TUTORIAL INSTALASI *MINDWAVE NEUROSKY, PYTHON* DAN PENJELASAN PADA *CODING PYTHON*

TUTORIAL INSTALASI *MINDWAVE NEUROSKY*, *PYTHON* DAN PENJELASAN PADA *CODING PYTHON*

Asep Setiawan

Member Informatics Research Center

1. *Install Mindwave Neurosky*
2. *Install Python* 2.7.11
3. Penjelasan dan Codingan *Python*

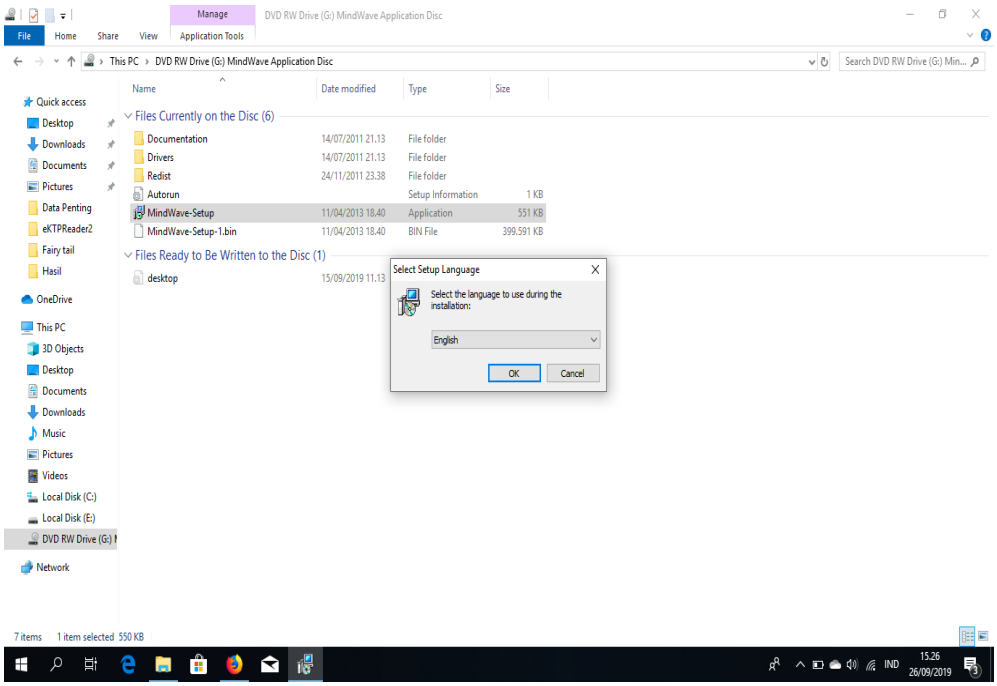
KATA PENGATAR

DAFTAR ISI

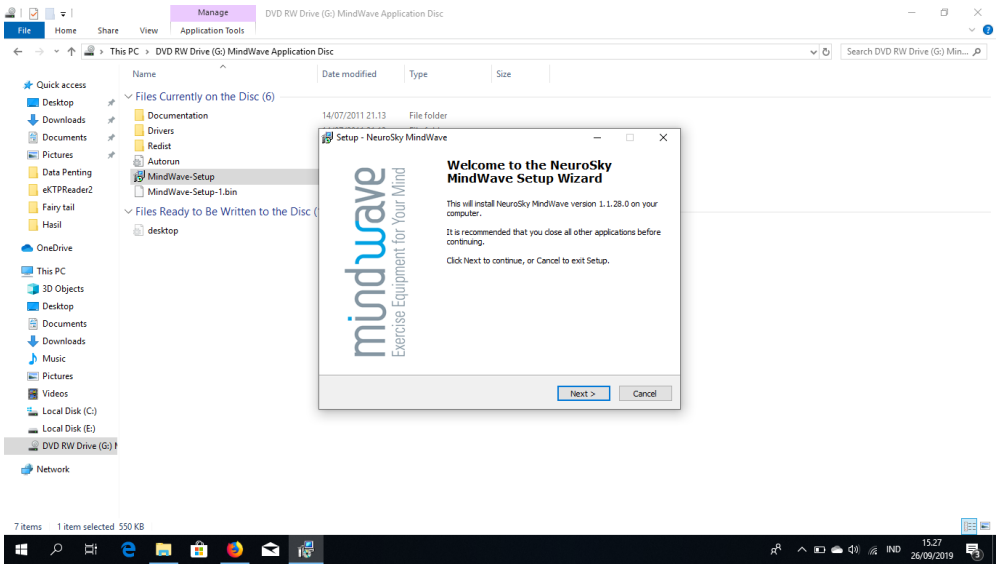
DAFTAR GAMBAR

INSTALL MINDWAVE NEUROSKY

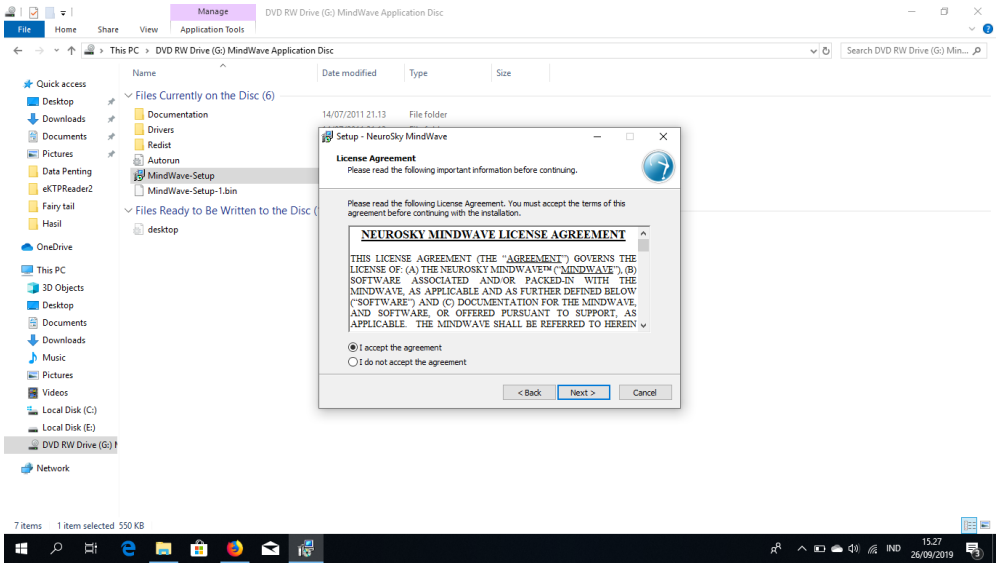
Pertama Install aplikasi mindwave neurosky, pilih ok dalam bahasa inggris.



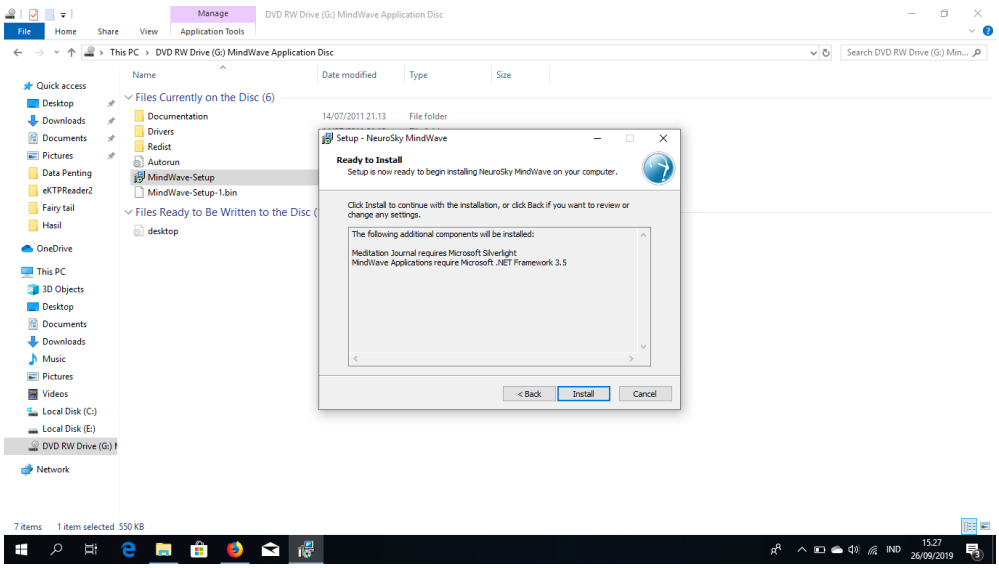
Kedua, pilih next untuk melanjutkan tahap kedua ini.



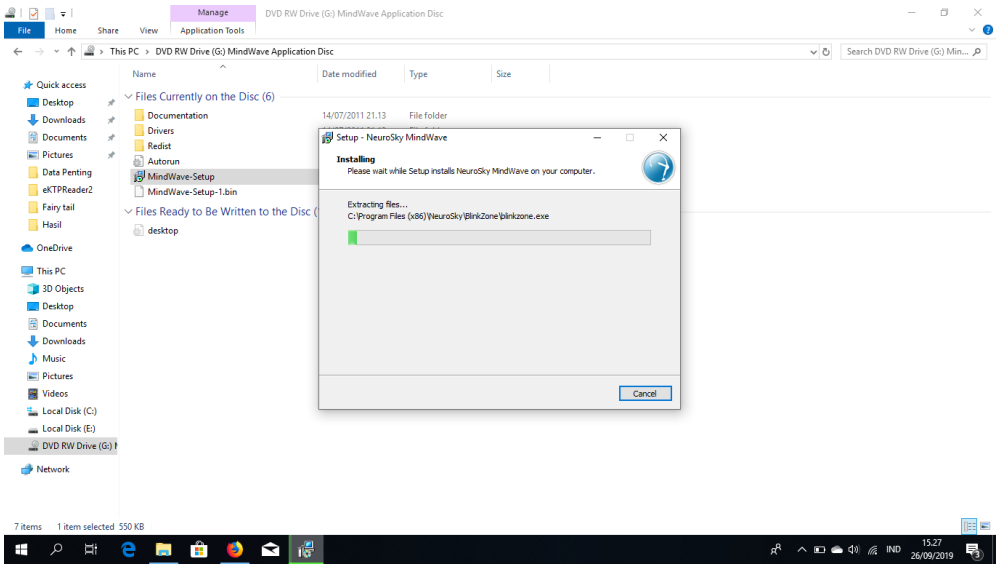
Ketiga, pilih “I accept the agreement” lalu klik Next.



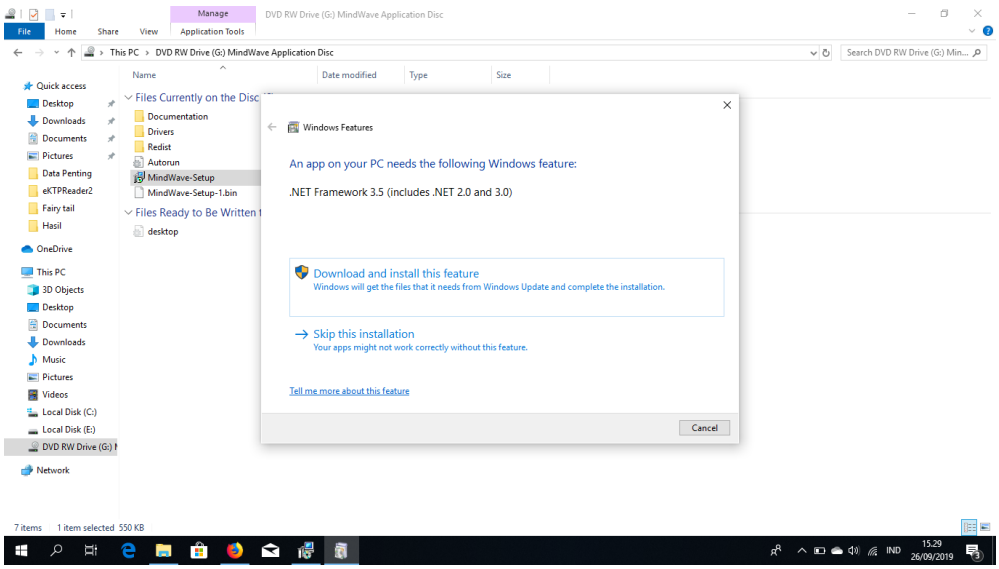
Keempat, ada tulisan yang mengarah untuk klik “Next”.



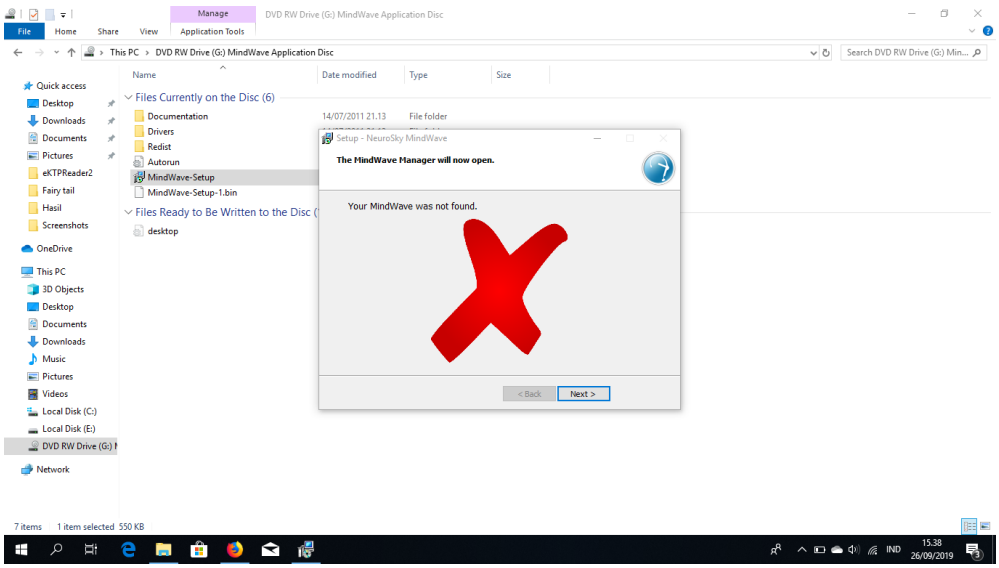
Kelima, tunggu sampai loding selesai.



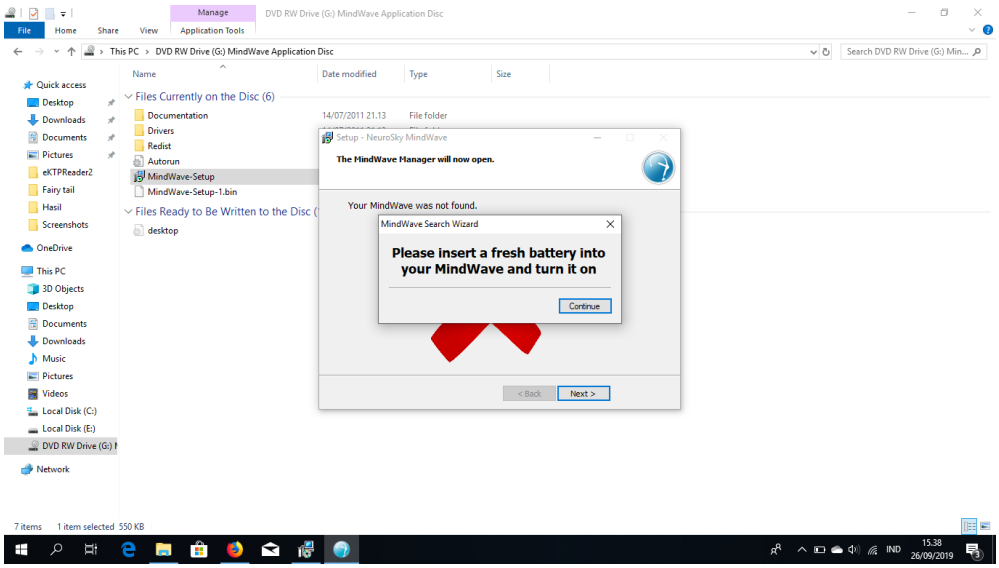
Keenam, saat pemilihan update NET. Framework 3.5 , kemudian pilih “download and install this feature”.



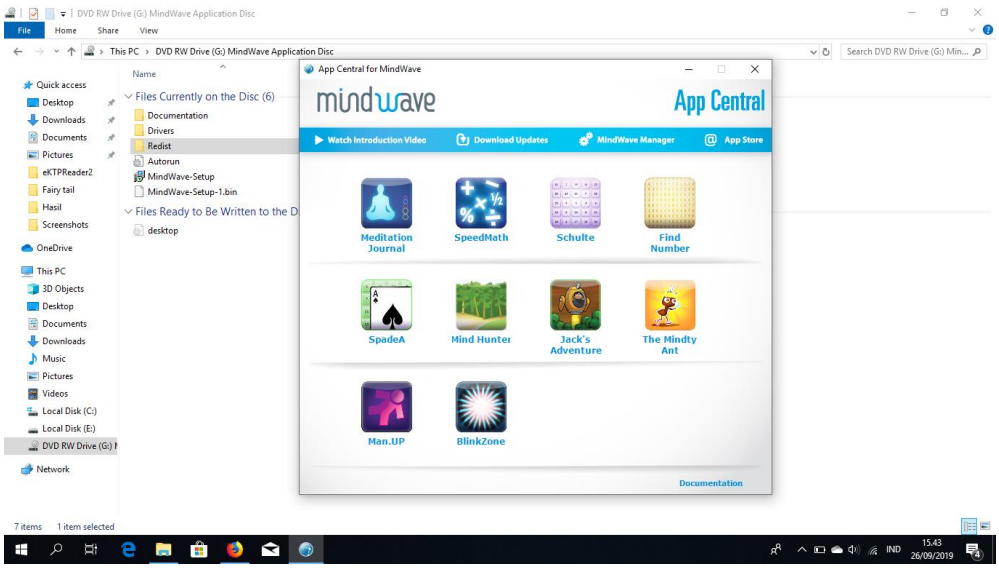
Ketujuh, jika selesai maka akan tampil yang dibawah ini, tanda (x) itu karena belum di pasang usb dari mindwavenya. Jika sudah terhubung maka akan ada tanda (√) berarti sudah terhubung, lalu klik next.



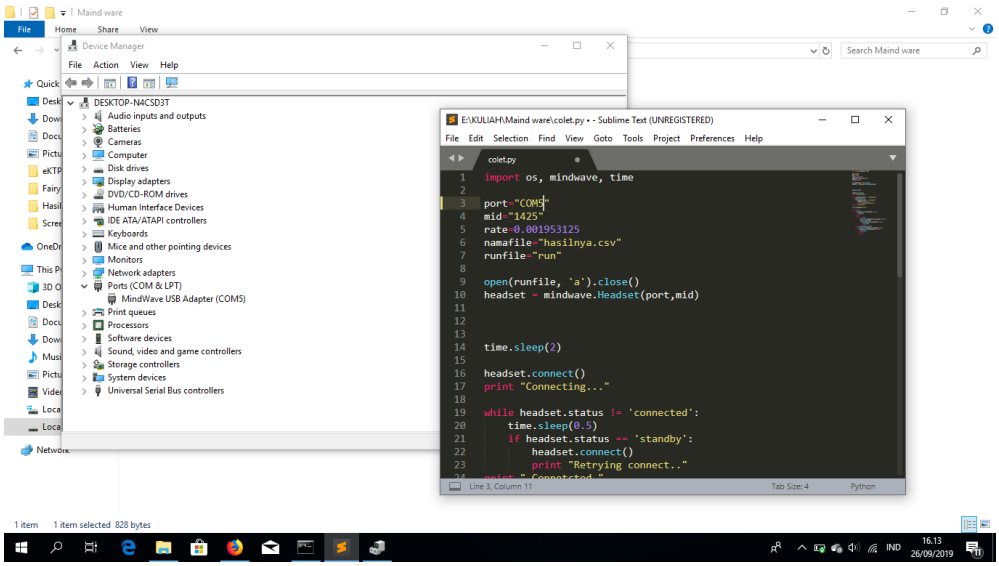
Kedelapan, muncul popup seperti dibawah kemudian klik “continue”.



Kesembilan, jika sudah proses penginstallan maka akan tampil seperti dibawah ini



Kesepuluh, sesuaikan portnya contoh dibawah ini port (COM5).

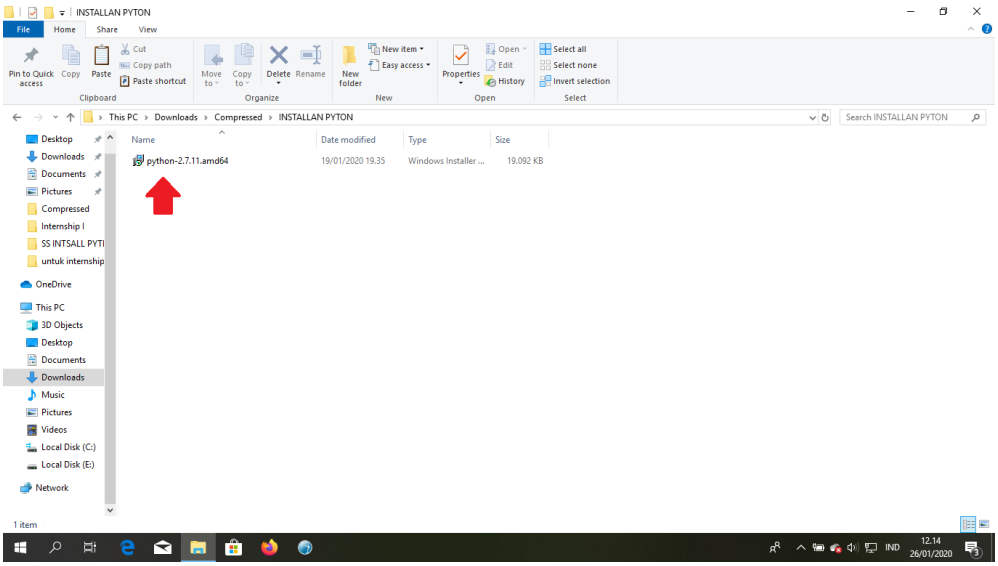


Selesai, penginstalan *mindwave neurosky*.

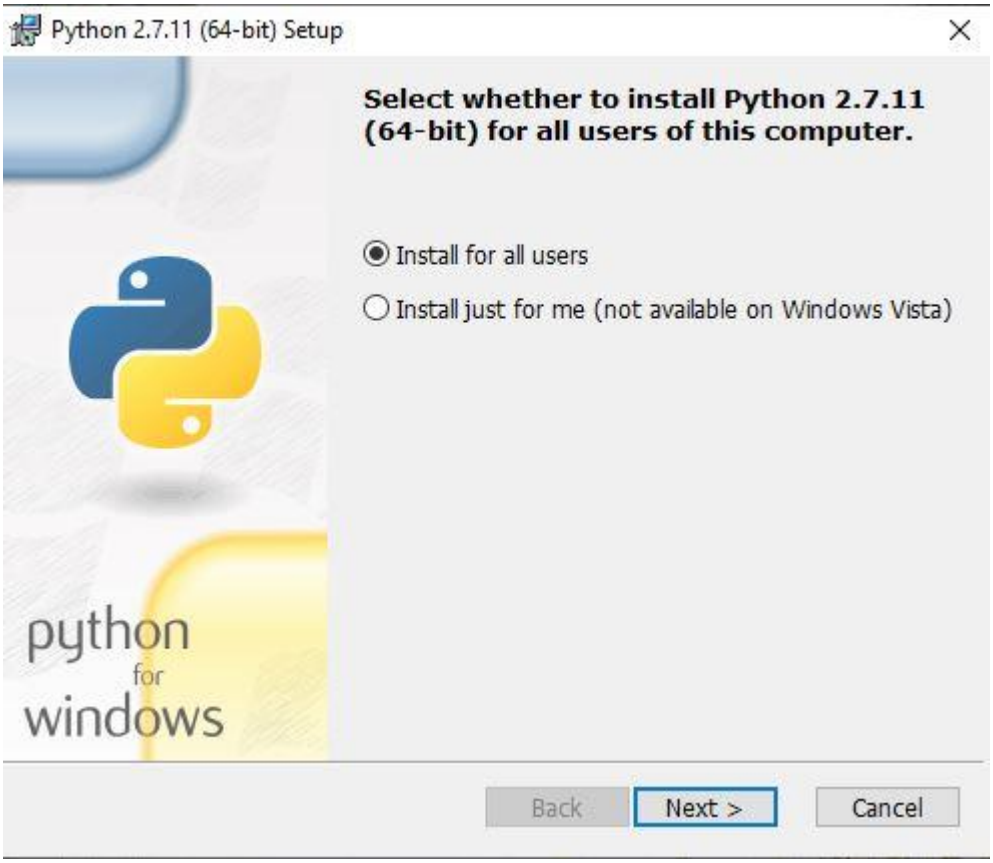
INSTALL PYTHON 2.7.11

Pertama download terlebih dahulu python 2.7.11 di link ini https://www.python.org/downloads/release/python-2711/

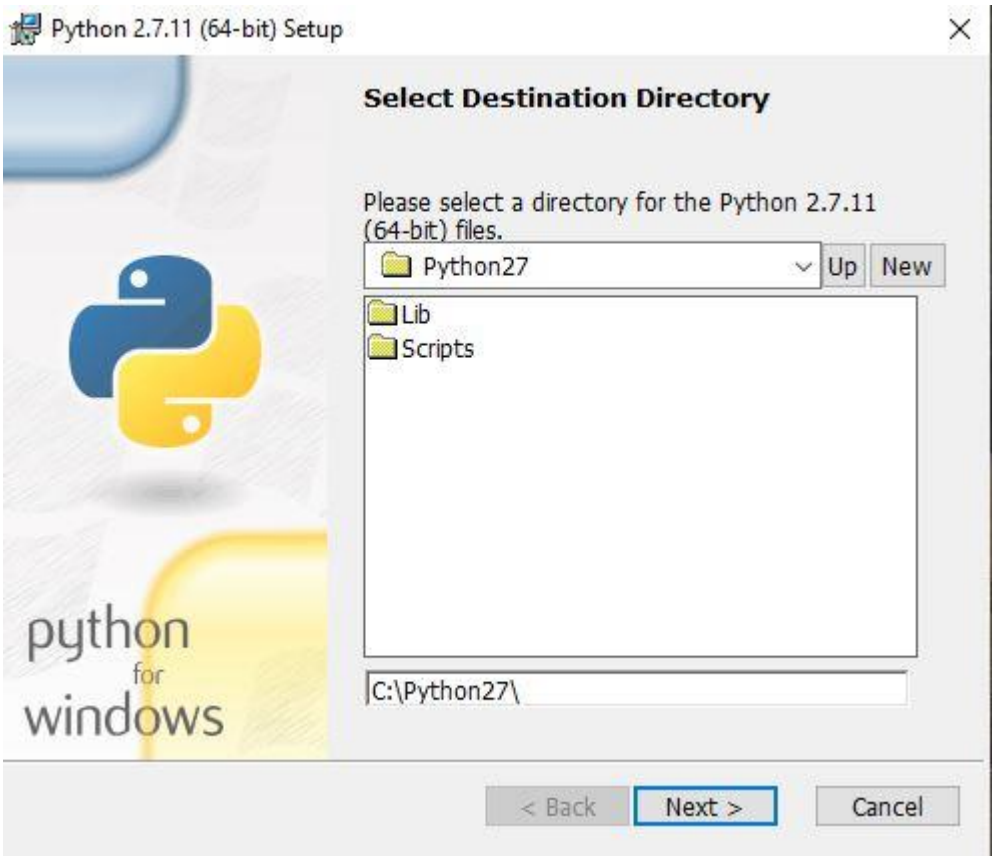
Kedua, Jika sudah didownload kemudian double klik pada python.



Ketiga, muncul seperti tampilan di bawah kemudian klik “Next”.

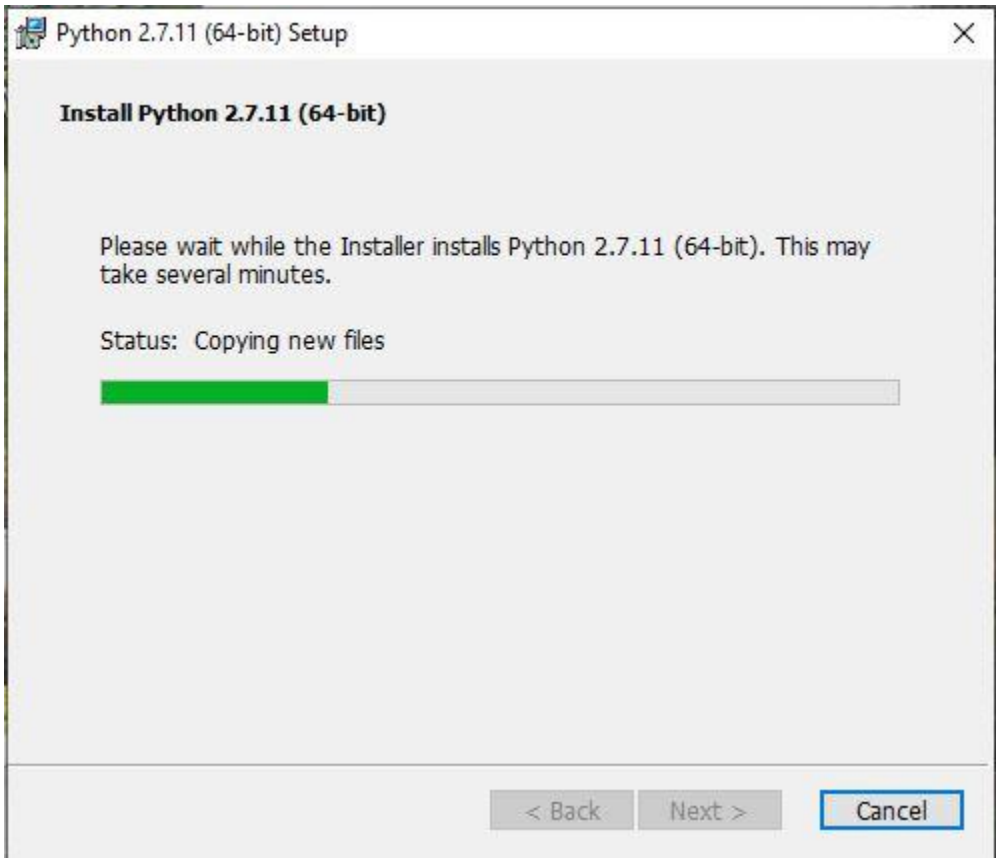


Keempat, pilih derektori sesuai yang diinginkan seperti dibawah ini, lalu klik “Next”.



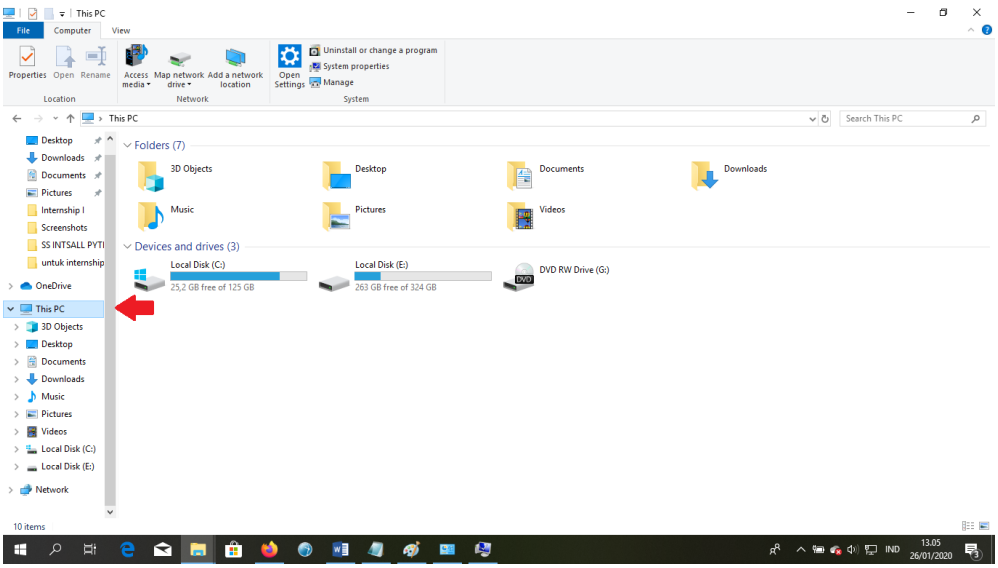
Kelima, cek pilihan yang diinstall jika sudah sesuai maka klik “Next”.

Keenam, tunggu sampai loding ini selesai.

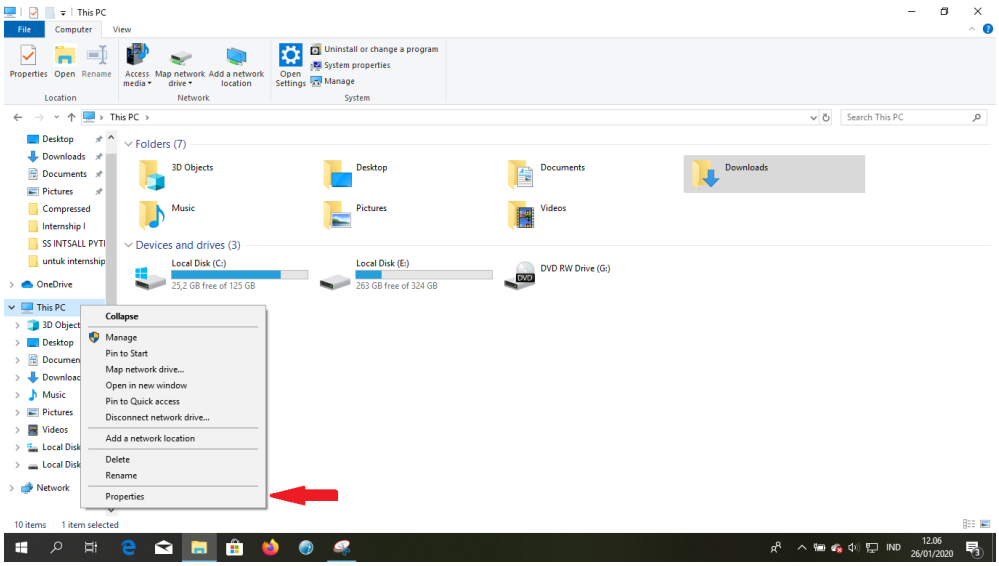


Ketujuh, selesai penginstallan *python*, kemudian klik “Finish”.

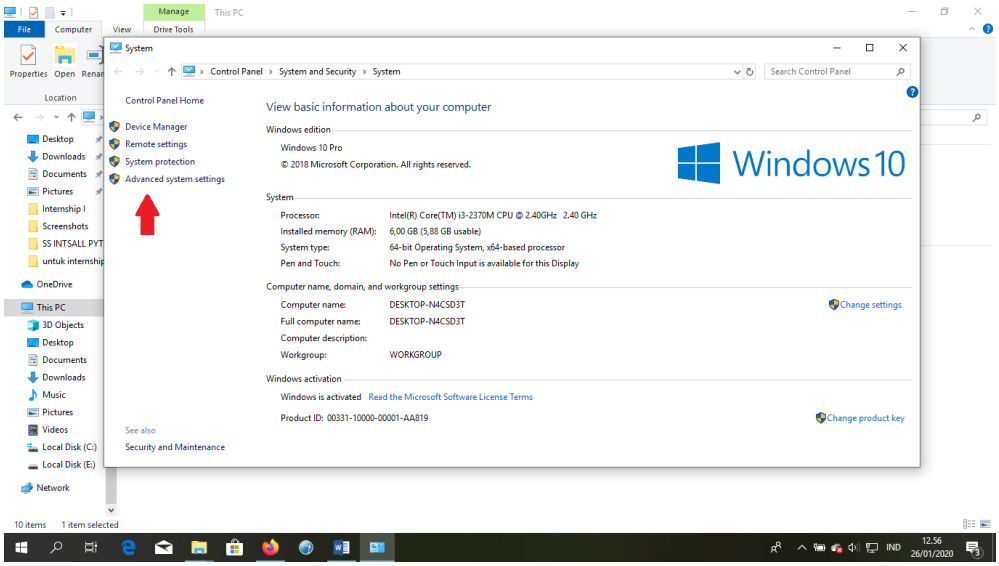
Kedelapan, atur environment variable klik kanan pada “This PC” seperti gambar dibawah ini.



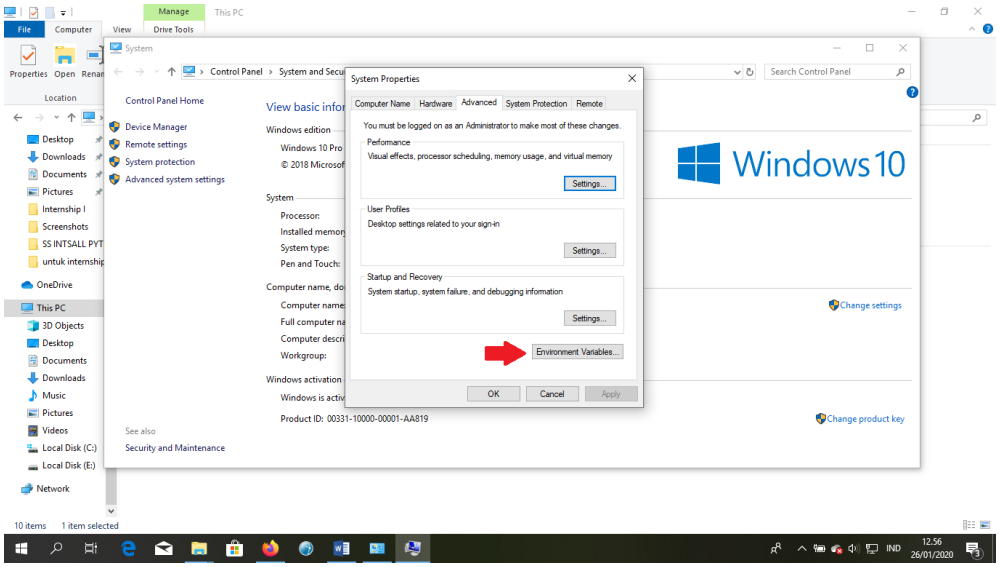
Kesembilan, kemudian klik pada “properties” sperti gambar dibawah ini.



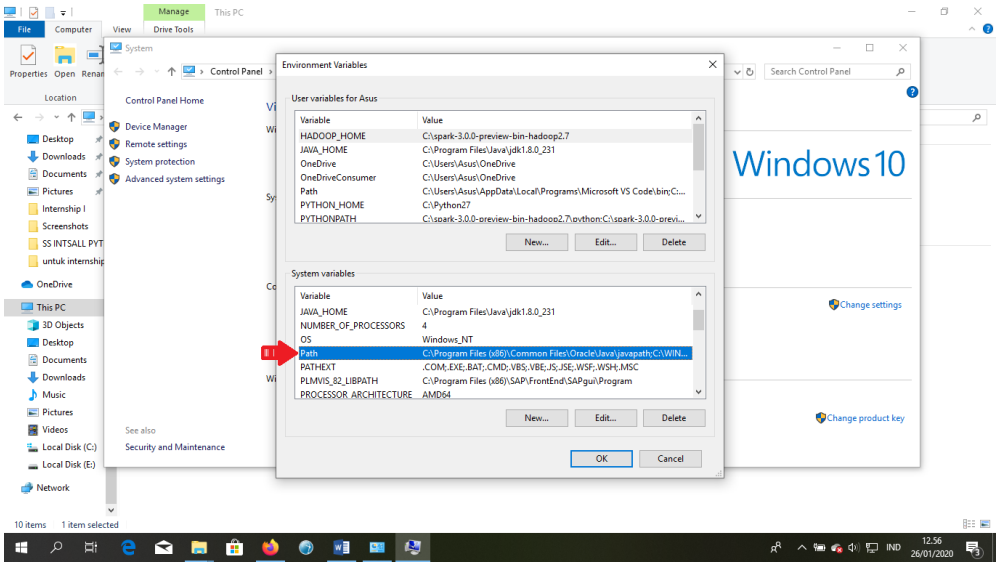
Kesepuluh, pilih “Advanced system setting”.



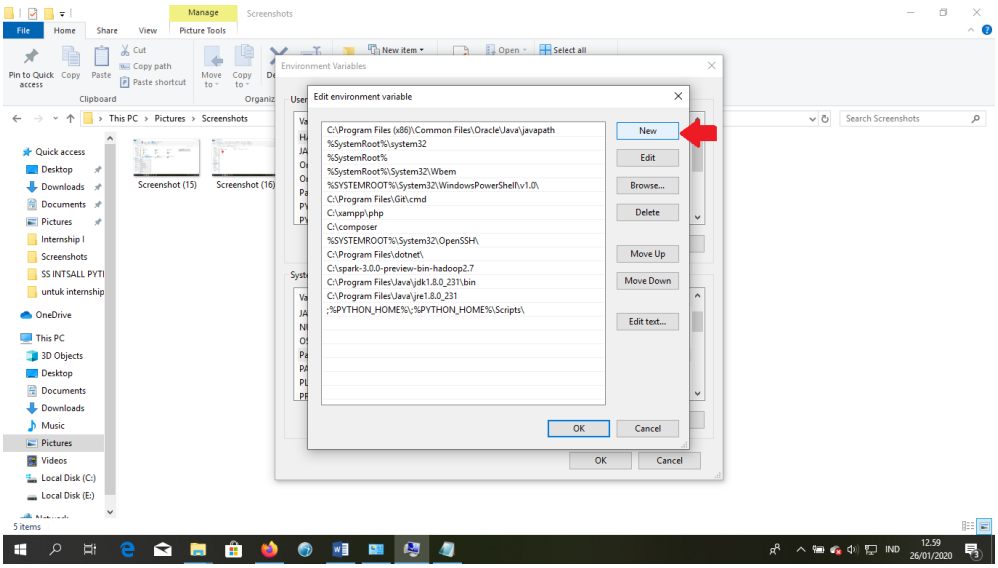
Kesebelas, klik “Enviroment Variables...”.



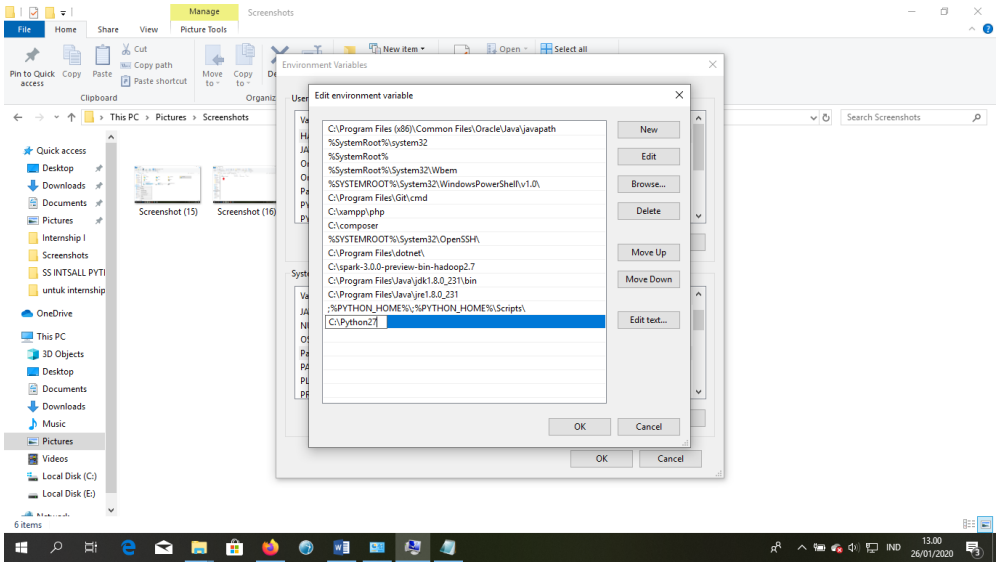
Keduabelas, lalu klik “Path”.

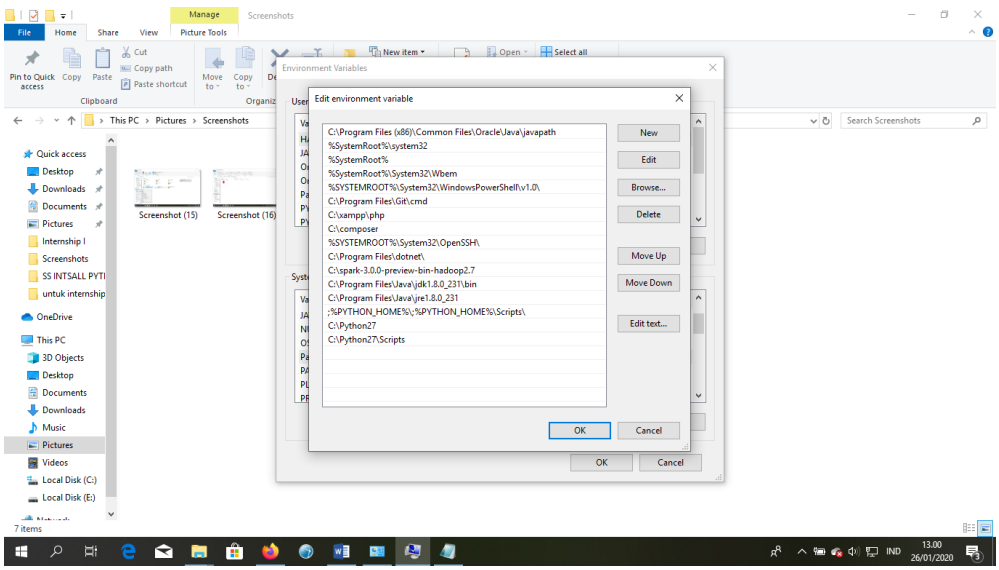


Ketigabelas, muncul seperti dibawah ini, kemudian klik “New” untuk membuat slot baru pada enviroment variable.

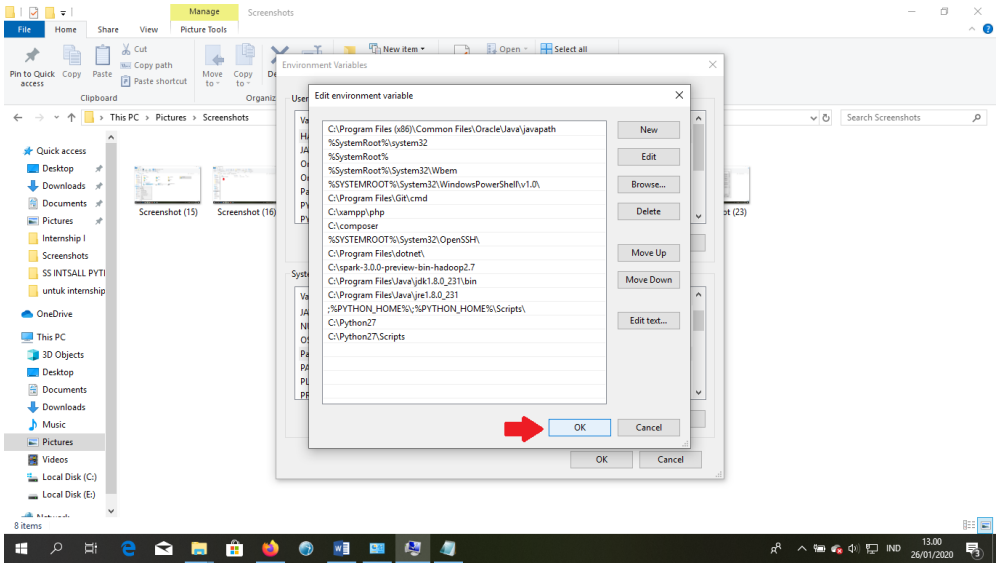


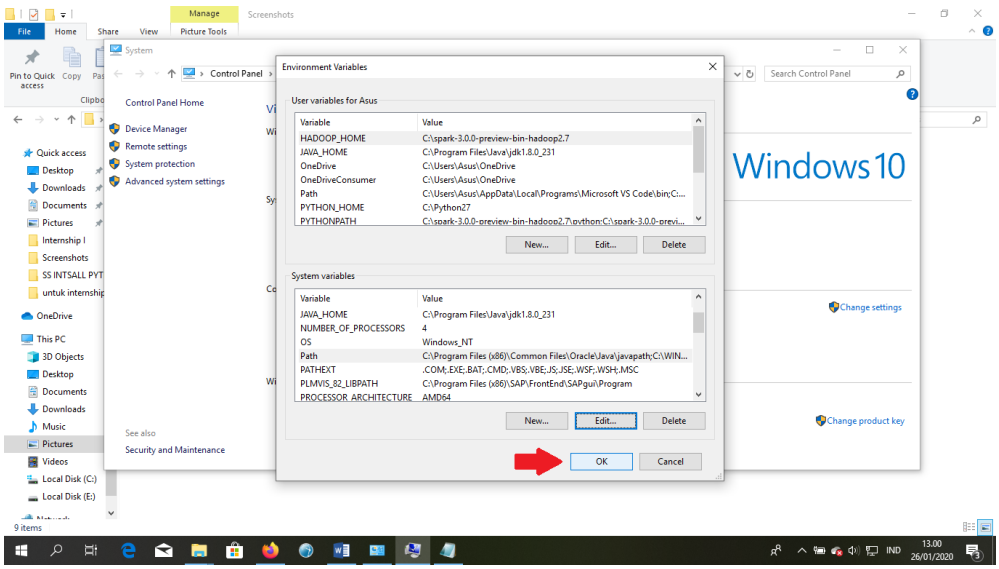
Keempatbelas, tuliskan atau ketik “C:\Python27” dan “C:\Python27\Scripts” seperti dibawah ini.



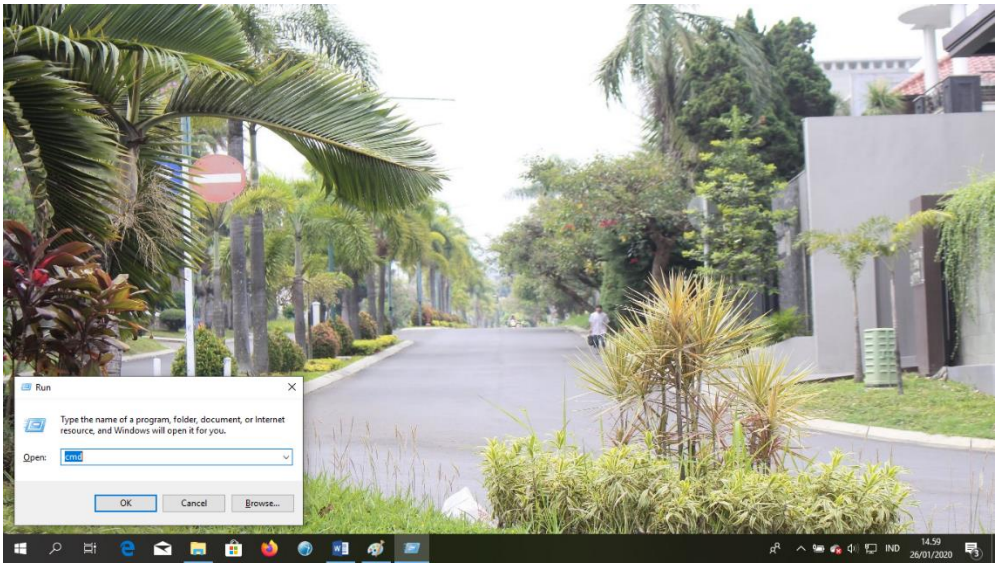


Kelimabelas, klik “Ok dan Ok”. Selesai membuat enviromentnya.

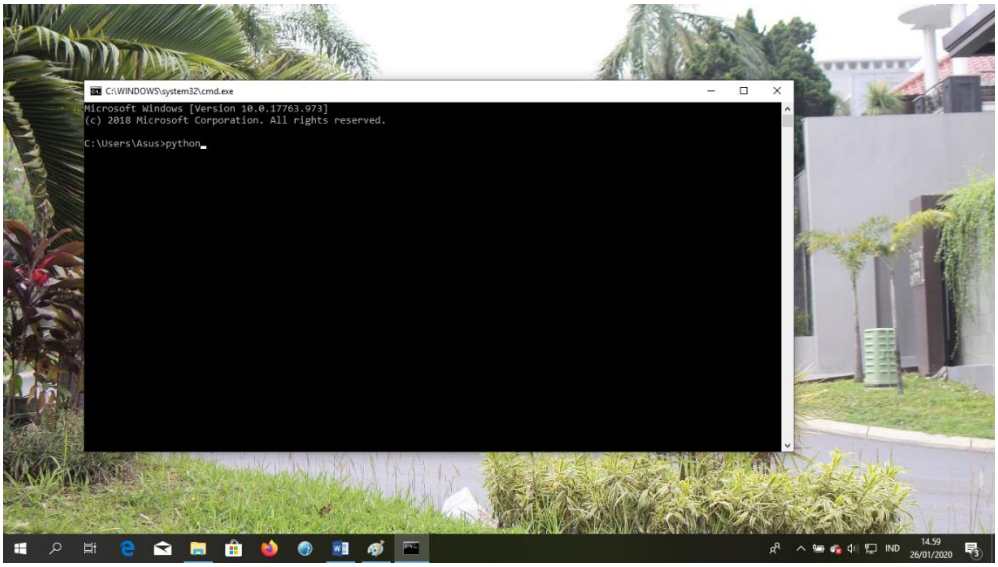




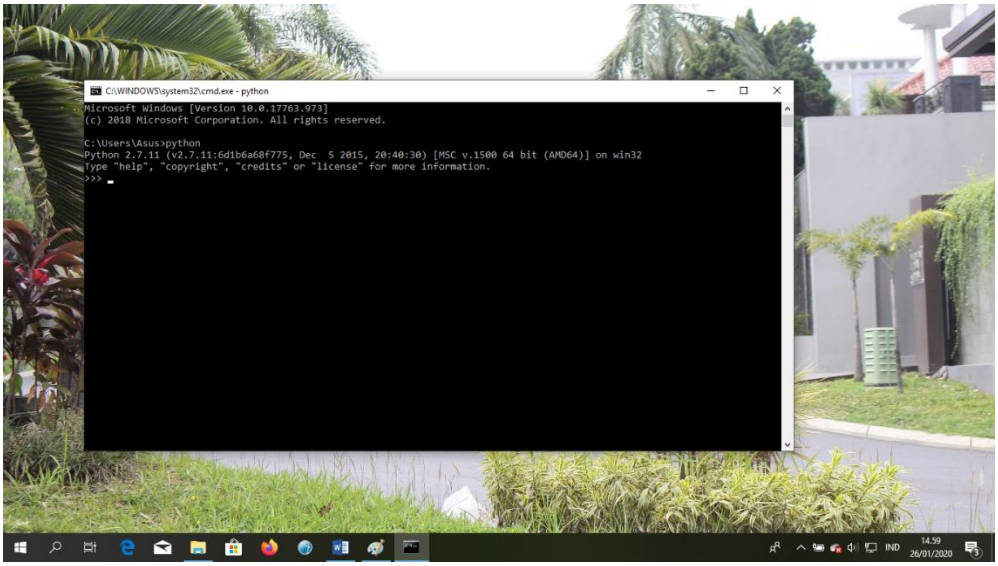
Terakhir kita jalankan python tersebut apakah sesuai atau tidak, Pertama, Run Administrator “CMD”.



Kedua, ketik printah “python” pada CMD.



Ketiga, jika tampil seperti digambar dibawah ini. Maka dapat disimpulkan penginginstallan dan pembuatan environment berhasil.



PENJELASAN DAN CODINGAN

PENJELASAN *PYTHON*

Python merupakan bahasa pemrograman yang dibuat dan dikembangkan oleh Guido Van Rossum pada tahun 1990 di Stichting Mathematisch Centrum (CWI). Bahasa pemrograman python itu bahasa pemrograman yang kekinian(terbaru), python juga bisa dikombinasikan dengan java dan arduino.

Python juga ada 2 versi yang berbeda yaitu versi *python* 2.x dan *python* 3.x, pada codingannya juga berbeda.

PERBEDAAN *PYTHON* 2.X DAN *PYTHON* 3.X

Perbedaan ypada codingan seperti dibawah ini.

Python 2.x :

Print ‘Hallo!’

Python 3.x :

Print (‘Hallo!’)

Kesimpulannya pada *python* 2.x tidak menggunakan “( )” akan tetapi *python* 3.x menggunakannya.

PERBEDAAN PYTHON DENGAN BAHASA PEMROGRAMAN LAIN.

Bahasa pemrograman python yaitu bahasa yang singkat, padat dan jelas. Berdasarkan codingan yang telah dibuat sperti dibawah ini.

Coding dari bahasa pemrograman C++ :

```
Int x = 2;
```

Coding dari bahasa pemrograman Python:

```
x = 2
```

perbedaan sudah jelas dimana C++ harus menggunakan class dan type data. Akan tetapi tidak menggunakan class dan tipe data.

MACAM – MACAM PACKAGE

FUNGSI PACKAGE

PERTAMA ADALAH CODINGAN MINDWAVE

```
import select, serial, threading

# Byte codes
CONNECT             = '\xc0'
DISCONNECT          = '\xc1'
AUTOCONNECT         = '\xc2'
SYNC                = '\xaa'
EXCODE              = '\x55'
POOR_SIGNAL         = '\x02'
ATTENTION           = '\x04'
MEDITATION          = '\x05'
BLINK               = '\x16'
HEADSET_CONNECTED   = '\xd0'
HEADSET_NOT_FOUND   = '\xd1'
HEADSET_DISCONNECTED = '\xd2'
REQUEST_DENIED      = '\xd3'
STANDBY_SCAN        = '\xd4'
RAW_VALUE           = '\x80'

# Status codes
STATUS_CONNECTED    = 'connected'
```

```
STATUS_SCANNING      = 'scanning'
STATUS_STANDBY       = 'standby'

class Headset(object):
    """
    A MindWave Headset
    """

    class DongleListener(threading.Thread):
        """
        Serial listener for dongle device.
        """
        def __init__(self, headset, *args, **kwargs):
            """Set up the listener device."""
            self.headset = headset
            super(Headset.DongleListener, self).__init__(*args,
**kwargs)

        def run(self):
            """Run the listener thread."""
            s = self.headset.dongle

            # Re-apply settings to ensure packet stream
```

```
s.write(DISCONNECT)
d = s.getSettingsDict()
for i in xrange(2):
    d['rtscts'] = not d['rtscts']
    s.applySettingsDict(d)

while True:
    # Begin listening for packets
    try:
        if s.read() == SYNC and s.read() == SYNC:
            # Packet found, determine plength
            while True:
                plength = ord(s.read())
                if plength != 170:
                    break
            if plength > 170:
                continue

            # Read in the payload
            payload = s.read(plength)

            # Verify its checksum
            val = sum(ord(b) for b in payload[:-1])
```

```
                val &= 0xff
                val = ~val & 0xff
                chksum = ord(s.read())

                #if val == chksum:
                if True: # ignore bad checksums
                    self.parse_payload(payload)
            except (select.error, OSError):
                break
            except serial.SerialException:
                s.close()
                break

    def parse_payload(self, payload):
        """Parse the payload to determine an action."""
        while payload:
            # Parse data row
            excode = 0
            try:
                code, payload = payload[0], payload[1:]
            except IndexError:
                pass
            while code == EXCODE:
```

```
            # Count excode bytes
            excode += 1
            try:
                code, payload = payload[0], payload[1:]
            except IndexError:
                pass
        if ord(code) < 0x80:
            # This is a single-byte code
            try:
                value, payload = payload[0], payload[1:]
            except IndexError:
                pass
            if code == POOR_SIGNAL:
                # Poor signal
                old_poor_signal = self.headset.poor_signal
                self.headset.poor_signal = ord(value)
                if self.headset.poor_signal > 0:
                    if old_poor_signal == 0:
                        for handler in \
                            self.headset.poor_signal_handlers:
                            handler(self.headset,
                                    self.headset.poor_signal)
                else:
```

```
            if old_poor_signal > 0:
                for handler in \
                    self.headset.good_signal_handlers:
                    handler(self.headset,
                            self.headset.poor_signal)
        elif code == ATTENTION:
            # Attention level
            self.headset.attention = ord(value)
            for handler in self.headset.attention_handlers:
                handler(self.headset, self.headset.attention)
        elif code == MEDITATION:
            # Meditation level
            self.headset.meditation = ord(value)
            for handler in self.headset.meditation_handlers:
                handler(self.headset, self.headset.meditation)
        elif code == BLINK:
            # Blink strength
            self.headset.blink = ord(value)
            for handler in self.headset.blink_handlers:
                handler(self.headset, self.headset.blink)
    else:
        # This is a multi-byte code
        try:
```

```
                vlength, payload = ord(payload[0]), payload[1:]
            except IndexError:
                continue
            value, payload = payload[:vlength],
payload[vlength:]
            # Multi-byte EEG and Raw Wave codes not included
            # Raw Value added due to Mindset Communications
Protocol
            if code == RAW_VALUE:
                try:
                    anu = value[0]
                    itu = value[1]
                except IndexError:
                    anu = "x"
                    itu = "x"
                raw=ord(anu)*256+ord(itu)
                if (raw>=32768):
                    raw=raw-65536
                self.headset.raw_value = raw
                for handler in self.headset.raw_value_handlers:
                    handler(self.headset, self.headset.raw_value)

            if code == HEADSET_CONNECTED:
                # Headset connect success
```

```
                    run_handlers = self.headset.status !=
STATUS_CONNECTED
                    self.headset.status = STATUS_CONNECTED
                    self.headset.headset_id = value.encode('hex')
                    if run_handlers:
                        for handler in \
                            self.headset.headset_connected_handlers:
                            handler(self.headset)
                elif code == HEADSET_NOT_FOUND:
                    # Headset not found
                    if vlength > 0:
                        not_found_id = value.encode('hex')
                        for handler in \
                            self.headset.headset_notfound_handlers:
                            handler(self.headset, not_found_id)
                    else:
                        for handler in \
                            self.headset.headset_notfound_handlers:
                            handler(self.headset, None)
                elif code == HEADSET_DISCONNECTED:
                    # Headset disconnected
                    headset_id = value.encode('hex')
                    for handler in \
```

```
                    self.headset.headset_disconnected_handlers:
                    handler(self.headset, headset_id)
                elif code == REQUEST_DENIED:
                    # Request denied
                    for handler in
self.headset.request_denied_handlers:
                    handler(self.headset)
                elif code == STANDBY_SCAN:
                    # Standby/Scan mode
                    try:
                        byte = ord(value[0])
                    except IndexError:
                        byte = None
                    if byte:
                        run_handlers = (self.headset.status !=
                                        STATUS_SCANNING)
                        self.headset.status = STATUS_SCANNING
                        if run_handlers:
                            for handler in self.headset.scanning_handlers:
                                handler(self.headset)
                    else:
                        run_handlers = (self.headset.status !=
                                        STATUS_STANDBY)
```

```
                    self.headset.status = STATUS_STANDBY
                    if run_handlers:
                        for handler in self.headset.standby_handlers:
                            handler(self.headset)


def __init__(self, device, headset_id=None, open_serial=True):
    """Initialize the  headset."""
    # Initialize headset values
    self.dongle = None
    self.listener = None
    self.device = device
    self.headset_id = headset_id
    self.poor_signal = 255
    self.attention = 0
    self.meditation = 0
    self.blink = 0
    self.raw_value = 0
    self.status = None

    # Create event handler lists
    self.poor_signal_handlers = []
    self.good_signal_handlers = []
```

```
        self.attention_handlers = []
        self.meditation_handlers = []
        self.blink_handlers = []
        self.raw_value_handlers = []
        self.headset_connected_handlers = []
        self.headset_notfound_handlers = []
        self.headset_disconnected_handlers = []
        self.request_denied_handlers = []
        self.scanning_handlers = []
        self.standby_handlers = []

        # Open the socket
        if open_serial:
            self.serial_open()

    def connect(self, headset_id=None):
        """Connect to the specified headset id."""
        if headset_id:
            self.headset_id = headset_id
        else:
            headset_id = self.headset_id
            if not headset_id:
                self.autoconnect()
```

```
            return
        self.dongle.write(''.join([CONNECT,
headset_id.decode('hex')]))

    def autoconnect(self):
        """Automatically connect device to headset."""
        self.dongle.write(AUTOCONNECT)

    def disconnect(self):
        """Disconnect the device from the headset."""
        self.dongle.write(DISCONNECT)

    def serial_open(self):
        """Open the serial connection and begin listening for
data."""
        # Establish serial connection to the dongle
        if not self.dongle or not self.dongle.isOpen():
            self.dongle = serial.Serial(self.device, 115200)

        # Begin listening to the serial device
        if not self.listener or not self.listener.isAlive():
            self.listener = self.DongleListener(self)
            self.listener.daemon = True
            self.listener.start()
```

```
def serial_close(self):
    """Close the serial connection."""
    self.dongle.close()
```

KEDUA CODINGAN CONNECTING

```
import serial

ser = serial.Serial("COM8", 9600)
temp ="
while 1:
    data=ser.readline()
    print data
    i = int(data, 16)
    data=data.script()
    print(data)
    if data[:1]=="[":
        print "\a"
```

KETIGA CODINGAN TESTING

```
import mindwave, time

headset = mindwave.Headset('COM8','1425')
time.sleep(2)

headset.connect()
print "Connecting..."

while headset.status != 'connected':
    time.sleep(0.5)
    if headset.status == 'standby':
        headset.connect()
        print "Retrying connect.."
print " Connetcted."

while True:
    #print "Auttention: %s, meditation: %s" % (headset.attention,
headset.meditation)
    #print headset.serial_open()
    print headset.raw_value#"raw_value: %s" %
(headset.raw_value)
    time.sleep(0.1) #pause 0.5 seconds
```

KEEMPAT CODINGAN RUNNING

```
import os, mindwave, time

port="COM8"
mid="1425"
rate=0.001953125
namafile="hasilnya.csv"
runfile="run"

open(runfile, 'a').close()
headset = mindwave.Headset(port,mid)



time.sleep(2)

headset.connect()
print "Connecting..."

while headset.status != 'connected':
    time.sleep(0.5)
    if headset.status == 'standby':
```

```
        headset.connect()
        print "Retrying connect.."
print " Connetcted."

f=open(namafile,'a+')

while True:
        if os.path.exists(runfile)!=True:
                f.close()
                break
        try:
                while True:
                        f.write(str(headset.raw_value)+',0\n')
                        time.sleep(rate)
                        if os.path.exists(runfile)!=True:
                                f.close()
                                break
        except KeyboardInterrupt:
                f.write(str(headset.raw_value)+',1\n')
                if os.path.exists(runfile)!=True:
                        f.close()
                        break
        continue
```

KEENAM CODINGAN GRAFIK

Codingan ini untuk menampilkan matplotlib yang nanti akan muncul grafik gelombang otak dan amplitudo.

CODING:

```
from matplotlib import pyplot as plt

from matplotlib import style

import numpy as np


style.use('ggplot')


x,y = np.loadtxt('asep.csv', unpack = True, delimiter = ',')


plt.plot(x,y)

plt.title('Brainwave data result test')

plt.ylabel('Y axis')

plt.xlabel('X axis')


plt.show()
```

PENJELASAN ALGORITMA C45

PENGERTIAN ALGORITMA C45

PERBEDAAN ALGORITMA C45 DENGAN YANG LAIN

PERSAMAAN ALGORITMA C45