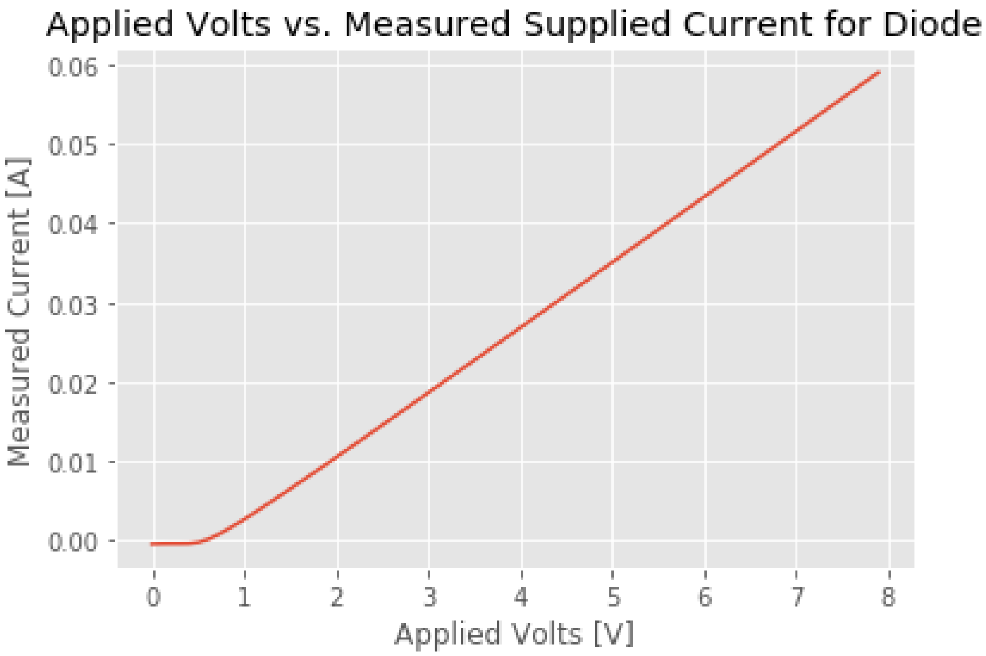
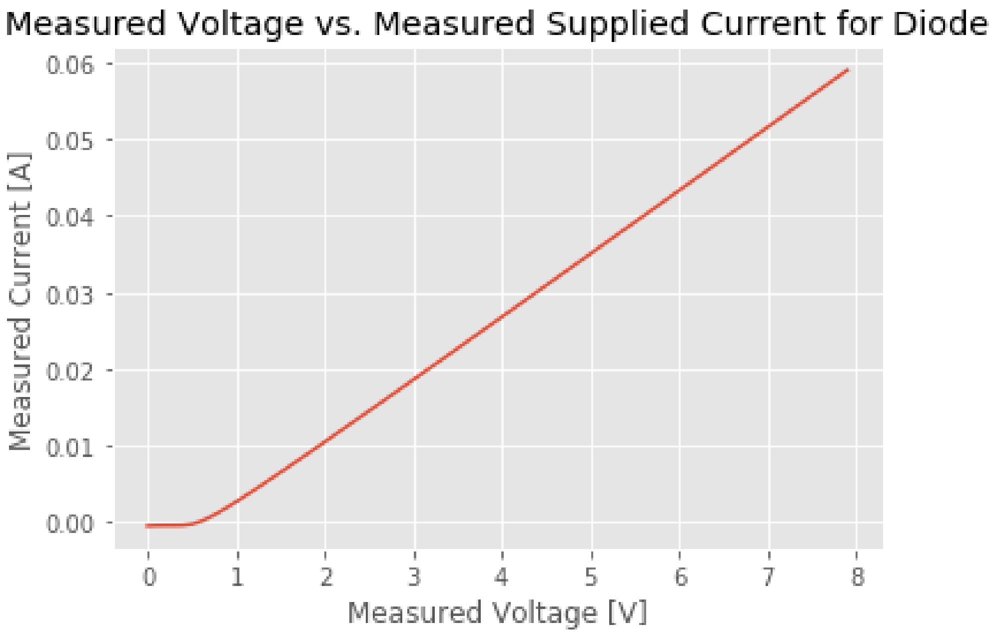
Name: Junzhe Wu   Net id: Junzhew3

1. Plot the current -voltage characteristic of the diode with a 0.1V step size





Code:

```
import visa
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import time
mpl.style.use('ggplot')
```

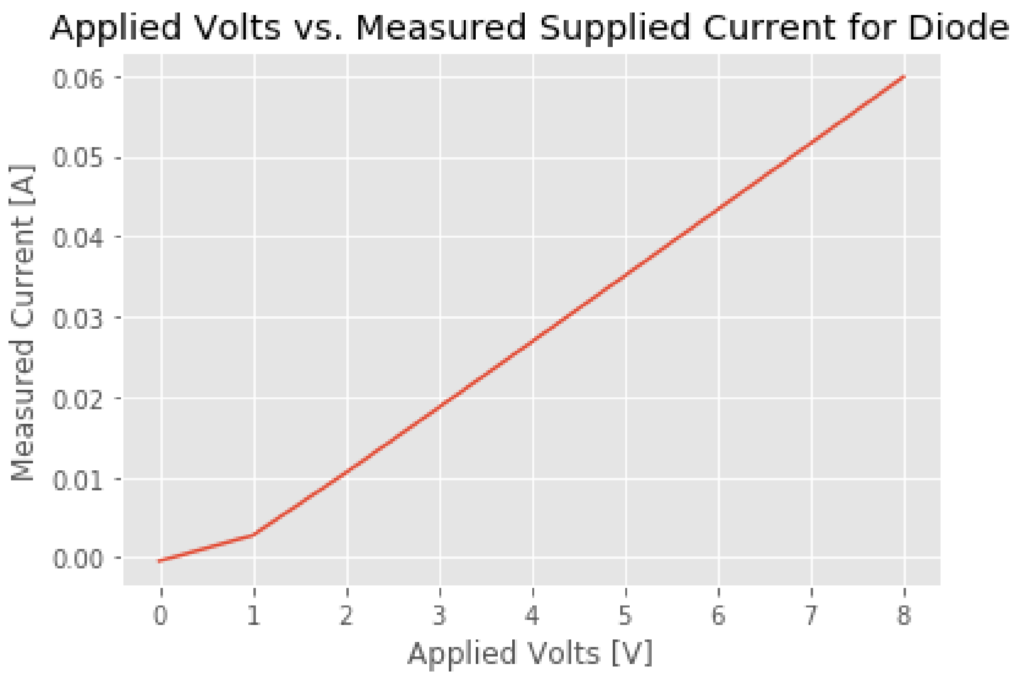
```
#%%
# This section of the code cycles through all USB connected devices to
the computer.
# The code figures out the USB port number for each instrument.
# The port number for each instrument is stored in a variable named
“instrument_id”
# If the instrument is turned off or if you are trying to connect to
the
# keyboard or mouse, you will get a message that you cannot connect on
that port.
device_manager = visa.ResourceManager()
devices = device_manager.list_resources()
number_of_device = len(devices)

power_supply_id = -1;
waveform_generator_id = -1;
digital_multimeter_id = -1;
oscilloscope_id = -1;

# assumes only the DC power supply is connected
for i in range (0, number_of_device):

# check that it is actually the power supply
    try:
        device_temp = device_manager.open_resource(devices[i])
        print("Instrument connect on USB port number [" + str(i) + "] is
" + device_temp.query("*IDN?"))
        if (device_temp.query("*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.2-
6.0-2.0\r\n'):
            power_supply_id = i
        if (device_temp.query("*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.0-
6.0-2.0\r\n'):
            power_supply_id = i
        if (device_temp.query("*IDN?") == 'Agilent
Technologies,33511B,MY52301259,3.03-1.19-2.00-52-00\n'):
            waveform_generator_id = i
        if (device_temp.query("*IDN?") == 'Agilent
Technologies,34461A,MY53207926,A.01.10-02.25-01.10-00.35-01-01\n'):
            digital_multimeter_id = i
        if (device_temp.query("*IDN?") == 'Keysight
Technologies,34461A,MY53212931,A.02.08-02.37-02.08-00.49-01-01\n'):
            digital_multimeter_id = i
        if (device_temp.query("*IDN?") == 'KEYSIGHT TECHNOLOGIES,MSO-X
3024T,MY54440281,07.10.2017042905\n'):
```

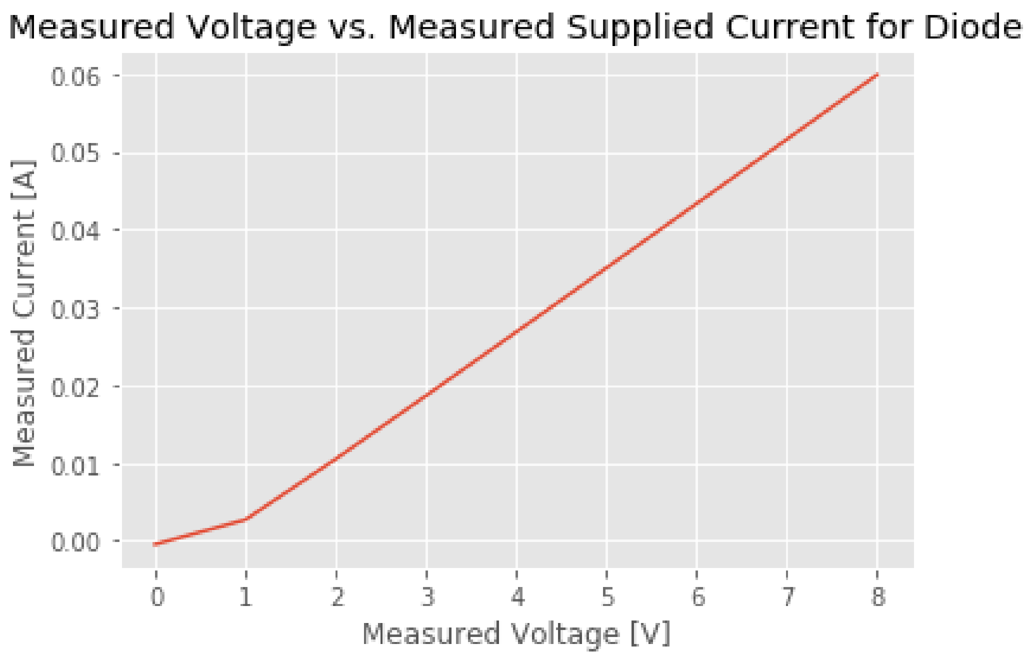
```
            oscilloscope_id = i
        device_temp.close()
    except:
        print("Instrument on USB port number [" + str(i) + "] cannot be
connected. The instrument might be powered of or you are trying to
connect to a mouse or keyboard.\n")


#%%
# Open the USB communication port with the power supply.
# The power supply is connected on USB port number power_supply_id.
# If the power supply ss not connected or turned off, the program will
exit.
# Otherwise, the power_supply variable is the handler to the power
supply

if (power_supply_id == -1):
    print("Power supply instrument is not powered on or connected to the
PC.")
else:
    print("Power supply is connected to the PC.")
    power_supply =
device_manager.open_resource(devices[power_supply_id])

#%%
# The power supply output voltage will be swept from 0 to 1.5V in steps
of 0.05V.
# This voltage will be applied on the 6V output ports.
# For each voltage applied on the 6V power supply, we will measure the
actual
# voltage and current supplied by the power supply.
# If your circuit operates correctly, the applied and measured voltage
will be the same.
# If the power supply reaches its maximum allowed current,
# then the applied voltage will not be the same as the measured
voltage.

    output_voltage = np.arange(0, 8, 0.1)
    measured_voltage = np.array([]) # create an empty list to hold our
values
    measured_current = np.array([]) # create an empty list to hold our
values
    power_supply.write("*CLS")
```

```
    print(power_supply.write("OUTPUT ON")) # power supply output is
turned on

    # loop through the different voltages we will apply to the power
supply
    # For each voltage applied on the power supply,
    # measure the voltage and current supplied by the 6V power supply
    for v in output_voltage:
        # apply the desired voltage on teh 6V power supply and limist
the output current to 0.5A
        power_supply.write("APPLy P25V, %0.2f, 0.06" % v)

        # pause 50ms to let things settle
        time.sleep(0.5)

        # read the output voltage on the 6V power supply
        measured_voltage_tmp = power_supply.query("MEASure:VOLTage:DC?
P25V")
        measured_voltage = np.append(measured_voltage,
measured_voltage_tmp)

        # read the output current on the 6V power supply
        measured_current_tmp = power_supply.query("MEASure:CURRent:DC?
P25V")
        measured_current = np.append(measured_current,
measured_current_tmp)

    # power supply output is turned off
    print(power_supply.write("OUTPUT OFF"))

    # close the power supply USB handler.
    # Otherwise you cannot connect to it in the future
    power_supply.close()

#%%
    # plot results (applied voltage vs measured supplied current)
    plt.figure()
    plt.plot(output_voltage,measured_current)
    plt.title("Applied Volts vs. Measured Supplied Current for Diode")
    plt.xlabel("Applied Volts [V]")
    plt.ylabel("Measured Current [A]")
    plt.draw()

    # plot results (measured voltage vs measured supplied current)
```

```
    plt.figure()
    plt.plot(measured_voltage,measured_current)
    plt.title("Measured Voltage vs. Measured Supplied Current for
Diode")
    plt.xlabel("Measured Voltage [V]")
    plt.ylabel("Measured Current [A]")
    plt.draw()

    # show all plots
    plt.show()
```

2. Plot the current -voltage characteristic of the diode with a 1V step size





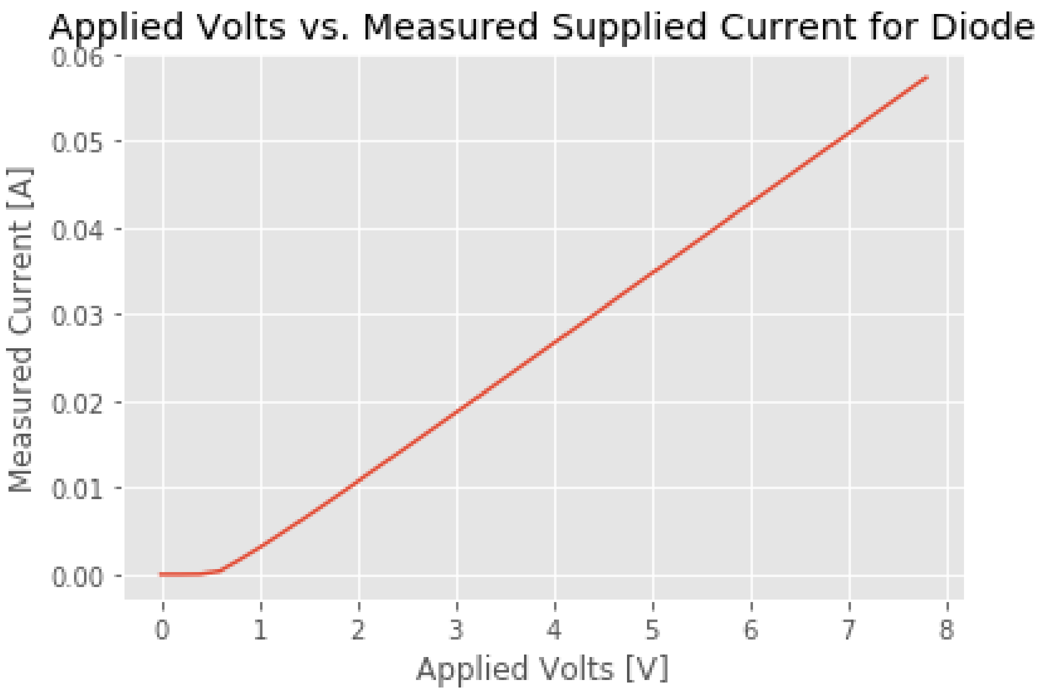
Difference:

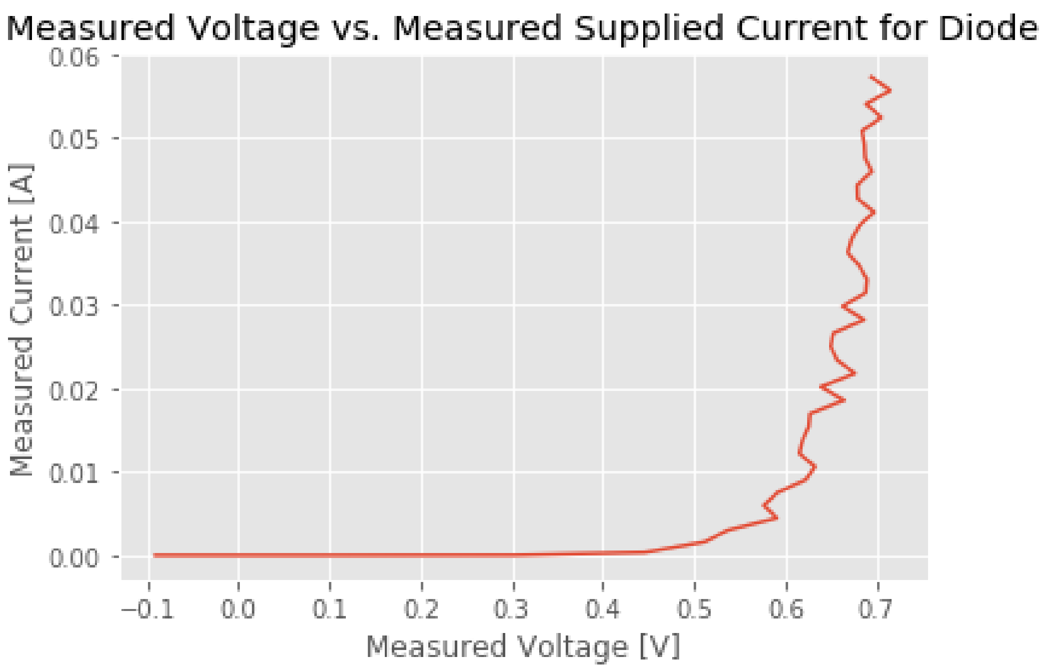
At the beginning, when the step size is 0.1V, the current -voltage characteristic curves are smoother and can better present the real situation of measured current and voltage change.

3. The tradeoffs when determining the increment size.

The reasonable step function should be small enough to make sure the current -voltage characteristic curves are smooth so that it can present the real situation of measured current and voltage change. But the step function cannot be too small since it will cost too much time in the program running.

4. Results:





Comment:
The Applied Volts vs. Measured Supplied Current curve is almost same to the curve in the checkpoint1. But the Measured Voltage vs. Measured Supplied Current curve is much different to the curve in the checkpoint1. The tendency is different.
Reason:
Voltage is measured by oscilloscope, so we can say the oscilloscope is not sensitive and accurate enough. It needs more time to react to the voltage change.
Code:

```
import visa
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import time
mpl.style.use('ggplot')

#%%
# This section of the code cycles through all USB connected devices to
the computer.
# The code figures out the USB port number for each instrument.
# The port number for each instrument is stored in a variable named
“instrument_id”
# If the instrument is turned off or if you are trying to connect to
the
# keyboard or mouse, you will get a message that you cannot connect on
that port.
device_manager = visa.ResourceManager()
devices = device_manager.list_resources()
number_of_device = len(devices)

power_supply_id = -1;
waveform_generator_id = -1;
digital_multimeter_id = -1;
oscilloscope_id = -1;

# assumes only the DC power supply is connected
for i in range (0, number_of_device):

# check that it is actually the power supply
    try:
        device_temp = device_manager.open_resource(devices[i])
        print("Instrument connect on USB port number [" + str(i) + "] is
" + device_temp.query("*IDN?"))
        if (device_temp.query("*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.2-
6.0-2.0\r\n'):
```

```
            power_supply_id = i
        if (device_temp.query("*IDN?") == 'HEWLETT-PACKARD,E3631A,0,3.0-
6.0-2.0\r\n'):
            power_supply_id = i
        if (device_temp.query("*IDN?") == 'Agilent
Technologies,33511B,MY52301259,3.03-1.19-2.00-52-00\n'):
            waveform_generator_id = i
        if (device_temp.query("*IDN?") == 'Agilent
Technologies,34461A,MY53207926,A.01.10-02.25-01.10-00.35-01-01\n'):
            digital_multimeter_id = i
        if (device_temp.query("*IDN?") == 'Keysight
Technologies,34461A,MY53212295,A.02.08-02.37-02.08-00.49-01-01\n'):
            digital_multimeter_id = i
        if (device_temp.query("*IDN?") == 'KEYSIGHT TECHNOLOGIES,MSO-X
3024T,MY55100341,07.10.2017042905\n'):
            oscilloscope_id = i
        device_temp.close()
    except:
        print("Instrument on USB port number [" + str(i) + "] cannot be
connected. The instrument might be powered of or you are trying to
connect to a mouse or keyboard.\n")


#%%
# Open the USB communication port with the power supply.
# The power supply is connected on USB port number power_supply_id.
# If the power supply ss not connected or turned off, the program will
exit.
# Otherwise, the power_supply variable is the handler to the power
supply
if (power_supply_id == -1):
    print("Power supply instrument is not powered on or connected to the
PC.")
else:
    print("Power supply is connected to the PC.")
    power_supply =
device_manager.open_resource(devices[power_supply_id])

if (digital_multimeter_id == -1):
    print("digital_multimeter instrument is not powered on or connected
to the PC.")
else:
    print("digital_multimeter is connected to the PC.")
```

```
    digital_multimeter =
device_manager.open_resource(devices[digital_multimeter_id])

if (oscilloscope_id == -1):
    print("oscilloscope instrument is not powered on or connected to the
PC.")
else:
    print("oscilloscope is connected to the PC.")
    oscilloscope =
device_manager.open_resource(devices[oscilloscope_id])


#%%
# The power supply output voltage will be swept from 0 to 1.5V in steps
of 0.05V.
# This voltage will be applied on the 6V output ports.
# For each voltage applied on the 6V power supply, we will measure the
actual
# voltage and current supplied by the power supply.
# If your circuit operates correctly, the applied and measured voltage
will be the same.
# If the power supply reaches its maximum allowed current,
# then the applied voltage will not be the same as the measured
voltage.

    output_voltage = np.arange(0, 8, 0.2)
    measured_voltage = np.array([]) # create an empty list to hold our
values
    measured_current = np.array([]) # create an empty list to hold our
values
    power_supply.write("*CLS")
    print(power_supply.write("OUTPUT ON")) # power supply output is
turned on

    # loop through the different voltages we will apply to the power
supply
    # For each voltage applied on the power supply,
    # measure the voltage and current supplied by the 6V power supply
    for v in output_voltage:
        # apply the desired voltage on teh 6V power supply and limist
the output current to 0.5A
        power_supply.write("APPLy P25V, %0.2f, 0.06" % v)

        # pause 50ms to let things settle
```

```
        time.sleep(0.5)

        # read the output voltage on the 6V power supply
        measured_voltage_tmp = oscilloscope.query("MEASure:VAVerage?")
        measured_voltage = np.append(measured_voltage,
measured_voltage_tmp)

        # read the output current on the 6V power supply
        measured_current_tmp =
digital_multimeter.query("MEASure:CURRent:DC?")
        measured_current = np.append(measured_current,
measured_current_tmp)

    # power supply output is turned off
    print(power_supply.write("OUTPUT OFF"))

    # close the power supply USB handler.
    # Otherwise you cannot connect to it in the future
    power_supply.close()

#%%
    # plot results (applied voltage vs measured supplied current)
    plt.figure()
    measured_current=[float(x) for x in measured_current]
    measured_voltage=[float(x) for x in measured_voltage]
    print(measured_current)
    print(measured_voltage)
    plt.plot(output_voltage,measured_current)
    plt.title("Applied Volts vs. Measured Supplied Current for Diode")
    plt.xlabel("Applied Volts [V]")
    plt.ylabel("Measured Current [A]")
    plt.draw()

    # plot results (measured voltage vs measured supplied current)
    plt.figure()
    plt.plot(measured_voltage,measured_current)
    plt.title("Measured Voltage vs. Measured Supplied Current for
Diode")
    plt.xlabel("Measured Voltage [V]")
    plt.ylabel("Measured Current [A]")
    plt.draw()

    plt.figure()
    plt.plot(output_voltage,measured_voltage)
```

```python
    plt.title("Applied Volts vs. Measured Supplied Voltage for Diode")
    plt.xlabel("Applied Volts [V]")
    plt.ylabel("Measured Voltage [A]")
    plt.draw()

    # show all plots
    plt.show()
```