Session 3

RAVINDRA KUDACHE

Python 3 and Python2 Diff

Install Required Packages

Use the following command to install prerequisites for Python before installing it.

```
# yum install gcc
```

Download Python 3.5.2

Download Python using following command from python official site. You can also download latest version in place of specified below.

```
# cd /usr/src
# wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
```

Now extract the downloaded package.

```
# tar xzf Python-3.5.2.tgz
```

Python 3 and Python2 Diff

Install Required Packages

Use the following command to install prerequisites for Python before installing it.

```
# yum install gcc
```

Download Python 3.5.2

Download Python using following command from python official site. You can also download latest version in place of specified below.

```
# cd /usr/src
# wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
```

Now extract the downloaded package.

```
# tar xzf Python-3.5.2.tgz
```

Python 3 and Python2 Diff

Compile Python Source

Use below set of commands to compile python source code on your system using altinstall.

```
# cd Python-3.5.2
# ./configure
# make altinstall
```

make altinstall is used to prevent replacing the default python binary file /usr/bin/python.

Now remove downloaded source archive file from your system

```
# rm Python-3.5.2.tgz
```

Check Python Version

Check the latest version installed of python using below command

```
# python3.5 -V

Python 3.5.2
```

Python 3 and Python2 Diff

Prerequisites

You will need a CentOS 7 computer with a non-root superuser account that is connected to the internet.

Step 1 — Preparing the System

We will be completing this installation through the command line. If your CentOS 7 computer starts up with a Graphical User Interface (GUI) desktop, you can gain access to the command line interface through the Menu, by navigating to Applications, then Utilities, and then clicking on Terminal. If you need more guidance on the terminal, be sure to read through the article "An Introduction to the Linux Terminal."

Before we begin with the installation, let's make sure to update the default system applications to have the latest versions available.

We will be using the open-source package manager tool **yum**, which stands for Yellowdog Updater Modified. This is a commonly used tool for working with software packages on Red Hat based Linux systems like CentOS. It will let you easily install and update, as well as remove software packages on your computer.

Let's first make sure that yum is up to date by running this command:

The `-y` flag is used to alert the system that we are aware that we are making changes, preventing the terminal from prompting us to confirm.

Next, we will install **yum-utils**, a collection of utilities and plugins that extend and supplement yum:

```
$ sudo yum -y install yum-utils
```

Finally, we'll install the CentOS Development Tools, which are used to allow you to build and compile software from source code:

```
$ sudo yum -y groupinstall development
```

Python 3 and Python2 Diff

Step 2 — Installing and Setting Up Python 3

CentOS is derived from RHEL (Red Hat Enterprise Linux), which has stability as its primary focus. Because of this, tested and stable versions of applications are what is most commonly found on the system and in downloadable packages, so on CentOS you will only find Python 2.

Since instead we would like to install the most current upstream stable release of Python 3, we will need to install **IUS**, which stands for Inline with Upstream Stable. A community project, IUS provides Red Hat Package Manager (RPM) packages for some newer versions of select software.

To install IUS, let's install it through `yum`:

```
$ sudo yum -y install https://centos7.iuscommunity.org/ius-release.rpm
```

Once IUS is finished installing, we can install the most recent version of Python:

```
$ sudo yum -y install python35u
```

When the installation process of Python is complete, we can check to make sure that the installation was successful by checking for its version number with the `python3.5` command:

```
$ python3.5 -V
```

With a version of Python 3.5 successfully installed, we will receive the following output:

```
Output
Python 3.5.3
```

Python 3 and Python2 Diff

The __future__ module

Python 3.x introduced some Python 2-incompatible keywords and features that can be imported via the in-built __future__ module in Python 2. It is recommended to use __future__ imports, if you are planning Python 3.x support for your code.

For example, if we want Python 3.x's integer division behavior in Python 2, add the following import statement.

```
from __future__ import division
```

The print Function

Most notable and most widely known change in Python 3 is how the **print** function is used. Use of parenthesis () with print function is now mandatory. It was optional in Python 2.

```
print "Hello World" #is acceptable in Python 2
print ("Hello World") # in Python 3, print must be followed by ()
```

The print() function inserts a new line at the end, by default. In Python 2, it can be suppressed by putting ',' at the end. In Python 3, "end=' '" appends space instead of newline.

```
print x,           # Trailing comma suppresses newline in Python 2
print(x, end=" ")  # Appends a space instead of a newline in Python 3
```

Python 3 and Python2 Diff

Reading Input from Keyboard

Python 2 has two versions of input functions, **input()** and **raw_input()**. The input() function treats the received data as string if it is included in quotes '' or "", otherwise the data is treated as number.

In Python 3, raw_input() function is deprecated. Further, the received data is always treated as string.

Integer Division

In Python 2, the result of division of two integers is rounded to the nearest integer. As a result, 3/2 will show 1. In order to obtain a floating-point division, numerator or denominator must be explicitly used as float. Hence, either 3.0/2 or 3/2.0 or 3.0/2.0 will result in 1.5

Python 3 evaluates 3 / 2 as 1.5 by default, which is more intuitive for new programmers.

Unicode Representation

Python 2 requires you to mark a string with a **u** if you want to store it as Unicode.

Python 3 stores strings as Unicode, by default. We have Unicode (utf-8) strings, and 2 byte classes: byte and byte arrays.

Python 3 and Python2 Diff

xrange() Function Removed

In Python 2 range() returns a list, and xrange() returns an object that will only generate the items in the range when needed, saving memory.

In Python 3, the range() function is removed, and xrange() has been renamed as range(). In addition, the range() object supports slicing in Python 3.2 and later .

raise exceprion

Python 2 accepts both notations, the 'old' and the 'new' syntax; Python 3 raises a SyntaxError if we do not enclose the exception argument in parenthesis.

```
raise IOError, "file error" #This is accepted in Python 2
raise IOError("file error") #This is also accepted in Python 2
raise IOError, "file error" #syntax error is raised in Python 3
raise IOError("file error") #this is the recommended syntax in Python 3
```

Python 3 and Python2 Diff

Arguments in Exceptions

In Python 3, arguments to exception should be declared with 'as' keyword.

```
except Myerror, err: # In Python2
except Myerror as err: #In Python 3
```

next() Function and .next() Method

In Python 2, next() as a method of generator object, is allowed. In Python 2, the next() function, to iterate over generator object, is also accepted. In Python 3, however, next(0 as a generator method is discontinued and raises **AttributeError.**

```
gen = (letter for letter in 'Hello World') # creates generator object
next(my_generator) #allowed in Python 2 and Python 3
my_generator.next() #allowed in Python 2. raises AttributeError in Python 3
```

2to3 Utility

Along with Python 3 interpreter, 2to3.py script is usually installed in tools/scripts folder. It reads Python 2.x source code and applies a series of fixers to transform it into a valid Python 3.x code.

```
Here is a sample Python 2 code (area.py):
def area(x,y=3.14):
    a=y*x*x
    print a
    return a
```