

# Lab – Basic Python Programming

## Objectives

**Part 1: Launch VirtualBox and Enter the I2IoT server VM**

**Part 2: Python Basics**

**Part 3: IDLE for Python**

## Background

Python, a programming language, allows for simpler statements. Python is very easy to use, powerful, and versatile. It has become the language of choice for many IoT developers. One of the main reasons for the popularity of Python is the developer community. Python developers have created and made available many specific modules that can be imported into any program to immediately lend added functionality.

## Scenario

In this lab, you will learn and practice some basic Python programming. More specifically, we will use Python version 3 in the lab.

## Required Resources

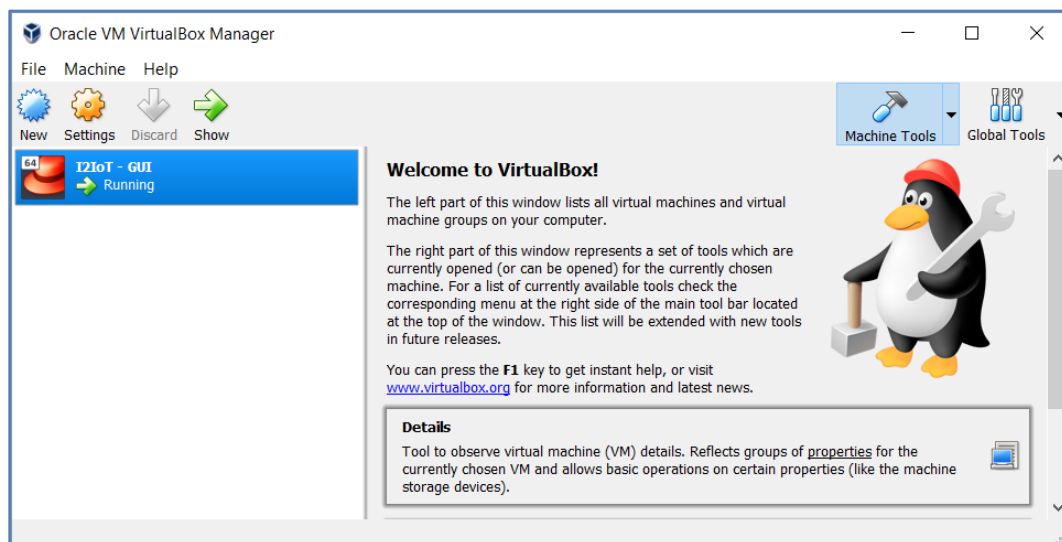
- A modern personal computer with internet access and sufficient RAM.
- VirtualBox with I2IoT server installed.

## Part 1: Launch VirtualBox and Enter the I2IoT server VM

In Part 1, you launch virtualization software VirtualBox and log in to the I2IoT server VM.

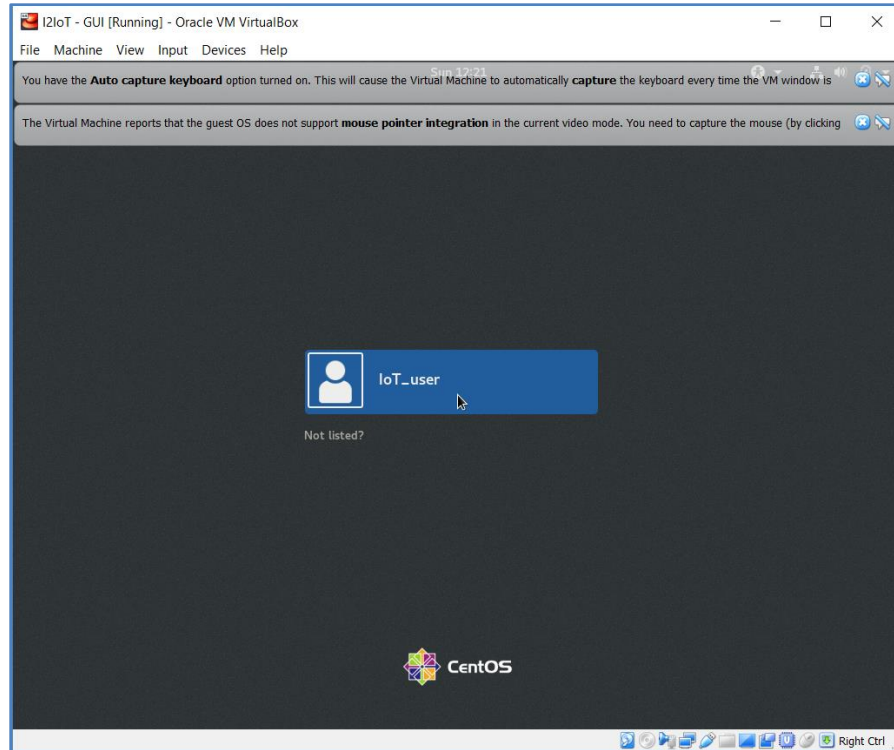
### Step 1: Launch VirtualBox.

- After VirtualBox is installed (see Lab 2.1.2.a), the VirtualBox icon should appear on the Desktop. Click the icon to launch the VirtualBox.

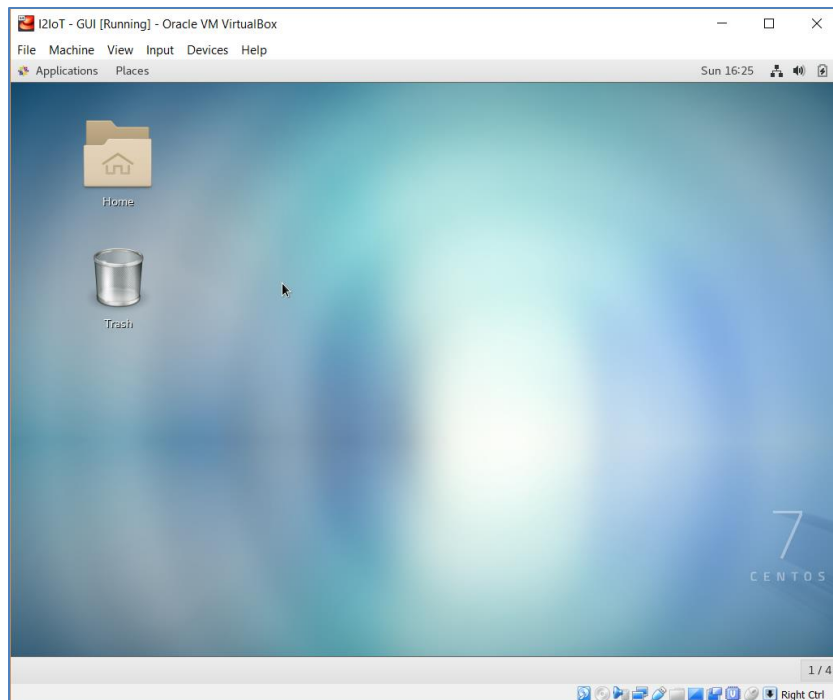


- Click **I2IoT – GUI** on the left pane to launch the server VM.

## Lab6 – Basic Python Programming

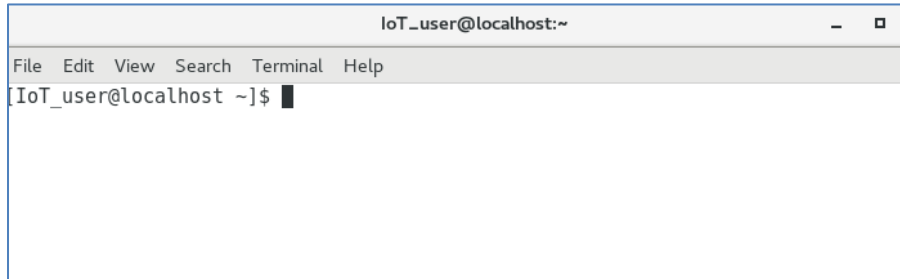


- c. The default username is IoT\_user with no password. Click the blue bar **IoT\_user** in the middle of the screen to login to the VM.



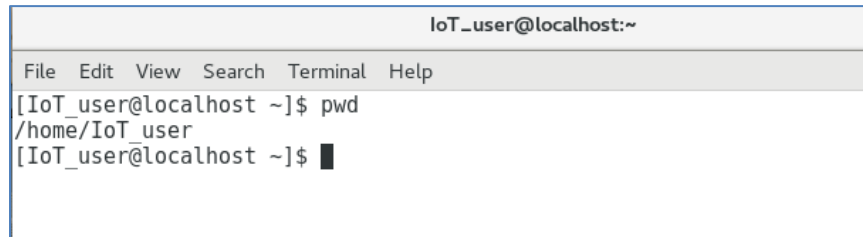
### Step 2: Navigate to the user Document directory

- a. To access the command line interface, click **Application** on the menu bar and choose **Terminal**.



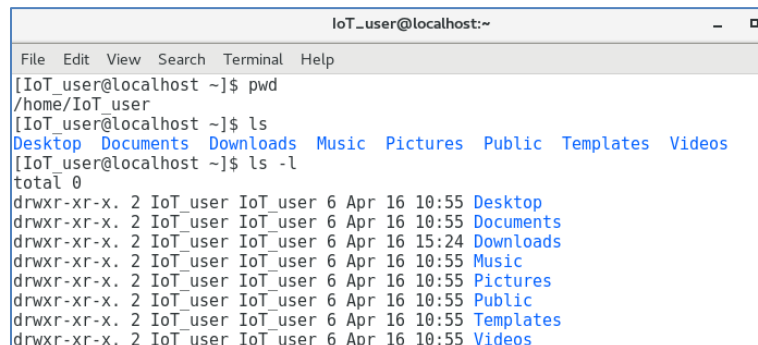
```
IoT_user@localhost:~  
File Edit View Search Terminal Help  
[IoT_user@localhost ~]$
```

- b. Use the **pwd** command to display the current directory.



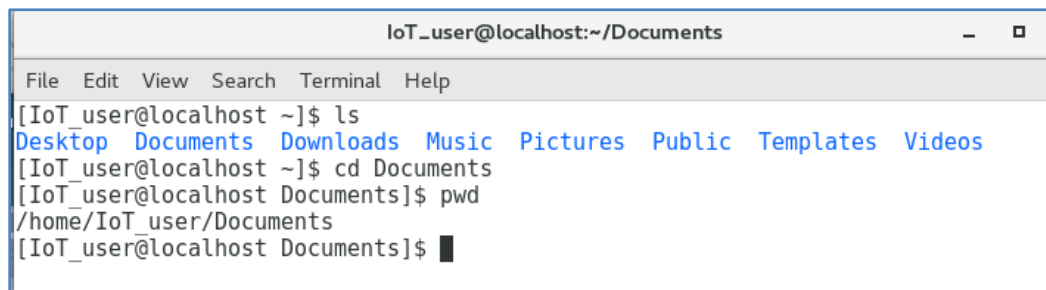
```
IoT_user@localhost:~  
File Edit View Search Terminal Help  
[IoT_user@localhost ~]$ pwd  
/home/IoT_user  
[IoT_user@localhost ~]$
```

- c. Use the **ls** command to show the list of contents in the current directory. Use the **ls** command with the **-l** option to show detailed information about the contents.



```
IoT_user@localhost:~  
File Edit View Search Terminal Help  
[IoT_user@localhost ~]$ pwd  
/home/IoT_user  
[IoT_user@localhost ~]$ ls  
Desktop Documents Downloads Music Pictures Public Templates Videos  
[IoT_user@localhost ~]$ ls -l  
total 0  
drwxr-xr-x. 2 IoT_user IoT_user 6 Apr 16 10:55 Desktop  
drwxr-xr-x. 2 IoT_user IoT_user 6 Apr 16 10:55 Documents  
drwxr-xr-x. 2 IoT_user IoT_user 6 Apr 16 15:24 Downloads  
drwxr-xr-x. 2 IoT_user IoT_user 6 Apr 16 10:55 Music  
drwxr-xr-x. 2 IoT_user IoT_user 6 Apr 16 10:55 Pictures  
drwxr-xr-x. 2 IoT_user IoT_user 6 Apr 16 10:55 Public  
drwxr-xr-x. 2 IoT_user IoT_user 6 Apr 16 10:55 Templates  
drwxr-xr-x. 2 IoT_user IoT_user 6 Apr 16 10:55 Videos
```

- d. Use the **cd Documents** command to change the directory to the /home/IoT\_user/Documents directory. Verify by using the command **pwd**.



```
IoT_user@localhost:~/Documents  
File Edit View Search Terminal Help  
[IoT_user@localhost ~]$ ls  
Desktop Documents Downloads Music Pictures Public Templates Videos  
[IoT_user@localhost ~]$ cd Documents  
[IoT_user@localhost Documents]$ pwd  
/home/IoT_user/Documents  
[IoT_user@localhost Documents]$
```

- e. To check the version of Python installed on the VM, issue the **python3 --version** command.

```
IoT_user@localhost:~/Documents
File Edit View Search Terminal Help
[IoT_user@localhost ~]$ cd Documents/
[IoT_user@localhost Documents]$ python3 --version
Python 3.6.5
[IoT_user@localhost Documents]$
```

## Part 2: Python Basics

In Part 2, you will learn and practice some basic Python programming.

### Step 1: Python programming in the interactive interpreter.

As an interpreted language, Python commands can be issued in an interactive interpreter.

- a. Use the **python3** command to start Python interpreter.

```
File Edit View Search Terminal Help
[IoT_user@localhost ~]$ cd Documents/
[IoT_user@localhost Documents]$ python3 --version
Python 3.6.5
[IoT_user@localhost Documents]$ python3
Python 3.6.5 (default, Apr 16 2018, 15:31:49)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

- b. Perform calculations.

```
>>> 1 + 2
3
>>> 2 * 4
8
>>> 6 / 2
3.0
>>>
```

- c. Print a text string.

```
>>> "How are you?"
'How are you?'
>>>
```

- d. Use the `type()` command to determine the basic data type: int, float, string, Boolean.

```
>>> type(65)
<class 'int'>
>>> type(45.6)
<class 'float'>
>>> type("Hi!")
<class 'str'>
>>> type(True)
<class 'bool'>
>>> 1<2
True
```

```
>>> 1<1
False
>>> 1==1
True
>>> 1>=1
True
>>>
```

- e. Create a variable.

```
>>> x=3
>>> x*5
15
>>> "Good!"*x
'Good!Good!Good! '
>>>
```

- f. Combine multiple strings together and print as one string.

```
>>> str1="Cisco"
>>> str2="Networking"
>>> str3="Academy"
>>> space=" "
>>> print(str1+space+str2+space+str3)
Cisco Networking Academy
>>>
```

- g. Convert data type from a numeric number to a string.

```
>>> x=5
>>> str(x)
'5'
>>> y=4.2
>>> str(y)
'4.2'
>>>
```

- h. Note that integers are not rounded up when converting from float. The decimal is ignored.

```
>>> int(8.21)
8
>>> int(8.99)
8
>>> int(8.21) + int(8.99)
16
>>>
```

- i. Convert an integer to a float.

```
>>> x=5
>>> x
5
>>> float(x)
5.0
>>> type(x)
<class 'int'>
>>> x=float(x)
>>> type(x)
<class 'float'>
```

```
>>> x
5.0
>>>
```

- j. Obtain user input.

```
>>> name=input("What is your name? ")
What is your name? John
>>> print("Hi " + name + ", it is nice to meet you!")
Hi John, it is nice to meet you!
>>>
```

- k. Use **quit()** to exit interactive interpreter.

### Part 3: IDLE for Python

IDLE stands for Integrated Development and Learning Environment. It is supported and included in the Python package. A few key features of IDLE for Python include these:

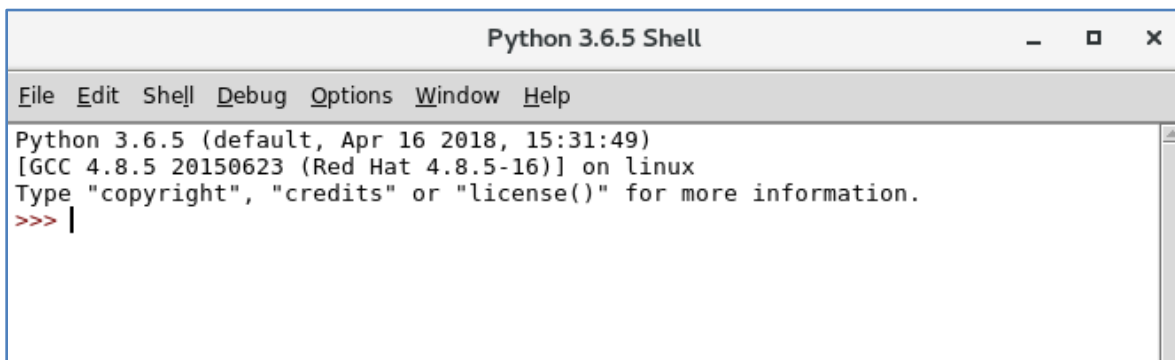
- A Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- A multiwindow text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- The ability to search within any window, replace within editor windows, and search through multiple files (grep)
- A debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- Configuration, browsers, and other dialogs

In Part 3, you will launch IDLE and create a simple script.

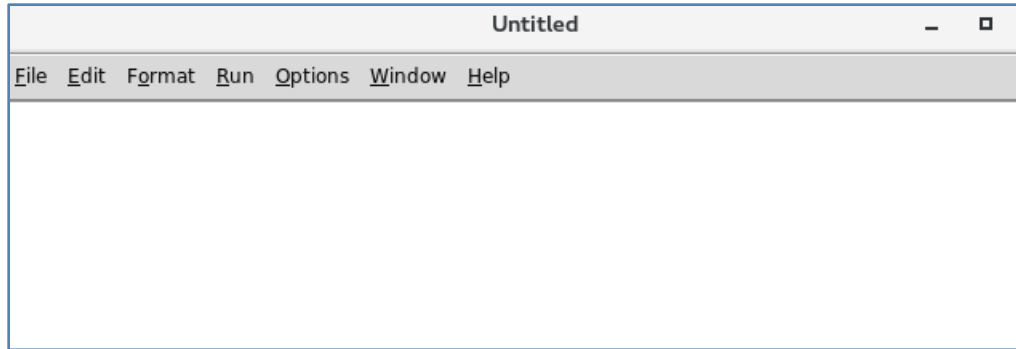
#### Step 1: Launch IDLE.

- a. Use the **idle3** command to launch IDLE. By default, it starts in Python Shell, or interactive interpreter, window. You are already familiar with the interactive interpreter.

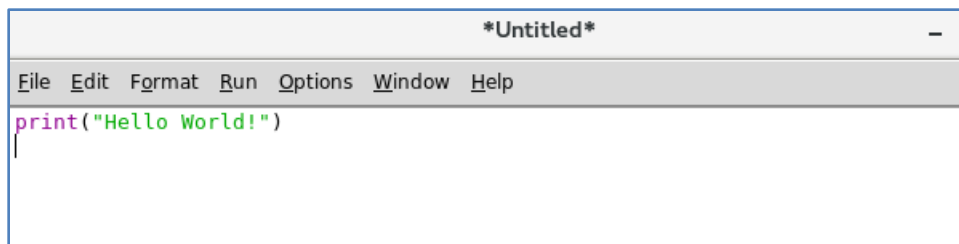
```
[IoT_user@stueverj-vm2 Documents]$ idle3
```



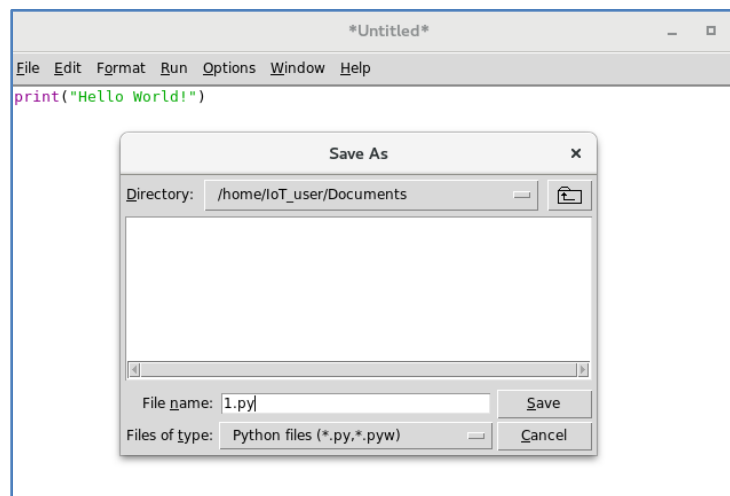
- b. Click **File -> New File** to open a new (untitled) Python script.



- c. Type the code in the script `print("Hello World!")`, note the codes are colored, and open and close parentheses are matched.



- d. Click **File** -> **Save**, save the current script as 1.py in the current directory. Click **Save** button.



- e. Click **Run** -> **Run Module** (or press F5). The shell window will display the result.

