How to Install Python3.6 in Ubuntu and CentOS7

lundi, 12 août 2019

20:47

**On Ubuntu:**

To remove python2 completely,

sudo apt purge python2.7-minimal

To install python3.6,

sudo add-apt-repository ppa:jonathonf/python-3.6

sudo apt-get update

sudo apt-get install python3.6

To check python3.6 version

python3.6 -V

To make python3 use the new installed python 3.6 instead of the default 3.5 release, run following 2 commands:

sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.5 1

sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.6 2

Finally switch between the two python versions for python3 via command

sudo update-alternatives --config python3

root@PyNetServer:~# python3 --version

Python 3.6.8

To set an alias for python3.6 as python

root@NornirServer:~# pwd

/root

root@NornirServer:~# nano .bashrc

##add this line###

alias python='python3.6'

root@NornirServer:~# source .bashrc

**To install openssh-server and start the service**

apt update

apt install openssh-server

service ssh start

service ssh status

**To install git and netmiko-tools on Ubuntu**

apt-get update

apt-get install git-core

Git --version

git clone <https://github.com/ktbyers/netmiko_tools/>

export PATH=/root/Nornir/netmiko\_tools/netmiko\_tools:$PATH

root@NornirServer:~# source .bashrc

You need to have netmiko.yml file in the home directory which contains the hosts/groups.

root@Nornir:~# env | grep HOME

HOME=/root/Nornir

root@Nornir:~# ls -al | grep netmiko.yml

-rw-r--r-- 1 1000 1000 767 Oct 17 15:22 netmiko.yml

root@Nornir:~# cat netmiko.yml

---

# Dictionaries are devices

L2SWA:

device\_type: cisco\_ios

ip: 192.168.122.243

username: cisco

password: 123!Cisco

# Lists are groups of devices

ios:

- L2SWA

On CentOS 8:

To set a hostname permanently

sudo hostnamectl set-hostname NEW\_HOSTNAME

To Install python

Sudo yum update

* This automatically installs python3.6

To install ipython

python3.6 -m pip install ipython==7.13.0

To install Nornir

python3.6 -m pip install nornir==2.4.0

To install sensible

python3.6 -m pip install ansible=2.9.0

**On CentOS7:**

**To Install Python**

sudo yum install -y <https://centos7.iuscommunity.org/ius-release.rpm>

sudo yum update

sudo yum install -y python36u python36u-libs python36u-devel python36u-pip

python3.6 -V

# start by registering python2 as an alternative

alternatives --install /usr/bin/python python /usr/bin/python2 50

# register python3.6 as an alternative

alternatives --install /usr/bin/python python /usr/bin/python3.6 60

# Select the python to use

alternatives --config python

**To install ipython**

python3.6 -m pip install ipython==7.7.0

**To install Nornir**

python3.6 -m pip install nornir==2.2.0

**To install tree**

python3.6 -m pip install tree

Class 1 - Reading Pending

lundi, 28 janvier 2019

16:16

1. Python variables can be Letters, numbers and underscore. You cannot use any other special characters

2. You cannot start the variable with a number

3. Python variables with leading and trailing underscore has special meaning so better avoid using variable names starting or ending with underscore.

4. If the variable name contains 2 words use underscore to seperate it. e.g my\_device, ip\_addr etc

5. If you want the system to use python to execute the code, then use #!/usr/bin/env python and change the permission. See this

root@PyNetServer:/home/pyscripts# cat scr1.py

#!/usr/bin/env python

print 'Do Nothing'

root@PyNetServer:/home/pyscripts# ./scr1.py

bash: ./scr1.py: Permission denied

root@PyNetServer:/home/pyscripts# chmod 755 scr1.py

root@PyNetServer:/home/pyscripts# ./scr1.py

Do Nothing

6. There is a difference in print function between python2 and python3. see the following. I am executing the script with python3 first and it shows me that parantheses are missing.

root@PyNetServer:/home/pyscripts# python3 scr1.py

File "scr1.py", line 2

print 'Do Nothing'

^

SyntaxError: Missing parentheses in call to 'print'

I am trying to execute the same with python2 and it works.

root@PyNetServer:/home/pyscripts# python2 scr1.py

Do Nothing

so for python3, the print function should be inside brackets like this.

root@PyNetServer:/home/pyscripts# cat scr1.py

#!/usr/bin/env python

print ('Do Nothing')

root@PyNetServer:/home/pyscripts# python3 scr1.py

Do Nothing

8. If you want to write a code that works with print function for both python2 and python3, do the following

#!/usr/bin/env python3

from \_\_future\_\_ import print\_function

print ('Do Nothing')

Going forward the print statements should be inside the brackets.

9. There is a difference between python2 and python3 in the input taken. See this.

root@PyNetServer:/home/pyscripts# cat scr1.py

#!/usr/bin/env python3

from \_\_future\_\_ import print\_function

dns = input("Enter google dns: ")

print (dns)

root@PyNetServer:/home/pyscripts# python3 scr1.py

Enter google dns: 8.8.8.8

8.8.8.8

But if I run the same script with python2, see what happens

root@PyNetServer:/home/pyscripts# python2 scr1.py

Enter google dns: 8.8.8.8

Traceback (most recent call last):

File "scr1.py", line 3, in <module>

dns = input("Enter google dns: ")

File "<string>", line 1

8.8.8.8

^

SyntaxError: invalid syntax

To solve this we need use raw\_input, like the following but this will not work with python3. see this

root@PyNetServer:/home/pyscripts# cat scr1.py

#!/usr/bin/env python3

from \_\_future\_\_ import print\_function

dns = raw\_input("Enter google dns: ")

print (dns)

root@PyNetServer:/home/pyscripts# python2 scr1.py

Enter google dns: 8.8.8.8

8.8.8.8

root@PyNetServer:/home/pyscripts# python3 scr1.py

Traceback (most recent call last):

File "scr1.py", line 3, in <module>

dns = raw\_input("Enter google dns: ")

NameError: name 'raw\_input' is not defined

so to write a code that supports the print function and input function on python2 and python3, do the following.

root@PyNetServer:/home/pyscripts# cat scr1.py

#!/usr/bin/env python3

from \_\_future\_\_ import print\_function

try:

dns = raw\_input("Enter google dns: ")

except NameError:

dns = input("Enter google dns: ")

print (dns)

root@PyNetServer:/home/pyscripts# python2 scr1.py

Enter google dns: 8.8.8.8

8.8.8.8

root@PyNetServer:/home/pyscripts# python3 scr1.py

Enter google dns: 8.8.8.8

8.8.8.8

10. We will look at the different methods/functions available for different data types. I am creating a string and I could see what are the methods/functions available for the data type string.

>>> my\_str = 'hello'

>>> dir(my\_str)

['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getat

tribute\_\_', '\_\_getitem\_\_', '\_\_getnewargs\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_iter\_\_', '\_\_le\_\_'

, '\_\_len\_\_', '\_\_lt\_\_', '\_\_mod\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_rmod\_\_',

'\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'capitalize', 'casefold', 'center', 'count',

'encode', 'endswith', 'expandtabs', 'find', 'format', 'format\_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal',

'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust',

'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',

'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

>>> my\_str.capitalize()

'Hello'

>>> my\_str.upper()

'HELLO'

>>> my\_str.lower()

'hello'

>>> my\_str.upper().lower().capitalize()

'Hello'

>>>

>>> my\_string = ' This line has extra space, strip '

>>> my\_string

' This line has extra space, strip '

>>> my\_string.strip()

'This line has extra space, strip'

The strip method has stripped the leading and trailing empty space.

11. If you would like to know what is the purpose of the method you can use the help function. Here I am checking what does the "my\_str.capitalize" function does.

>>> help(my\_str.capitalize)

Help on built-in function capitalize:

capitalize() method of builtins.str instance

Return a capitalized version of the string.

More specifically, make the first character have upper case and the rest lower.

12. The variable will be referencing a memory . For example

>>> ip\_addr = '192.168.1.1'

>>> id(ip\_addr)

1503336691184

>>> ip\_addr2 = ip\_addr

>>> id(ip\_addr)

1503336691184

>>> id(ip\_addr2)

1503336691184

You can see the variables ip\_addr and ip\_addr2 has the same value and pointing to the same memory.

But once you change the value assigned to the variable it automatically changes the memory reference.

>>> ip\_addr = '10.1.1.1'

>>> id(ip\_addr)

1503373540144

>>> id(ip\_addr2)

1503336691184

>>> ip\_addr

'10.1.1.1'

>>> ip\_addr2

'192.168.1.1'

13. In Python2, the strings are by default ascii strings but in python3, by default strings are unicode strings. The difference between ascii and unicode is, in unicode you can have more number of characters.

so to create ascii strings in python3, you need to make them byte strings.

In [1]: ip\_addr = '10.1.1.1'

In [2]: type(ip\_addr)

Out[2]: str

In [5]: ip\_addr = b'10.1.1.1'

In [6]: type(ip\_addr)

Out[6]: bytes

so to write a code that supports both python2 and python3 type of strings, do import unicode\_literals

#!/usr/bin/env python3

from \_\_future\_\_ import print\_function, unicode\_literals

ip\_addr = '10.1.1.1'

print(type(ip\_addr))

PS C:\Users\vignesh.sethuraman\My PyScripts> python pyscript1.py

<class 'str'>

14. The following are boolean operators

= assignment

== comparison

!= not equals

>, >= greather than or equal to

<, <= less than or equal to

You can also see if the sub string is present or not present in the variable

In [1]: ip\_addr = '10.1.1.1'

In [4]: '10.1' in ip\_addr

Out[4]: True

In [5]: '15.1' in ip\_addr

Out[5]: False

In [6]: '15.1' not in ip\_addr

Out[6]: True

15. Strings will also have indexes. The indexes starts from 0

In [1]: ip\_addr = '10.1.1.1'

In [7]: ip\_addr[0]

Out[7]: '1'

In [8]: ip\_addr[1]

Out[8]: '0'

You can also get the last character in the string.

In [9]: ip\_addr[-1]

Out[9]: '1'

In [10]: ip\_addr[-2]

Out[10]: '.'

In [11]: ip\_addr[-3]

Out[11]: '1'

16. To know the length of the string

In [12]: len(ip\_addr)

Out[12]: 8

17. you can do string concatenation

In [17]: sub\_mask = ' 255.255.255.0'

In [18]: ip\_addr + sub\_mask

Out[18]: '10.1.1.1 255.255.255.0'

18. The \n and \t has special meaning so when we need to use them in the strings as normal strings, you need to make them as raw strings.

>>> my\_path = "C:\Users\newdir\test"

File "<stdin>", line 1

SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position 2-3: truncated \UXXXXXXXX escape

>>> my\_path = r"C:\Users\newdir\test"

>>> print(my\_path)

C:\Users\newdir\test

19. If you want to enter strings in multiple lines use triple quotes.

>>> my\_string = ''' This is multiple line

... line 1

... line 2

... line 3'''

>>> print (my\_string)

This is multiple line

line 1

line 2

line 3

>>>

To know how the strings looks like from the python shell perspective, you can use this "repr"

>>> print(repr(my\_string))

' This is multiple line\nline 1\nline 2\nline 3'