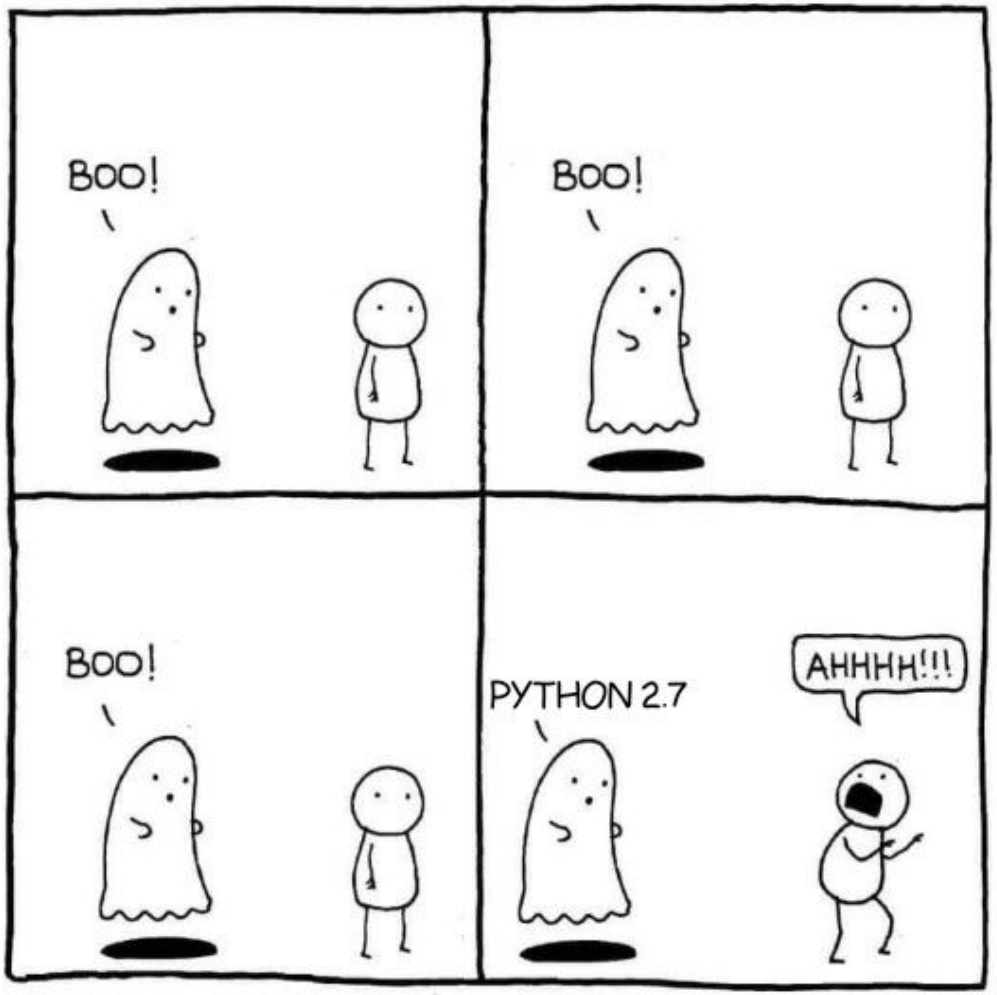
How to Python Development

and a bit of git

Contents

- Python
- Integrated Development Environments (IDEs)
- Version Control with git
- Scapy
- Workflow Demo

Python 3.x



Python 3.x

Install Python 3 on Windows, Linux, macOS

realpython.com/installing-python

A word of warning

Familiarize yourself with Python.

Before you start!

Getting started

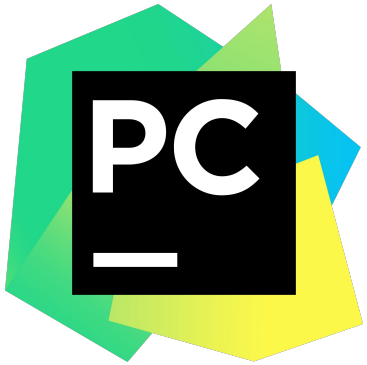
Beginners Guide

learnpython.org

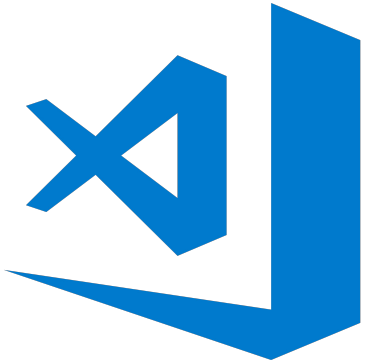
Advanced/Refresh Guide

learnxinyminutes.com/docs/python3

From Text Editors to IDE's



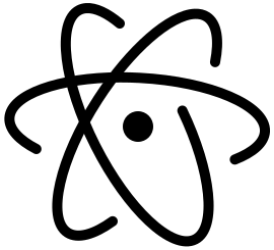
JetBrains PyCharm



Visual Studio Code

if you like it a bit slimmer...

Sublime Text



atom.io

Vim

(+ a lot of plugins)

...

Getting started

Create JetBrains account with your ETH mail

jetbrains.com/shop/eform/students

Install PyCharm

jetbrains.com/pycharm/download

Version Control with git

git Tracks Changes in Source Code

Without git

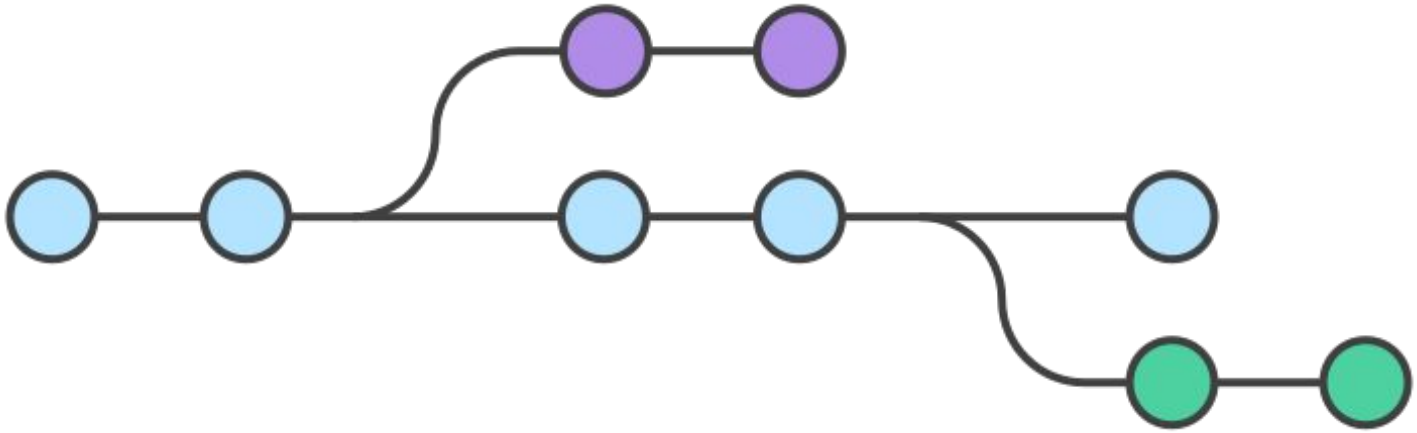
Everyone works on the same file and uploads it to the server

The version uploaded last overwrites all other changes.

With git

Everyone works on the same file and pushes the changes to the git repository.

All changes are combined, nothing is lost.



git Workflow

1. Create a repository for your group

gitlab.ethz.ch/projects/new

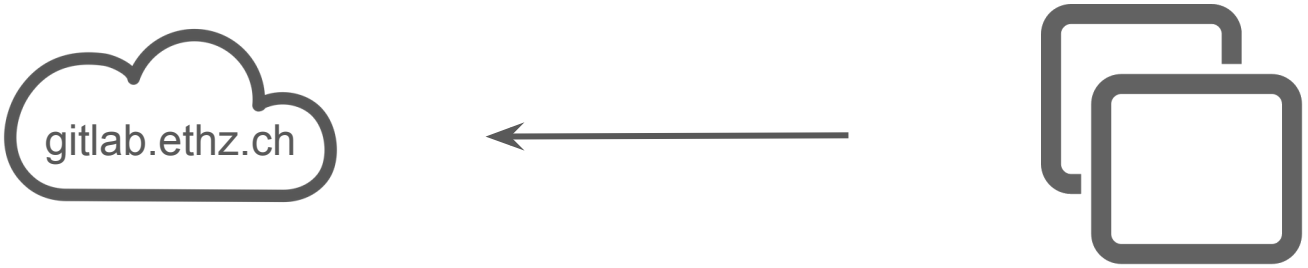
2. Invite group members

Settings -> Members

git Workflow

3. Create ssh key on VM and upload to Gitlab

docs.gitlab.com/ee/gitlab-basics/create-your-ssh-keys.html



4. Initialise your repository on the VM

Follow GitLab instructions for “Existing Folder”

git Workflow

5. Clone your repository to your machine

```
git clone <repository>
```

6. Commit your changes locally

```
git add <file>
```

```
git commit -m “Describe what you are committing”
```

git Workflow

7. Download changes from GitLab

```
git pull
```

8. Upload your changes to GitLab

```
git push
```

git Tips and Tricks

- No branching required for the assignment
- Run the git commands from the right directory
- Always pull before you push

Cheat Sheet & Installation Guide

rogerdudler.github.io/git-guide

Make you own packets with Scapy

Sending and Receiving Packets in Python

```
from scapy.all import send, IP, TCP

payload = b"This is some binary test data."

packet = IP(src="192.168.0.1", dst="8.8.8.8") / TCP() / payload

send(packet)
```


Combine headers with the divison operator

Sending and Receiving Packets in Python

Show summary and details

```
print(packet.summary())
```

```
print(packet.show())
```

Access headers and data

```
from scapy.all import IP
```

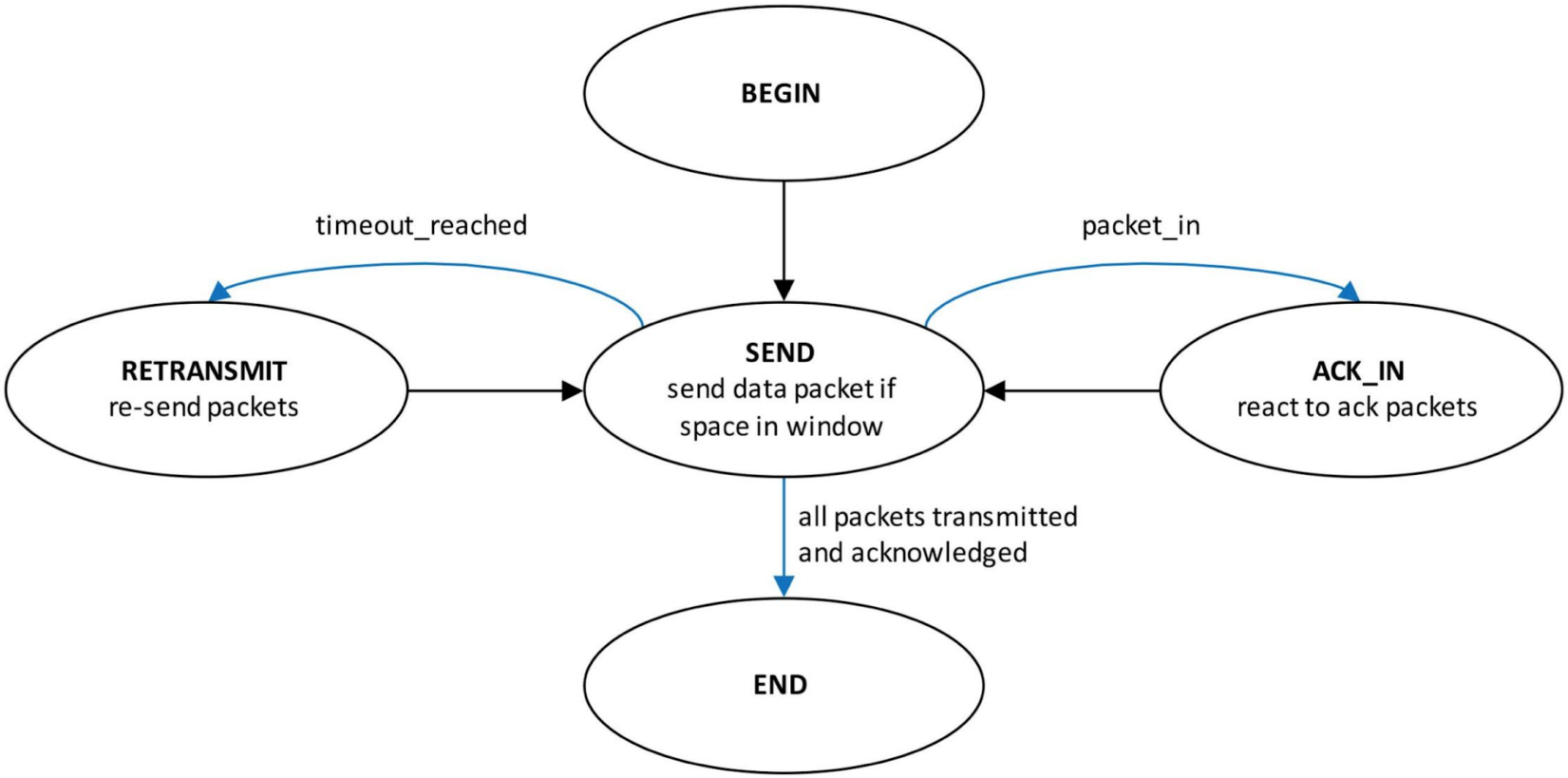
```
ip_header = packet.getlayer(IP)
source_address = ip_header.src
payload = ip_header.payload
```

Building you own Headers

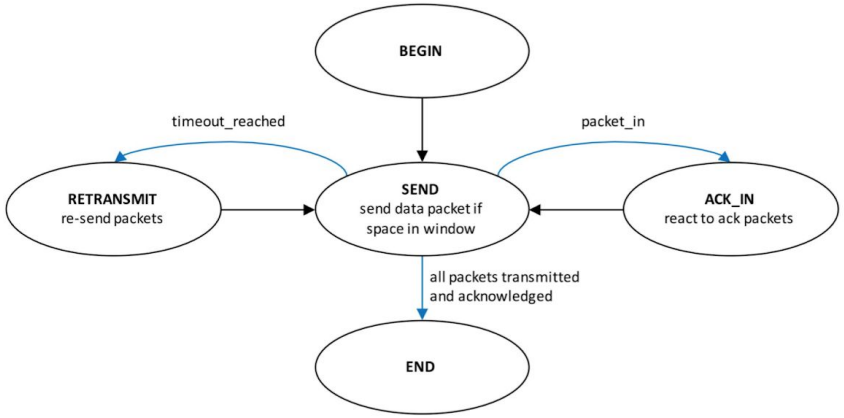
```
from scapy.all import Packet, bind_layers, BitEnumField, BitField


class GBN(Packet):
    name = 'GBN'
    fields_desc = [
        BitEnumField("type", 0, 1, {0: "data", 1: "ack"}),
        BitField("options", 0, 7),
        # other fields ...
    ]
```

GBN Automaton of the assignment



GBN Automaton in Scapy




```
from scapy.all import Automaton, ATMT


class GBNSender(Automaton):
    @ATMT.state(initial=1)
    def BEGIN(self):
        raise self.SEND()

    @ATMT.state()
    def SEND(self):
        # Your code!
```

Transitions are triggered by exceptions

Demo Time