# sparkfun_de2120_py

***Release 0.0.01***

**SparkFun Electronics**

**May 26, 2021**

# CONTENTS:

Python module for the SparkFun 2D Barcode Scanner Breakout - DE2120.

This python package is a port of the existing SparkFun DE2120 Arduino Library

CHAPTER

# ONE

## CONTENTS

CHAPTER

TWO

# SUPPORTED PLATFORMS

The Qwiic Button Python package current supports the following platforms:

- Raspberry Pi

CHAPTER

THREE

# DEPENDENCIES

This driver package depends on the pyserial package.

CHAPTER

FOUR

# DOCUMENTATION

The SparkFun 2D Barcode Scanner Breakout module documentation is hosted at ReadTheDocs

CHAPTER

FIVE

# INSTALLATION

## 5.1 PyPi Installation

This repository is hosted on PyPi as the de2120-barcode-scanner package. On systems that support PyPi installation via pip, this library is installed using the following commands

For all users (note: the user must have sudo privileges):

```
sudo pip install de2120-barcode-scanner
```

For the current user:

```
pip install de2120-barcode-scanner
```

To install, make sure the setuptools package is installed on the system.

Direct installation at the command line:

```
python setup.py install
```

To build a package for use with pip:

```
python setup.py sdist
```

A package file is built and placed in a subdirectory called dist. This package file can be installed using pip.

```
cd dist
pip install de2120-barcode-scanner-<version>.tar.gz
```

CHAPTER

# SIX

# EXAMPLE USE

See the examples directory for more detailed use examples.

```
from __future__ import print_function
import de2120_barcode_scanner
import time
import sys

def run_example():

    print("\nSparkFun DE2120 Barcode Scanner Breakout Example 1")
    my_scanner = de2120_barcode_scanner.DE2120BarcodeScanner()

    if my_scanner.begin() == False:
        print("\nThe Barcode Scanner module isn't connected correctly to the system.␣
↪Please check wiring", \
            file=sys.stderr)
        return
    print("\nScanner ready!")

    scan_buffer = ""

    while True:
        scan_buffer = my_scanner.read_barcode()
        if scan_buffer:
            print("\nCode found: " + str(scan_buffer))
            scan_buffer = ""

        time.sleep(0.02)

if __name__ == '__main__':
    try:
        run_example()
    except(KeyboardInterrupt, SystemExit) as exErr:
        print("\nEnding Example 1")
        sys.exit(0)
```

CHAPTER

SEVEN

# TABLE OF CONTENTS

## 7.1 API Reference

### 7.1.1 de2120_barcode_scanner

Python module for the 2D Barcode Scanner.

This python package is a port of the exisiting [SparkFun DE2120 Arduino Library](https://github.com/sparkfun/SparkFun_DE2120_Arduino_Library)

**class** de2120_barcode_scanner.**DE2120BarcodeScanner**(*hard_port=None*)

Initialize the library with the given port.

**Parameters** **hard_port** – The port to use to communicate with the module, this is a serial port at 9600 baud rate.

**Returns** The DE2120BarcodeScanner object.

**Return type** Object

**USB_mode**(*mode*)

Enable USB communication and set the mode. THIS WILL MAKE THE MODULE UNRESPONSIVE ON COM PORT

**Parameters** **mode** – string defining what USB mode to set the module in. Valid arguments are "KBD", "HID", "232".

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

**available**()

**Returns** the number of bytes in the serial receive buffer

**Return type** int

**begin**()

Initializes the device with basic settings. Calls the is_connected() function

**Returns** Returns true if initialization was successful

**Return type** bool

**change_baud_rate**(*baud*)

Change the serial baud rate for the barcode module. Default 115200

**Parameters** **baud** – baud rate to change to

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

`change_buzzer_tone`(*tone*)
Change the buzzer frequency between low, med, and high

**Parameters** `tone` – int that's 1 = low, 2 = med, 3 = high frequency

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

`change_reading_area`(*percent*)
Change the percentage of the frame to scan for barcodes

**Parameters** `percent` – Percentage of frame to scan. Valid values are 100, 80, 60, 40, 20 as stated in the DE2120 Scan Setting Manual

**Returns** true if command successfully sent, false otherwise

**Return type** bool

`disable_all_1D`()
Disable decoding of all 1D symbologies

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

`disable_all_2D`()
Disable decoding of all 2D symbologies

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

`disable_boot_beep`()
Disable beep on module startup

**Returns** true if command successfully sent, false otherwise

**Return type** bool

`disable_decode_beep`()
Disable beep on successful read

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

`disable_image_flipping`()
Disable mirror image reading

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

`enable_all_1D`()
Enable decoding of all 1D symbologies

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

`enable_all_2D`()
Enable decoding of all 2D symbologies

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

`enable_boot_beep()`
Enable beep on module startup

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

`enable_continuous_read(repeat_interval=2)`
Enable continuous reading of barcodes and set the time interval for same-code reads

**Parameters** `repeat_interval` – int parameter. 0: same code output 1 times 1: continuous output with same code without interval 2: continuous output with same code, 0.5 second interval (default) 3: continuous output with same code, 1 second interval

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

`enable_decode_beep()`
Enable beep on successful read

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

`enable_image_flipping()`
Enable mirror image reading

**Returns** true if the command successfully sent, false otherwise

**Return type** bool

`enable_manual_trigger()`
Disable the motioin sensitive and continuous read mode. Return to the default trigger mode.

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

`enable_motion_sense(sensitivity=20)`
Enable the motion sensitive read mode and set sensitivity level

**Parameters** `sensitivity` – int value. The smaller the sensitivity, the more sensitive. Values are taken from the DE2120 settings manual. Valid arguments are: 15 (very high), 20 (high/default), 30 (little high), 50 (general), 100 (low sensitivity)

**Returns** true if command is successfully sent, false otherwise

**Return type** bool

`factory_default()`
Send command to put the module back into facory default settings. This will disconnect the module from the serial port.

**Returns** True if command successfully received, false

otherwise. :rtype: bool

`is_connected()`
Ask the DE2120 for the firmware version.

**Returns** Returns true if the DE2120 responds with an ACK.

Retruns false otherwise. :rtype: bool

`light_off()`
Turn white illumination LED off

**Returns** true if command successfully sent, false otherwise

**Return type** bool

`light_on()`
Turn white illumination LED on

**Returns** true if command successfully sent, false otherwise

**Return type** bool

`read()`

**Returns** the first byte on the serial port

**Return type** int

`read_barcode()`
Read from the serial buffer until we hit a new line character

**Returns** the string in the serial buffer

**Return type** bool

`reticle_off()`
Turn red scan line off

:return true if command successfully sent, false otherwise :rtype: bool

`reticle_on()`
Turn red scan line on

**Returns** true if command successfully sent, false otherwise

**Return type** bool

`send_command(cmd, arg='')`
Create command string and send to DE2120 over serial port. Check serial buffer for a response

**Parameters**

- **cmd** – The command name
- **arg** – The command variation, if there is one

**Returns** True if the response from DE2120 contains the

ACK character, false otherwise. :rtype: bool

`start_scan()`
Start reading when in trigger mode (default)

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

`stop_scan()`
Stop reading when in trigger mode. Module will automatically stop reading after a few seconds

**Returns** true if the command is successfully sent, false otherwise

**Return type** bool

## 7.2 Example One - Serial Scan

Listing 1: examples/de2120_ex1_serial_scan.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# qde2120_ex1_serial_scan.py
#-----------------------------------------------------------------------
#
# Written by Priyanka Makin @ SparkFun Electronics, April 2021
#
# This example demonstrates how to get the scanner connected and will output
# and barcode it sees.
#
# NOTE: you must put the module into COM mode by scanning the PORVIC barcode
# in the datasheet. This will put the module in the correct mode to receive
# and transmit serial.
#
# This package has been developed on a Raspberry Pi 4. Connect the DE2120 Barcode
# Scanner Breakout directly to your Pi using a USB-C cable
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#==================================================================================
# Copyright (c) 2021 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==================================================================================
# Example 1

from __future__ import print_function
import de2120_barcode_scanner
import time
import sys

def run_example():
```

(continues on next page)

(continued from previous page)

```

    print("\nSparkFun DE2120 Barcode Scanner Breakout Example 1")
    my_scanner = de2120_barcode_scanner.DE2120BarcodeScanner()

    if my_scanner.begin() == False:
        print("\nThe Barcode Scanner module isn't connected correctly to the system. Please check wiring", \
            file=sys.stderr)
        return
    print("\nScanner ready!")

    scan_buffer = ""

    while True:
        scan_buffer = my_scanner.read_barcode()
        if scan_buffer:
            print("\nCode found: " + str(scan_buffer))
            scan_buffer = ""

        time.sleep(0.02)

if __name__ == '__main__':
    try:
        run_example()
    except(KeyboardInterrupt, SystemExit) as exErr:
        print("\nEnding Example 1")
        sys.exit(0)
```

## 7.3 Example 2 - Serial Settings

Listing 2: examples/de2120_ex2_serial_settings.py

```
#!/usr/bin/env python
#-----------------------------------------------------------------------------
# de2120_ex2_serial_settings.py
#-----------------------------------------------------------------------
#
# Written by Priyanka Makin @ SparkFun Electronics, April 2021
#
# This example demonstrates how to configure the settings of the DE2120 Breakout
#
# NOTE: you must put the module into COM mode by scanning the PORVIC barcode
# in the datasheet. This will put the module in the correct mode to receive
# and transmit serial.
#
# This package has been developed on a Raspberry Pi 4. Connect the DE2120 Barcode
# Scanner Breakout directly to your Pi using a USB-C cable
#
# Do you like this library? Help support SparkFun. Buy a board!
#
```

(continues on next page)

(continued from previous page)

```
#==============================================================================
# Copyright (c) 2021 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#==============================================================================
# Example 2

from __future__ import print_function
import de2120_barcode_scanner
import time
import sys
import serial

def flash_light(bar_scanner):
    print("\n")
    print("\n-------------------------------------------------")
    print("\n1) Enable flash light")
    print("\n2) Disable flash light")
    print("\n-------------------------------------------------")

    val = input("\nSelect an option number: ")

    if val == '1':
        print("\nWhite scan light on")
        bar_scanner.light_on()
    elif val == '2':
        print("\nWhite scan light off")
        bar_scanner.light_off()
    else:
        print("\nCommand not recognized")

def reticle(bar_scanner):
    print("\n")
    print("\n-------------------------------------------------")
    print("\n1) Enable reticle")
    print("\n2) Disable reticle")
```

(continues on next page)

(continued from previous page)

```
    print("\n-------------------------------------------------")

    val = input("\nSelect an option number: ")

    if val == '1':
        print("\nRed scan reticle on")
        bar_scanner.reticle_on()
    elif val == '2':
        print("\nRed scan reticle off")
        bar_scanner.reticle_off()
    else:
        print("\nCommand not recognized")

def decode_beep(bar_scanner):
    print("\n")
    print("\n-------------------------------------------------")
    print("\n1) Enable decode beep")
    print("\n2) Disable decode beep")
    print("\n-------------------------------------------------")

    val = input("\nSelect an option number: ")

    if val == '1':
        print("\nDecode beep turned on")
        bar_scanner.enable_decode_beep()
    elif val == '2':
        print("\nDecode beep turned off")
        bar_scanner.disable_decode_beep()
    else:
        print("\nCommand not recognized")

def boot_beep(bar_scanner):
    print("\n")
    print("\n-------------------------------------------------")
    print("\n1) Enable beep on module power on")
    print("\n2) Disable beep on module power off")
    print("\n-------------------------------------------------")

    val = input("\nSelect an option number: ")

    if val == '1':
        print("\nBeep on power on enabled")
        bar_scanner.enable_boot_beep()
    elif val == '2':
        print("\nBeep on power on disabled")
        bar_scanner.disable_boot_beep()
    else:
        print("\nCommand not recognized")

def change_buzz_freq(bar_scanner):
    print("\n")
    print("\n-------------------------------------------------")
```

(continues on next page)

(continued from previous page)

```
    print("\n1) Passive low frequency")
    print("\n2) Passive medium frequency")
    print("\n3) Passive high frequency")

    val = input("\nSelect an option number: ")

    if val == '1':
        bar_scanner.change_buzzer_tone(int(val))
    elif val == '2':
        bar_scanner.change_buzzer_tone(int(val))
    elif val == '3':
        bar_scanner.change_buzzer_tone(int(val))
    else:
        print("\nCommand not recognized")

def image_flip(bar_scanner):
    print("\n")
    print("\n-------------------------------------------------")
    print("\n1) Turn on image flipping")
    print("\n2) Turn off image flipping (default)")
    print("\n-------------------------------------------------")

    val = input("\nSelect an option number: ")

    if val == '1':
        bar_scanner.enable_image_flipping()
    elif val == '2':
        bar_scanner.disable_image_flipping()
    else:
        print("\nCommand not recognized")
def reading_area(bar_scanner):
    print("\n")
    print("\n-------------------------------------------------")
    print("\n1) Full width (default)")
    print("\n2) Center 80%")
    print("\n3) Center 60%")
    print("\n4) Center 40%")
    print("\n5) Center 20%")
    print("\n-------------------------------------------------")

    val = input("\nSelect an option number: ")

    if val == '1':
        print("\nScanning 100% of frame")
        bar_scanner.change_reading_area(100)
    elif val == '2':
        print("\nScanning center 80% of frame")
        bar_scanner.change_reading_area(80)
    elif val == '3':
        print("\nScanning center 60% of frame")
        bar_scanner.change_reading_area(60)
    elif val == '4':
```

(continues on next page)

(continued from previous page)

```
        print("\nScanning center 40% of frame")
        bar_scanner.change_reading_area(40)
    elif val == '5':
        print("\nScanning center 20% of frame")
        bar_scanner.change_reading_area(20)
    else:
        print("\nCommand not recognized")

def reading_mode(bar_scanner):
    print("\n")
    print("\n-----------------------------------------------------")
    print("\n1) Manual read mode (default)")
    print("\n2) Continuous read mode")
    print("\n3) Motion sensor mode")
    print("\n-----------------------------------------------------")

    val = input("\nSelect an option number: ")

    if val == '1':
        print("\nManual trigger mode enabled")
        bar_scanner.enable_manual_trigger()
    elif val == '2':
        print("\nContinuous read mode enabled")
        bar_scanner.enable_continuous_read(1)
    elif val == '3':
        print("\nMotion trigger mode enabled")
        bar_scanner.enable_motion_sense()
    else:
        print("\nCommand not recognized")

def symbologies(bar_scanner):
    print("\n")
    print("\n-----------------------------------------------------")
    print("\n1) Enable all 1D symbologies")
    print("\n2) Disable all 1D symbologies")
    print("\n3) Enable all 2D symbologies")
    print("\n4) Disable all 2D symbologies")
    print("\n-----------------------------------------------------")

    val = input("\nSelect an option number: ")

    if val == '1':
        print("\n1D symbologies enabled")
        bar_scanner.enable_all_1D()
    elif val == '2':
        print("\n1D symbologies disabled")
        bar_scanner.disable_all_1D()
    elif val == '3':
        print("\n2D symbologies enabled")
        bar_scanner.enable_all_2D()
    elif val == '4':
        print("\n2D symbologies disabled")
```

(continues on next page)

(continued from previous page)

```
        bar_scanner.disable_all_2D()
    else:
        print("\nCommand not recognized")

def run_example():

    print("\nSparkFun DE2120 Barcode Scanner Breakout Example 2")
    my_scanner = de2120_barcode_scanner.DE2120BarcodeScanner()

    if my_scanner.begin() == False:
        print("\nThe Barcode Scanner module isn't connected correctly to the system. Please check wiring", \
            file=sys.stderr)
        return
    print("\nScanner ready!")

    while True:

        print("\n")
        print("\nSparkFun DE2120 Barcode Scanner Python Package")
        print("\n------------------------------------------------")
        print("\n1) Start Scan")
        print("\n2) Stop Scan")
        print("\n3) Enable/Disable Flashlight")
        print("\n4) Enable/Disable Aiming Reticle")
        print("\n5) Enable/Disable Decode Beep")
        print("\n6) Enable/Disable Start Up Beep")
        print("\n7) Change Buzzer Frequency")
        print("\n8) Enablde/Disable Image Flipping")
        print("\n9) Set Reading Area")
        print("\n10) Set Reading Mode")
        print("\n11) Enable/Disable Symbologies")
        print("\n------------------------------------------------")

        val = input("\nSelect an option number: ")

        if val == '1':
            my_scanner.start_scan()
        elif val == '2':
            my_scanner.stop_scan()
        elif val == '3':
            flash_light(my_scanner)
        elif val == '4':
            reticle(my_scanner)
        elif val == '5':
            decode_beep(my_scanner)
        elif val == '6':
            boot_beep(my_scanner)
        elif val == '7':
            change_buzz_freq(my_scanner)
        elif val == '8':
            image_flip(my_scanner)
```

(continues on next page)

(continued from previous page)

```
        elif val == '9':
            reading_area(my_scanner)
        elif val == '10':
            reading_mode(my_scanner)
        elif val == '11':
            symbologies(my_scanner)
        else:
            print("\nCommand not recognized")

if __name__ == '__main__':
    try:
        run_example()
    except (KeyboardInterrupt, SystemExit) as exErr:
        print("\nEnding Example 2")
        sys.exit(0)
```

## 7.4 Example 3 - Send Command

Listing 3: examples/de2120_ex3_send_command.py

```
#!/usr/bin/env python
#-------------------------------------------------------------------------------
# de2120_ex3_send_command.py
#-------------------------------------------------------------------------------
#
# Written by Priyanka Makin @ SparkFun Electronics, April 2021
#
# This example demonstrates how to use the "send_command()" method to send
# arbitrary serial commands to the barcode reader. It also demonstrates the "CIDENA"
# or "Code ID Enable" function, which includes the barcode type when transmitting the
# decoded string.
#
# send_command() takes two strings as arguments, concatenate them, adds the command
# prefix "^_^" and the command suffix "." and then transmits the command to the module.
# For example, to enable matrix 2 of 5 scanning, which is done using the command
# "^_^M25ENA1." you would make the call "my_scanner.send_command("M25ENA", 1)"
#
# While it is valid to call "my_scanner.send_command("M25ENA1")", the former method
# is preferred in many cases.
#
# NOTE: you must put the module into COM mode by scanning the PORVIC barcode
# in the datasheet. This will put the module in the correct mode to receive
# and transmit serial.
#
# This package has been developed on a Raspberry Pi 4. Connect the DE2120 Barcode
# Scanner Breakout directly to your Pi using a USB-C cable
#
# Do you like this library? Help support SparkFun. Buy a board!
#
#===============================================================================
```

(continues on next page)

(continued from previous page)

```
# Copyright (c) 2021 SparkFun Electronics
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#===============================================================================
# Example 3

from __future__ import print_function
import de2120_barcode_scanner
import time
import sys

def run_example():

    print("\nSparkFun DE2120 Barcode Scanner Breakout Example 3")
    my_scanner = de2120_barcode_scanner.DE2120BarcodeScanner()

    if my_scanner.begin() == False:
        print("\nThe Barcode Scanner module isn't connected correctly to the system. Please check wiring", \
            file=sys.stderr)
        return
    print("\nScanner ready!")

    print("\n")
    print("\nTransmit Code ID with Barcode? (y/n)")
    print("\n-----------------------------------------")

    val = input("\nType 'y' or 'n' or scan a barcode: ")

    if val == 'y':
        print("\nCode ID will be displayed on scan")
        my_scanner.send_command("CIDENA", "1")
    elif val == 'n':
        print("\nCode ID will NOT be displayed on scan")
        my_scanner.send_command("CIDENA", "0")
    else:
```

(continues on next page)

(continued from previous page)

```
        print("\nCommand not recognized")

    scan_buffer = ""

    while True:
        scan_buffer = my_scanner.read_barcode()
        if scan_buffer:
            print("\nCode found: ")
            print("\n" + str(scan_buffer))
            scan_buffer = ""

        time.sleep(0.02)


if __name__ == '__main__':
    try:
        run_example()
    except (KeyboardInterrupt, SystemExit) as exErr:
        print("\nEnding Example 3")
        sys.exit(0)
```

CHAPTER

EIGHT

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## d

# INDEX

## S



## U