết nối TCP tới Proxy Server và gửi yêu cầu HTTP để lấy đối tượng từ Proxy Server.
- B2. Proxy Server kiểm tra xem có bản sao của đối tượng được lưu trữ trong bộ nhớ cục bộ của chúng hay không. Nếu có, Proxy Server trả về đối tượng trong bản tin phản hồi HTTP tới Client.
- B3. Nếu Proxy Server không có bản sao của đối tượng, Proxy Server thiết lập kết nối TCP với Origin Web Server (máy chủ gốc) và gửi yêu cầu HTTP để lấy đối tượng từ Origin Web Server. Sau khi nhận được yêu cầu này, Origin Web Server gửi đối tượng trả về đối tượng trong bản tin phản hồi HTTP tới Proxy Server.
- B4. Khi Proxy Server nhận được đối tượng, nó lưu trữ bản sao trên bộ nhớ cục bộ của chúng và gửi bản sao này trong bản tin phản hồi HTTP tới Client thông qua kết nối TCP hiện tại giữa Client và Proxy Server.

Hãy tạo tập tin Hello.html với nội dung “Hello world! Network programming” trong đường dẫn (nơi lưu trữ) của Origin Web Server. Thử nghiệm 2 kịch bản để kiểm chứng sự chính xác của chương trình: Bộ nhớ cục bộ của Proxy Server (1) **chứa** tập tin Hello.html; (2) **không chứa** tập tin Hello.html. Gợi ý: Viết chương trình cho Origin Web Server; Proxy Server; còn Client là trình duyệt (ví dụ: Chrome).

Bài 8. Tương tự Bài 6, nhưng viết chương trình cho Client thay vì dùng trình duyệt.

Bài 9. Viết chương trình Web Server gửi cho Client (chạy trên trình duyệt) nội dung được lấy từ URL: http://gaia.cs.umass.edu/kurose_ross/about.php. Yêu cầu Client hiển thị nội dung mà Server đã gửi. Socket của Web Server sử dụng địa chỉ localhost.

(Gợi ý: Path name là: `gaia.cs.umass.edu/kurose_ross/about.php`)

BÀI THỰC HÀNH SỐ 3

LẬP TRÌNH MẠNG VỚI HỆ THỐNG PHÂN GIẢI TÊN MIỀN

Mục tiêu

Bài thực hành này giúp sinh viên có thể:

- Viết được chương trình để truy vấn các bản ghi được lưu trữ trên máy chủ DNS
- Nắm bắt được cách hoạt động của hệ thống phân giải tên miền
- Vận dụng thành thạo kỹ năng lập trình mạng để giải quyết các bài toán liên quan đến ứng dụng mạng trong thực tế

Một số đặc điểm và hàm (phương thức) điển hình được sử dụng trong bài

```
import dns.resolver

qtype = ['AAAA', 'A', 'CNAME', 'MX', 'NS']
hostname = "google.com"
answer = dns.resolver.resolve(hostname,
qtype[0], raise_on_no_answer=False)

infolist = socket.getaddrinfo( hostname,
'www', 0, socket.SOCK_STREAM, 0,
socket.AI_ADDRCONFIG | socket.AI_V4MAPPED |
socket.AI_CANONNAME,)
```

- *import dns.resolver*: khai báo thư viện phân giải tên miền
- *qtype*: loại bản ghi DNS
- *resolve()*: truy vấn hostname với loại bản ghi DNS chỉ định
- *getaddrinfo()*: trả về danh sách dữ liệu chứa thông tin về socket

Bài tập mẫu

Bài 1. Viết chương trình in ra các thông tin liên kết với các hostname sau: google.com; microsoft.com. Thông tin này bao gồm (nếu có): Địa chỉ IPv4 và thời gian tồn tại của bản ghi.

```
import dns.resolver

def lookup(hostname):
    qtype = 'A'
    answer = dns.resolver.resolve(hostname, qtype, raise_on_no_answer=False)
    if answer.rrset is not None:
        print(f"Loại bản ghi: {qtype}")
        print(f"Thời gian tồn tại của bản ghi: {answer.rrset.ttl}")
        print(f"Địa chỉ IPv4:")
        for item in answer.rrset.items:
            print(" ", item)
    pass
```



```
hostname = 'dhcnhn.vn'
print(f"Hostname: {hostname}")
lookup(hostname)
```

Kết quả in ra màn hình:

```
Hostname: google.com
Loại bản ghi: A
Địa chỉ IPv4: {<DNS IN A rdata: 142.250.207.78>: None}
Thời gian tồn tại của bản ghi: 208
```

Bài 2. Viết chương trình in ra tên máy chủ thư điện tử (MCTĐT) của các tên miền sau: python.org. Nếu có MCTĐT được trả về, in ra địa chỉ IPv6 ứng với máy chủ đó. Nếu không có MCTĐT thì in ra thông báo không có MCTĐT. Nếu có MCTĐT mà không có địa chỉ IPv6 thì in ra thông báo không có địa chỉ IPv6.

```
import dns.resolver

def lookup(hostname):
    answer = dns.resolver.resolve(hostname, 'MX', raise_on_no_answer=False)
    if answer.rrset is not None:
        for ans in answer.rrset:
            Email_server = ans.exchange.to_text(omit_final_dot=True)
            IPv6 = dns.resolver.resolve(Email_server, 'AAAA',
            raise_on_no_answer=False)
            if IPv6.rrset is not None:
                print(f"Địa chỉ IPv6 tương ứng với MCTĐT {Email_server}:
                {IPv6.rrset[0]}")
            else:
                print(f"Không có địa chỉ IPv6 tương ứng với MCTĐT {Email_server}")
                pass
            pass
        else:
            print(f"Không có MCTĐT tương ứng với hostname {hostname}")
            pass

hostname = 'python.org'
print(f"Hostname: {hostname}")
lookup(hostname)
```

Kết quả in ra màn hình:

```
Hostname: python.org
Địa chỉ IPv6 tương ứng với MCTĐT mail.python.org: 2a03:b0c0:2:d0::71:1
```

Bài luyện tập

Bài 3. Viết chương trình sử dụng `dns.resolver` để in ra bản ghi (records) phản hồi của Server đối với các hostname `fee.hauai.edu.vn`; `python.org`. In ra tất cả bản ghi phản hồi của DNS Server, bao gồm các loại bản ghi sau (nếu có): A, AAAA, CNAME, MX, NS.

Bài 4. Viết chương trình lấy bản ghi của hostname `fee.hauai.edu.vn` (gợi ý: sử dụng phương thức `getaddrinfo`). Hãy in ra các bản ghi của hostname được trả về bao gồm:

- Các địa chỉ IP mà Client có thể kết nối đến.
- Các địa chỉ IP phiên bản 4.
- Các địa chỉ IP mà Client có thể kết nối đến; Các địa chỉ IP phiên bản 4; và Tên chính tắc của máy chủ (Canonical Hostname).

Nếu không có bản ghi nào được trả về, hãy in ra lỗi của Socket (Gợi ý: `socket.error` chứa thông tin lỗi của Socket).

Bài 5. Viết chương trình lấy thông tin như địa chỉ IP, port number của hostname `fee.hauai.edu.vn` (gợi ý: sử dụng phương thức `getaddrinfo`). Hãy tạo Socket với các thông tin vừa lấy được để kết nối với hostname này. In ra trạng thái kết nối: Thành công hoặc Lỗi (Gợi ý: `socket.error` chứa thông tin lỗi của Socket).

Bài 6. Viết chương trình in ra tất cả tên chính tắc (Canonical Name) của máy chủ thư điện tử STMP liên kết với tên miền `gmail.com`. Nếu không có bản ghi nào được trả về, in ra thông báo rằng tên miền này không có tên chính tắc của máy chủ thư điện tử STMP.

Bài 7. Viết chương trình in ra tất cả tên chính tắc của máy chủ thư điện tử SMTP liên kết với tên miền `iana.org`; `sv.dhcnhn.vn`; `google.com`. Nếu không có bản ghi tên chính tắc nào được trả về, in ra thông báo rằng tên miền này không có tên chính tắc của máy chủ thư điện tử SMTP. Nếu có bản ghi chính tắc được trả về, hãy phân dải tên chính tắc của từng bản ghi dùng loại bản ghi A. Sau đó, in ra địa chỉ IPv4 tương ứng với từng tên chính tắc.

Bài 8. Viết chương trình in ra tất cả tên chính tắc của máy chủ thư điện tử SMTP liên kết với tên miền `ibm.com`; `sv.dhcnhn.vn`. Nếu không có bản ghi tên chính tắc nào được trả về, in ra thông báo rằng tên miền này không có tên chính tắc của máy chủ thư điện tử SMTP. Nếu có bản ghi chính tắc được trả về, hãy phân dải tên chính tắc của từng bản ghi dùng các loại bản ghi với mức độ ưu tiên lần lượt như sau: A, AAAA, CNAME. Sau đó, in ra kết quả tương ứng với từng loại bản ghi.

Bài 9. Viết chương trình in ra tất cả tên chính tắc của máy chủ thư điện tử SMTP liên kết với tên miền microsoft.com; egov.hauai.edu.vn. Nếu không có bản ghi tên chính tắc nào được trả về, in ra thông báo rằng tên miền này không có tên chính tắc của máy chủ thư điện tử SMTP. Nếu có bản ghi chính tắc được trả về, hãy phân dải tên chính tắc của từng bản ghi dùng các loại bản ghi với mức độ ưu tiên lần lượt như sau: A, AAAA, CNAME. Sau đó, in ra kết quả tương ứng với từng loại bản ghi.

Lưu ý: Nếu bất kỳ loại bản ghi có mức độ ưu tiên cao hơn đạt được thì dùng chương trình.

Bài 10. Viết chương trình in ra tất cả tên chính tắc của máy chủ thư điện tử SMTP liên kết với tên miền iana.org; fee.hauai.edu.vn.

- A. Nếu không có bản ghi tên chính tắc nào được trả về, hãy phân dải **tên miền** dùng các loại bản ghi với mức độ ưu tiên lần lượt như sau: A, AAAA, CNAME. Hãy in ra kết quả tương ứng với từng loại bản ghi.
- B. Nếu có bản ghi chính tắc được trả về, hãy phân dải **tên chính tắc của từng bản ghi được trả về** dùng các loại bản ghi với mức độ ưu tiên lần lượt như sau: A, AAAA, CNAME. Hãy in ra kết quả tương ứng với từng loại bản ghi.

Lưu ý: Nếu bất kỳ loại bản ghi có mức độ ưu tiên cao hơn đạt được thì dùng chương trình.

BÀI THỰC HÀNH SỐ 4

LẬP TRÌNH MẠNG VỚI THƯ ĐIỆN TỬ

Mục tiêu

Bài thực hành này giúp sinh viên có thể:

- Viết được chương trình gửi thư điện tử tới các địa chỉ email
- Nắm bắt được cách hoạt động của các giao thức SMTP, POP3 và IMAP
- Vận dụng thành thạo kỹ năng lập trình mạng để giải quyết các bài toán liên quan đến ứng dụng mạng trong thực tế

Một số đặc điểm và hàm (phương thức) điển hình được sử dụng trong bài

EMAIL = 'laptrinhmang@outlook.com' DESTINATION_EMAIL = 'laptrinhmang.havi@gmail.com' PASSWORD = 'LTM.DHCNHN.HaUI' mailServer = 'smtp.office365.com' mailPort = 587	- Thông tin tài khoản email được cấp
Thư viện smtplib	
import smtplib connection = smtplib.SMTP(mailServer, mailPort) connection.starttls() connection.login(EMAIL, PASSWORD) connection.sendmail(from_addr=EMAIL, to_addrs=DESTINATION_EMAIL, msg=msg) connection.quit() msg = EmailMessage() msg['To'] = DESTINATION_EMAIL msg['From'] = EMAIL msg['Subject'] = SUBJECT_EMAIL msg.set_content(BODY_EMAIL) msg.add_attachment(data, maintype='text', subtype='plain', filename=f.name)	- <i>starttls()</i> : vào chế độ TLS (Transport Layer Secure) - <i>login()</i> : đăng nhập email - <i>sendmail()</i> : gửi email - <i>quit()</i> : thoát phiên SMTP - <i>set_content()</i> : thiết lập nội dung - <i>add_attachment()</i> : thêm dữ liệu đính kèm
Thư viện poplib	

<pre> import poplib p = poplib.POP3_SSL(mailServer) p.user(EMAIL) p.pass_(PASSWORD) p.quit() status = p.stat() response, listings, octet_count = p.list() </pre>	<ul style="list-style-type: none"> - <i>Stat()</i>: trả về tổng số hòm thư và kích thước tổng của hòm thư - <i>List()</i>: trả về (1) tổng số hòm thư và kích thước tổng của hòm thư trả về; (2) mã số và kích thước thư
Thư viện imapclient	
<pre> import imapclient IMAP_object = imapclient.IMAPClient(mailServer, ssl=True,) IMAP_object.login(EMAIL, PASSWORD) IMAP_object.logout() IMAP_object.list_folders() IMAP_object.capabilities() IMAP_object.select_folder(folder_name, readonly=True) </pre>	

Bài tập mẫu

Bài 1. Viết chương trình gửi thư điện tử tới email sau: laptrinhmang.haui@gmail.com; với nội dung như sau:

Thân gửi bạn An,

Nội dung thư điện tử.

Trân trọng.

Tài khoản để gửi email: laptrinhmang@outlook.com

Mật khẩu: LTM.DHCNHN.HaUI

```
import smtplib
from email.message import EmailMessage

EMAIL = 'laptrinhmang@outlook.com'
PASSWORD = 'LTM.DHCNHN.HaUI'
DESTINATION_EMAIL = 'laptrinhmang.haui@gmail.com'

SUBJECT_EMAIL = "Tiêu đề: Lập trình mạng"
BODY_EMAIL = "Thân gửi bạn An,\n\nNội dung thư điện tử.\n\nTrân trọng."

msg = EmailMessage()
msg['To'] = DESTINATION_EMAIL
msg['From'] = EMAIL
msg['Subject'] = SUBJECT_EMAIL
msg.set_content(BODY_EMAIL)

mailServer = 'smtp.office365.com'
mailPort = 587

connection = smtplib.SMTP(mailServer, mailPort)
connection.starttls()
connection.login(EMAIL, PASSWORD)
connection.send_message(msg=msg, from_addr=EMAIL, to_addrs=DESTINATION_EMAIL)
connection.quit()
```

Kết quả sau khi gửi Email:

Tiêu đề: Lập trình mạng



Lap Trinh Mang <laptrinhmang@outlook.com>

9:30 PM

To: laptrinhmang.haui@gmail.com

Thân gửi bạn An,

Bài ví dụ gửi file đính kèm.

Trân trọng.

Bài 2. Viết chương trình phân tích hòm thư với các yêu cầu như sau:

- In ra kích thước của từng tin nhắn.
- In ra tất cả tên của thư mục trong hòm thư.

Tài khoản để gửi email: laptrinhmang@outlook.com

Mật khẩu: LTM.DHCNHN.HaUI

```
import poplib, imapclient
EMAIL = 'laptrinhmang@outlook.com'
PASSWORD = 'LTM.DHCNHN.HaUI'
mailServer = 'outlook.office365.com'
POP_object = poplib.POP3_SSL(mailServer)
IMAP_object = imapclient.IMAPClient(mailServer, ssl=True,)
try:
    POP_object.user(EMAIL)
    POP_object.pass_(PASSWORD)
    IMAP_object.login(EMAIL, PASSWORD)
except:
    print("Đăng nhập không thành công")
else:
    response, listings, octet_count = POP_object.list()
    if not listings:
        print("Không có hòm thư nào")
    for listing in listings:
        number, size = listing.decode().split()
        print(f"Hòm thư thứ {number} có kích thước {size} bytes")
        pass
    print()
    data = IMAP_object.list_folders()
    for flags, delimiter, folder_name in data:
        print(flags[0].decode(), delimiter.decode(), folder_name)
        pass
finally:
    POP_object.quit()
    IMAP_object.logout()
```

Kết quả in ra màn hình:

Hòm thư thứ 1 có kích thước 174407 bytes
Hòm thư thứ 2 có kích thước 55293 bytes
...
Hòm thư thứ 44 có kích thước 122026 bytes

\HasChildren / Archive
\HasNoChildren / Archive/ABC
\HasNoChildren / Archive/Thu muc con cua Archive
\HasNoChildren / Deleted
\HasNoChildren / Drafts
\Marked / Inbox
\HasNoChildren / Junk
\HasNoChildren / Notes
\HasNoChildren / Outbox
\HasNoChildren / Sent
\HasChildren / Sync Issues
\HasNoChildren / Sync Issues/Conflicts

Bài luyện tập

Bài 3. Viết chương trình mô tả sự tương tác SMTP giữa Client và Server. Client kết nối với, nhận phản hồi từ Server để gửi thư điện tử (e-mail). Client đăng nhập tài khoản Microsoft để gửi đến các máy chủ email khác nhau (ví dụ: @gmail.com; @hau.edu.vn; @icloud.com; @outlook.com). Biết tên máy chủ SMTP của Microsoft là smtp.office365.com được gán với Port 587.

Bài 4. Viết chương trình để gửi email từ tài khoản Microsoft đến các máy chủ email khác nhau (ví dụ: @outlook.com; @gmail.com; @icloud.com; @hau.edu.vn). Biết tên máy chủ SMTP của Microsoft là smtp.office365.com được gán với Port 587. Mở rộng: Gửi đồng thời 4 e-mail trong 1 lần chạy chương trình.

Bài 5. Viết chương trình để gửi email từ tài khoản Microsoft đến các máy chủ email khác nhau (ví dụ: @outlook.com; @gmail.com; @icloud.com; @hau.edu.vn). Biết tên máy chủ SMTP của Microsoft là smtp.office365.com được gán với Port 587. Nội dung e-mail viết bằng Tiếng Việt. Mở rộng: Gửi đồng thời 4 e-mail trong 1 lần chạy chương trình.

Bài 6. Viết chương trình để gửi email từ tài khoản Microsoft đến các máy chủ email khác nhau (ví dụ: @outlook.com; @gmail.com; @icloud.com; @hau.edu.vn). Biết tên máy chủ SMTP của Microsoft là smtp.office365.com được gán với Port 587. Nội dung e-mail viết bằng Tiếng Việt gửi cùng file đính kèm .txt và .rar. Mở rộng: Gửi đồng thời 4 e-mail trong 1 lần chạy chương trình.

Bài 7. Viết chương trình đăng nhập vào tài khoản Microsoft và thống kê trạng thái hòm thư bao gồm số tin nhắn và kích thước hòm thư (đơn vị bytes). Biết tên máy chủ POP của Microsoft là outlook.office365.com.

Bài 8. Viết chương trình đăng nhập vào tài khoản Microsoft và in ra kích thước của từng tin nhắn (đơn vị bytes).

Bài 9. Viết chương trình đăng nhập vào tài khoản Microsoft và in ra thông tin của tất cả tin nhắn, bao gồm: Địa chỉ email gửi, nhận, tiêu đề của email, nội dung của email. Biết tên máy chủ POP của Microsoft là outlook.office365.com.

Bài 10. Viết chương trình đăng nhập vào tài khoản Microsoft và in ra thông tin của tất cả tin nhắn, bao gồm: Địa chỉ email gửi, nhận, tiêu đề của email, nội dung của email có số (message number) là 1. Biết tên máy chủ POP của Microsoft là outlook.office365.com.

Bài 11. Viết chương trình đăng nhập vào tài khoản Microsoft và in ra thông tin của tất cả tin nhắn, bao gồm: Địa chỉ email gửi, nhận, tiêu đề của email, nội dung của email có số (message number) là 2. Sau đó, xóa email có message number là 5. Biết tên máy chủ POP của Microsoft là outlook.office365.com.

Bài 12. Viết chương trình đăng nhập vào tài khoản Microsoft và in ra thông tin trạng thái phản hồi và tất cả tên thư mục, hòm thư trong tài khoản đó sử dụng phương thức tạo đối tượng **IMAP4_SSL** và **IMAPClient**. Biết tên máy chủ IMAP của Microsoft là outlook.office365.com.

Bài 13. Viết chương trình đăng nhập vào tài khoản Microsoft và in ra thông tin tổng hợp của thư mục “Inbox” trong hòm thư của tài khoản. Biết tên máy chủ IMAP của Microsoft là outlook.office365.com.

BÀI THỰC HÀNH SỐ 5

LẬP TRÌNH MẠNG VỚI TRUYỀN NHẬN TẬP TIN

Mục tiêu

Bài thực hành này giúp sinh viên có thể:

- Viết được chương trình truyền nhận tập tin giữa máy khách và máy chủ
- Nắm bắt được cách hoạt động của giao thức FTP
- Vận dụng thành thạo kỹ năng lập trình mạng để giải quyết các bài toán liên quan đến ứng dụng mạng trong thực tế

Một số đặc điểm và hàm (phương thức) điển hình được sử dụng trong bài

Thư viện ftplib	
<pre>import ftplib host = 'ftp.ibiblio.org' ftp = ftplib.FTP(host) ftp.login() ftp.quit() ftp.getwelcome() ftp.pwd() ftp.nlst() ftp.cwd()</pre>	<ul style="list-style-type: none"> - <i>pwd()</i>: trả về đường dẫn hiện tại - <i>nlst()</i>: trả về danh sách thông tin các tập tin trong đường dẫn hiện tại - <i>cwd()</i>: thay đổi đường dẫn
Trích xuất dữ liệu từ máy chủ FTP	
<pre>with open(file, 'w') as f: def writeline(data): f.write(data) # f.write(os.linesep) f.write('\n') pass ftp.retrlines(f"RETR {file}", writeline) pass with open(file, 'wb') as f: ftp.retrbinary(f"RETR {file}", f.write) pass ftp.voidcmd("TYPE I") socket, size = ftp.ntransfercmd(f"RETR {file}") data = socket.recv(2048) ftp.voidresp() entries = [] ftp.dir(entries.append)</pre>	<ul style="list-style-type: none"> - <i>retrlines()</i>: truy xuất dữ liệu ở chế độ dòng - <i>retrbinary()</i>: truy xuất dữ liệu ở chế độ nhị phân - <i>voidcmd()</i>: gửi lệnh và đợi phản hồi bắt đầu với '2' - <i>ntransfercmd()</i>: bắt đầu truyền qua kết nối dữ liệu - <i>voidresp()</i>: đợi phản hồi bắt đầu với '2'

Bài tập mẫu

Bài 1. Viết chương trình trích xuất tập tin từ máy chủ sử dụng giao thức FTP. Biết tên máy chủ là ibiblio.org. Yêu cầu:

- In ra thông báo lời chào của máy chủ FTP/
- In ra đường dẫn hiện tại và sau khi đổi thành /pub/linux/kernel.
- Kết nối tới máy chủ thông qua đường dẫn /pub/linux/kernel để trích xuất tập tin profil.tgz và lưu tập tin vào bộ nhớ cục bộ

```
import ftplib, os

host = 'ftp.ibiblio.org'
filename = 'profil.tgz'

ftp = ftplib.FTP(host)
print("Welcome:", ftp.getwelcome())
ftp.login()

print(f"Đường dẫn hiện tại: {ftp.pwd()}")
ftp.cwd('/pub/linux/kernel')
print(f"Đường dẫn sau khi đổi: {ftp.pwd()}")

if os.path.exists(filename):
    print(f"Ghi đè tập tin {filename}")

with open(filename, 'wb') as f:
    ftp.retrbinary(f"RETR {filename}", f.write)
    pass

ftp.quit()
```

Kết quả in ra màn hình:

Welcome: 220 ProFTPD Server

Đường dẫn hiện tại: /

Đường dẫn sau khi đổi: /pub/linux/kernel

Bài luyện tập

Bài 2. Viết chương trình kết nối với máy chủ sử dụng giao thức FTP. Biết tên máy chủ: ibiblio.org. Yêu cầu:

- In ra thông báo lời chào của máy chủ FTP
- In ra đường dẫn hiện tại và sau khi đổi thành /pub/linux/kernel. Sau đó in ra danh sách các tập tin và thư mục tại đường dẫn sau khi được đổi (sử dụng phương thức `ftp.nlst()`).

Bài 3. Viết chương trình trích xuất tập tin từ máy chủ sử dụng giao thức FTP. Biết tên máy chủ: ibiblio.org. Yêu cầu:

- In ra thông báo lời chào của máy chủ FTP
- In ra đường dẫn hiện tại và sau khi đổi thành /pub/linux/kernel.
- Kết nối tới máy chủ thông qua đường dẫn /pub/linux/kernel để trích xuất tập tin README theo chế độ ASCII và lưu tập tin vào bộ nhớ cục bộ.

Bài 4. Viết chương trình trích xuất tập tin từ máy chủ sử dụng giao thức FTP. Biết tên máy chủ: ibiblio.org. Yêu cầu:

- In ra thông báo lời chào của máy chủ FTP
- In ra đường dẫn hiện tại và sau khi đổi thành /pub/linux/kernel.
- Kết nối tới máy chủ thông qua đường dẫn /pub/linux/kernel để trích xuất tập tin getkernel-1.1.1.tar.gz theo chế độ Binary và lưu tập tin vào bộ nhớ cục bộ.

Bài 5. Viết chương trình trích xuất tập tin từ máy chủ sử dụng giao thức FTP. Biết tên máy chủ: ibiblio.org. Yêu cầu:

- Kết nối tới máy chủ thông qua đường dẫn /pub/linux/kernel để trích xuất tập tin getkernel-1.1.1.tar.gz theo chế độ Binary và lưu tập tin vào bộ nhớ cục bộ.
- Kiểm soát số lượng bytes cho mỗi lần nhận (tối đa 2048 bytes cho 1 lần nhận)
- Sau mỗi lần nhận số bytes tối đa được cho phép, in ra tổng số bytes nhận được và phần trăm trên tổng số bytes đã nhận.

Gợi ý: Sử dụng phương thức `ftp.ntransfercmd()` để nhận socket, sau đó kiểm soát số lượng bytes nhận mỗi lần bằng phương thức `socket.recv()`

Bài 6. Viết chương trình kết nối với máy chủ sử dụng giao thức FTP. Biết tên máy chủ: ibiblio.org. Yêu cầu:

- In ra thông báo lời chào của máy chủ FTP
- In ra đường dẫn hiện tại và sau khi đổi thành /pub/linux/kernel. Sau đó in ra danh sách các tập tin và thư mục tại đường dẫn sau khi được đổi (sử dụng phương thức ftp.dir ())
- In ra ngày chỉnh sửa lần cuối của tập tin thứ 5

Bài 7. Cài đặt Dịch vụ thông tin Internet (IIS: Internet Information Service) và máy chủ FTP trên máy tính cá nhân

- Máy tính đóng vai trò máy chủ FTP chứa thư mục và tập tin
- Viết chương trình trích xuất tập tin từ máy chủ. Yêu cầu:
 - In ra thông báo lời chào của máy chủ FTP/
 - In ra đường dẫn trước và sau khi đổi
 - Kết nối tới máy chủ để trích xuất và lưu tập tin vào bộ nhớ cục bộ

PHỤ LỤC






Phụ lục A: Hướng dẫn cài đặt PyCharm và thư viện

Phụ lục này hướng dẫn cài đặt PyCharm, nhập thông tin cài đặt cho PyCharm, thiết lập môi trường quản lý thư viện, mở dự án Python có sẵn và thêm dự án có sẵn vào cửa sổ làm việc hiện tại của PyCharm.

1. Cài đặt PyCharm

Đầu tiên, tải tất cả các tập tin trong thư mục tại địa chỉ Google Drive sau (các tập tin như trong Hình 1):

https://drive.google.com/drive/folders/14pwQ_v4LHY7vAxyy4GVSOgPNoiYf36bR?usp=sharing

Phan mem va huong dan cai dat			
Tên ↑	Chủ sở hữu	Sửa đổi lần cuối	Kích cỡ tệp
 Huong dan cai dat Pycharm va thu vien.pdf		19:24	1,1 MB
 Miniconda3-latest-Windows-x86_64 Python 3.9.rar		17:31	56,7 MB
 NetworkProgramming.rar		16:19	73,1 MB
 pycharm-community-2021.2.1.rar		17:32	372 MB
 settings_1309.2022.zip		16:17	19 KB

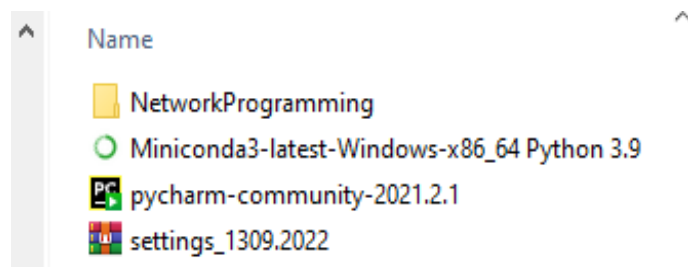
Hình 1: Tập tin cài đặt PyCharm và thư viện.

Tiến hành cài đặt phần mềm theo các bước sau:

B1. Giải nén tất cả tập tin nén, ngoại trừ tập tin *settings_1309.2022.rar*. Sau khi giải nén xong được các tập tin và thư mục như trong Hình 2. **Lưu ý: Thư mục sau khi giải nén không được để trong đường dẫn có ký tự đặc biệt hoặc để Tiếng Việt có dấu.**

B2. Mở tập tin *pycharm-community-2021.2.1.exe* và chọn Next để tiến hành cài đặt.


B3. Mở tập tin *Miniconda3-latest-Windows-x86_64 Python 3.9.exe* và chọn Next để tiến hành cài đặt.



Hình 2: Tập tin sau khi giải nén

2. Nhập thông tin cài đặt của PyCharm

Thông tin cài đặt bao gồm giao diện, phông chữ, ... được nhập vào PyCharm bằng cách thực hiện các bước sau:

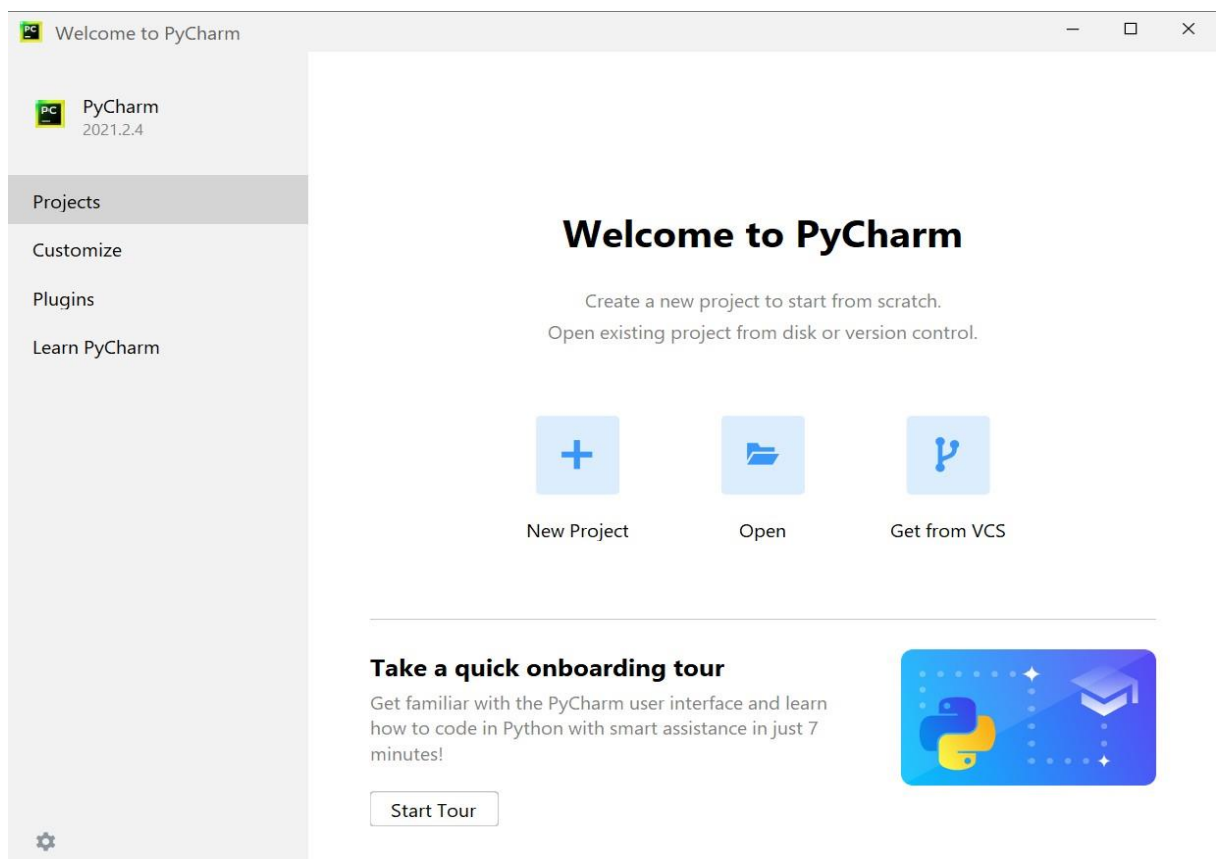
B1. Khởi động phần mềm PyCharm (có biểu tượng ). Giao diện sau khi khởi động lần đầu tiên sẽ hiển thị tương tự như trong Hình 3.

B2. Chọn *Customize* rồi chọn *Import Settings* như Hình 4.

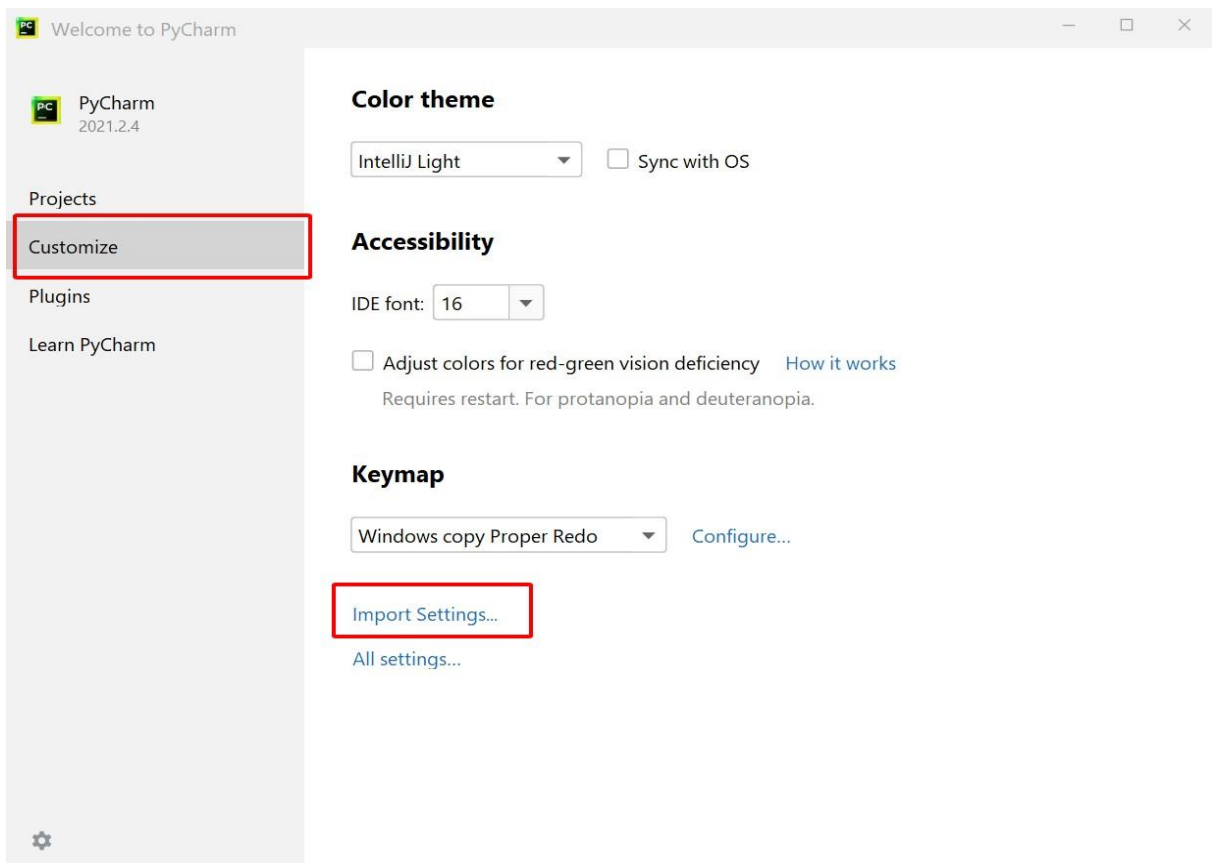
B3. Chọn tập tin *settings_1309.2022.rar* rồi chọn *OK* như Hình 5.

B4. Chọn *Select All* và *OK* như Hình 6.

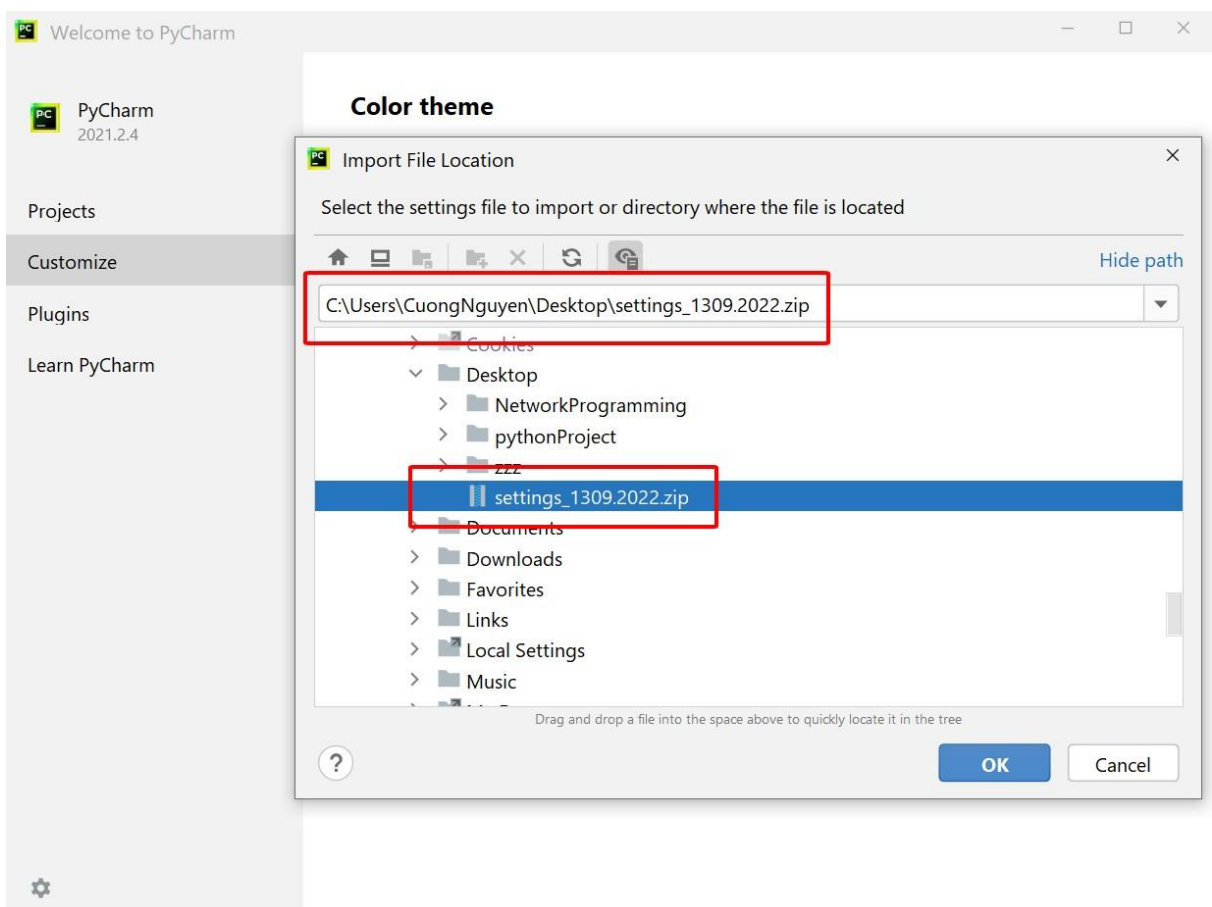
B5. Chọn Import and Restart như Hình 7.



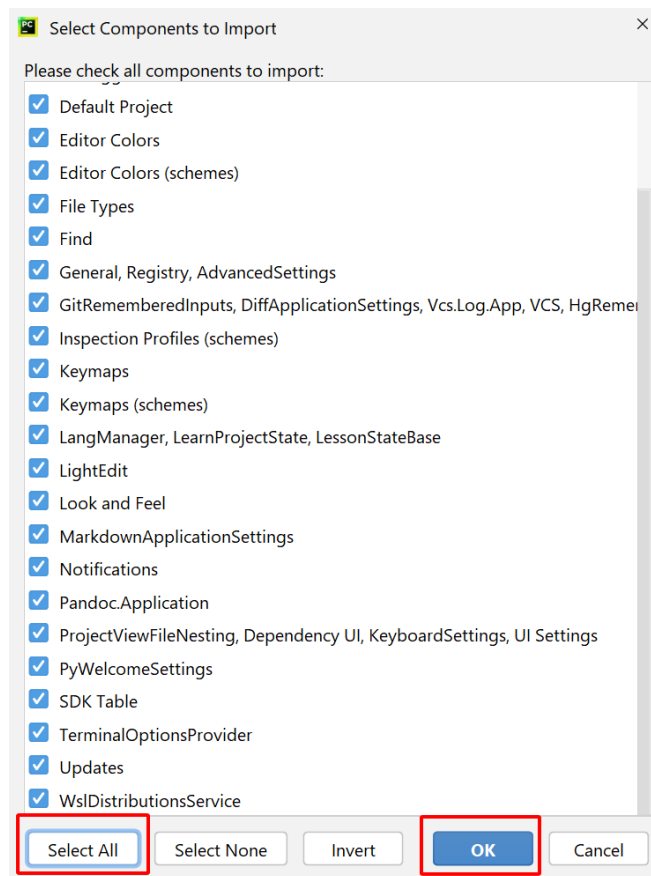
Hình 3: Màn hình sau khi khởi động PyCharm.



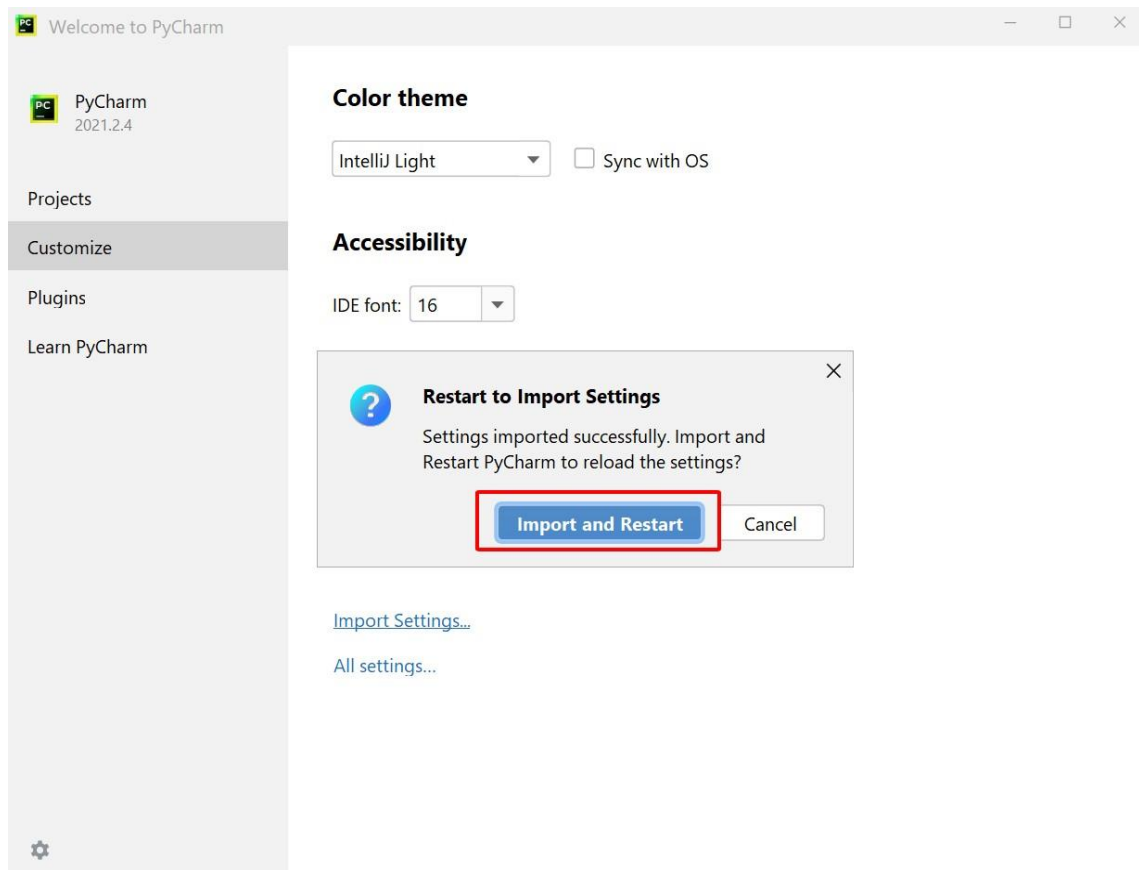
Hình 4: Chọn Customize và Import Settings.



Hình 5: Chọn tập tin để nhập thông tin cài đặt vào PyCharm



Hình 6: Thông tin cài đặt được nhập vào PyCharm.



Hình 7: Nhập thông tin cài đặt và khởi động lại PyCharm.

3. Cài đặt thư viện

Thư mục *NetworkProgramming* đã chứa bộ thư viện được cài đặt để phục vụ cho học phần Lập trình mạng. Tài liệu này hướng dẫn thiết lập thư viện được cài đặt sẵn. Các bước để thiết lập thư viện này như sau:

B1. Khởi động phần mềm PyCharm. Giao diện sau khi khởi động sẽ hiển thị tương tự như trong Hình 8.

B2. Chọn *New Project* để tạo dự án mới.

B3. Đặt tên và vị trí lưu dự án. Theo như ví dụ trong Hình 9, tên dự án là *pythonProject*, được lưu tại đường dẫn *C:\Users\CuongNguyen\Desktop\pythonProject*.

B4. Mở rộng phần **Python Interpreter** sẽ được kết quả tương tự Hình 10. Chọn *Previously configured interpreter (1)* rồi chọn ô vuông ... (2) như trong Hình 10.

B5. Chọn môi trường Conda và trình thông dịch thứ tự 1->3 như trong Hình 11.

B6. Chọn đường dẫn đến trình thông dịch *NetworkProgramming* tương tự như trong Hình 12. Hãy đường dẫn mà thư mục *NetworkProgramming* được giải nén.

Đường dẫn trong ví dụ này là:

C:\Users\CuongNguyen\Desktop\NetworkProgramming\python.exe

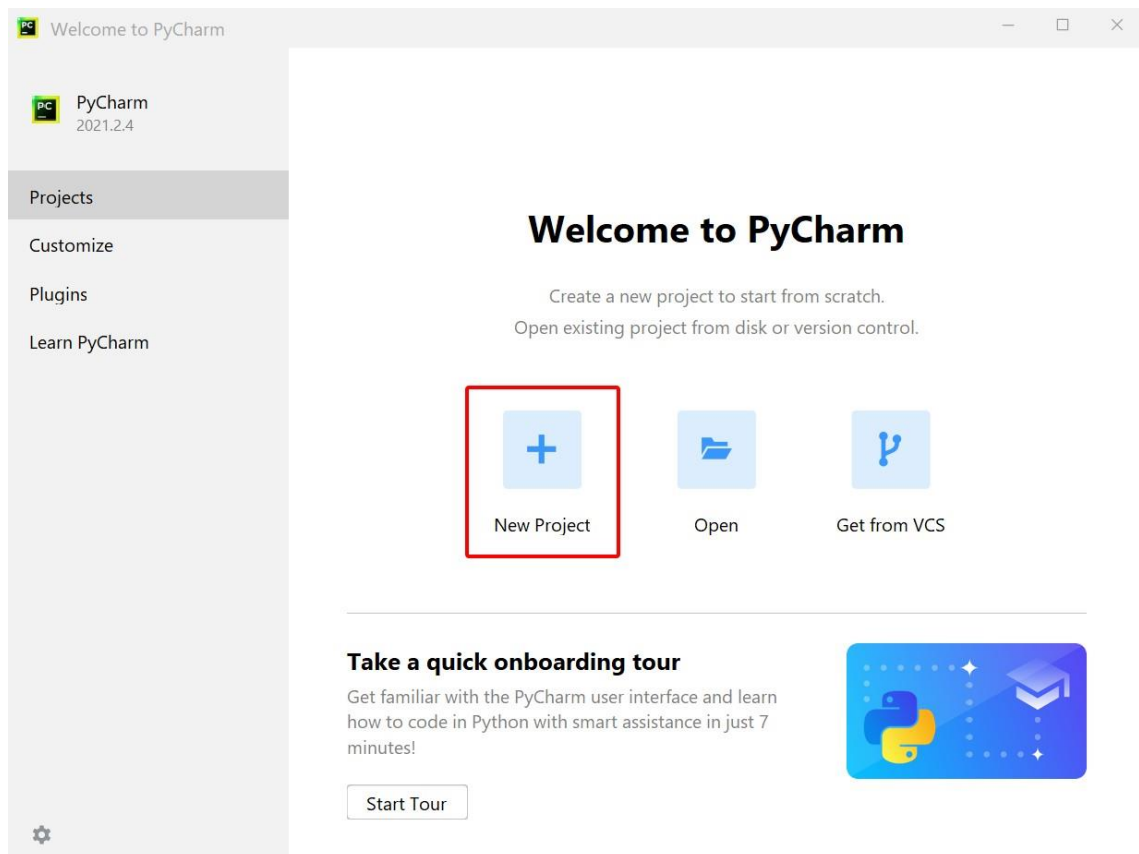
B7. Kết quả sau khi chọn xong đường dẫn đến trình thông dịch *NetworkProgramming* được minh họa như trong Hình 13.

B8. Sau khi thiết lập môi trường xong, tạo dự án mới bằng chọn *Create* như Hình 14.

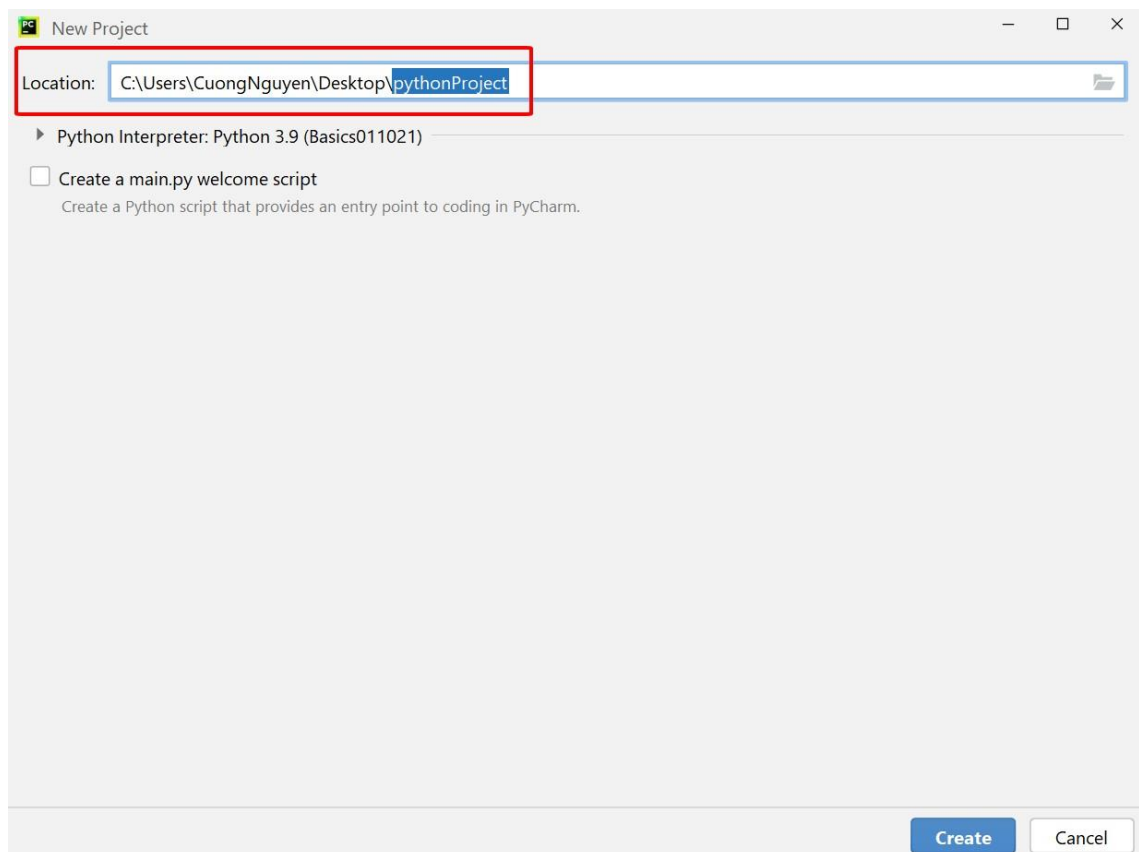
B9. Tạo tập tin và đặt tên tập tin .py để lập trình ngôn ngữ Python như Hình 15 và Hình 16.

B10. Chạy chương trình trong tập tin *HelloWorld.py* bằng cách nhấp chuột phải vào màn hình lập trình, và chọn *Run*.

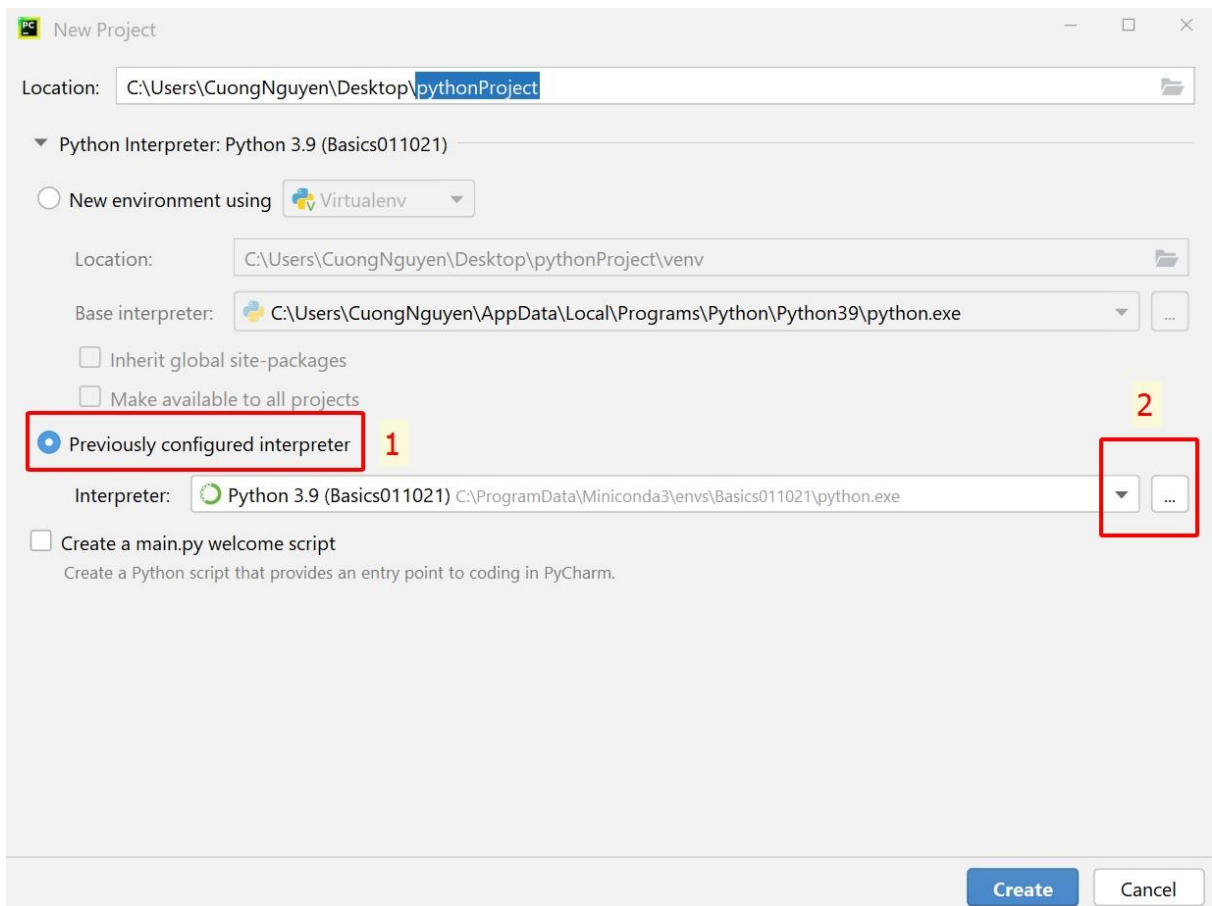
B11. Kết quả trả về sau khi thực thi chương trình được hiển thị tại cửa sổ *Run*.



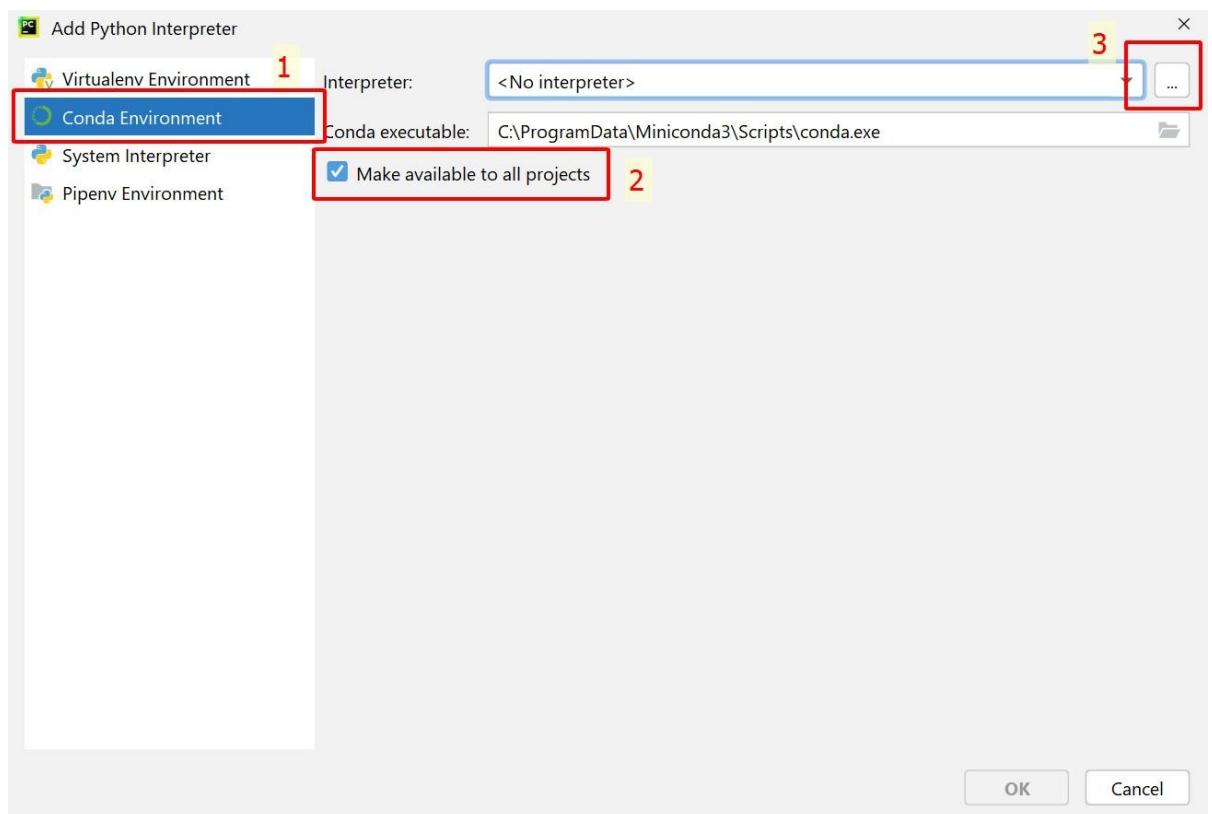
Hình 8: Màn hình sau khi khởi động PyCharm.



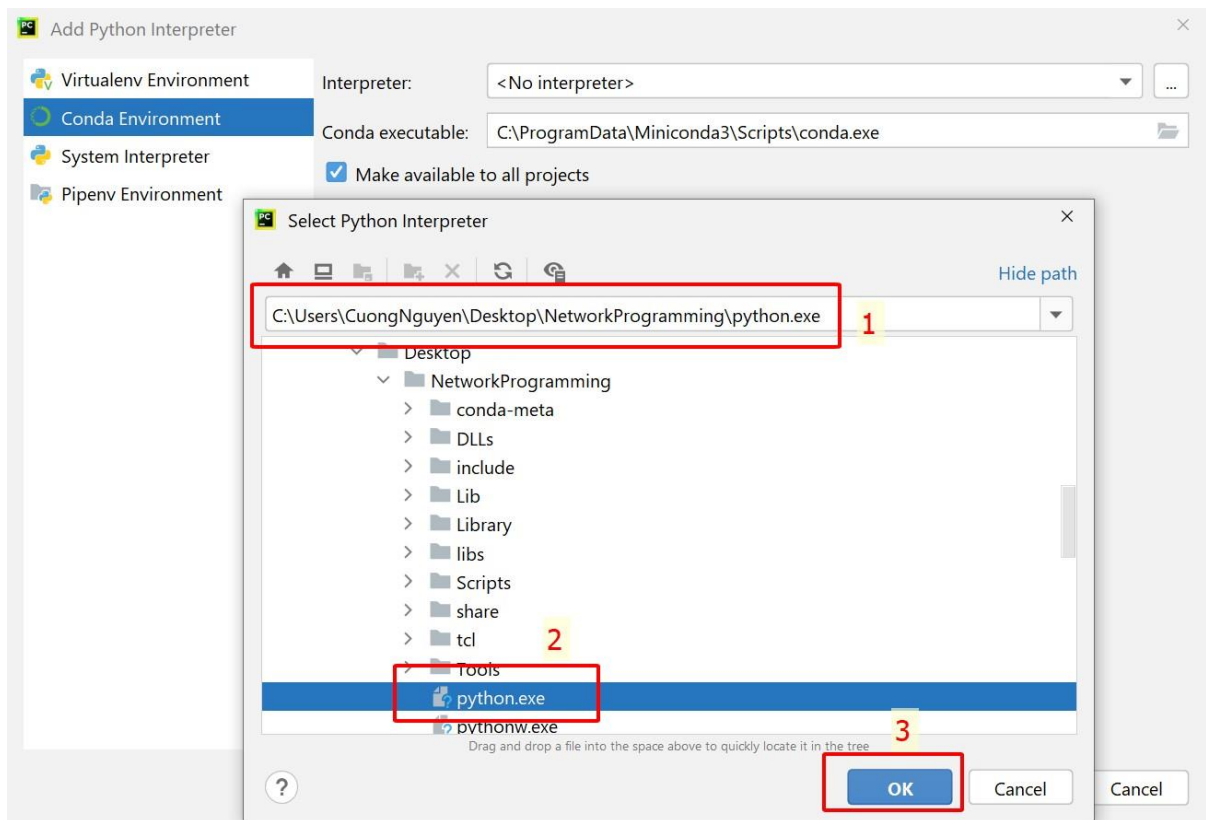
Hình 9: Tạo dự án mới với tên là pythonProject.



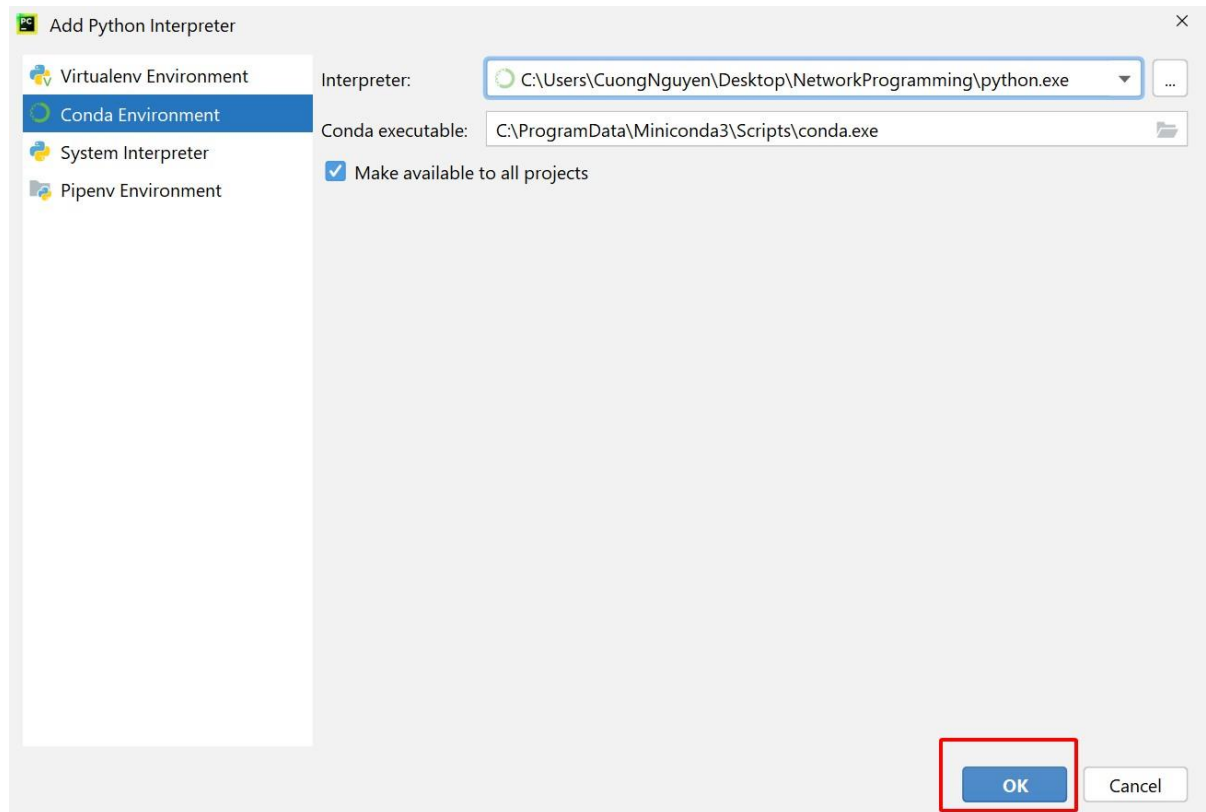
Hình 10: Chọn cấu hình trình thông dịch có sẵn.



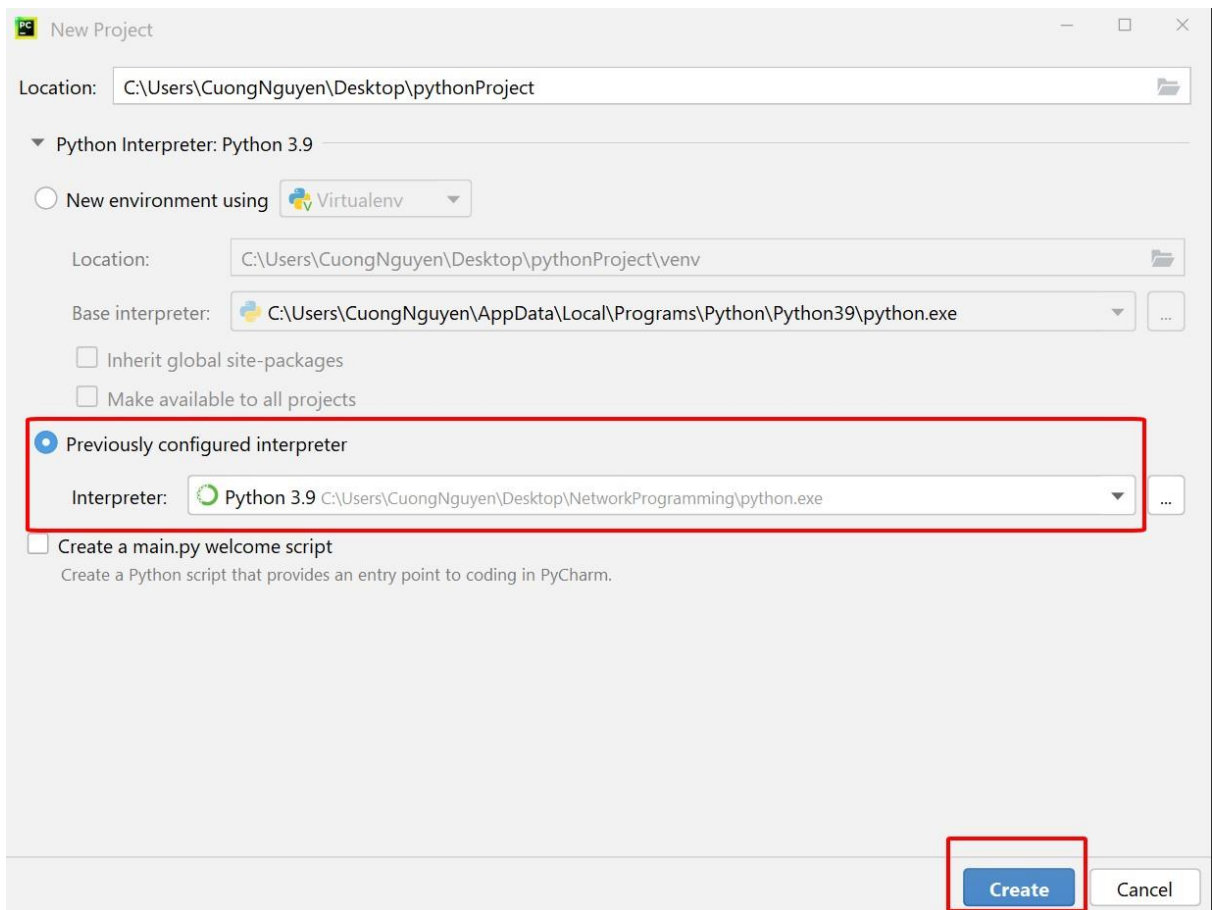
Hình 11: Chọn môi trường Conda và trình thông dịch.



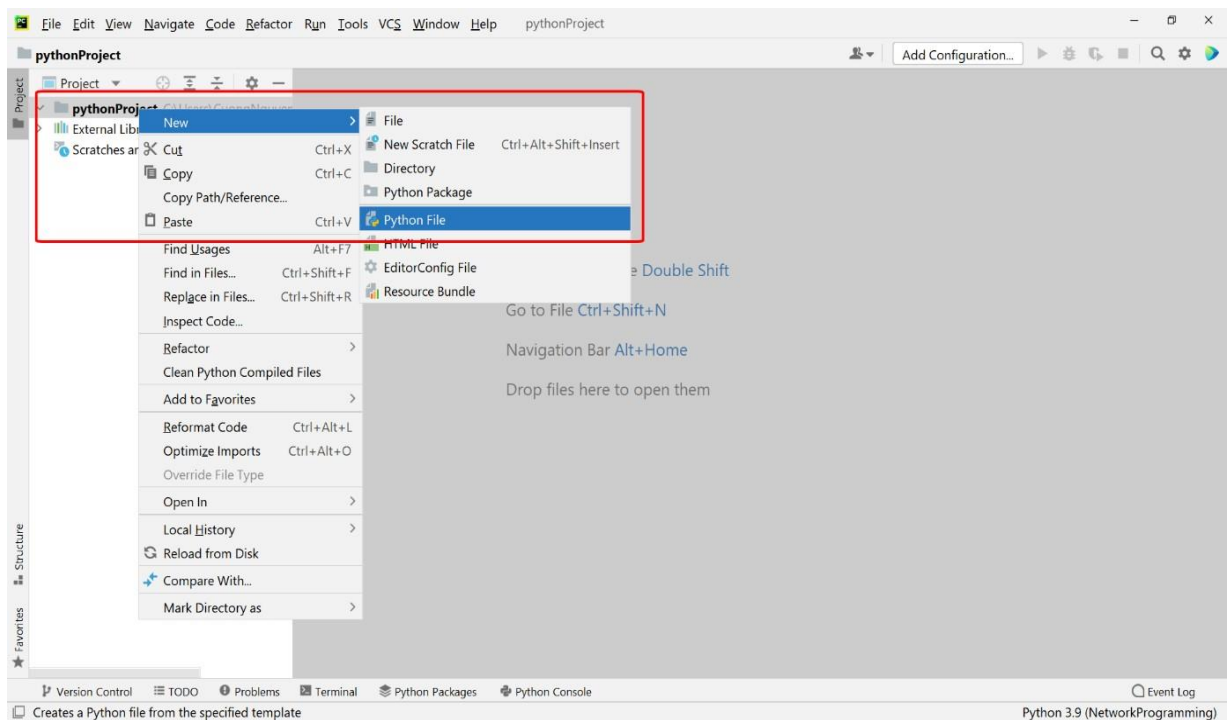
Hình 12: Chọn đường dẫn đến trình thông dịch NetworkProgramming.



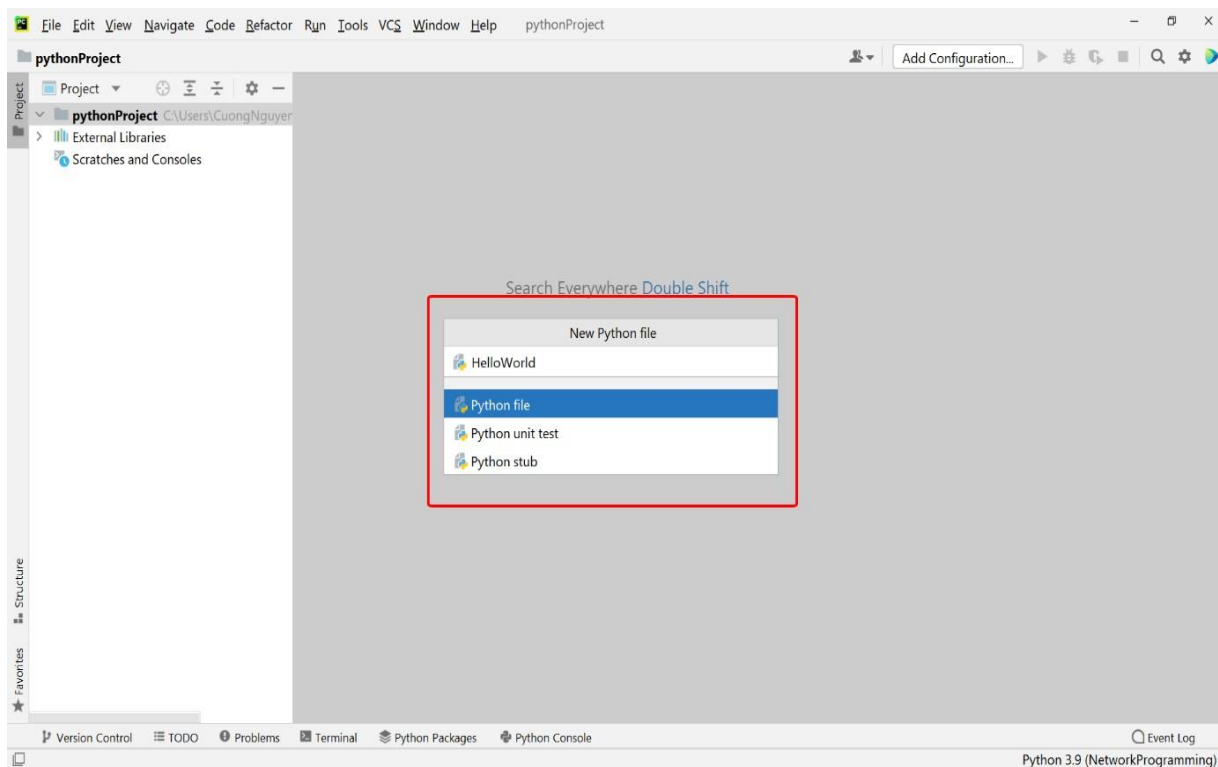
Hình 13: Chọn xong đường dẫn đến trình thông dịch NetworkProgramming.



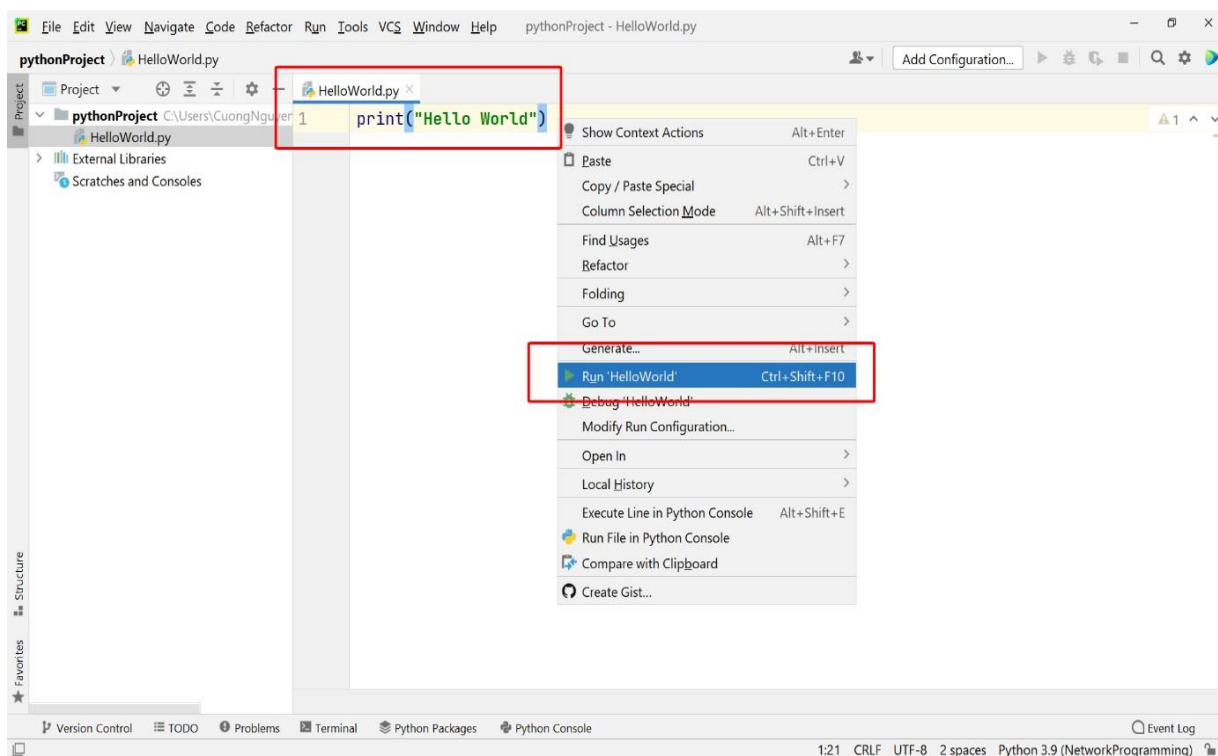
Hình 14: Tạo dự án pythonProject với trình thông dịch NetworkProgramming.



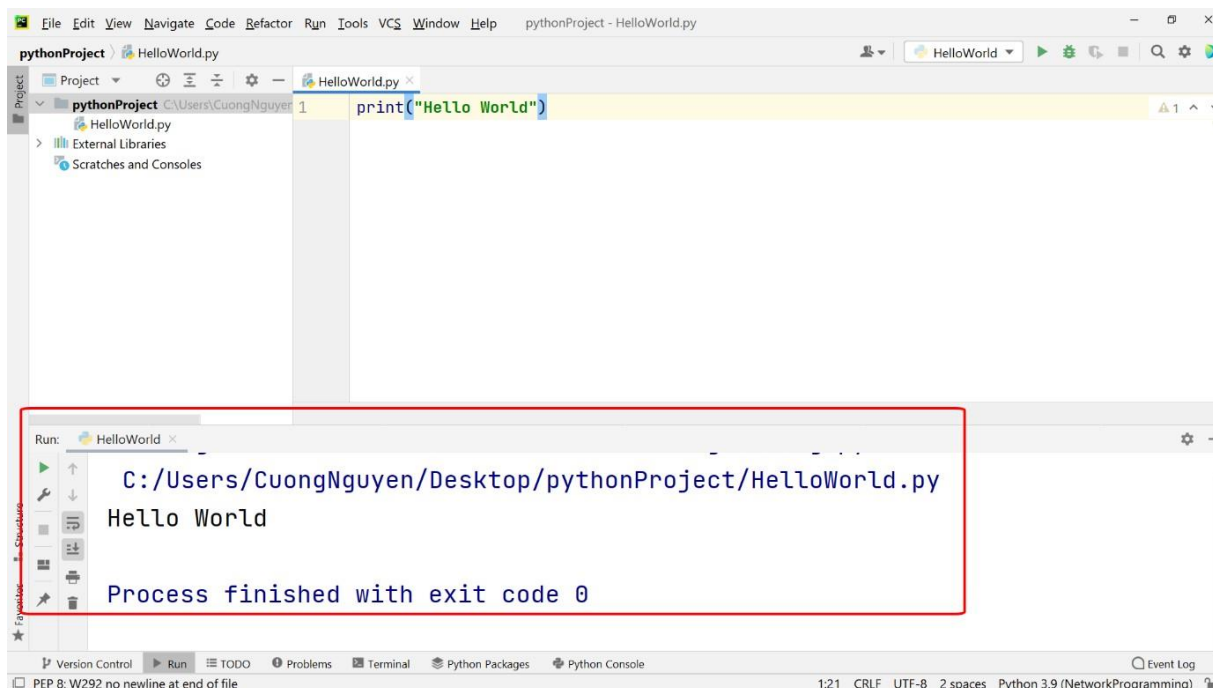
Hình 15: Tạo tập tin .py để lập trình ngôn ngữ Python.



Hình 16: Đặt tên tin .py cho dự án.



Hình 17: Chạy tập tin chương trình đã tạo.

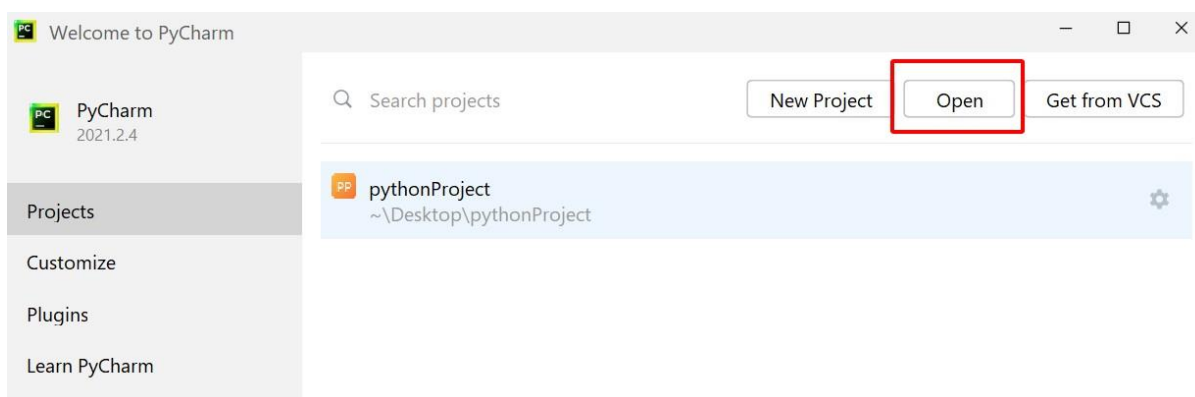


Hình 18: Kết quả trả về sau khi thực thi chương trình.

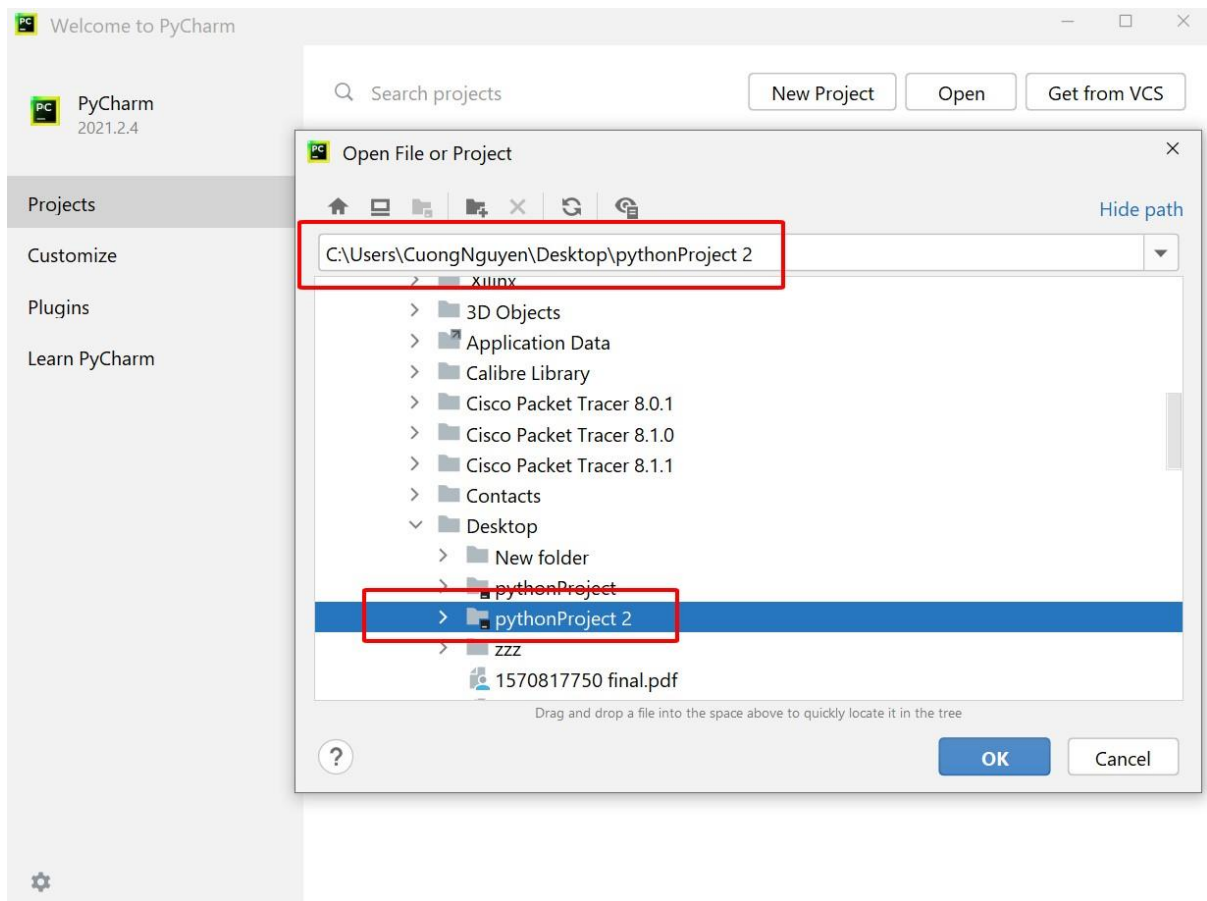
4. Mở dự án có sẵn

Để mở dự án có sẵn, hãy thực hiện các bước sau:

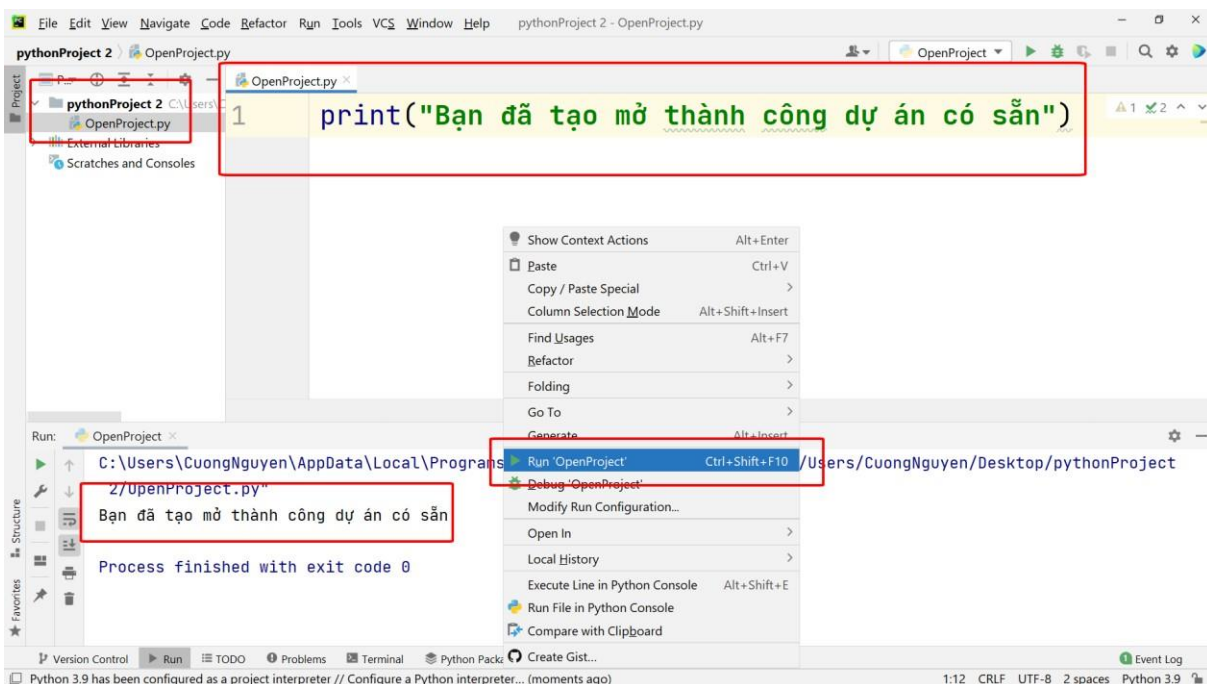
- B1.** Khởi động phần mềm PyCharm. Chọn **Open** để mở dự án có sẵn như Hình 19.
- B2.** Chọn đường dẫn đến dự án có sẵn. Hình 20 minh họa ví dụ chọn đường dẫn đến dự án **pythonProject2**.
- B3.** Chạy dự án đã mở như Hình 21.



Hình 19: Giao diện của PyCharm sau khi khởi động.



Hình 20: Chọn đường dẫn đến dự án có sẵn.



Hình 21: Chạy dự án đã được mở.

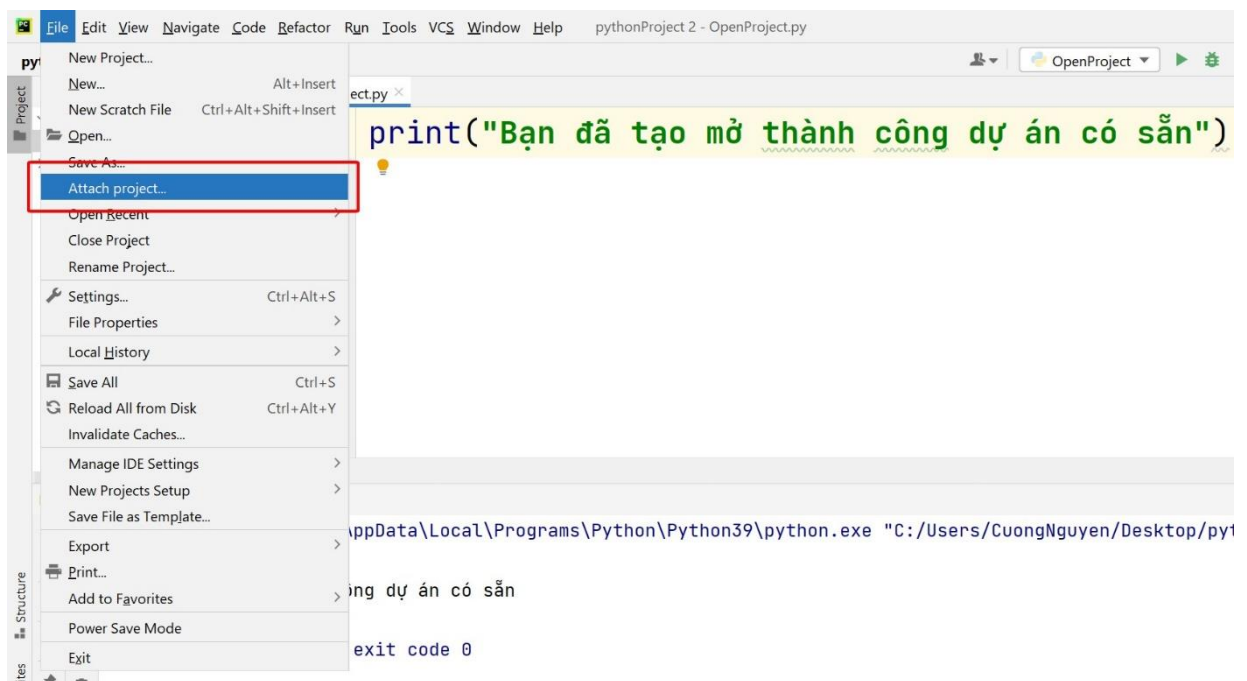
5. Thêm dự án vào cửa sổ làm việc hiện tại

Để thêm dự án có sẵn vào cửa sổ làm việc hiện tại, hãy thực hiện các bước sau:

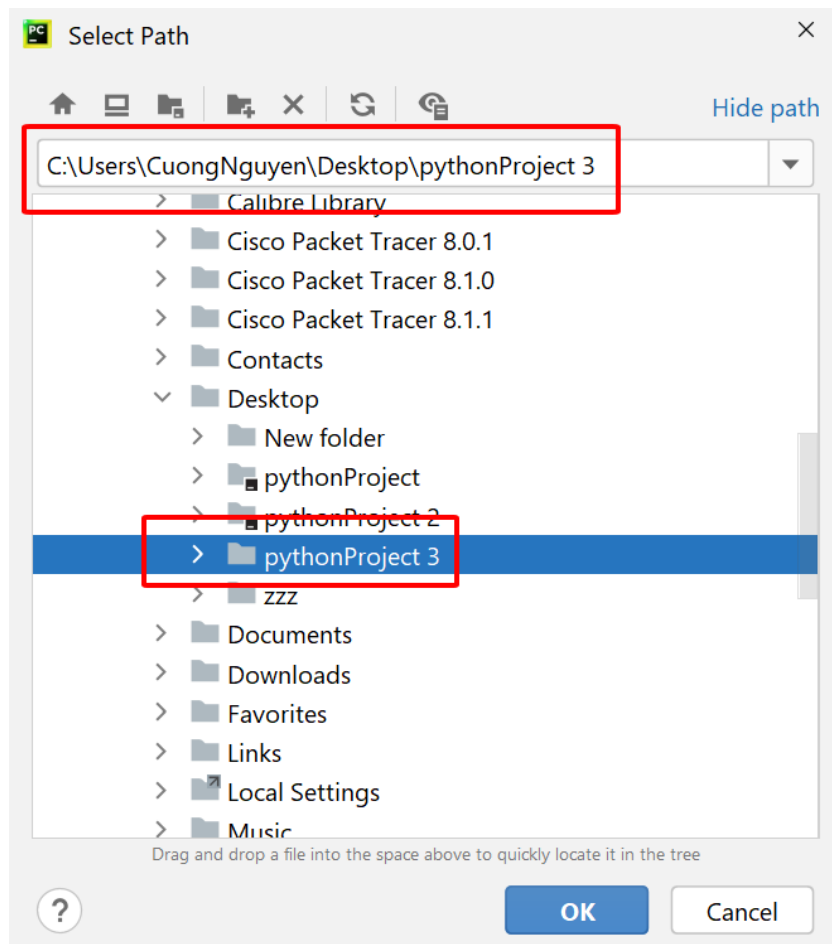
B1. Thêm dự án mới vào cửa sổ làm việc hiện tại bằng cách chọn *Attach project* như trong Hình 22.

B2. Chọn đường dẫn đến dự án có sẵn (*pythonProject3*) để thêm vào cửa sổ làm việc hiện tại như Hình 23.

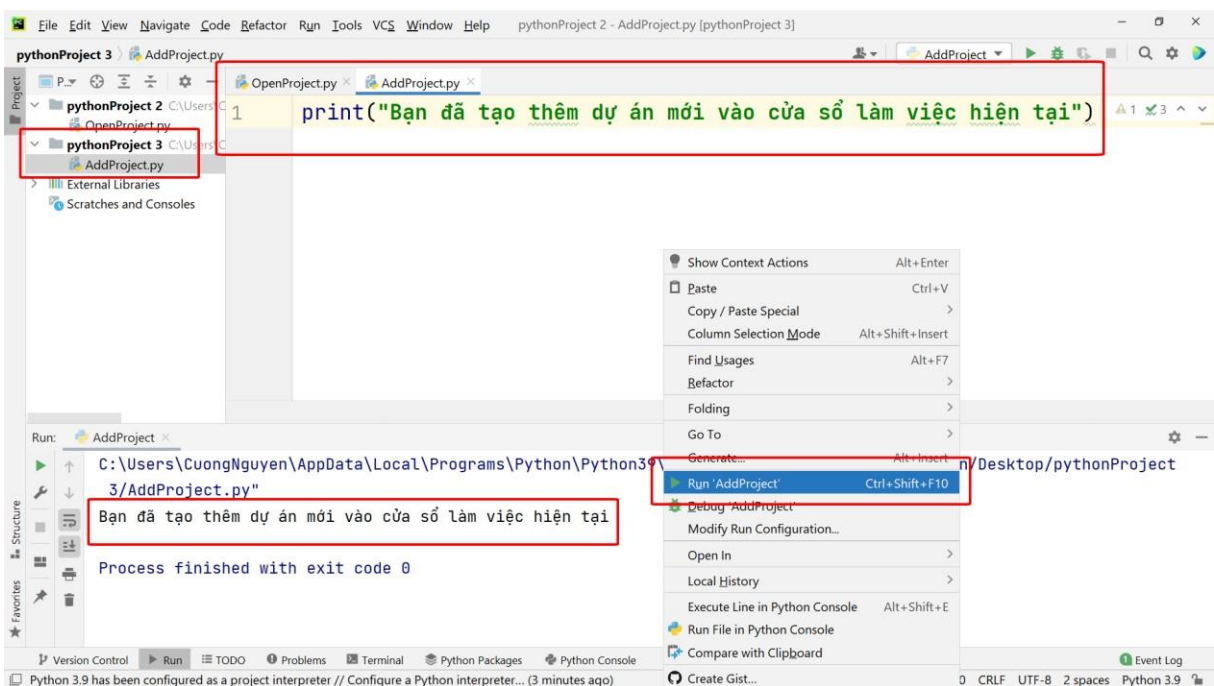
B3. Chạy dự án đã được thêm vào như Hình 24.



Hình 22: Thêm dự án mới vào cửa sổ làm việc hiện tại.



Hình 23: Chọn dự án cần thêm.



Hình 24: Chạy dự án đã được thêm vào.


Phụ lục B: Hướng dẫn Debug trên PyCharm IDE

Debug là việc dừng chương trình tại điểm được chỉ định trước. Mục đích của debug để gỡ lỗi (nếu có) và hiểu cách chương trình tính toán bằng cách xem thuộc tính của các biến, đối tượng trong chương trình. Phụ lục này hướng dẫn debug trên PyCharm IDE.

Để tiến hành debug chương trình, hãy thực hiện các bước như sau:

B1. Click chuột trái vào lề trái của dòng muốn debug hoặc nhấn tổ hợp phím tắt **Ctrl+F8** (theo mặc định). Lúc này, **breakpoint** màu đỏ sẽ xuất hiện, click thêm lần nữa để xóa như trong Hình 25.








Để cấu hình chi tiết về **breakpoint**, tại điểm **breakpoint** click chuột **phải** chọn **More (Ctrl+Shift+F8)**, hộp thoại **breakpoint** xuất hiện như Hình 26. Có thể áp dụng điều kiện cho điểm **breakpoint** tại ô **Condition** với điều kiện tương tự cú pháp của câu lệnh điều kiện **if**.

B2. Tại cửa sổ chương trình, click chuột phải chọn **Debug** hoặc ở góc trên cùng bên phải chọn biểu tượng  như Hình 27. Theo mặc định, chương trình sẽ chạy hết chương trình trước điểm **breakpoint** và dừng tại hàng được đặt dấu **breakpoint**. Cửa sổ debug tương tự Hình 28 sẽ xuất hiện. Trong đó,

- (1) **Debug** hiển thị các chương trình đang được debug (TCP_Server trong ví dụ này).
- (2) Cho phép chọn **Debugger** hoặc **Console**: **Debugger** hiển thị các thông tin debug như trong Hình 28 còn **Console** hiển thị màn hình đầu ra của chương trình.
- (3) **Frames** cho phép chọn luồng trong chương trình để hiển thị thuộc tính tại cửa sổ 4 và 5 (ví dụ này hiển thị cho module TCP_Server).
- (4) **Variables** hiển thị thuộc tính của các biến, đối tượng tại thời điểm chương trình dừng (Tất cả biến đang hiển thị trong Hình 28 là giá trị tại thời điểm sau khi chạy qua dòng số 7 của chương trình).
- (5) **Watches** hiển thị thuộc tính của các biến, đối tượng do người dùng chỉ định. Đặc biệt, tại đây có thể cho phép thực hiện các biểu thức như gán biến, gọi phương thức... (ví dụ trong Hình 28 đang xem thuộc tính của biến **serverPort** và đối tượng **serverSocket**).

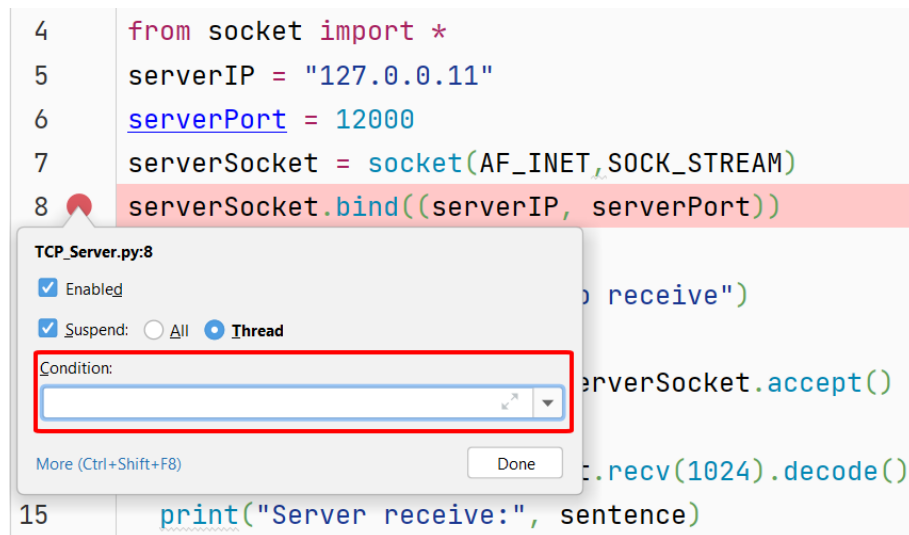
Tiếp theo, để tiếp tục debug chương trình tại các dòng sau thì có thể tham khảo bước 3.

B3. Trong cửa sổ Debug Tool, tùy chọn các biểu tượng sau để thực hiện các chức năng tương ứng:

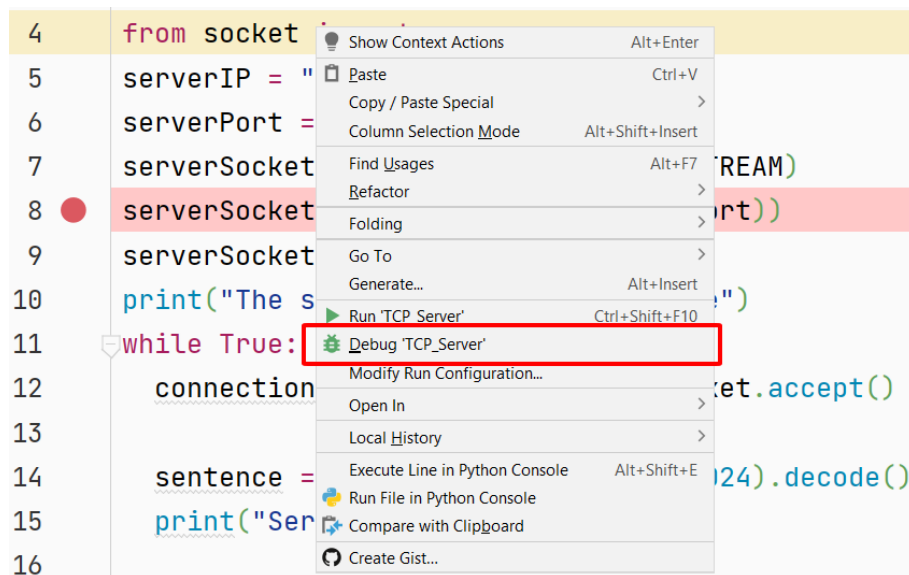
- Nhấn vào biểu tượng  (**Step Over**) hoặc phím tắt **F8**, chương trình thực thi lần lượt từng dòng lệnh.
- Nhấn vào biểu tượng  (**Step Into**) hoặc phím tắt **F7**, chương trình nhảy vào trong hàm con của dòng lệnh đang dừng (nếu không có hàm tại dòng đang dừng thì thực hiện chức năng như **Step Over**).
- Nhấn vào biểu tượng  (**Step Into My Code**) hoặc tổ hợp phím tắt **Alt+Shift+F7**, chương trình nhảy vào trong hàm và quay về nơi gọi hàm để thực thi dòng lệnh tiếp theo.
- Nhấn vào biểu tượng  (**Step Out**) hoặc tổ hợp phím tắt **Shift+F8**, chương trình nhảy ra khỏi hàm và quay trở về nơi gọi hàm.
- Nhấn vào biểu tượng  (**Run to Cursor**) hoặc tổ hợp phím tắt **Alt+F9**, chương trình được thực thi liên tục cho tới dòng lệnh trước con trỏ.
- Nhấn vào biểu tượng  (**Show Execution Point**) hoặc tổ hợp phím tắt **Alt+F10**, để quay trở về dòng lệnh con trỏ vừa thực hiện.
- Nhấn vào biểu tượng  (**Evaluate Expression**) hoặc tổ hợp phím tắt **Alt+F8**, chương trình tính toán biểu thức, có thể nhập tên biến bất kỳ để tính toán.
-

```
4   from socket import *
5   serverIP = "127.0.0.11"
6   serverPort = 12000
7   serverSocket = socket(AF_INET, SOCK_STREAM)
8   serverSocket.bind((serverIP, serverPort))
9   serverSocket.listen(1)
10  print("The server is ready to receive")
11  while True:
12      connectionSocket, addr = serverSocket.accept()
13
14      sentence = connectionSocket.recv(1024).decode()
15      print("Server receive:", sentence)
```

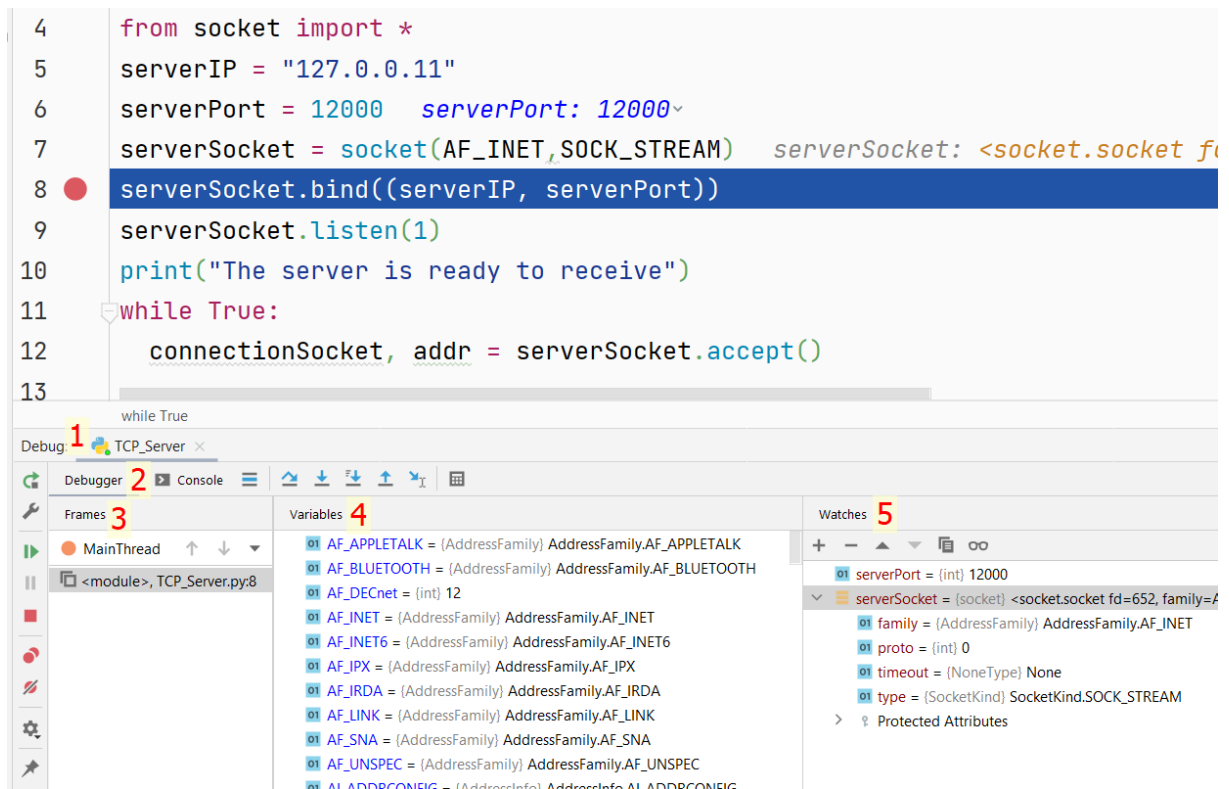
Hình 25: Đặt breakpoint để debug chương trình.



Hình 26: Hộp thoại Breakpoint.





Hình 27: Chọn chế độ Debug tại cửa sổ chương trình



Hình 28: Cửa sổ Debug

B4. Ngoài ra, có thể

- (1) chọn **Rerun**  để thực hiện chạy lại chương trình
- hoặc (2) chọn **Resume Program**  để tiếp tục chương trình hay thực thi chương trình cho đến khi gặp điểm **breakpoint** tiếp theo (nếu không có thì sẽ chạy hết chương trình)
- và (3) chọn **Stop** để dừng chương trình 