Seven Segment Display

Purpose Of This Page

This page explains how to show numbers on the CodyNick Seven Segment Display using Python.

The Seven Segment Display is useful for showing:

- Scores.
- Counts.
- Sensor values.
- Timer values.
- Simple numeric feedback.

By the end of this page, students will be able to:

- Display a positive number.
- Display a negative number.
- Display a decimal number.
- Update the displayed value inside a Python program.

CodyNick Seven Segment Display

Figure 1 - CodyNick Seven Segment Display for showing numeric values.

Start Code

Every CodyNick Python script should import the CodyNick library and connect to the CodyNick device.

```
import CodyNick


cn = CodyNick.CN()
```

The variable `cn` represents the connected CodyNick device.

Display A Number

Use `Seven_Segment.display()` to show a number.

Function format:

```
CodyNick.Seven_Segment.display(cn, value)
```

Example:

```
import CodyNick

cn = CodyNick.CN()

CodyNick.Seven_Segment.display(cn, 1234)
```

This displays `1234`.

Display A Negative Number

The display can show negative values.

Example:

```
import CodyNick

cn = CodyNick.CN()

CodyNick.Seven_Segment.display(cn, -123)
```

This displays `-123`.

Display A Decimal Number

The display can also show decimal values.

Example:

```
import CodyNick

cn = CodyNick.CN()

CodyNick.Seven_Segment.display(cn, 1.234)
```

This displays `1.234` if the value fits on the display.

Using Strings Or Numbers

The value can be passed as a number:

```
CodyNick.Seven_Segment.display(cn, 1234)
```

It can also be passed as a string:

```
CodyNick.Seven_Segment.display(cn, "1234")
```

Both examples display the same value.

Display Range

The display is designed for short numeric values.

Typical useful values are:

Type	Example
Positive integer	`1234`
Negative integer	`-123`
Decimal number	`1.234`
Small decimal	`-1.23`

Very large or very small values may not fit on the display.

In the CodyNick Python library, values outside the normal display range are treated as out of range.

Example: Count Up

This example counts from `0` to `9`.

```
import CodyNick
import time

cn = CodyNick.CN()

for number in range(10):
    CodyNick.Seven_Segment.display(cn, number)
    time.sleep(0.5)
```

The number changes every half second.

Example: Countdown

This example counts down from `5` to `0`.

```
import CodyNick
import time

cn = CodyNick.CN()
```

```
for number in range(5, -1, -1):
    CodyNick.Seven_Segment.display(cn, number)
    time.sleep(1)
```

The third value in `range(5, -1, -1)` means the loop counts down by `1`.

Example: Show Decimal Values

This example displays a few decimal values.

```
import CodyNick
import time

cn = CodyNick.CN()

values = [1.234, 2.5, 3.75, -1.23]

for value in values:
    CodyNick.Seven_Segment.display(cn, value)
    time.sleep(1)
```

Function Summary

Function	Purpose	Example
`Seven_Segment.display(cn, value)`	Display a number	`display(cn, 1234)`

Practice Tasks

Try these exercises:

1. Display `1234`.
2. Display `-123`.
3. Display `1.234`.
4. Count from `0` to `9`.
5. Count down from `9` to `0`.
6. Display a list of decimal values.
7. Create a timer that counts seconds.

Common Mistakes

The first argument must be the connected CodyNick device:

```
CodyNick.Seven_Segment.display(cn, 1234)
```

This is not correct:

```
CodyNick.Seven_Segment.display(1234)
```

If a value is too long, it may not appear as expected on the display.

Use short numeric values:

```
CodyNick.Seven_Segment.display(cn, 1234)
```

When using a loop, add a delay so each value can be seen:

```
time.sleep(0.5)
```

> **Page summary:**
> The Seven Segment Display shows short numeric values. Use `Seven_Segment.display(cn, value)` to display integers, negative numbers, or decimal numbers from Python.

Revision #1
Created 2026-05-15 04:54:31 UTC by Admin
Updated 2026-05-15 04:55:15 UTC by Admin