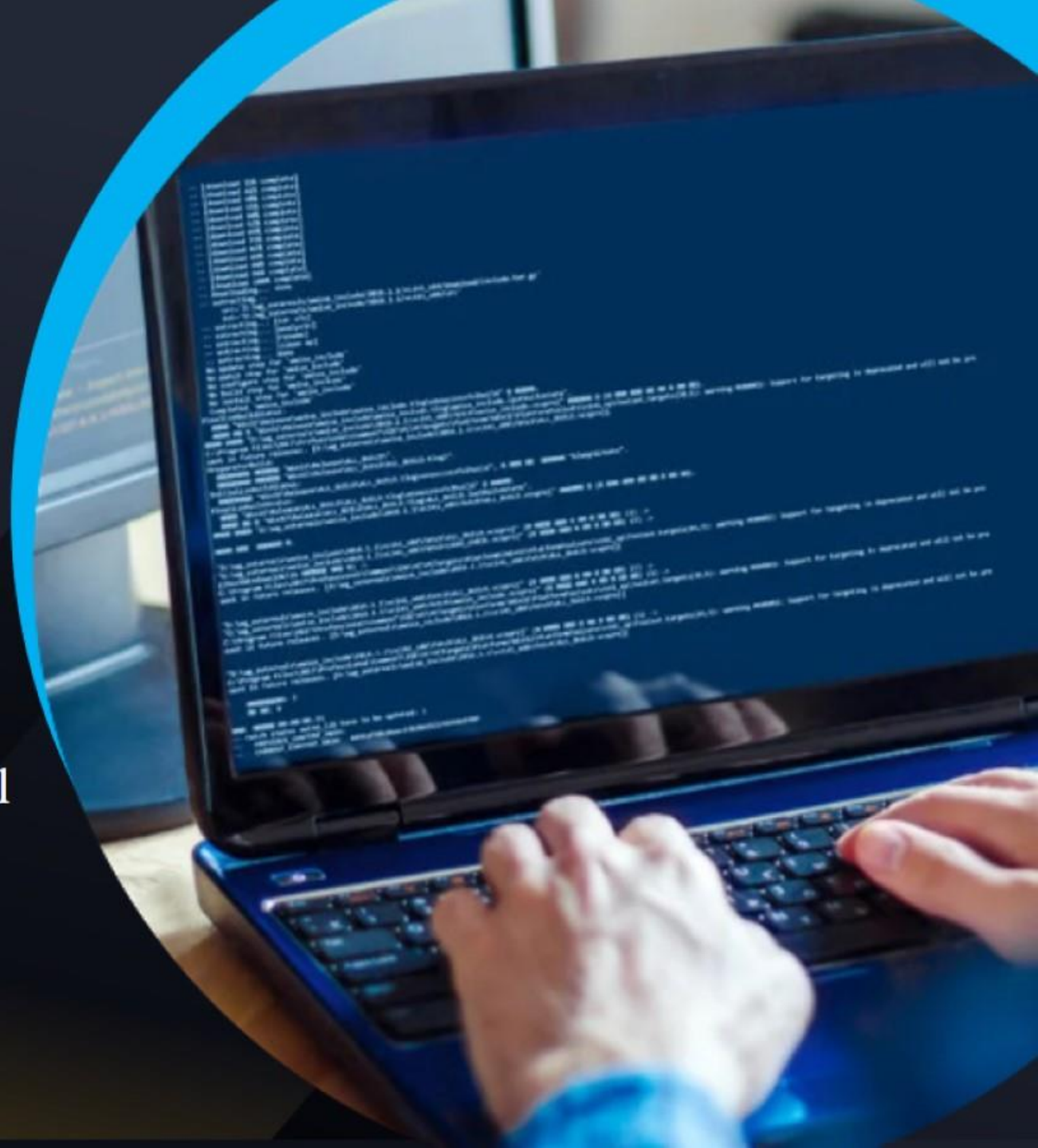


Network Programming in Python

Prepared by Dr. Hamada I. AbdulWakel

Lecture 1





Basics of Computer Networking

Lecture 1

Prepared by
Dr. Hamada I. AbdulWakel

Lecture

01

LECTURE MAIN POINTS

- Network definition.
- Importance of networks.
- Networking components and types.
- Network programming and programming levels.
- Dealing with virtual environments.

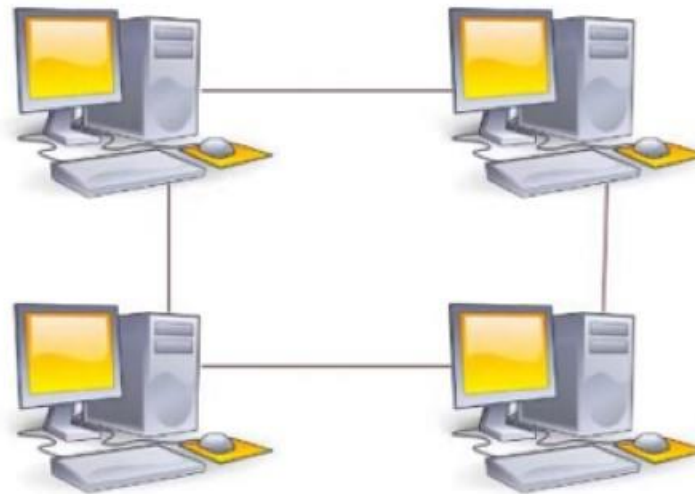
Lecture

01

NETWORK DEFINITION

- A collection of devices connected to one another allowing for data to be shared and used.

- Devices count.
 - At least 2 devices.
- Device type:
 - Any device.
- Connection type:
 - Wired.
 - Wireless.

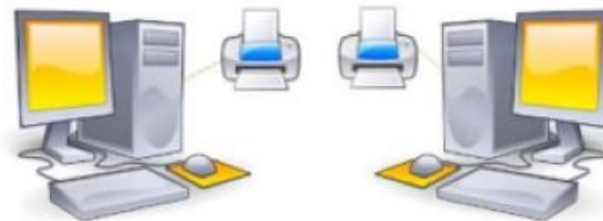
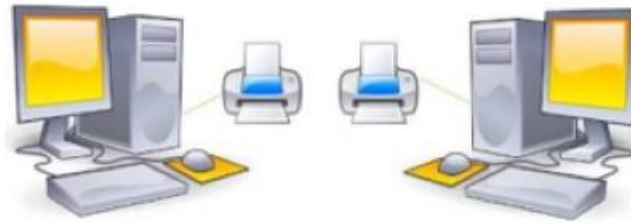


Lecture

01

IMPORTANCE OF NETWORKS

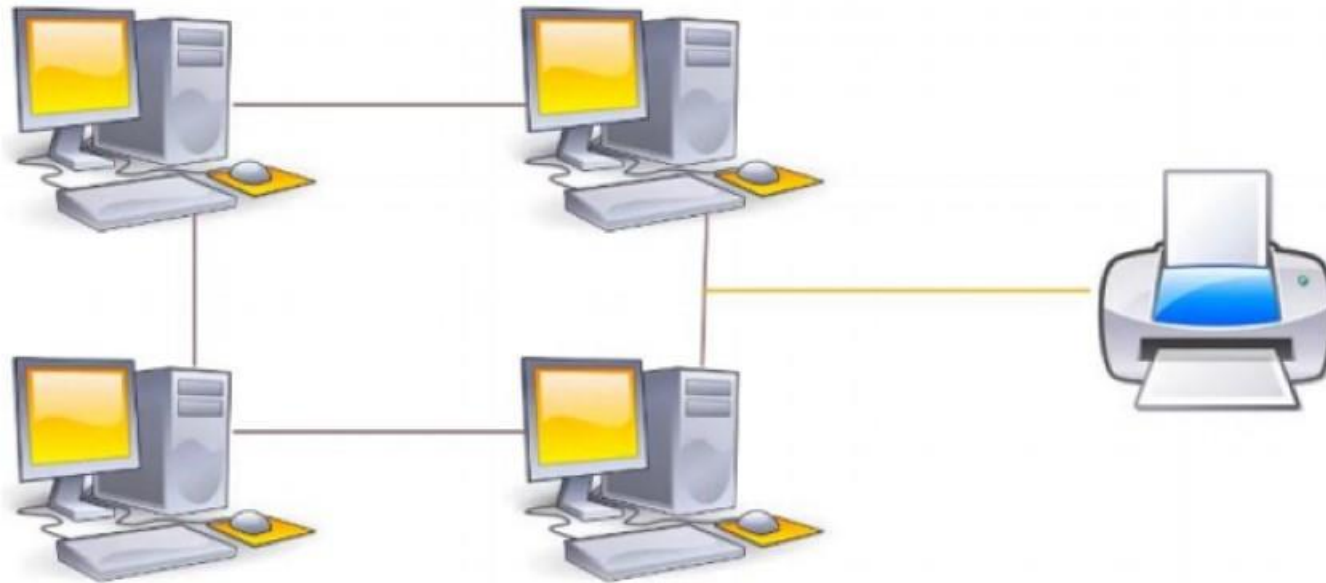
- Scenario 1: without networks
 - Violating efficient use of resources.



IMPORTANCE OF NETWORKS

Cont'd

- Scenario 1: with networks
 - Efficient use of resources.



IMPORTANCE OF NETWORKS

Cont'd

- Scenario 2: without networks
 - Reducing teamwork collaboration and data synchronization.



IMPORTANCE OF NETWORKS

Cont'd

- Scenario 2: with networks
 - Increasing teamwork collaboration and data synchronization.



Lecture

01

IMPORTANCE OF NETWORKS

Cont'd

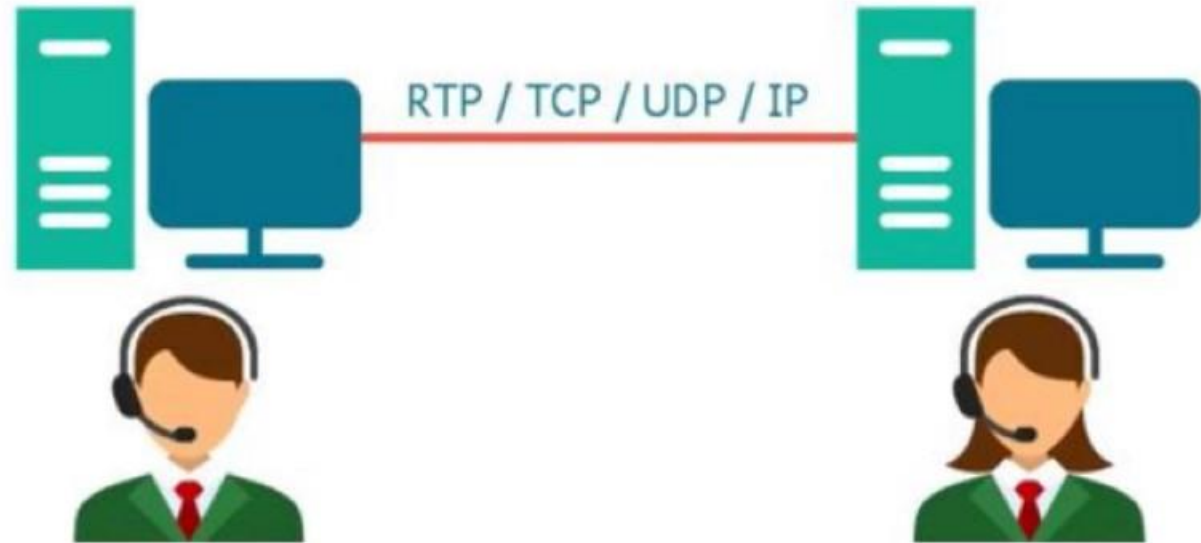
- Share resources such as Data and Hardware.
- Save a lot of money.
- Increase workgroups collaborations.
- Increase productivity.
- Protect company data.

Lecture

01

NETWORK COMPONENTS

- Any network must have:
 - Computer systems.
 - Network media.
 - Cables:
 - Electrical signals.
 - Pulses of light.
 - Wireless:
 - Radio frequencies.



Lecture

01

NETWORK COMPONENTS

Cont'd

- Any network must have:
 - Network interface.
 - A point of interconnection between a computer and a network.
 - Physical form or implemented in software.
 - Convert data to suitable formats for media used.



NETWORK COMPONENTS

Cont'd

- Any network must have:
 - Network protocol.
 - **A protocol** is a standard rules for how the connected devices are communicate and exchange information.
 - IP: a set of standards for addressing and routing data on the Internet.

106.241.27.244



133.83.189.60



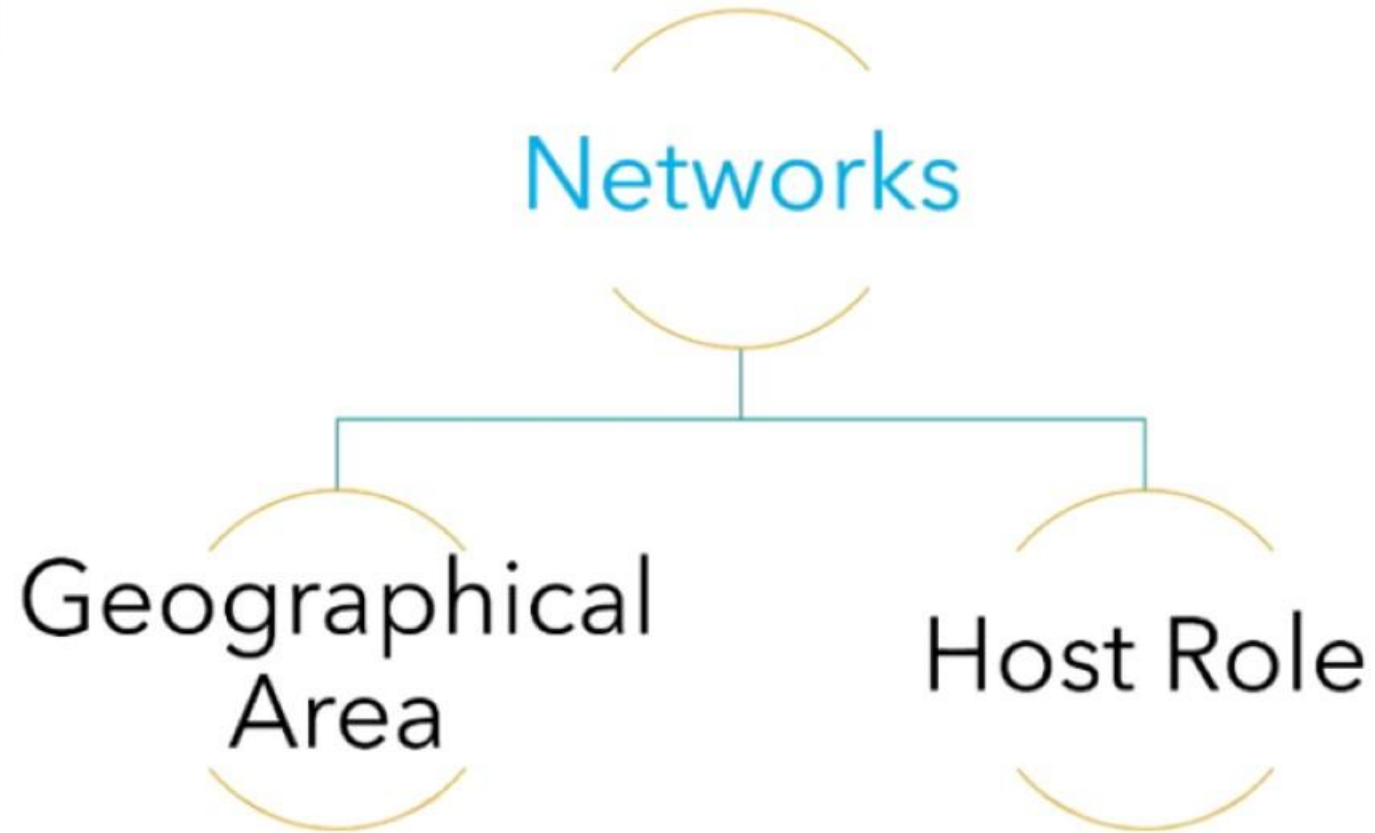
240.101.163.19



Lecture

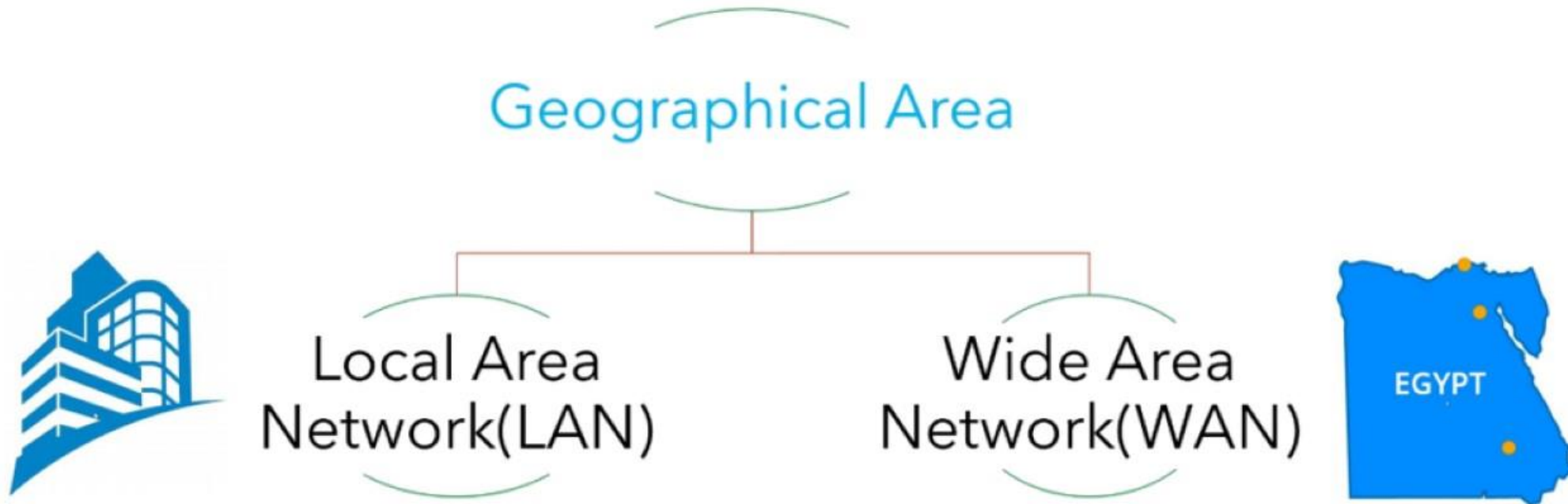
01

NETWORK TYPES



NETWORK TYPES

Cont'd



NETWORK TYPES

Cont'd

- The LAN network features:
 - Short for Local Area Network.
 - Connects a group of computers within a limited geographic area such as school, university, or hospital.
 - High bandwidth for data transfer.
 - Owned by private companies or individuals.
 - Limited to 100 to 1000 meters.
 - Lower setup cost due to inexpensive devices.

NETWORK TYPES

Cont'd

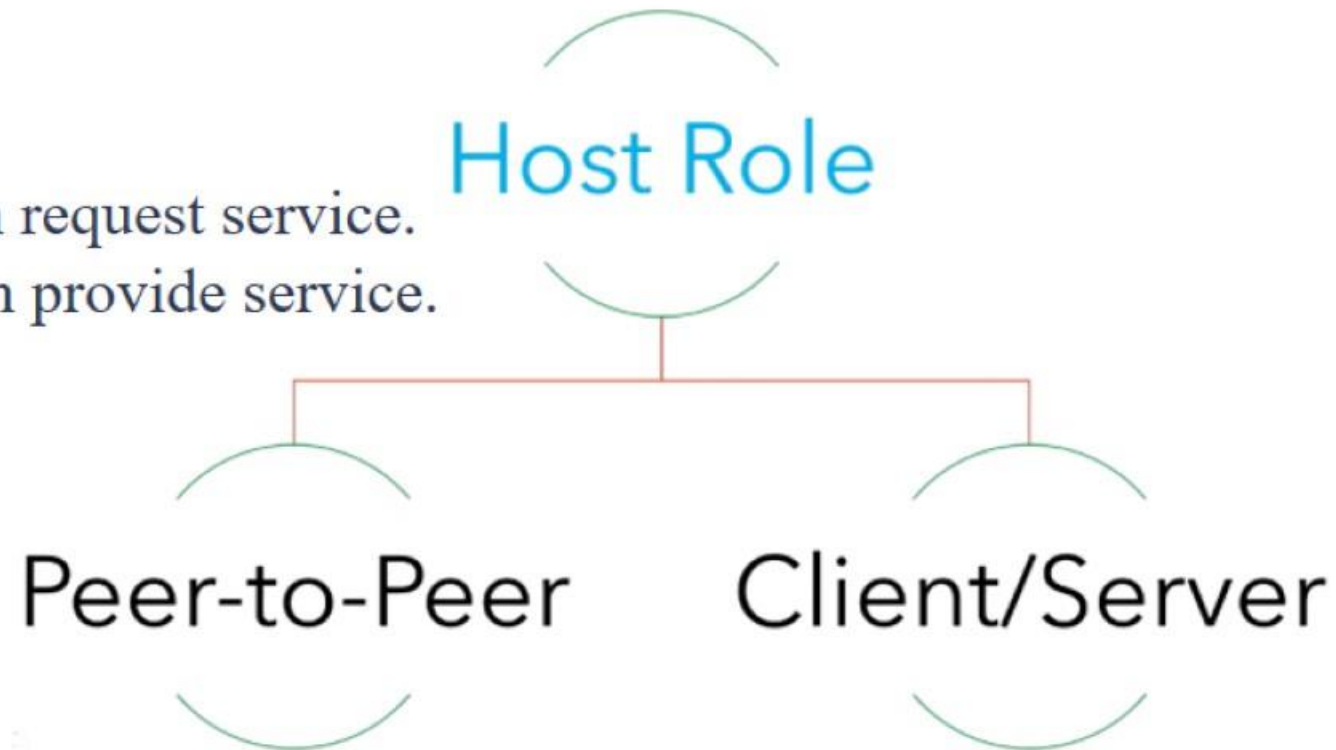
- The WAN network features:
 - Short for Wide Area Network.
 - Covers a large geographic area such as state, or country.
 - Low bandwidth for data transfer.
 - Established under distributed ownership.
 - Spans a huge area of 100000 kilometers.
 - Higher setup cost than LAN.

NETWORK TYPES

Cont'd

Host Roles:

- 1- Client which request service.
- 2- Server which provide service.



NETWORK TYPES

Cont'd

Peer-to-Peer



Lecture

01

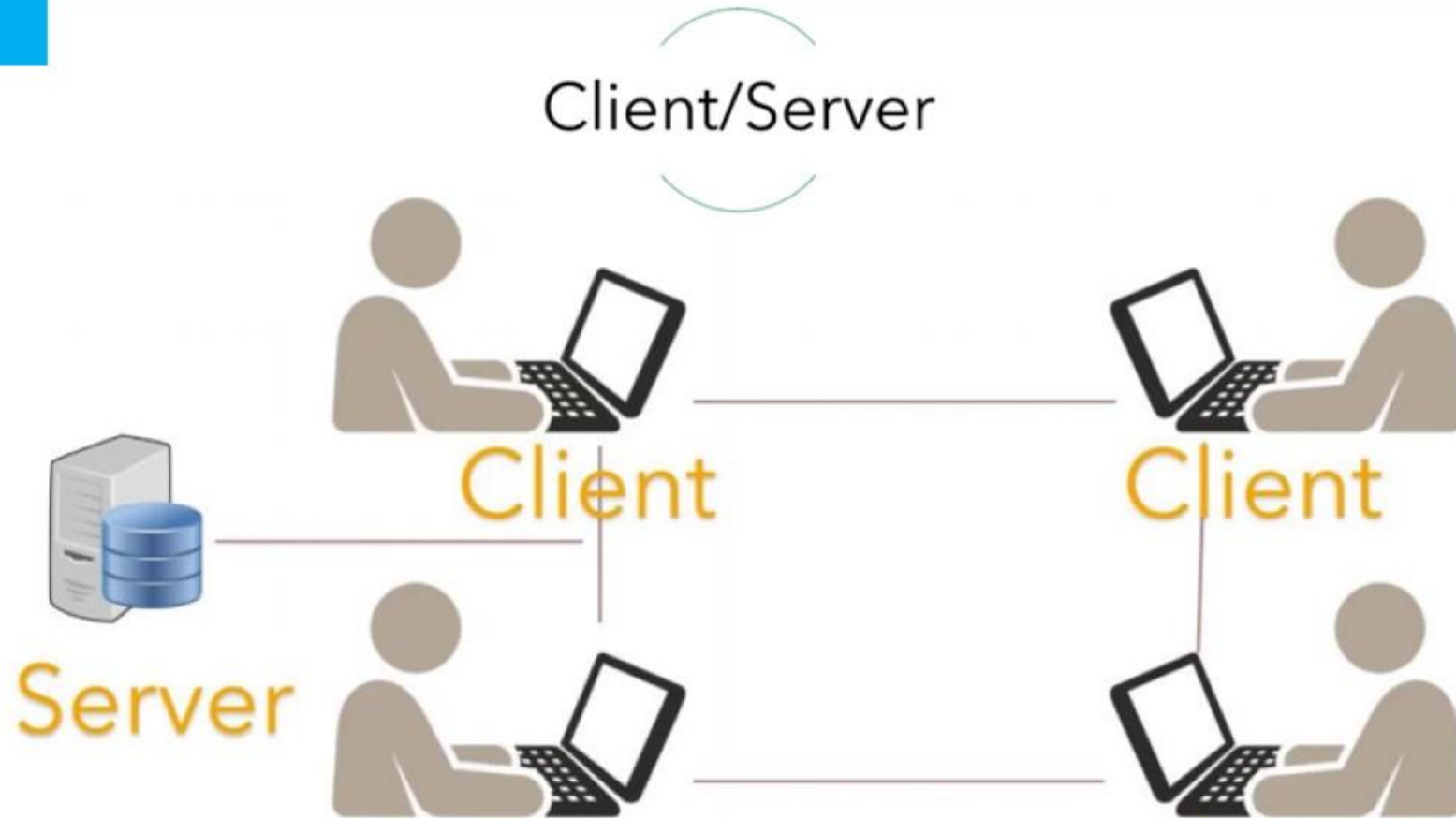
NETWORK TYPES

Cont'd

- The peer-to-peer network features:
 - Simple network forms.
 - Less costs to forming.
 - Roles could be switched.
 - No centralized administration.
 - Security is its main problem.

NETWORK TYPES

Cont'd



NETWORK TYPES

Cont'd

- The client/server network features:
 - Scalable network forms.
 - The server costs is high and requires experience to manage.
 - Roles do not switch.
 - The server considers the centralized administration.
 - Security maintained by the server.

Lecture

01

NETWORK PROGRAMMING

- A network allows arbitrary applications to communicate.
- Network programming is the act of using **computer code** to write **programs** that can **communicate** with other programs across a **network**.
- A **"programmable" network** has programs in its infrastructure that the developers can use to program applications and other components to interact directly with the network.

Lecture

01

REASONS FOR USING PYTHON ON NETWORK PROGRAMMING

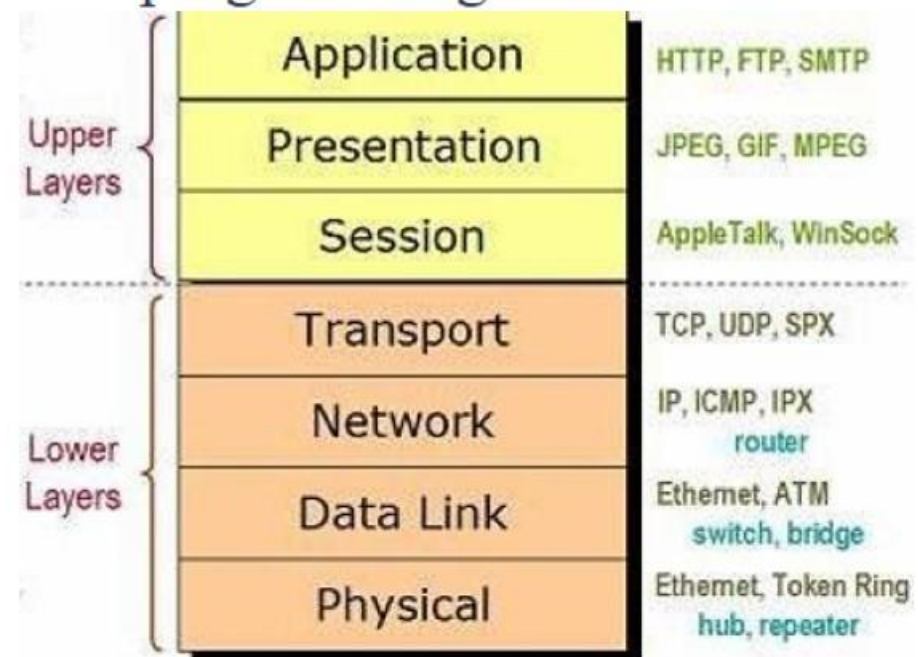
- The simplicity and high readability.
- The most powerful language.
- Helpful libraries for network engineers.
- Python increases network reliability.

Lecture

01

NETWORK PROGRAMMING
LEVELS

- Python provides two levels of access to network programming:
 - **Low-level access.**
 - **High-level access.**



NETWORK PROGRAMMING LEVELS

Cont'd

- Python provides two levels of access to network programming:
 - **Low-level access:**
 - Access the basic `socket` support for the operating system using Python's libraries.
 - Implement client and server for both connection-oriented and connectionless protocols.
 - **High-level access:**
 - Implement protocols like HTTP, FTP, etc. using Python's libraries.

Lecture

01

REQUIREMENTS FOR PYTHON NETWORK PROGRAMMING

- Python basics and OOP.
- PyPi [website](#) to find, install Python packages.
- Integrated Development Environment such as [VScode](#) or [Pycharm](#).

Lecture

01

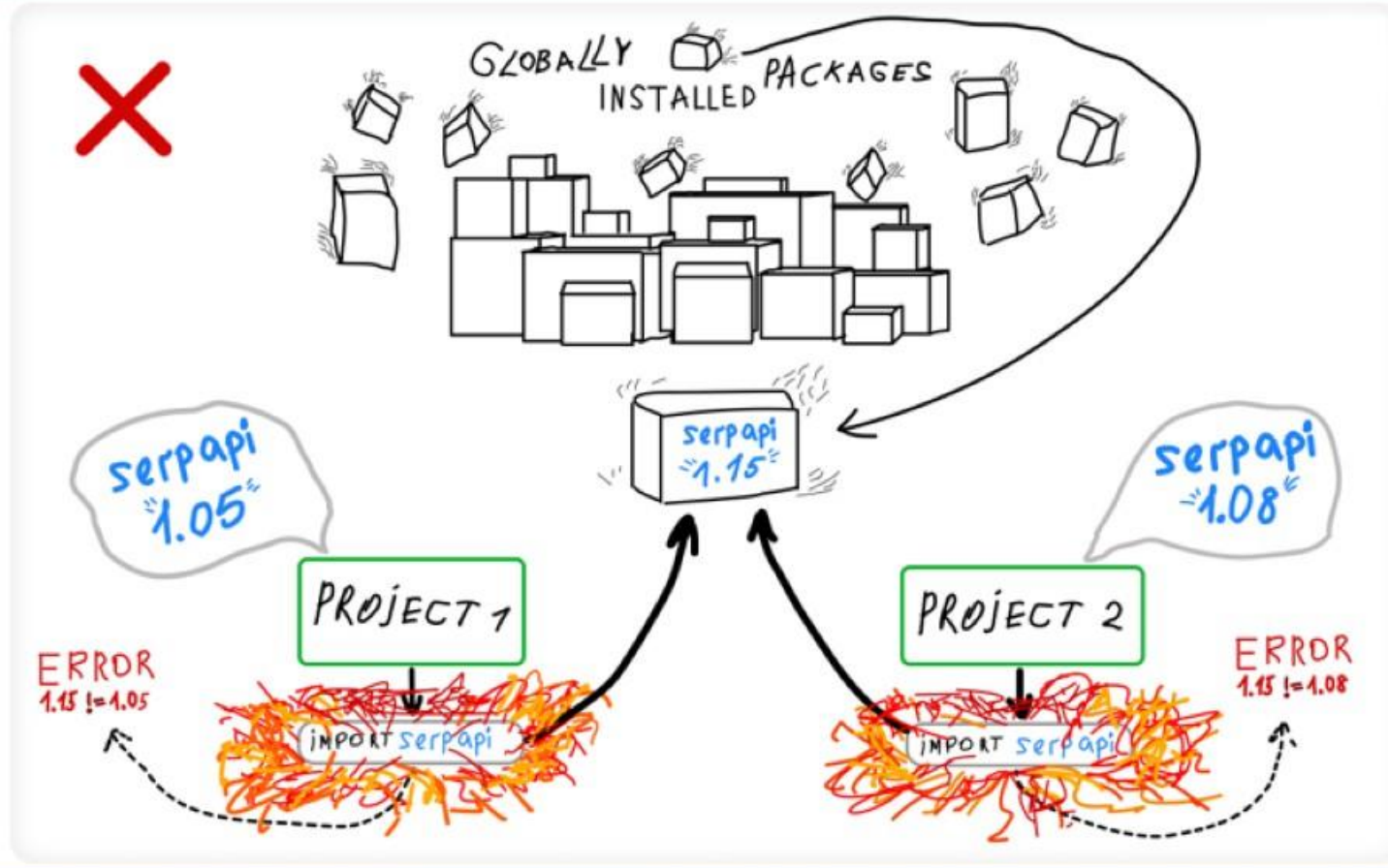
REQUIREMENTS FOR PYTHON
NETWORK PROGRAMMING

- Python package installer (pip).
 - Linux OS:
`sudo apt install python3-pip`
 - Windows OS steps:
 - Download **get-pip.py** then open **CMD** and run:
 - `curl -sS https://bootstrap.pypa.io/get-pip.py -o get-pip.py` (OR here)
 - Run the Script:
 - `python get-pip.py`
 - Verify Installation:
 - `pip --version`

Lecture

01

VIRTUAL ENVIRONMENTS



VIRTUAL ENVIRONMENTS

Cont'd

- Python considers a package-based language.
- When a package is installed globally, it's made **available to all users**. Typically, that means Python and all packages will get installed to a directory under:
 - On Linux:
`/usr/local/bin/`
 - On Windows:
`\Program Files\`

VIRTUAL ENVIRONMENTS

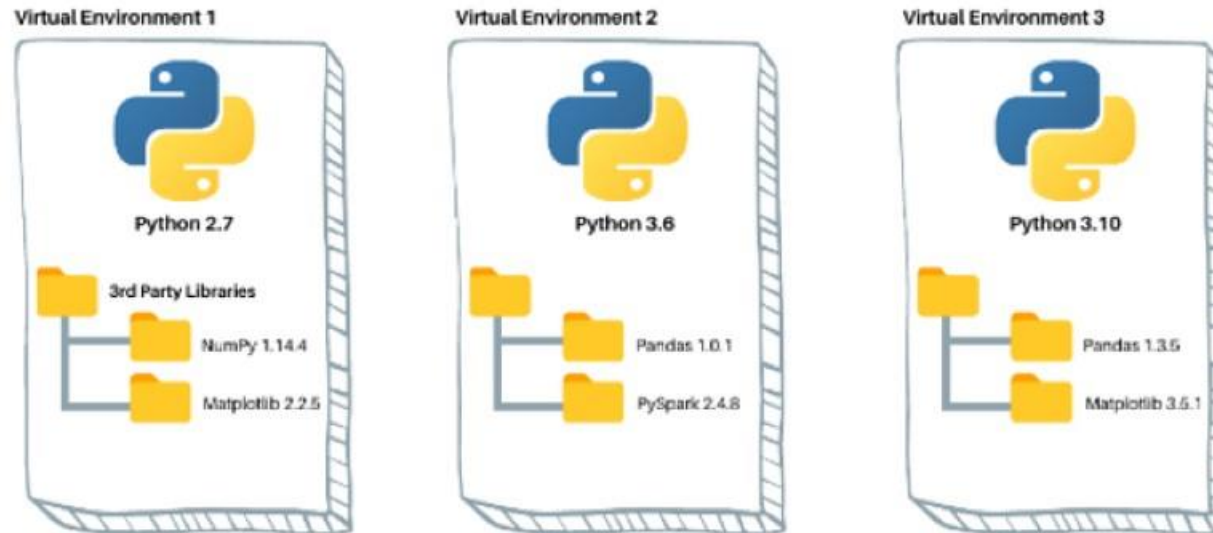
Cont'd

- **Virtual environment (Venv)** is a tool that helps to keep dependencies required by different projects separate by creating **isolated** python virtual environments for them.
- **Isolation** means any changes on dependencies installed in a virtual environment don't affect the dependencies of the other virtual environments or the system-wide (default python) libraries.
- We can create **multiple** virtual environments with different/same Python versions and different/same libraries versions on the same machine.

VIRTUAL ENVIRONMENTS

Cont'd

- Any virtual environment consists of **two components**:
 - The python interpreter.
 - A folder contains the third-party libraries installed in the venv.



Lecture

01

VIRTUAL ENVIRONMENTS

Cont'd

- Venv **does not** related to your created codes, it just for **handling packages** needed to run/use in your codes.
- Program codes **must not** be inside the Venv folder.

VIRTUAL ENVIRONMENTS

Cont'd

- Using a module named `virtualenv` to create isolated python environments.
- Install virtual environment:
 - Linux (Terminal):

```
sudo apt install python3-venv  
virtualenv --version
```
 - Windows (CMD):
 - Python 3.3+ includes venv by default. Verify by running:

```
python -m venv --help
```

Lecture

01

CREATING VIRTUAL ENVIRONEMNT

Cont'd

- Create a folder for your virtual environments in your computer. Call it **Myenvs**.
- Virtual environment steps:
 - Windows (CMD):
 - Create: `python -m venv Your_venv_Name`
 - Activate: `Your_venv_Name\Scripts\activate`
 - Deactivate: `deactivate`
 - Linux (Terminal):
 - Create: `python3 -m venv Your_venv_Name`
 - Activate: `source Your_venv_Name/bin/activate`
 - Deactivate: `deactivate`

PACKAGE ISOLATION IN VIRTUAL ENVIRONMENTS

Cont'd

- Run the below command to see the list of installed packages.
 - On default python path.
`pip list`

```
C:\>pip list
```

Package	Version
absl-py	0.9.0
alabaster	0.7.12
anaconda-client	1.7.2
anaconda-navigator	1.9.7
anaconda-project	0.8.3
asn1crypto	1.0.1
astor	0.7.1
astroid	2.3.1
astropy	3.2.1
atomicwrites	1.3.0
attrs	19.2.0
Babel	2.7.0
backcall	0.1.0
backports.functools-lru-cache	1.5

..... and more

PACKAGE ISOLATION IN VIRTUAL ENVIRONMENTS

Cont'd

- Run the below command to see the list of installed packages.
 - On virtual environment.

`pip list`

```
(alpha-venv) ~ % pip list
```

```
Package      Version
```

```
-----
```

```
pip          21.3.1
```

```
setuptools  58.1.0
```

PACKAGE ISOLATION IN VIRTUAL ENVIRONMENTS

Cont'd

- Install a package in your venv.

```
pip install ascii-train
```

- Make a python code file and write the following code.

```
import ascii_train  
print(ascii_train.train("Midocean University"))
```

- Run the code. If you got an error, then make sure you **activate the correct venv**.
`Your_venv_Name/bin/python3 "/path/to/code file/test.py"`

Lecture

01

QUESTIONS FOR ONLINE DISCUSSION

- Suggest a scenario for using networks which highlights its importance.
- Is there is any other types of networks?
- Give an example for the low-level and high-level network programming?
- Give a real-world example which can be solving using virtual environments in your work?



Thank You

