School of Computing Sciences Pak-Austria Fachhochschule:
Institute of Applied Sciences and Technology, Haripur, Pakistan

Lab 04
Introduction to Socket Programming (Trivial TCP Server)

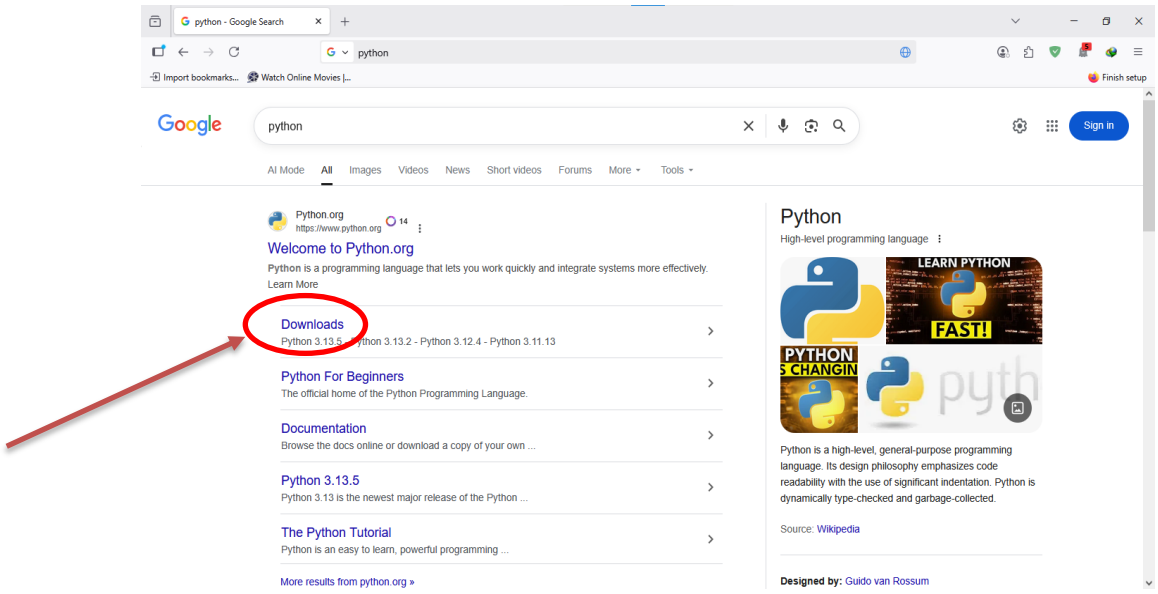
Course: COMP-352L (Computer Networks Lab)

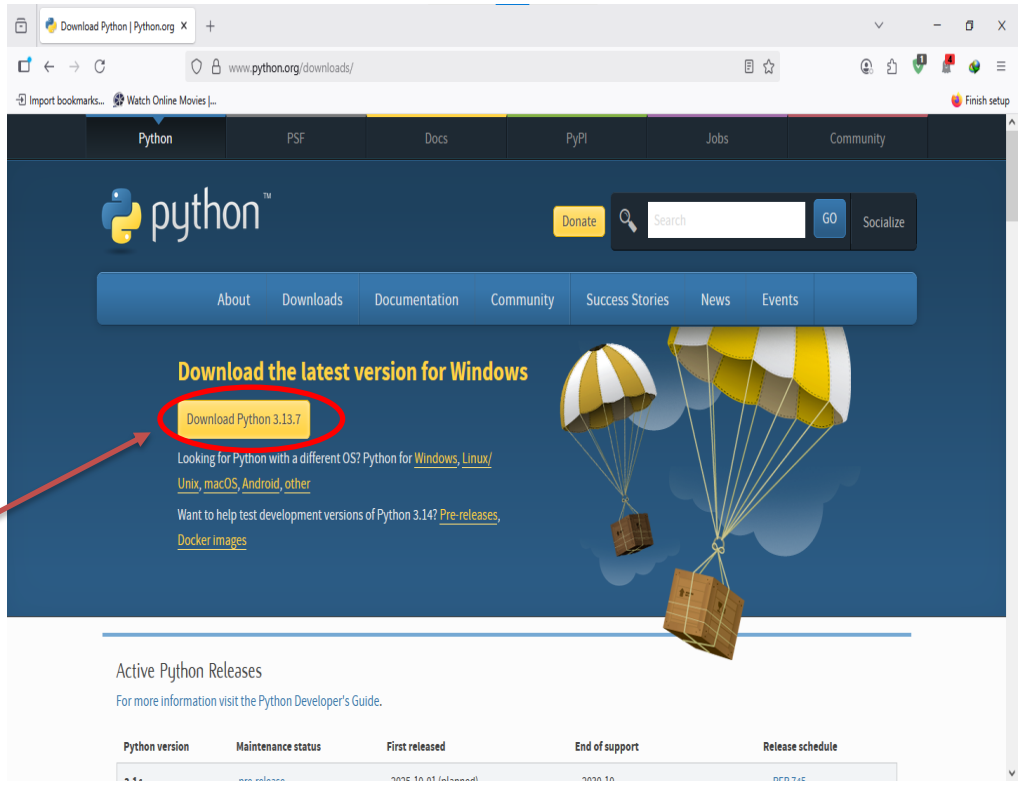
Lab Demonstrator: Jarullah Khan

Setting up Development Environment

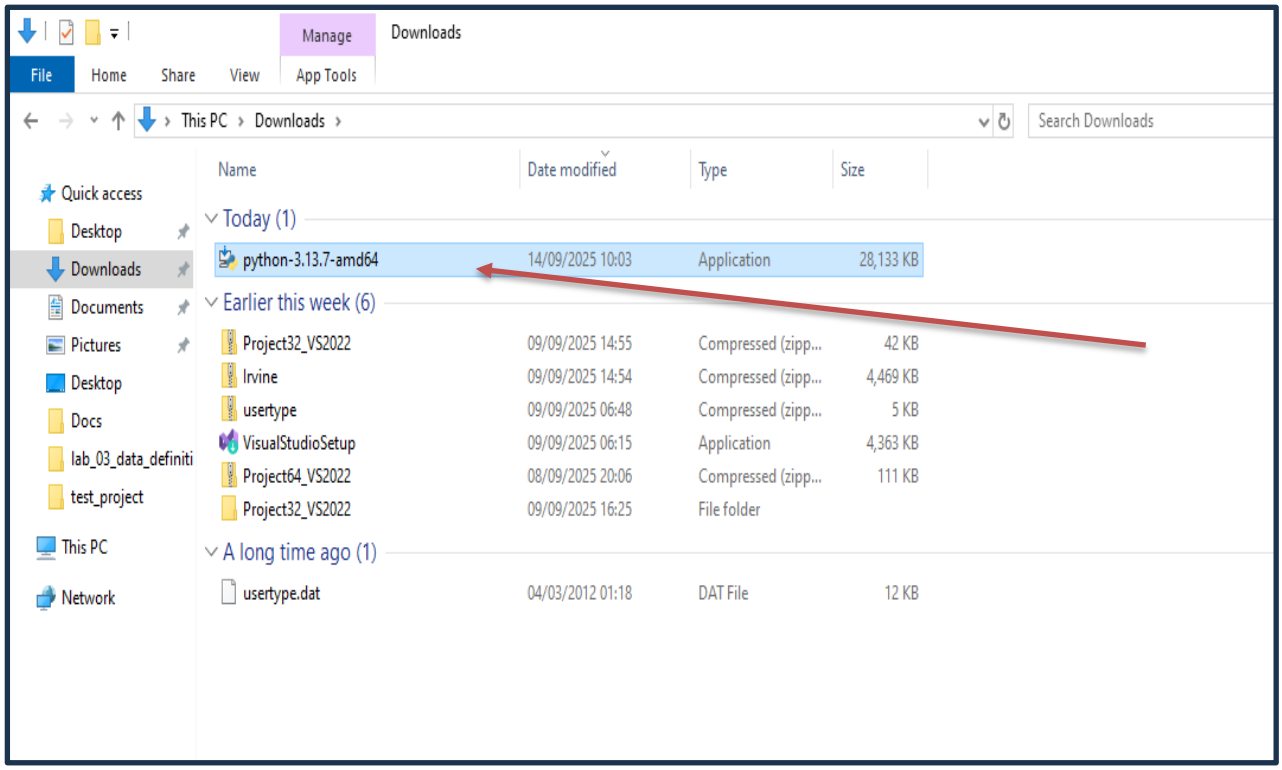
1. Installing Python

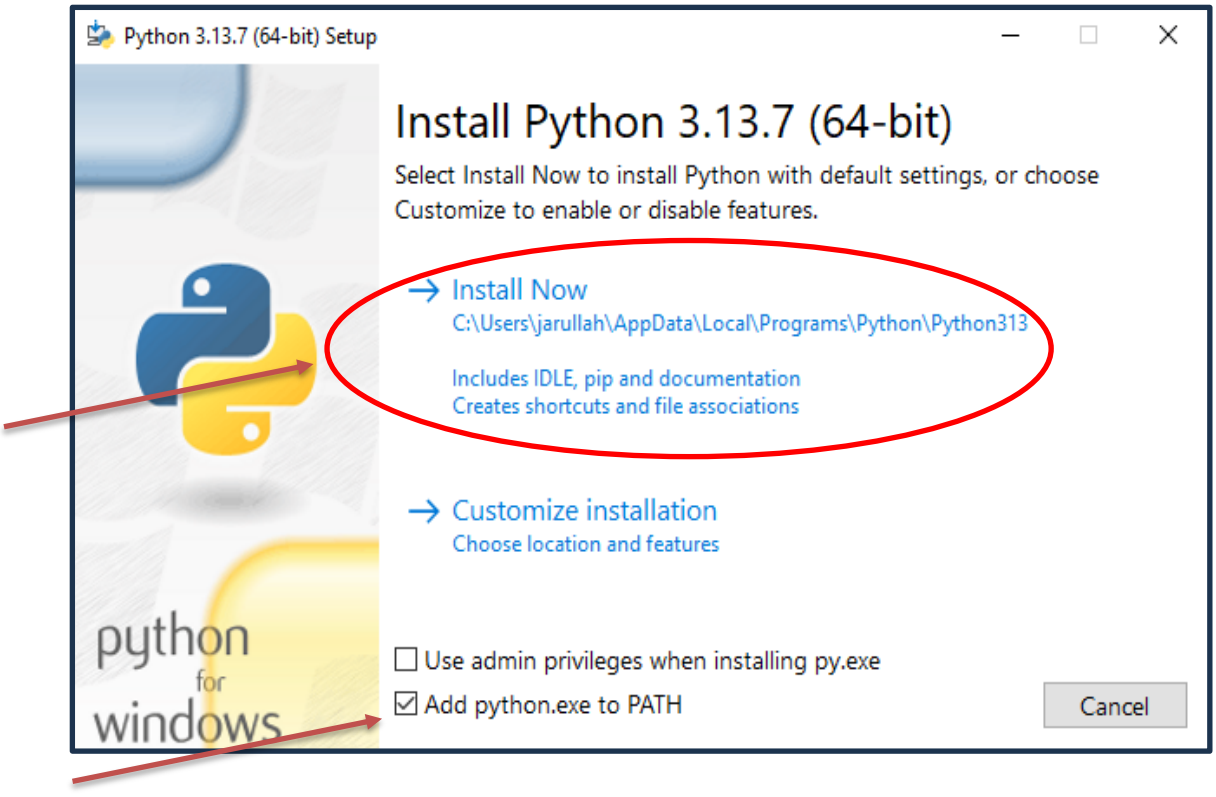
a. Download python from https://www.python.org/downloads/

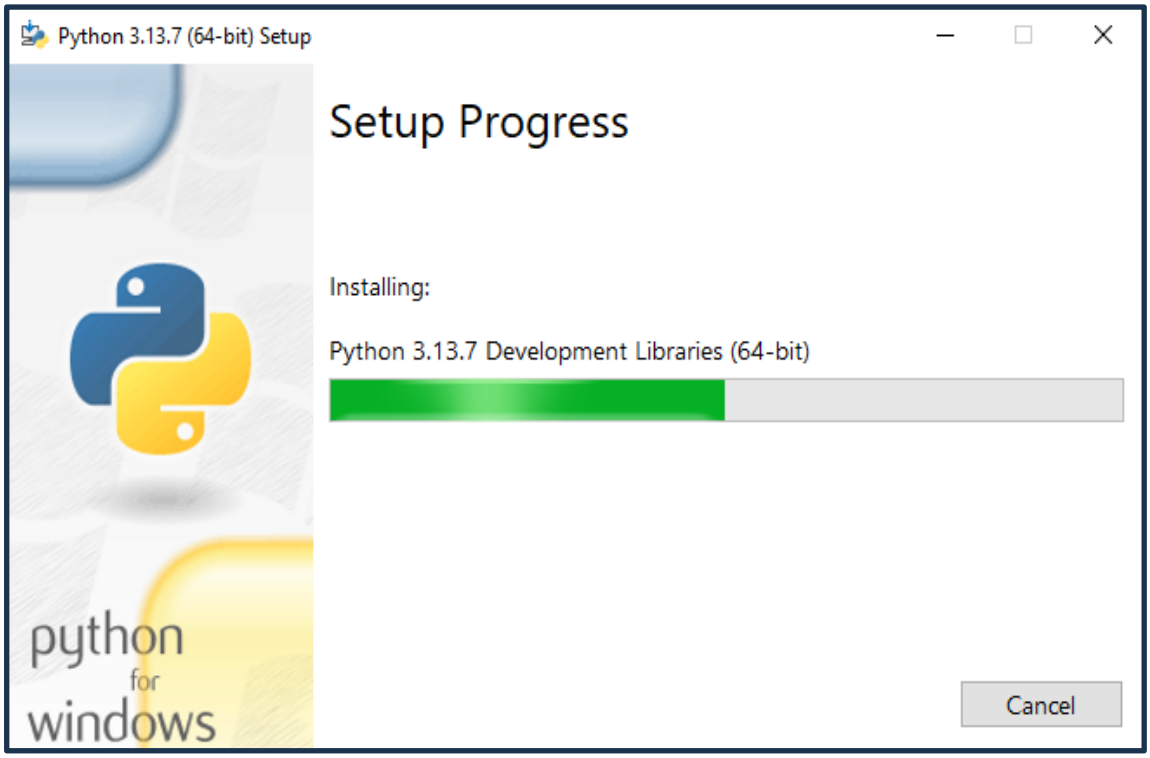


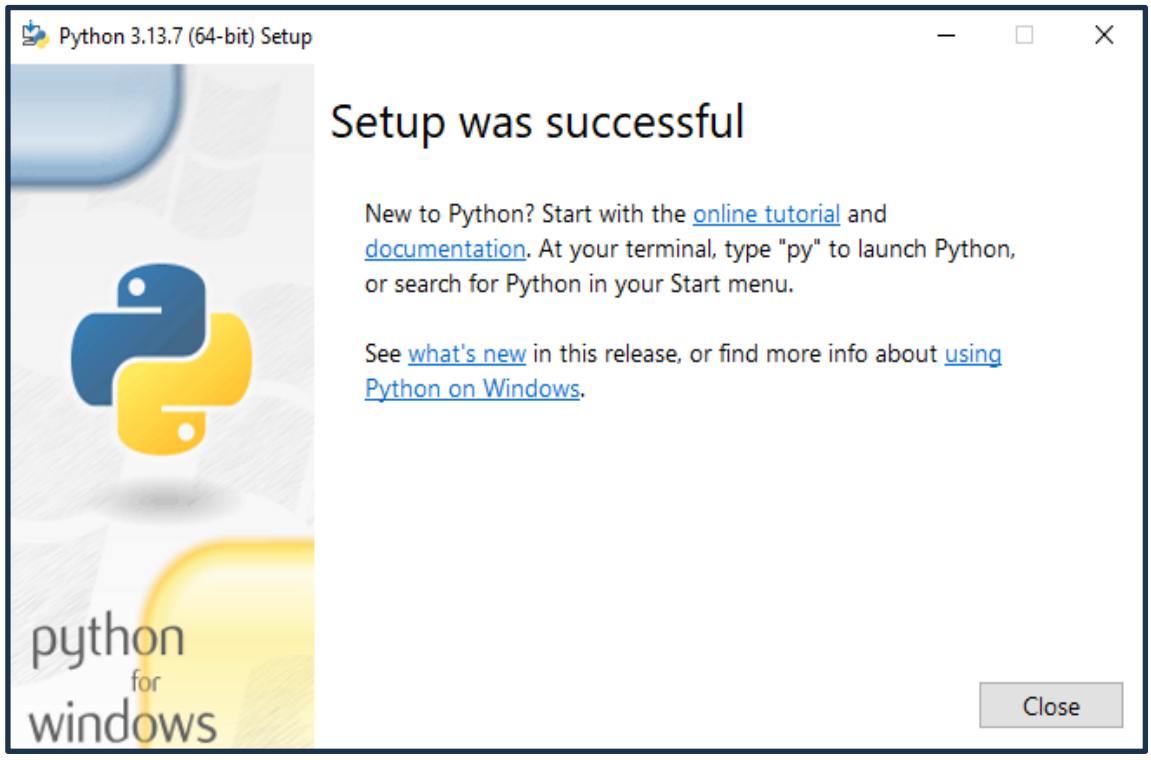


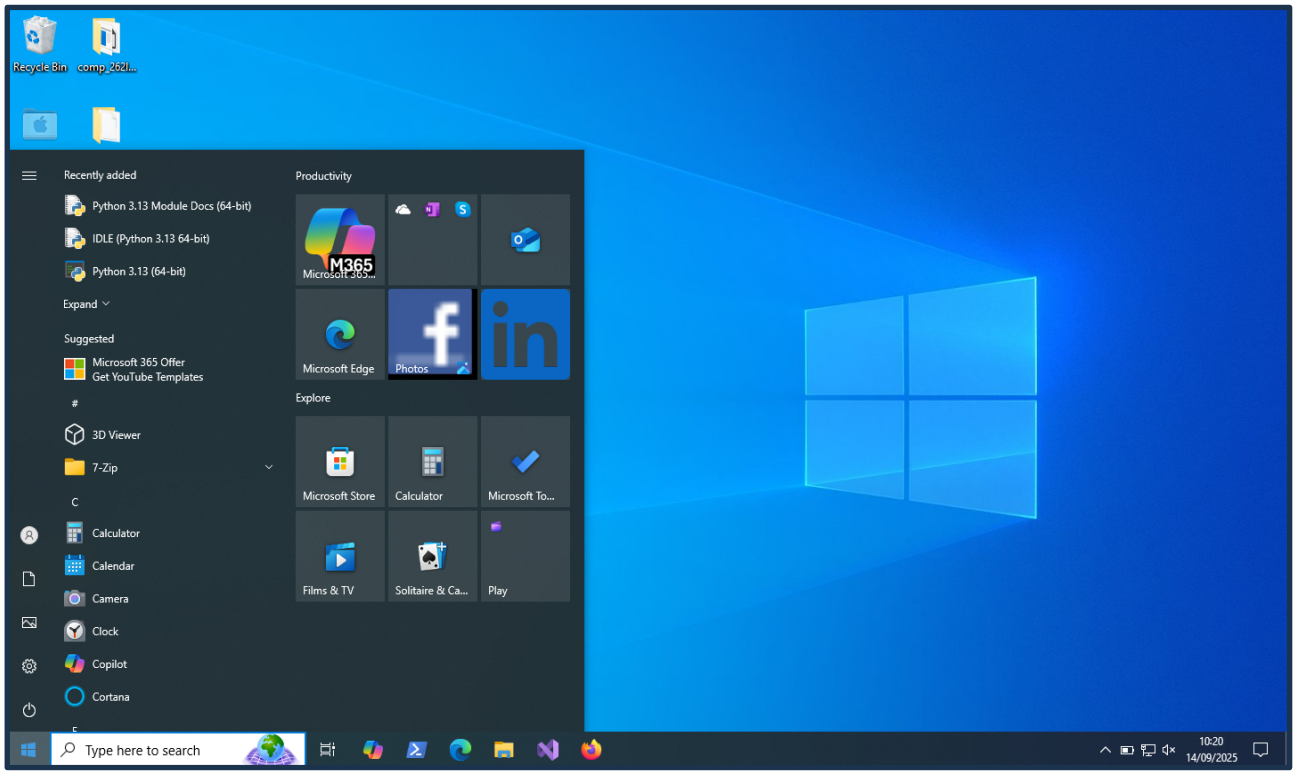
b. Run the installer and follow on screen instructions.

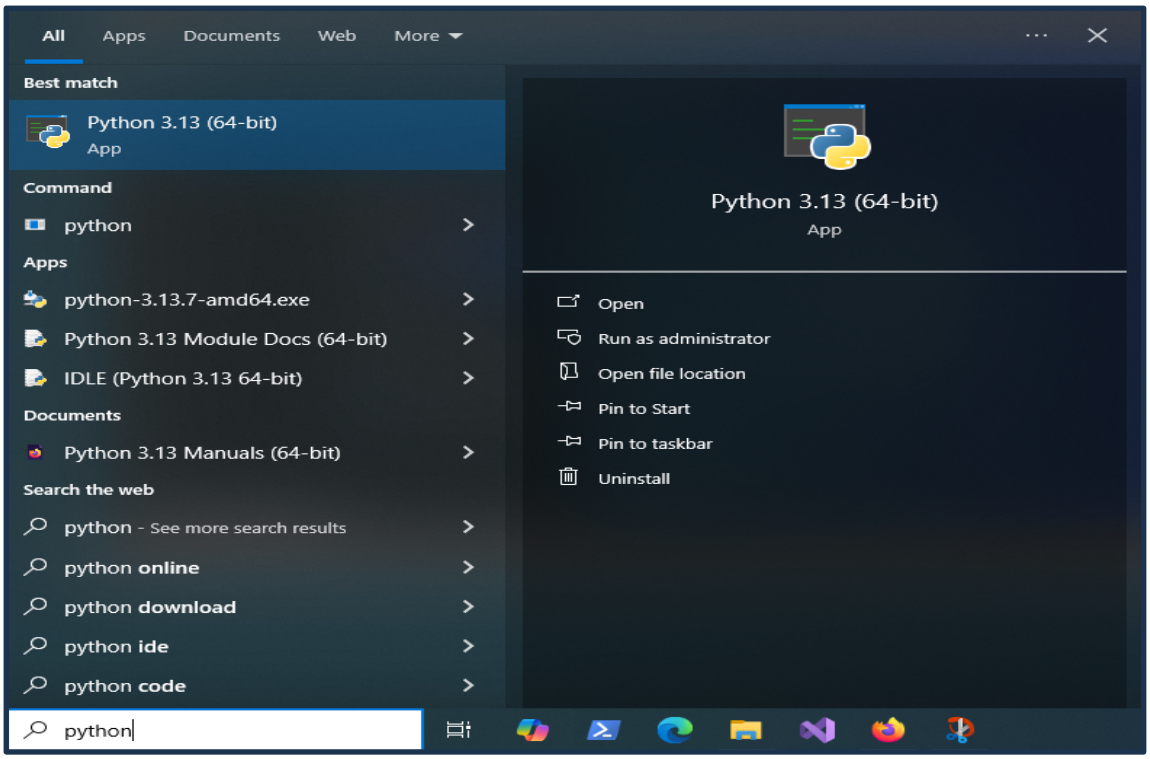


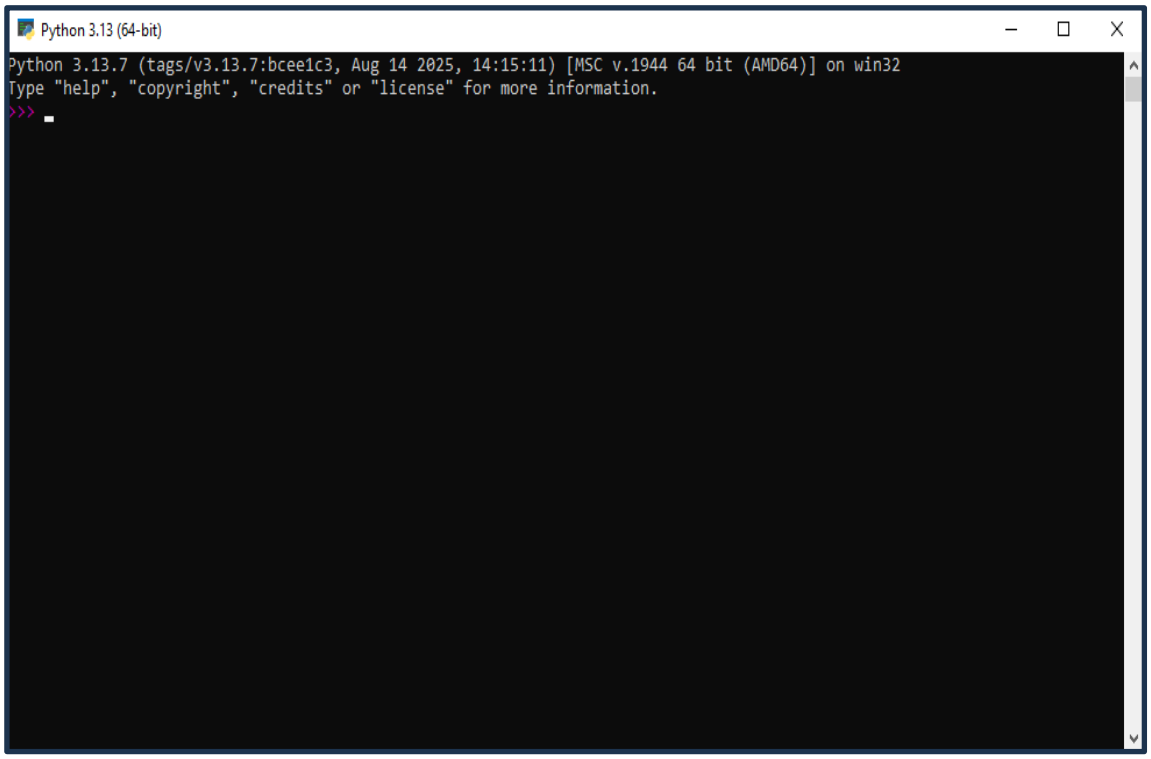



Python 3.13.7 (64-bit) Setup
Setup Progress
Installing:
Python 3.13.7 Development Libraries (64-bit)
python
for
windows
Cancel


Python 3.13.7 (64-bit) Setup
Setup was successful
New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.
See what's new in this release, or find more info about using Python on Windows.
python
for
windows
Close


Recycle Bin
Recently added
Python 3.13 Module Docs (64-bit)
IDLE (Python 3.13 64-bit)
Python 3.13 (64-bit)
Expand
Suggested
Microsoft 365 Offer
Get YouTube Templates
3D Viewer
7-Zip
Calculator
Calendar
Camera
Clock
Copilot
Cortana
Productivity
Microsoft Edge
Photos
Explore
Microsoft Store
Calculator
Films & TV
Solitaire & Ca...
Play
Type here to search
10:20
14/09/2025

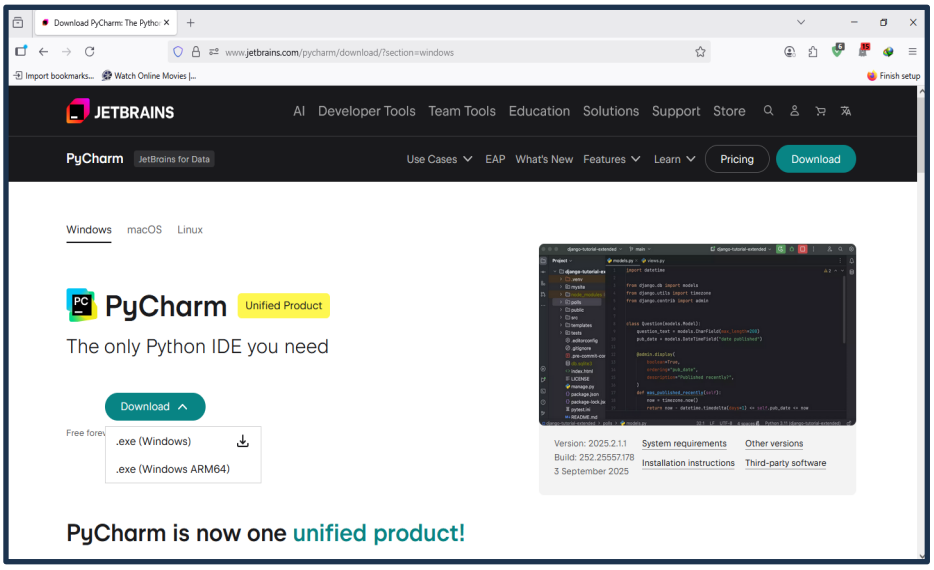

All
Apps
Documents
Web
More
Best match
Python 3.13 (64-bit)
App
Command
python
Apps
python-3.13.7-amd64.exe
Python 3.13 Module Docs (64-bit)
IDLE (Python 3.13 64-bit)
Documents
Python 3.13 Manuals (64-bit)
Search the web
python - See more search results
python online
python download
python ide
python code
python
Python 3.13 (64-bit)
App
Open
Run as administrator
Open file location
Pin to Start
Pin to taskbar
Uninstall



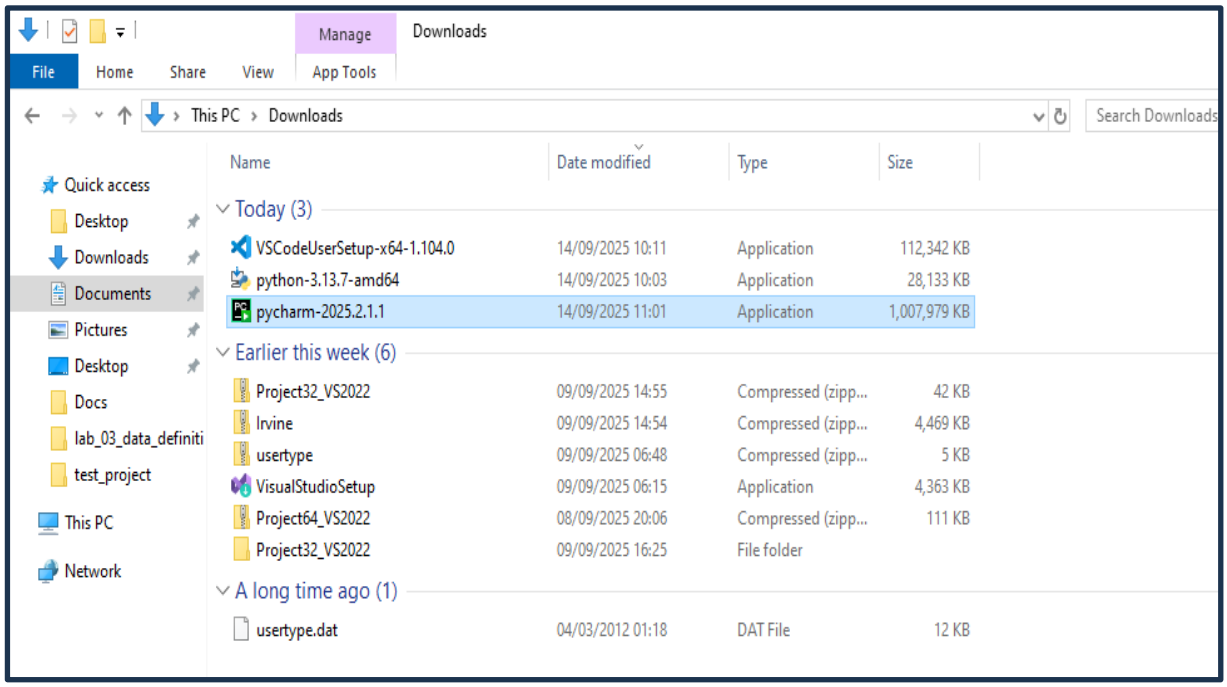
2. Installing PyCharm

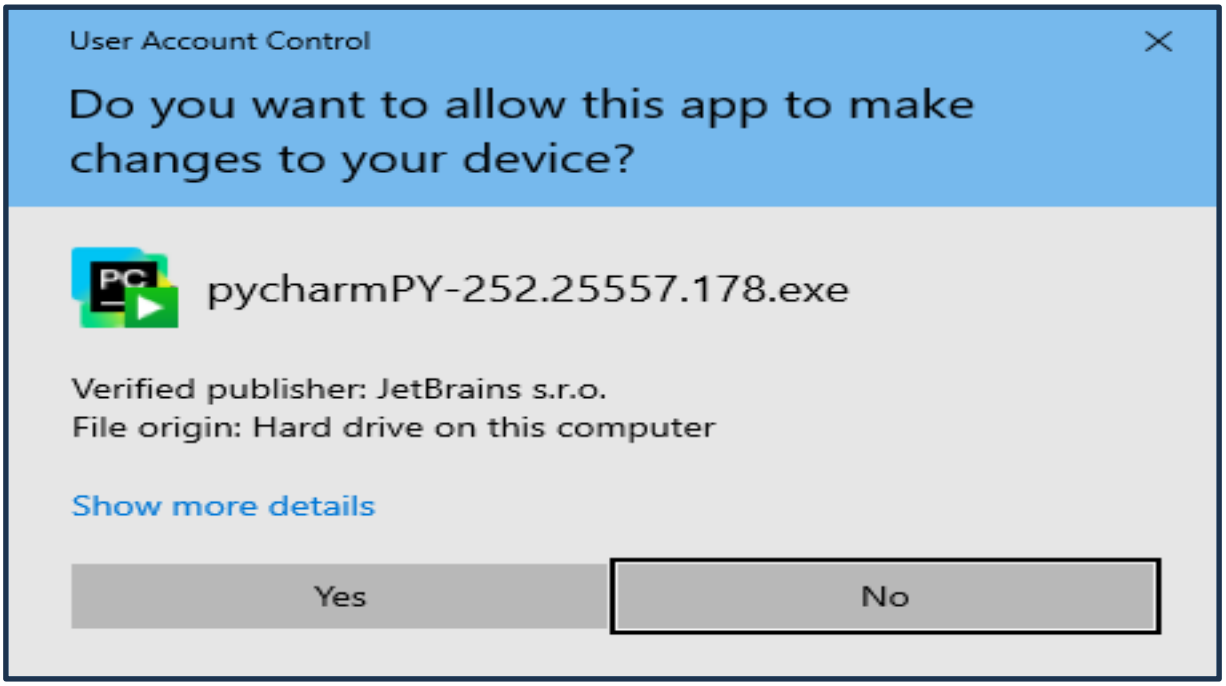
a. Download PyCharm from

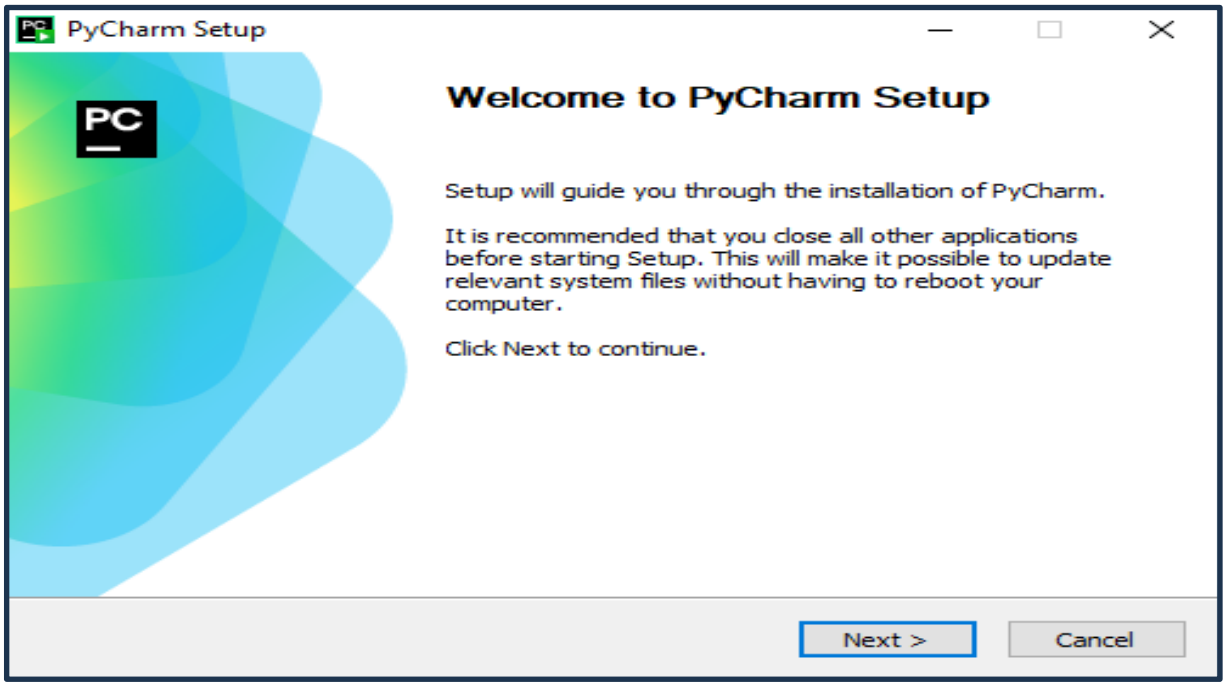
https://www.jetbrains.com/pycharm/download/?section=windows

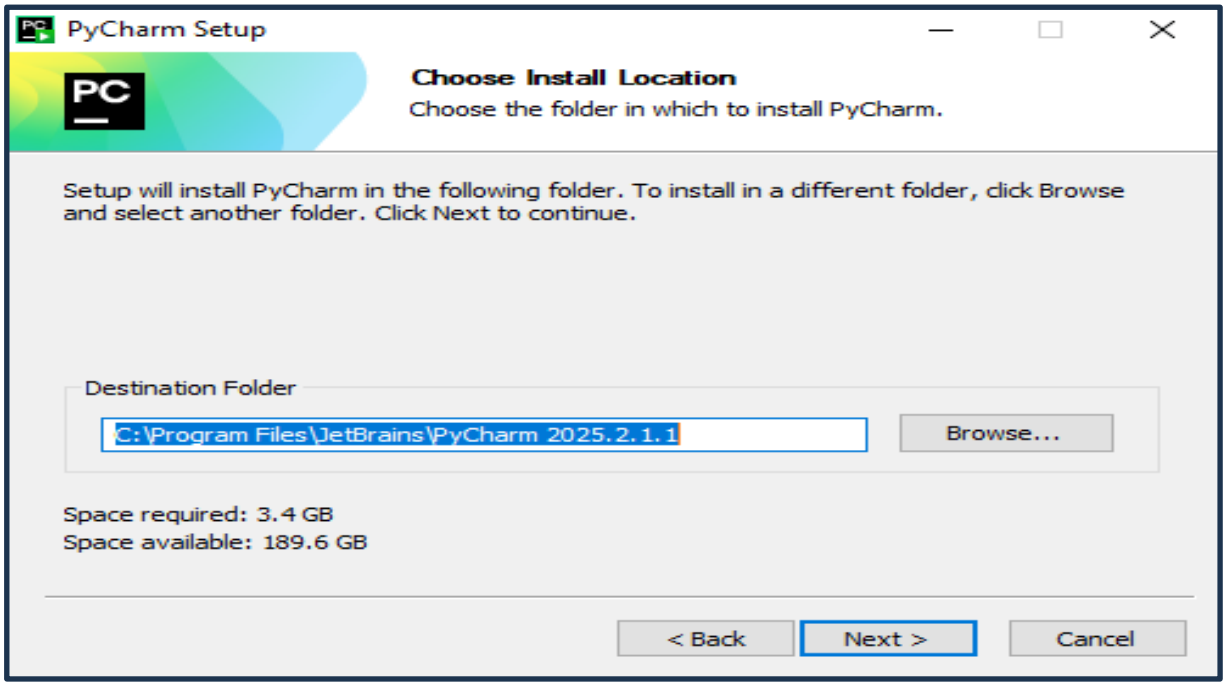


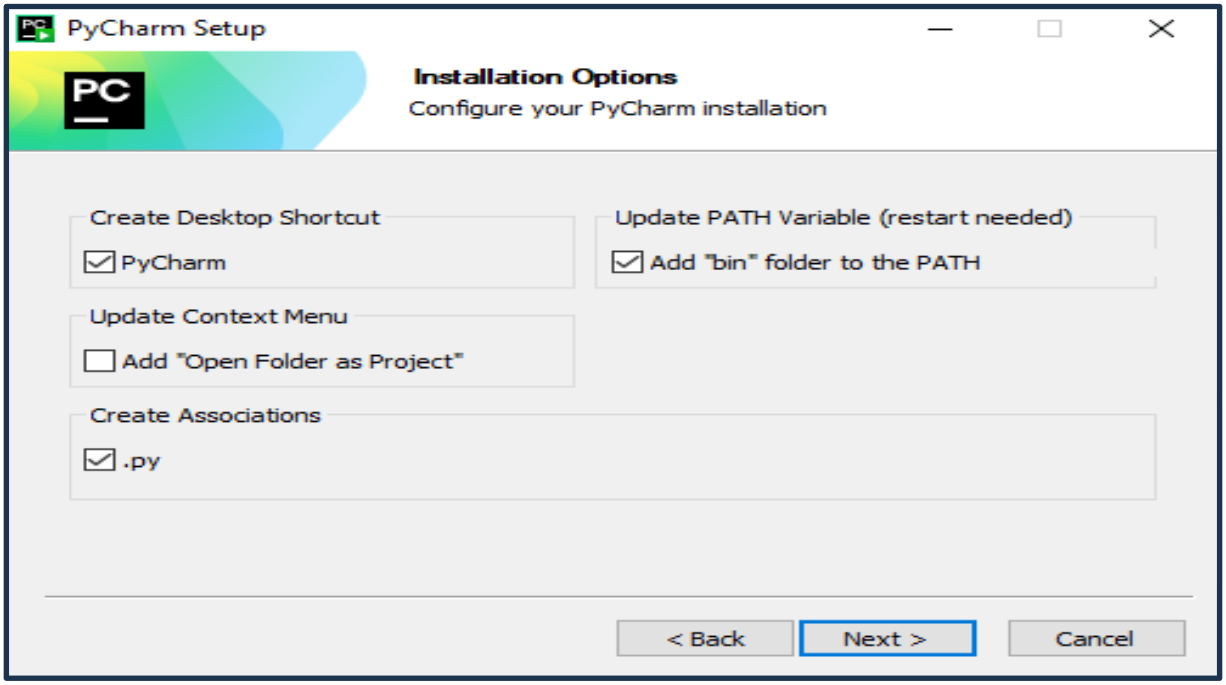
b. Run the installer and follow on screen instructions

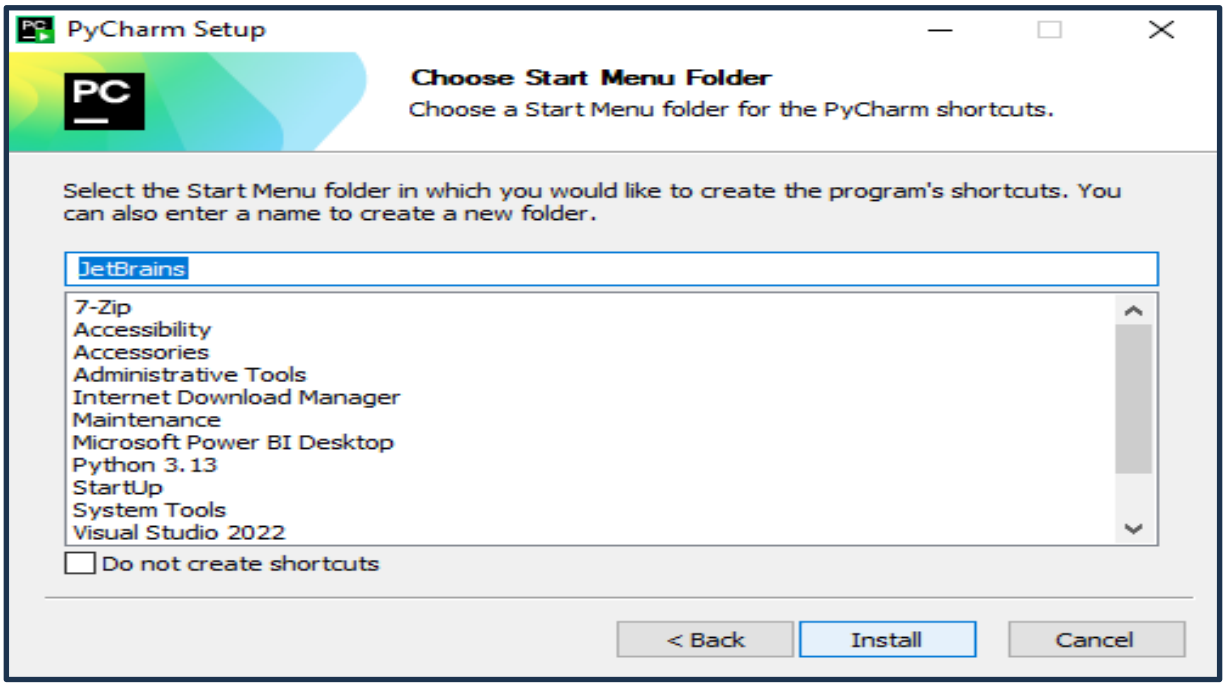


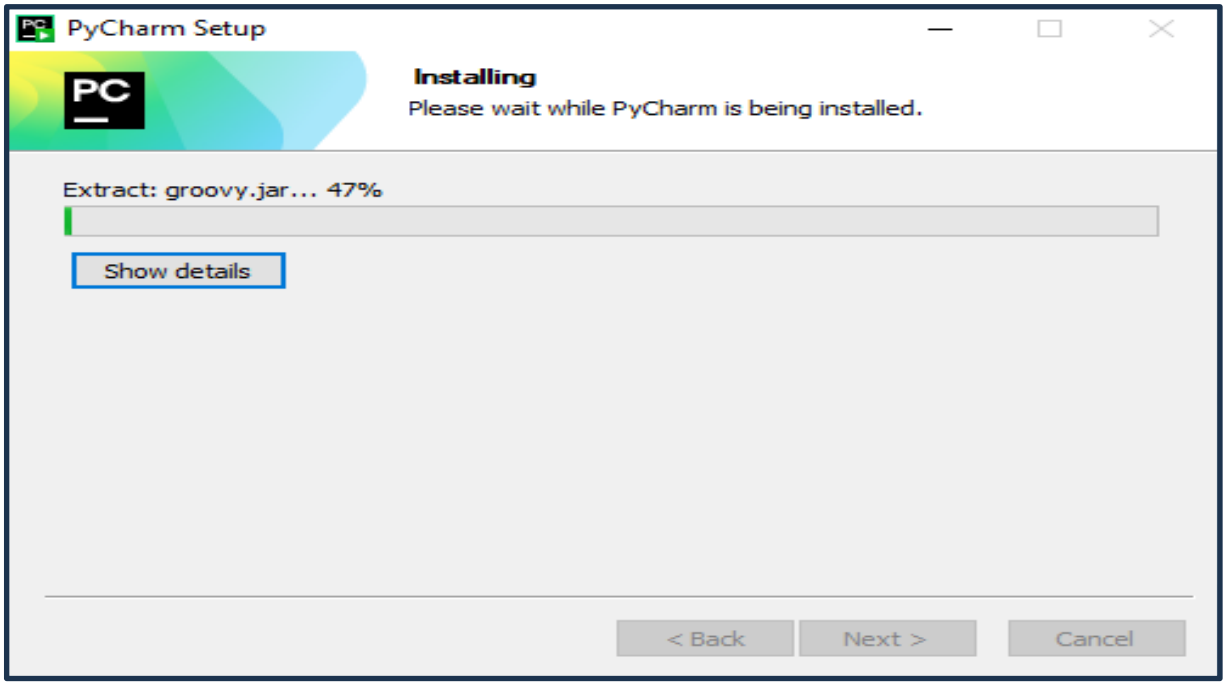


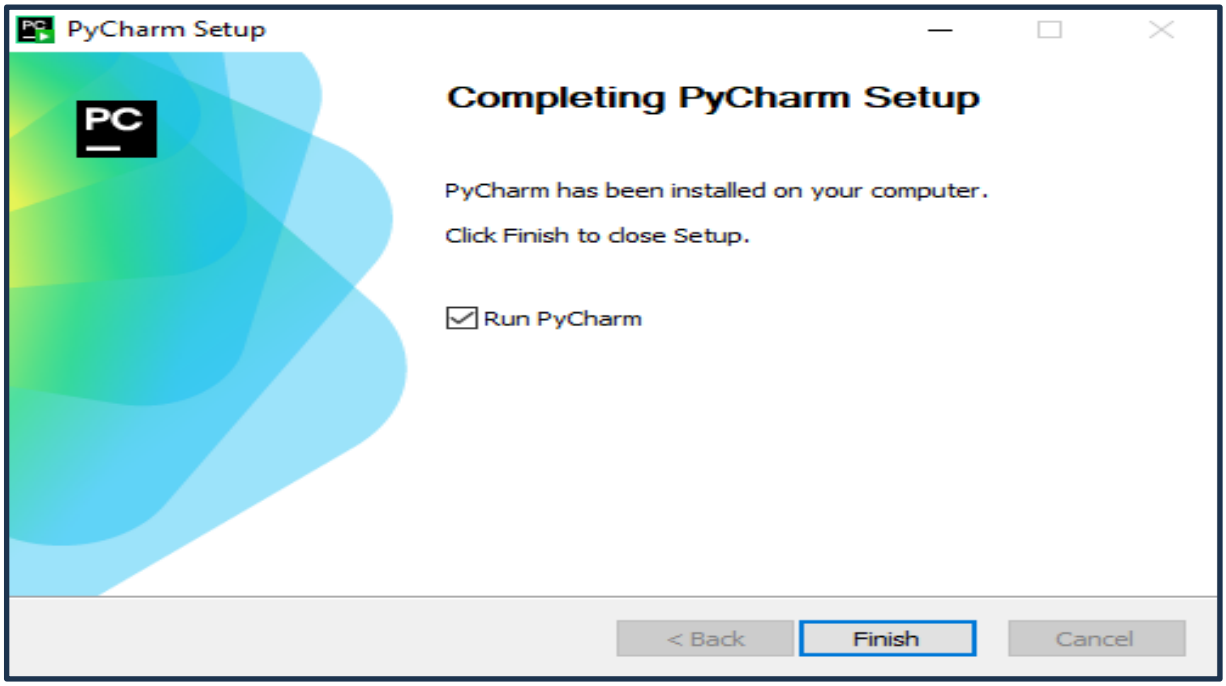

PyCharm Setup
Welcome to PyCharm Setup
Setup will guide you through the installation of PyCharm.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Next >
Cancel

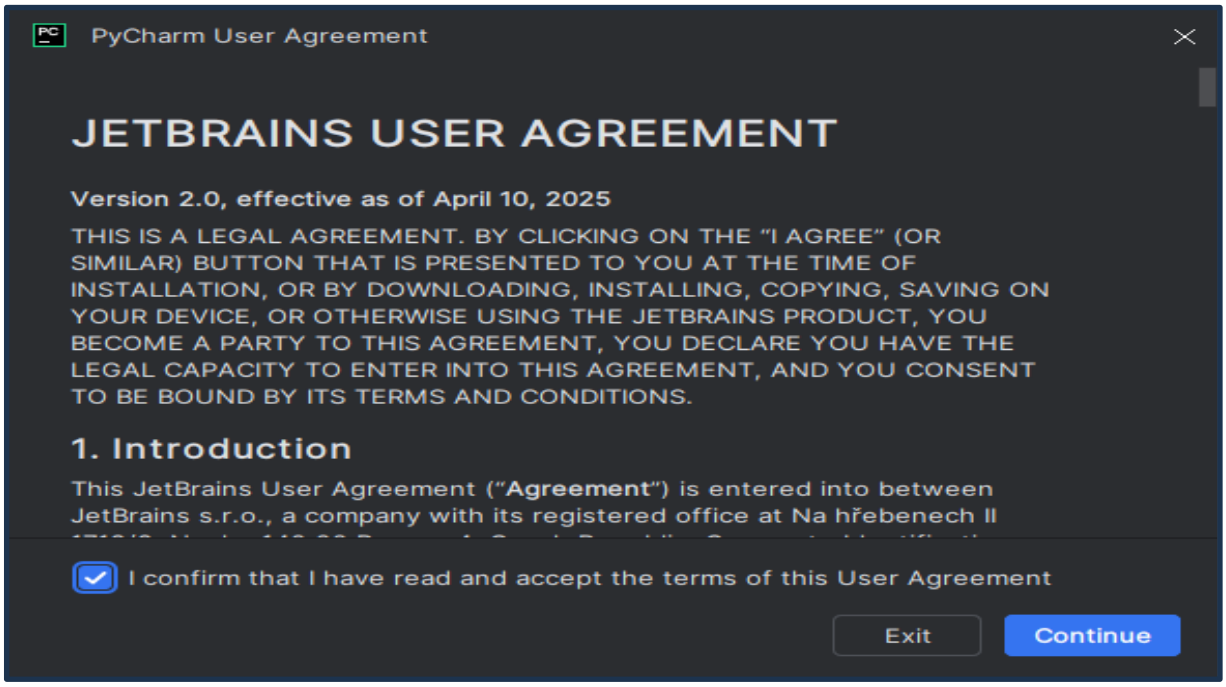

PyCharm Setup
Choose Install Location
Choose the folder in which to install PyCharm.
Setup will install PyCharm in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Program Files\JetBrains\PyCharm 2025.2.1.1
Browse...
Space required: 3.4 GB
Space available: 189.6 GB
< Back
Next >
Cancel

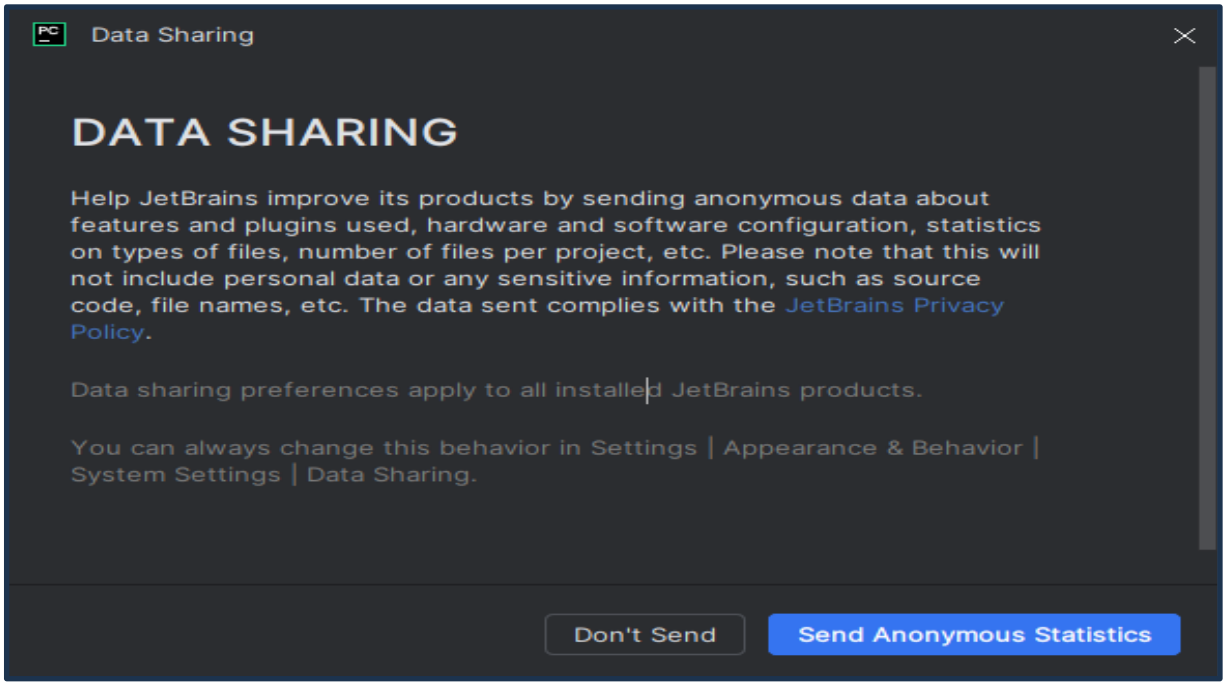

PyCharm Setup
Installation Options
Configure your PyCharm installation
Create Desktop Shortcut
PyCharm
Update PATH Variable (restart needed)
Add "bin" folder to the PATH
Update Context Menu
Add "Open Folder as Project"
Create Associations
.py
< Back
Next >
Cancel

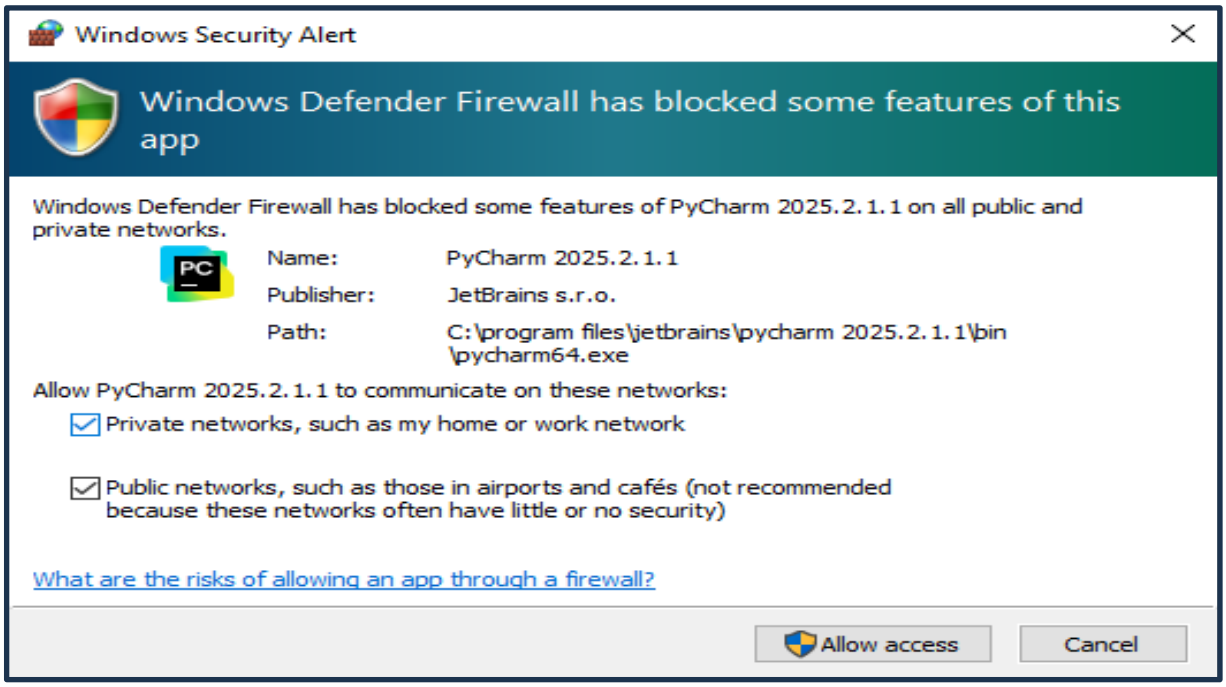

PyCharm Setup
Choose Start Menu Folder
Choose a Start Menu folder for the PyCharm shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
JetBrains
7-Zip
Accessibility
Accessories
Administrative Tools
Internet Download Manager
Maintenance
Microsoft Power BI Desktop
Python 3.13
StartUp
System Tools
Visual Studio 2022
Do not create shortcuts
< Back
Install
Cancel


PyCharm Setup
Installing
Please wait while PyCharm is being installed.
Extract: groovy.jar... 47%
Show details
< Back
Next >
Cancel


PyCharm Setup
Completing PyCharm Setup
PyCharm has been installed on your computer.
Click Finish to close Setup.
Run PyCharm
< Back
Finish
Cancel


PyCharm User Agreement
JETBRAINS USER AGREEMENT
Version 2.0, effective as of April 10, 2025
THIS IS A LEGAL AGREEMENT. BY CLICKING ON THE "I AGREE" (OR SIMILAR) BUTTON THAT IS PRESENTED TO YOU AT THE TIME OF INSTALLATION, OR BY DOWNLOADING, INSTALLING, COPYING, SAVING ON YOUR DEVICE, OR OTHERWISE USING THE JETBRAINS PRODUCT, YOU BECOME A PARTY TO THIS AGREEMENT, YOU DECLARE YOU HAVE THE LEGAL CAPACITY TO ENTER INTO THIS AGREEMENT, AND YOU CONSENT TO BE BOUND BY ITS TERMS AND CONDITIONS.
1. Introduction
This JetBrains User Agreement ("Agreement") is entered into between JetBrains s.r.o., a company with its registered office at Na hřebenech II
I confirm that I have read and accept the terms of this User Agreement
Exit
Continue


Data Sharing
DATA SHARING
Help JetBrains improve its products by sending anonymous data about features and plugins used, hardware and software configuration, statistics on types of files, number of files per project, etc. Please note that this will not include personal data or any sensitive information, such as source code, file names, etc. The data sent complies with the JetBrains Privacy Policy.
Data sharing preferences apply to all installed JetBrains products.
You can always change this behavior in Settings | Appearance & Behavior | System Settings | Data Sharing.
Don't Send
Send Anonymous Statistics

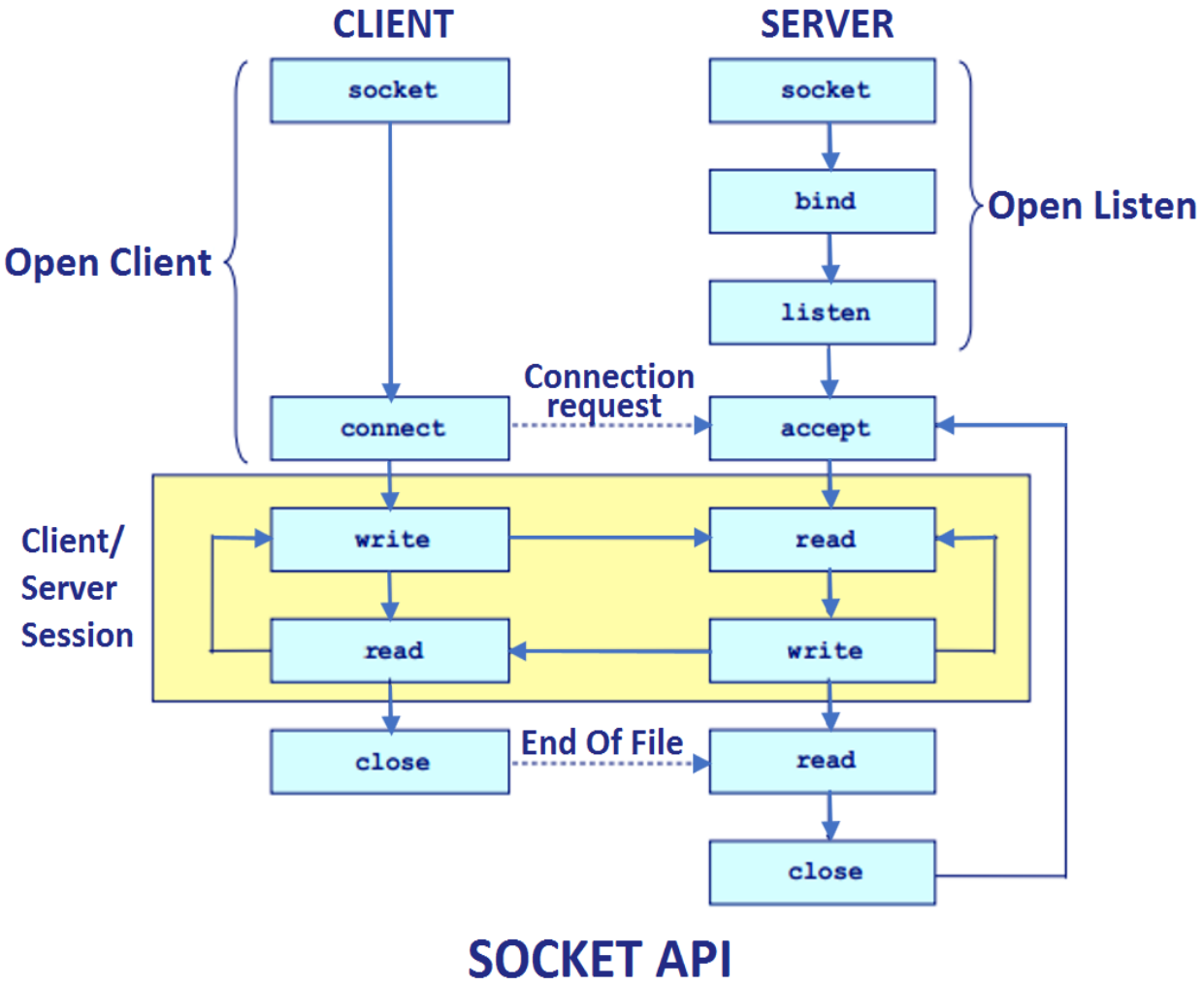

Windows Security Alert
Windows Defender Firewall has blocked some features of this app
Windows Defender Firewall has blocked some features of PyCharm 2025.2.1.1 on all public and private networks.
Name: PyCharm 2025.2.1.1
Publisher: JetBrains s.r.o.
Path: C:\program files\jetbrains\pycharm 2025.2.1.1\bin\pycharm64.exe
Allow PyCharm 2025.2.1.1 to communicate on these networks:
Private networks, such as my home or work network
Public networks, such as those in airports and cafés (not recommended because these networks often have little or no security)
What are the risks of allowing an app through a firewall?
Allow access
Cancel


Welcome to PyCharm
PyCharm
2025.2.1.1
Projects
Customize
Plugins
Learn
Welcome to PyCharm
Create a new project to start from scratch.
Open existing project from disk or version control.
New Notebook
New Script
New Project
Open
Clone Repository
Take a quick onboarding tour
New to PyCharm? Get the most out of your IDE. Become acquainted with its tools and basic workflows.
Start Tour


Welcome to PyCharm
PyCharm
2025.2.1.1
Projects
Customize
Plugins
Learn
Welcome to PyCharm
Create a new project to start from scratch.
Open existing project from disk or version control.
New Notebook
New Script
New Project
Open
Clone Repository
Take a quick onboarding tour
New to PyCharm? Get the most out of your IDE. Become acquainted with its tools and basic workflows.
Start Tour

What are Sockets?

- A socket is an endpoint in a two-way communication link between two programs running on a network.
- Sockets offer a mechanism for inter-process communication by establishing named contact points through which communication takes place.
- Sockets enable different devices and applications to exchange data over a network, whether it be a local network or the internet.



- The sockets API(Application Programming Interface) provides an interface for porgrams to communicate with eachother over the TCP/IP protocol suite.

- The sockets API hides or abstarcts away the underlying complexities of the network stack (network layers) and provides us with a clean interface for creating network applications.
- Sockets can use either TCP or UDP for transporting data.
- Sockets are used to program both server and client applications.

Python Socket API Overview

Python's socket module provides an interface to the Berkeley sockets API. This is the module that we will use.

The primary socket API functions and methods we will be using are:

- socket()
- .bind()
- .listen()
- .accept()
- .connect()
- .connect_ex()
- .send()
- .recv()
- .close()

Python socket module functions reference

1. Creating a Socket:

```
import socket

# Create a TCP/IP socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Create a UDP socket
s_udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

- `socket.socket(family, type, proto)`: This constructor creates a new socket object.
  - `family`: Specifies the address family, e.g., `socket.AF_INET` for IPv4, `socket.AF_INET6` for IPv6.
  - `type`: Specifies the socket type, e.g., `socket.SOCK_STREAM` for TCP (connection-oriented), `socket.SOCK_DGRAM` for UDP (connectionless).
  - `proto`: Specifies the protocol, usually not needed as it's automatically chosen by `type`.

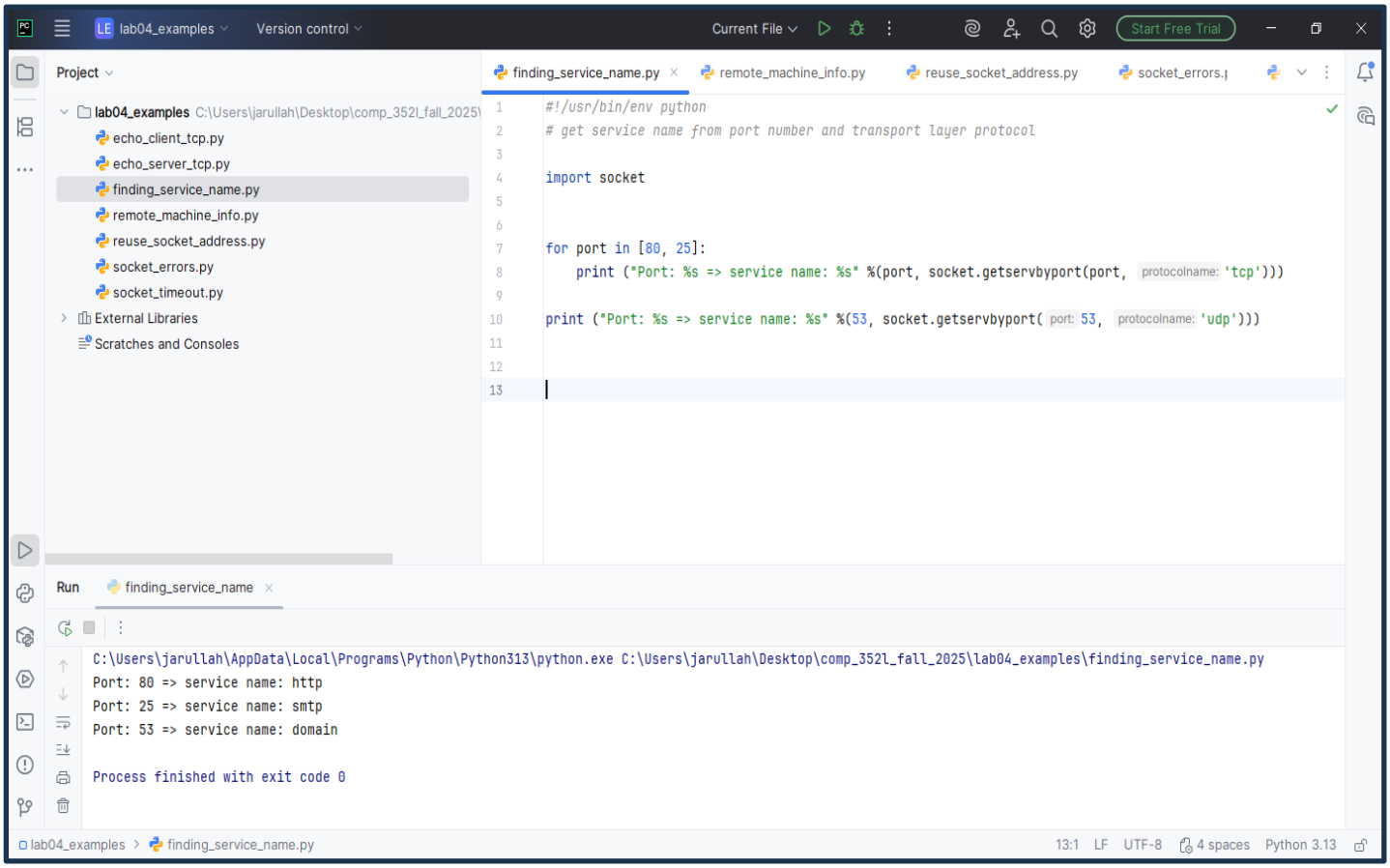
2. Server-Side Operations:
   - `s.bind((host, port))`: Binds the socket to a specific address and port.
   - `s.listen(backlog)`: Puts the server socket into listening mode, waiting for incoming connections. `backlog` specifies the maximum number of queued connections.
   - `conn, addr = s.accept()`: Accepts a new connection. Returns a new socket object `conn` representing the connection to the client, and the client's address `addr`.
3. Client-Side Operations:
   - `s.connect((host, port))`: Connects the client socket to a remote server at the specified address and port.
4. Data Transfer:
   - `s.send(data)`: Sends data over a connected socket.
   - `s.sendall(data)`: Sends all data over a connected socket, repeatedly calling `send` until all data is sent.
   - `s.recv(bufsize)`: Receives data from a connected socket. `bufsize` is the maximum amount of data to receive at once.
   - `s.sendto(data, address)`: Sends data to a specific address using a UDP socket.
   - `data, address = s.recvfrom(bufsize)`: Receives data and the sender's address from a UDP socket.

5. Socket Options and Configuration:
   - `s.settimeout(timeout)`: Sets a timeout for blocking socket operations.
   - `s.setblocking(flag)`: Sets the socket to blocking (`True`) or non-blocking (`False`) mode.
   - `s.setsockopt(level, optname, value)`: Sets the value of a given socket option.
   - `s.getsockopt(level, optname, buflen)`: Returns the value of a given socket option.
6. Utility Functions:
   - `socket.gethostname()`: Returns the local hostname.
   - `socket.gethostbyname(hostname)`: Translates a hostname to an IPv4 address.
   - `socket.getaddrinfo(host, port, family=0, type=0, proto=0, flags=0)`: Translates host/port arguments into a sequence of 5-tuples for creating sockets.
   - `s.close()`: Closes the socket.

Example Programs

finding_service_name.py

```python
#!/usr/bin/env python
# get service name from port number and transport layer protocol

import socket


for port in [80, 25]:
    print ("Port: %s => service name: %s" %(port, socket.getservbyport(port, 'tcp')))

print ("Port: %s => service name: %s" %(53, socket.getservbyport(53, 'udp')))
```

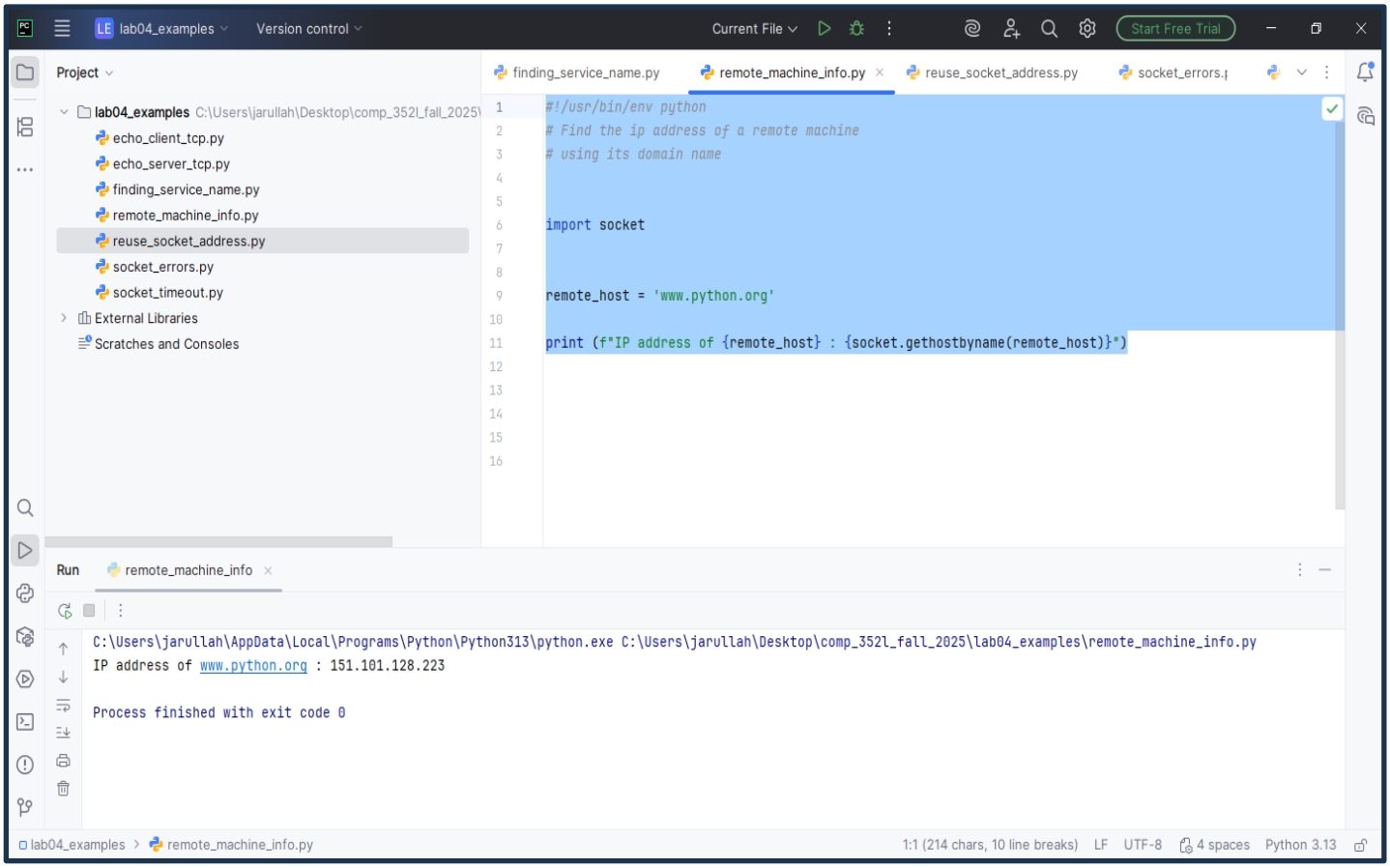


Task 1:	Get service name for 10 different port numbers.

remote_machine_info.py

```python
#!/usr/bin/env python
# Find the ip address of a remote machine
# using its domain name
```

```python
import socket

remote_host = 'www.python.org'

print (f"IP address of {remote_host} : {socket.gethostbyname(remote_host)}")
```



Task 2:	Find the IP address of 10 different domains.

A Trivial TCP server and client

"A tirvial server terminates after responding to a single request."

echo_server.py

```
# A Trivial TCP Server using sockets
# It receives a message from a client.
# Echoes the message back and terminates.

import socket


host = 'localhost'
data_payload = 2048 # how many bytes to receive
port = 8000



# Create a TCP socket
# AF.INET: IPv4, SOCK_STREAM: TCP
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Enable reuse address/port
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
# Bind the socket to the port
server_address = (host, port)
print (f"Starting up echo server  on {host} port {port}")
sock.bind(server_address)
# Listen to clients
sock.listen()

print ("Waiting to receive message from client")
client, address = sock.accept()
data = client.recv(data_payload)

print (f"Data: {data}")
client.send(data)
print (f"sent {len(data)} bytes back to {address}")
# end connection
client.close()
sock.close()
```

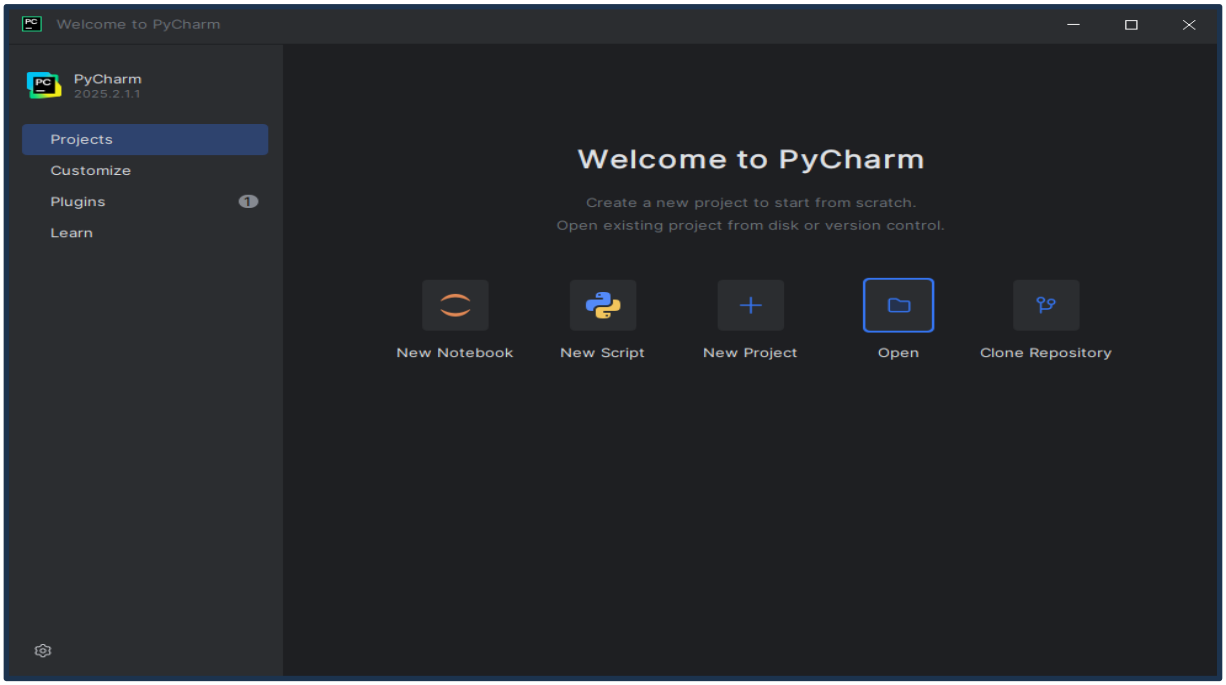
echo_client.py

```python
# A simple TCP client using sockets
# Establishes a connection to a server.
# Sends a message.
# Receives a reply and terminates.


import socket


host = 'localhost'
port = int(input("Enter server port: "))

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# Connect the socket to the server
server_address = (host, port)
print (f"Connecting to {host} port {port}")
sock.connect(server_address)

# Send data

# Send data
message = "Test message. This will be echoed"
print (f"Sending {message}")
sock.sendall(message.encode('utf-8'))
# Look for the response

data = sock.recv(256)

print (f"Received: {data}")


print ("Closing connection to the server")
sock.close()
```

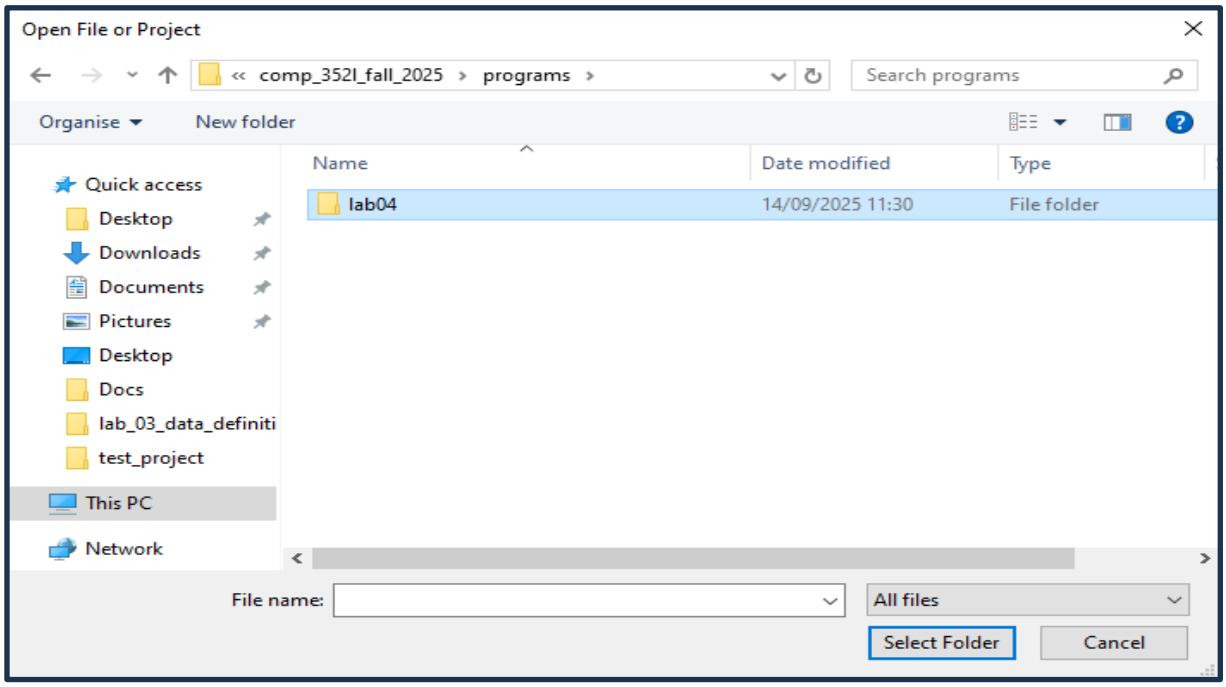
How to run these programs

1. Open Pycharm

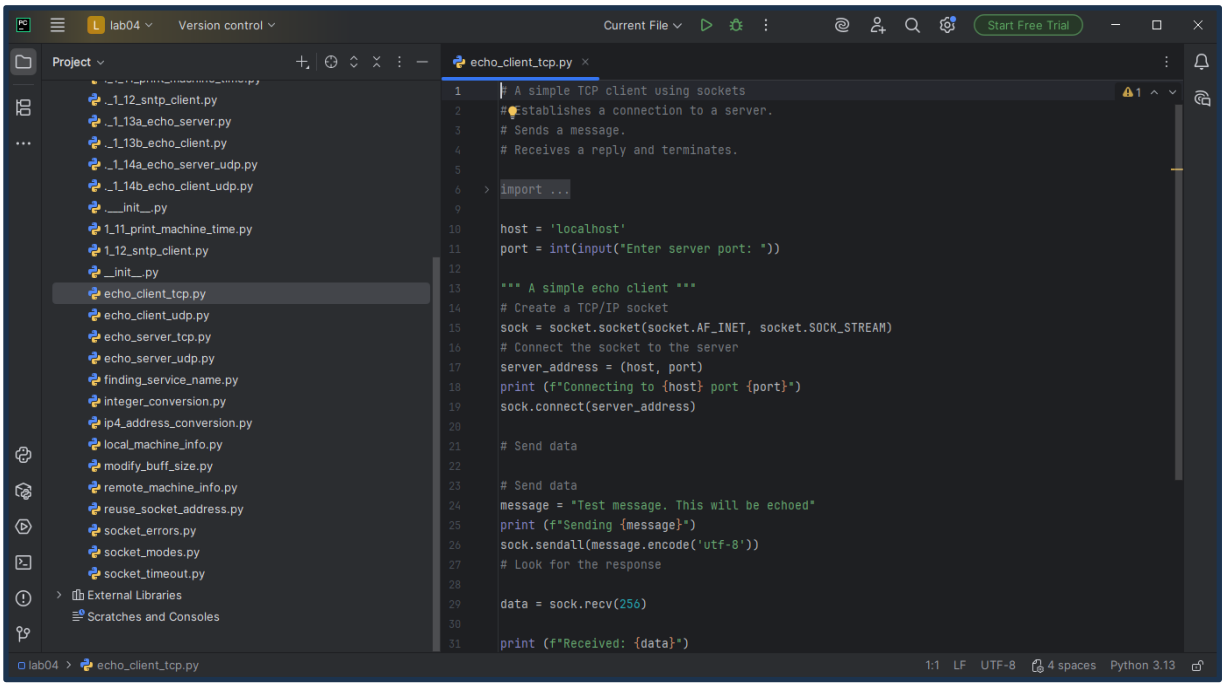
2. **Select Open**

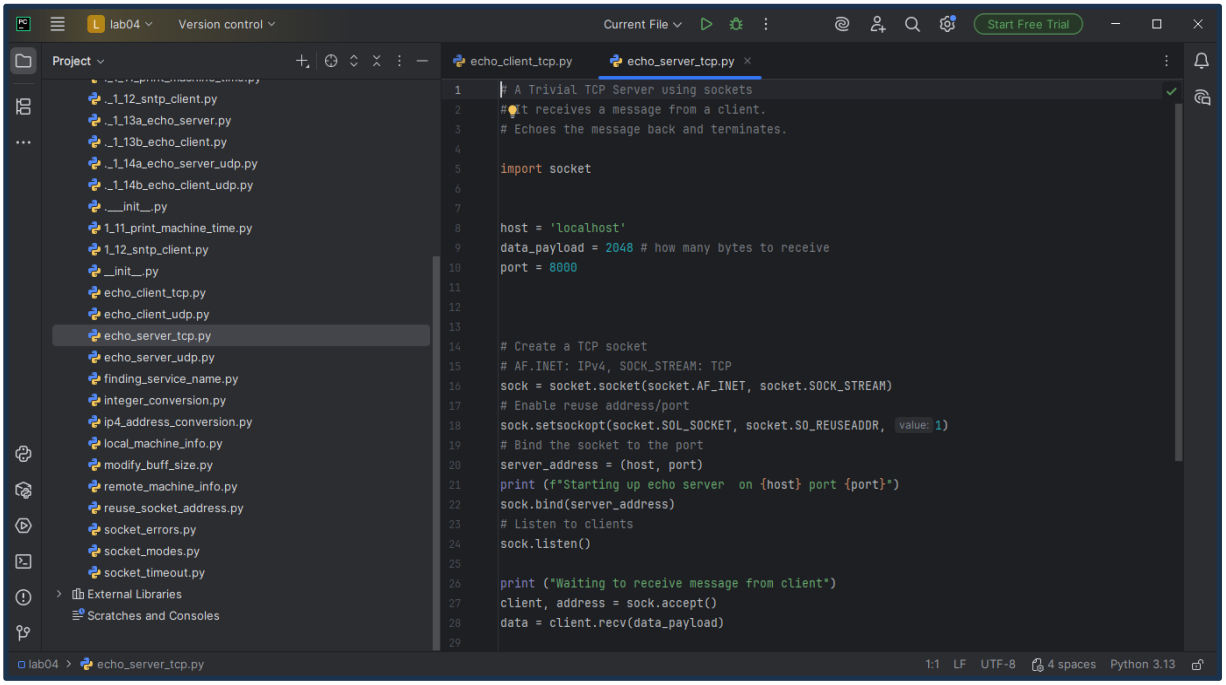


3. Select the folder containing python source files.

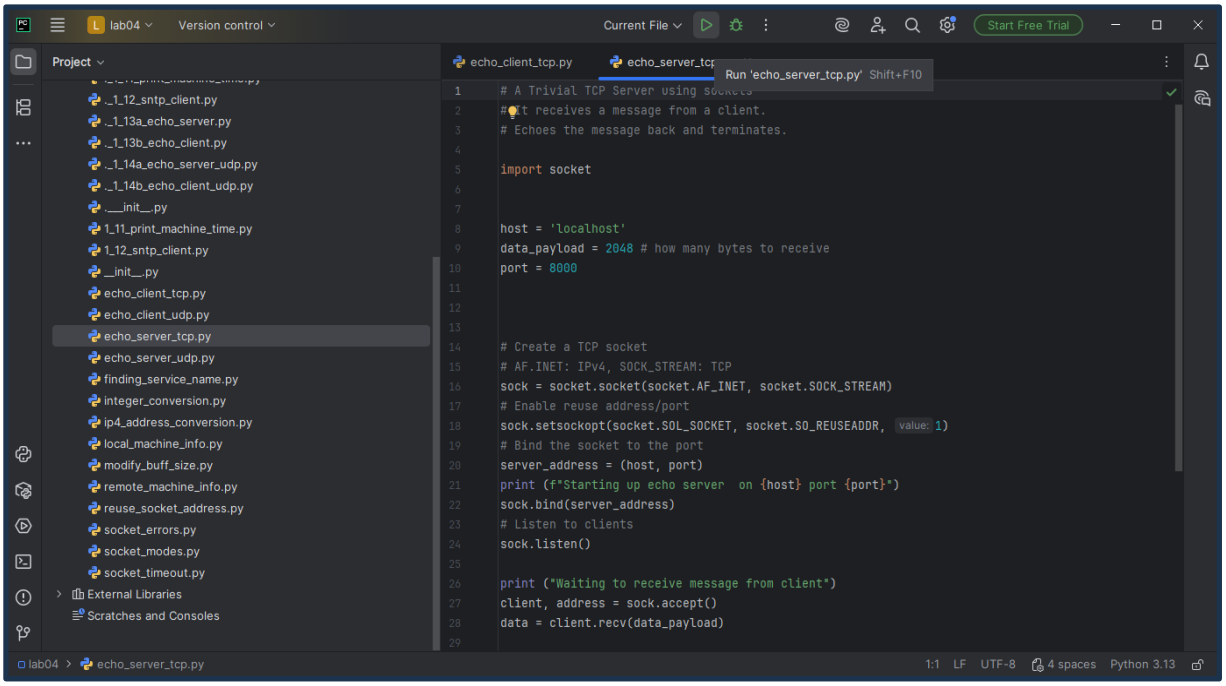


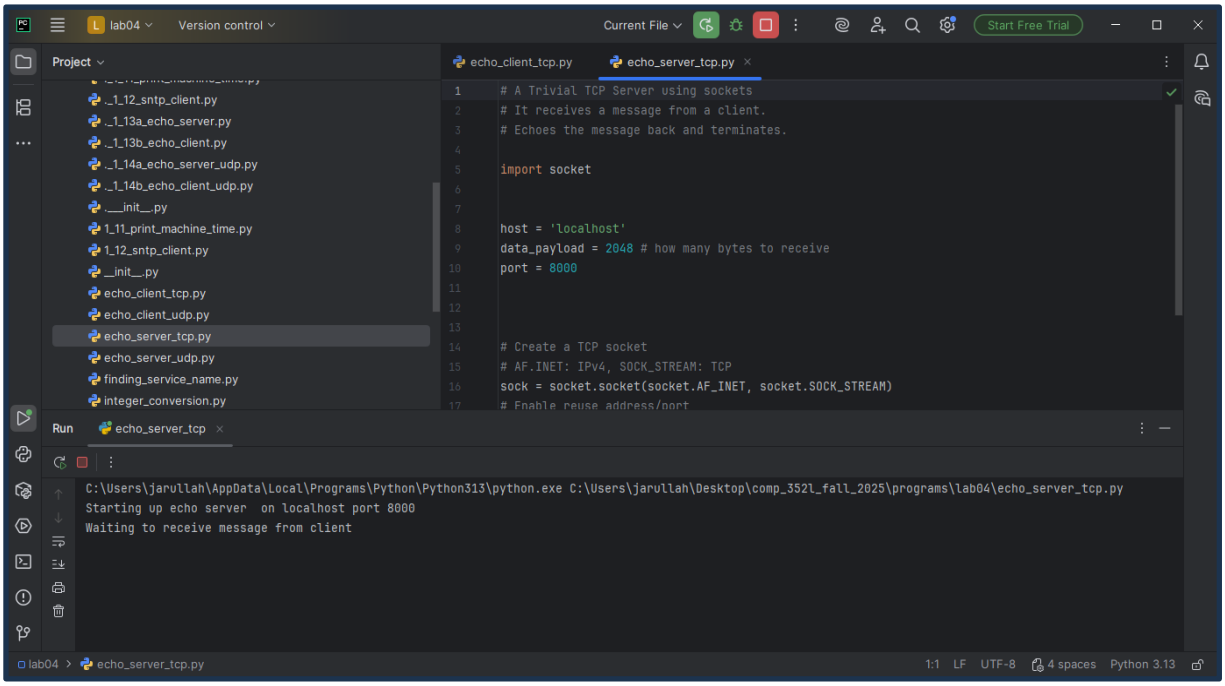
4. Double click on the desired source file to open it in the editor (open both the *`echo_server_tcp.py`* and *`echo_client_tcp.py`* files in the editor)



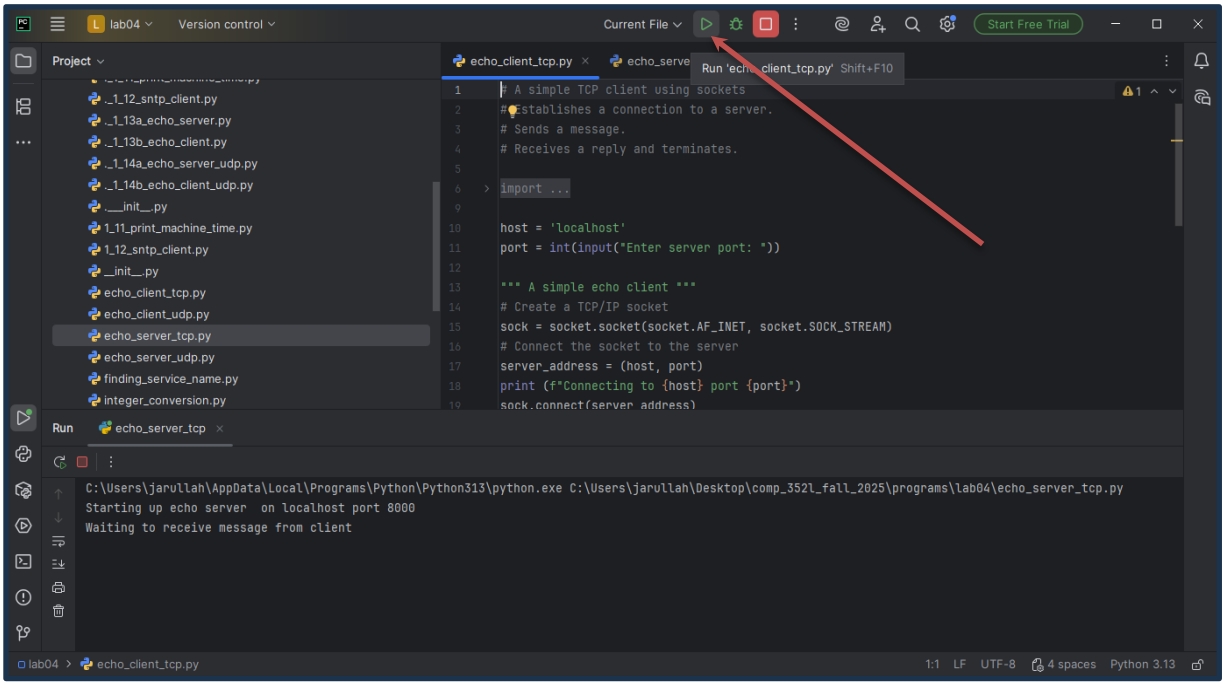


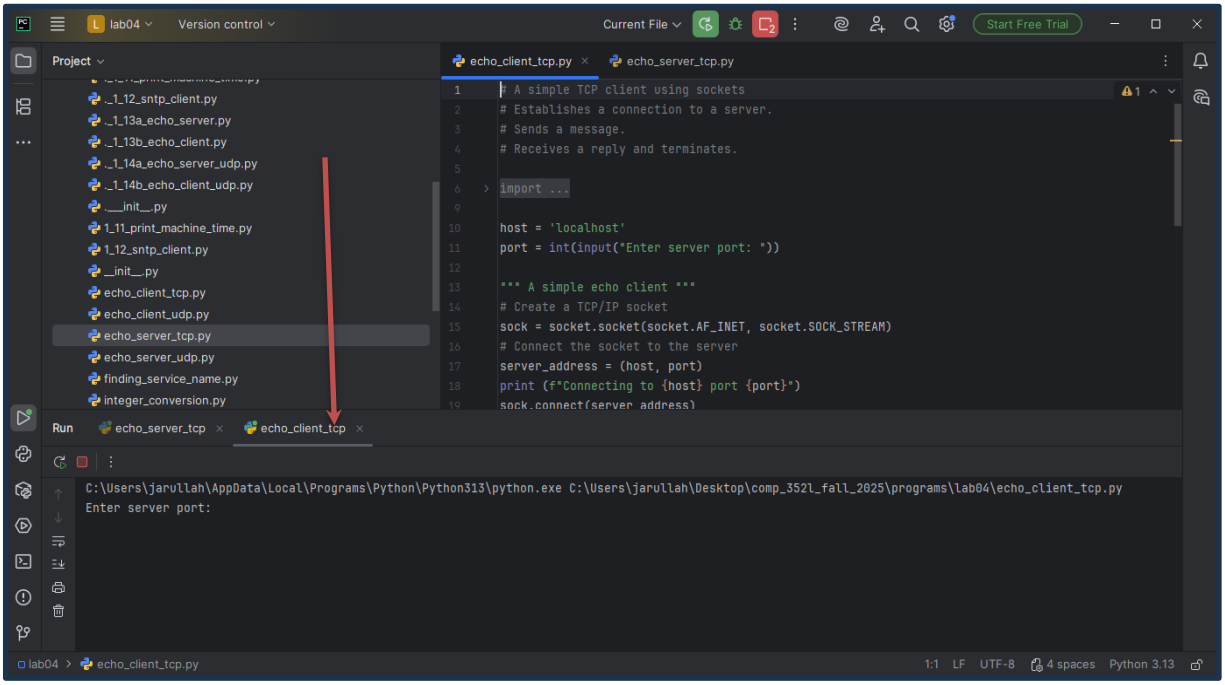
5. Select the 'echo_server_tcp.py' file in the editor and click the run button.



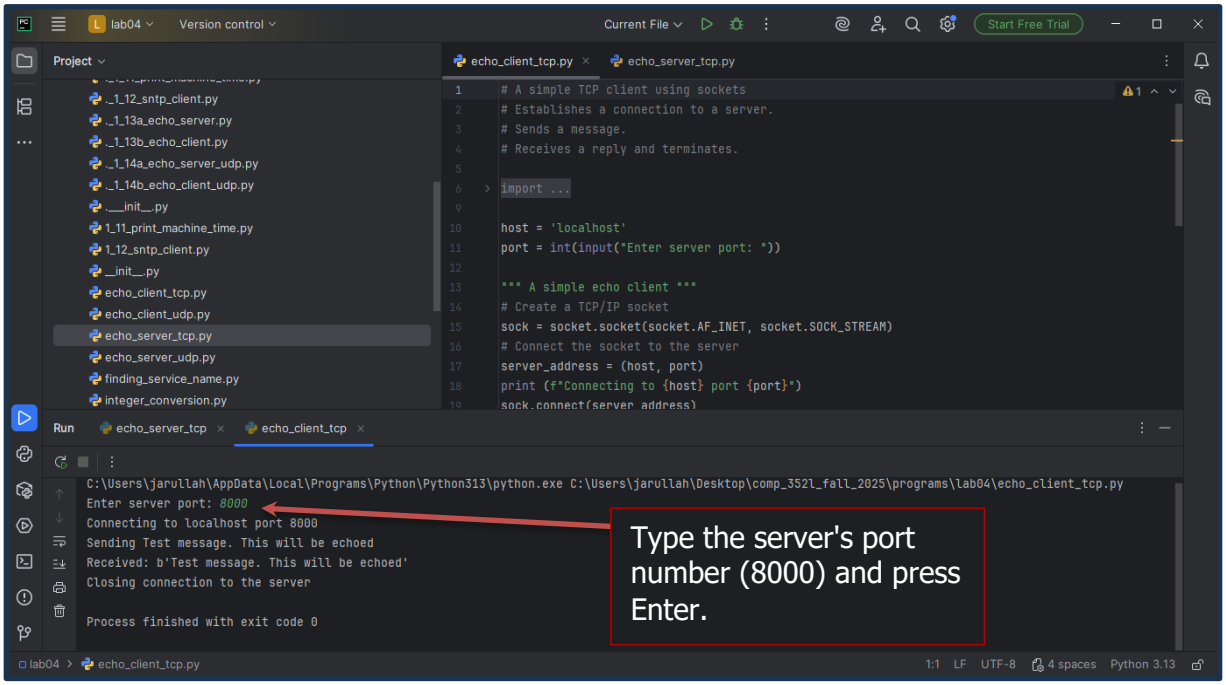


6. Now select the 'echo_client_tcp.py' file in the editor and click the run button.





7. **There are two tabs in the window showing running programs. Select the client program's tab. The prompt is asking for a port number. Type the server's port number (8000 in this case) and press Enter.**



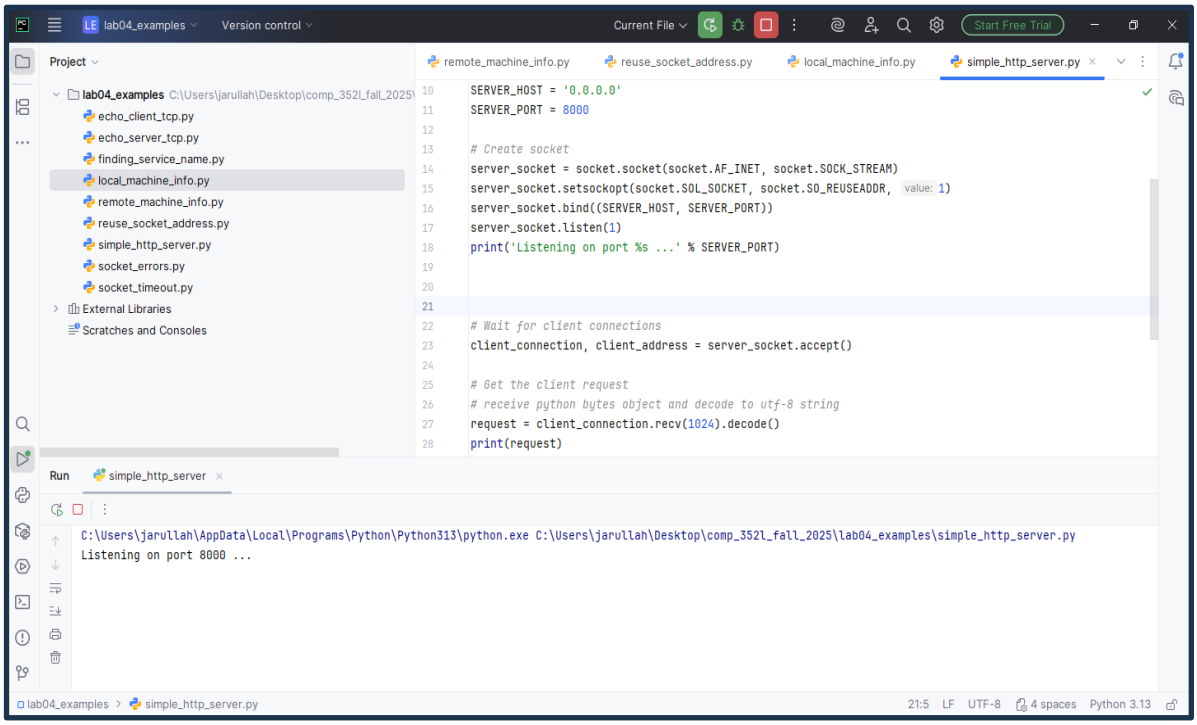
8. **Both the client and server programs exchange messages and terminate.**

Task 3:	Modify the client so that the user provides the message to be sent to the server.
Task 4:	Modify the server to display the address of the machine it is running on.
Task 5:	Modify the client and server programs so that: The client sends a message to the server. If the message contains the user's name, the server responds with a greeting message containing the user's name. If the user's name is not in the message, the server responds with a message like 'Unknown user' Test your program with and without user's name in client message.

simple_http_server.py

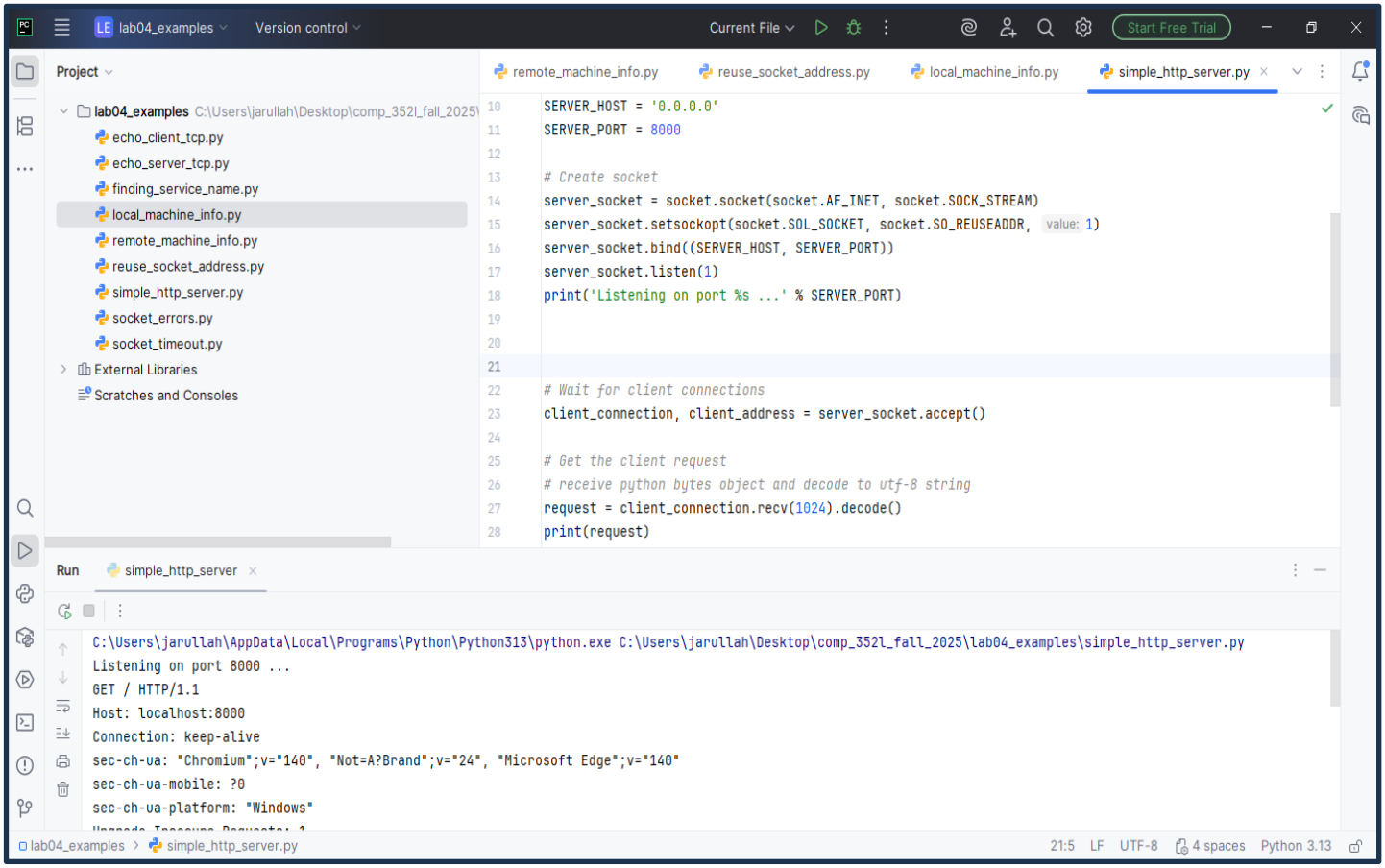
```
"""
 Implements a simple HTTP/1.0 Server

"""

import socket


# Define socket host and port
SERVER_HOST = '0.0.0.0'
SERVER_PORT = 8000

# Create socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server_socket.bind((SERVER_HOST, SERVER_PORT))
server_socket.listen(1)
print('Listening on port %s ...' % SERVER_PORT)



# Wait for client connections
client_connection, client_address = server_socket.accept()

# Get the client request
# receive python bytes object and decode to utf-8 string
request = client_connection.recv(1024).decode()
print(request)

# Send HTTP response
response = 'HTTP/1.0 200 OK\n\nHello World'
# Encode string to byte object (utf-8)
client_connection.sendall(response.encode())
client_connection.close()

# Close socket
server_socket.close()
```

- Run the Server



- Open your browser and navigate to *'localhost:8000'*.

- Check the server's output in PyCharm.



Task:	Modify the simple_http_server.py so that it sends a proper html response along with the status code. (i.e. instead of just 'Hello World' in the response the server should send something like the following: <html> <head> <title> Hello Server<title> <head> <body> <h1> Hello World ! <\h1> <h2> A trivial server using sockets API with Python! <h2> <\body>)