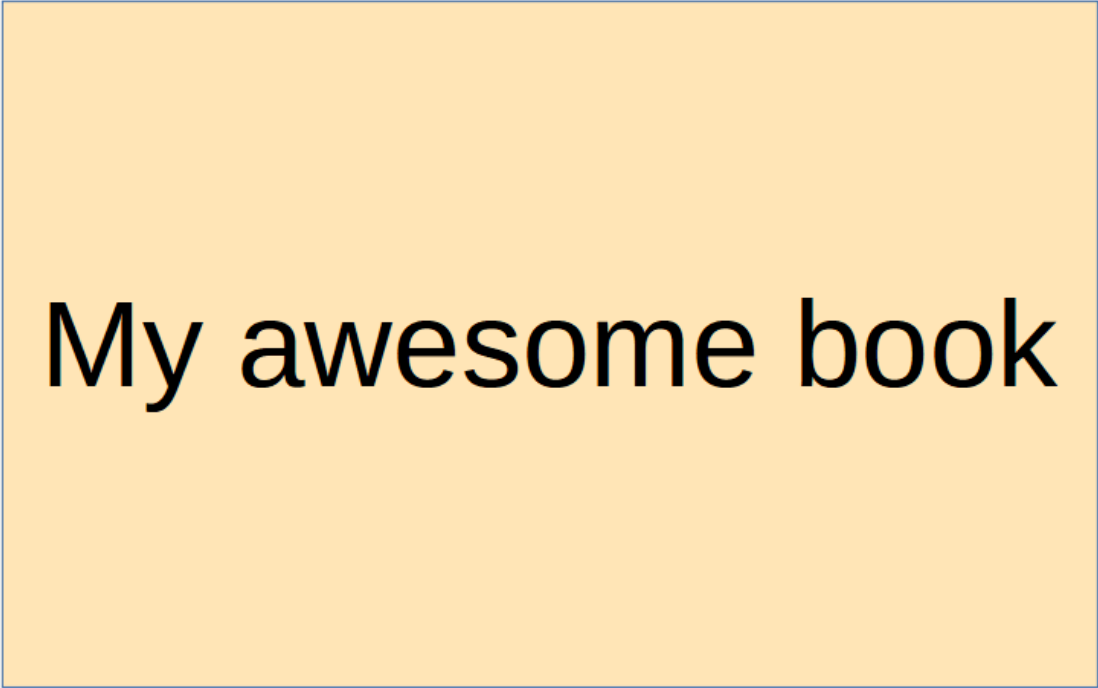


Python Skills for Network Engineers

Network Programmability

Syed Asif

Nov 22, 2023



My awesome book

Table of contents

1	Introduction	4
1.1	What you'll learn	4
1.1.1	About the Author	4
1.2	Conventions	4
2	Python	5
2.1	Example	5
3	Ruby	6
3.1	Example	6
4	CLI	7
4.1	Quote	7
4.2	Links	7
5	Back Ground	8
6	Syntax	9
7	Bullets	10
7.1	Picture	10
8	Conclusion	11
9	Appendix A: Additional Resources	12
9.1	Pandoc Command	12
9.2	Py pandoc	12

1 Introduction

Learn network programmability with Python, GNS3, and Cisco devices.

1.1 What you'll learn

1. Python fundamentals
2. Network automation with Python

1.1.1 About the Author

Associate Engineer (DAE in Electronics), learning network automation as a hobby after 25 years of job in the field of Computer Networking. Skill as an Associate Engineer:

- Routing and Switching
- OFC/LAN Networking
- IP Addressing and Sub-netting
- Computer Basics - Windows 7/10
- Linux and Ubuntu Desktop/Server
- VMware/KVM/VirtualBox
- Docker/Vagrant - Hands On
- Ansible for Network Automation
- Python for Network Automation

1.2 Conventions

This book is a guide for network engineers and is made for networks to write casual code, so there's not much time spent on style and beauty. Programming concepts like object-oriented programming aren't covered in detail because of their complexity. But this book is mainly concerned with getting programs to work with the minimum amount of effort to automate network equipment.

2 Python

- Lists are declared within `[]` and elements are separated by `,`
- Each element can be of any data type, including list data type

2.1 Example

Use `for` loop to iterate over a list.

```
numbers = [2, 12, 3, 25, 624, 21, 5, 9, 12]
odd_numbers = []
even_numbers = []

for num in numbers:
    odd_numbers.append(num) if (num % 2) else even_numbers.append(num)

print(f'numbers:      {numbers}')
print(f'odd_numbers:  {odd_numbers}')
print(f'even_numbers: {even_numbers}')
```

3 Ruby

- Arrays are declared within `[]` and elements are separated by `,`
- Each element can be of any data type, including array data type

3.1 Example

Use `each` method to iterate over an array.

```
numbers = [2, 12, 3, 25, 624, 21, 5, 9, 12]
odd_numbers = []
even_numbers = []

numbers.each {
  |n| n.even? ? even_numbers.append(n) : odd_numbers.append(n)
}

puts "numbers:      #{numbers}"
puts "odd_numbers:  #{odd_numbers}"
puts "even_numbers: #{even_numbers}"
```

4 CLI

Executing the Python and Ruby programs mentioned in previous chapters:

```
$ python3.7 list_looping.py
numbers:      [2, 12, 3, 25, 624, 21, 5, 9, 12]
odd_numbers:  [3, 25, 21, 5, 9]
even_numbers: [2, 12, 624, 12]

$ ruby array_looping.rb
numbers:      [2, 12, 3, 25, 624, 21, 5, 9, 12]
odd_numbers:  [3, 25, 21, 5, 9]
even_numbers: [2, 12, 624, 12]
```

4.1 Quote

This is a quote

4.2 Links

This is a sample [GitHub style markdown](#) file. Top level headers are chapters and other headings are for sub-sections.

5 Back Ground

By default, `background` is set to `null`.

Can be changed to different color to suit the chosen theme.

```
# sample comment: a/b != a\b
```

```
>>> 2 ** 5
```

```
32
```

Executing Python program from shell:

```
$ python3.7 list_looping.py
```

```
numbers:      [2, 12, 3, 25, 624, 21, 5, 9, 12]
```

```
odd_numbers:  [3, 25, 21, 5, 9]
```

```
even_numbers: [2, 12, 624, 12]
```

Just another line.

6 Syntax

- Using python syntax highlighting

```
# remove first two columns where : is delimiter
>>> re.sub(r'\A(?:.+){2}', r'', 'foo:123:bar:baz', count=1)
'bar:baz'
>>> ''.join(re.findall(r'\b\w', 'sea eat car rat eel tea'))
'secret'
# match 'dog' only if it is not preceded by 'parrot'
>>> bool(regex.search(r'(?<!parrot.*)dog', 'fox,cat,dog,parrot'))
True
```

- Using ruby syntax highlighting

```
# remove first two columns where : is delimiter
>>> re.sub(r'\A(?:.+){2}', r'', 'foo:123:bar:baz', count=1)
'bar:baz'
>>> ''.join(re.findall(r'\b\w', 'sea eat car rat eel tea'))
'secret'
# match 'dog' only if it is not preceded by 'parrot'
>>> bool(regex.search(r'(?<!parrot.*)dog', 'fox,cat,dog,parrot'))
True
```

7 Bullets

- fruit
 - apple
 - mango
 - grape
 - ★ green
 - ★ black seedless
- pet
 - cat
 - dog
 - parrot

7.1 Picture



Figure 1: Picture

8 Conclusion

This sample file helps you see a demo for `markdown` to `pdf` conversion using `pandoc`.

9 Appendix A: Additional Resources

This section contains additional resources related to the content of the book.

9.1 Pandoc Command

Use this command to create a pdf book.

```
pandoc chapters/*.md -o output.pdf \  
--metadata-file=metadata.yml \  
--resource-path=. --toc --number-sections \  
--include-in-header main.tex \  
--highlight-style pygments.theme \  
-V geometry:a5paper \  
-V geometry:margin=2cm \  
-V fontsize=12pt
```

9.2 Pypandoc

Pypandoc provides a thin wrapper for pandoc, a universal document converter.

```
import os  
import pypandoc  
  
input_dir = "chapters"  
output_dir = "my_book"  
output_filename = "book_output_02.pdf"  
  
# Ensure the output directory exists  
os.makedirs(output_dir, exist_ok=True)  
  
# Build the pandoc options as a string  
pandoc_cmd = [  
    "--toc",  
    "--number-sections",  
    "--resource-path=.",  
    "--metadata-file", "metadata.yml",  
    "--include-in-header", "main.tex",  
    "--highlight-style", "pygments.theme",  
    "-V", "toc-title=Table of contents",  
    "-V", "linkcolor:blue",  
    "-V", "geometry:a4paper",  
    "-V", "geometry:margin=1.8cm",  
    "-V", "mainfont=DejaVu Serif",  
    "-V", "monofont=SauceCodePro Nerd Font",  
    "--pdf-engine=xelatex"  
]  
  
output_path = os.path.join(output_dir, output_filename)  
# Convert all markdown files in the chapters/ subdirectory.  
pypandoc.convert_file(  
    os.path.join(input_dir, '*.md'), 'pdf',  
    outfile=output_path,  
    extra_args=pandoc_cmd  
)  
  
print(f"Conversion completed. Output saved to: {output_path}")
```