

Python Fundamentals

Lab 2



Learning outcomes:

At the end of this Lab 2, you should be able to:

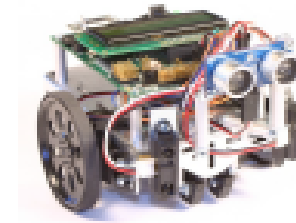
- **understand Python basics**
- **why Python**
- **features**
- **use editors**
- **complete examples**

Python features:

- Python is a general purpose interpreted high-level programming language.
- Current version is 3.4.3 available for UNIX, Mac and Windows
- It is case sensitive (print and Print are different)
- **Python is Interpreted:** Processed at runtime by the interpreter (similar to PERL and PHP).
- **Python is Interactive:** You can write your programs at command prompt.
- **Easy-to-learn:** It has relatively few keywords, simple structure, and a clearly defined syntax.
- Interpreter & IDE are free.

Who uses Python?

- On-line games
- Web services
- Applications
- Science
- Instrument control
- Embedded systems
- Google extensively uses Python



Running Python

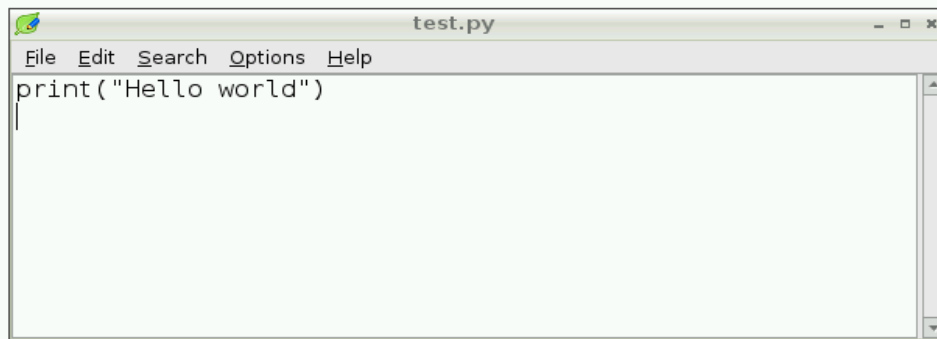
There are three different ways to start Python:

1. **Interactive Interpreter:** You can enter python and start coding right away in the interactive interpreter by starting it from the command line. Command-line interpreter or shell window. Launch LX terminal then **launch a shell, type**
pi@raspberrypi ~ \$ python
>>> print ("Hello, world!")
Hello, world!
>>>
2. **Save as files (using text editor Leafpad)**
3. **Integrated Development Environment (IDLE) - IDLE:** You can run Python code from a graphical user interface (GUI) environment. All you need is a GUI application on your system that supports Python.

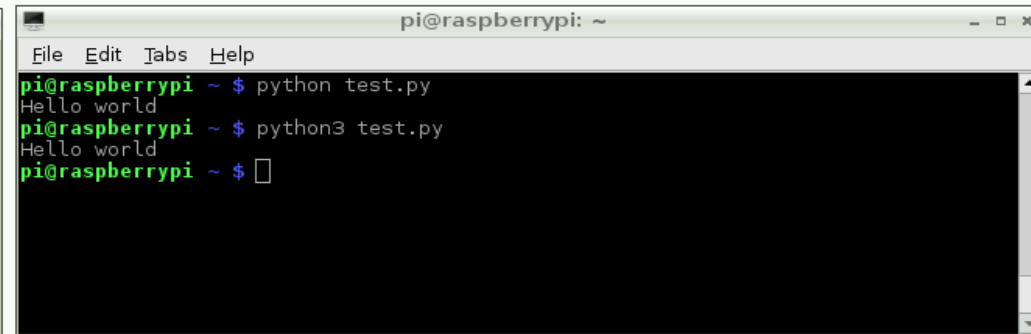
2. Executing python script using a command line

You can write a Python file in a standard editor like Vim, Nano or LeafPad, and run it as a Python script from the command line.

Just navigate to the directory the file is saved (use `cd` and `ls` for guidance) and run with `python`, e.g. `python hello.py`.



```
test.py
File Edit Search Options Help
print("Hello world")
```



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi ~ $ python test.py
Hello world
pi@raspberrypi ~ $ python3 test.py
Hello world
pi@raspberrypi ~ $
```

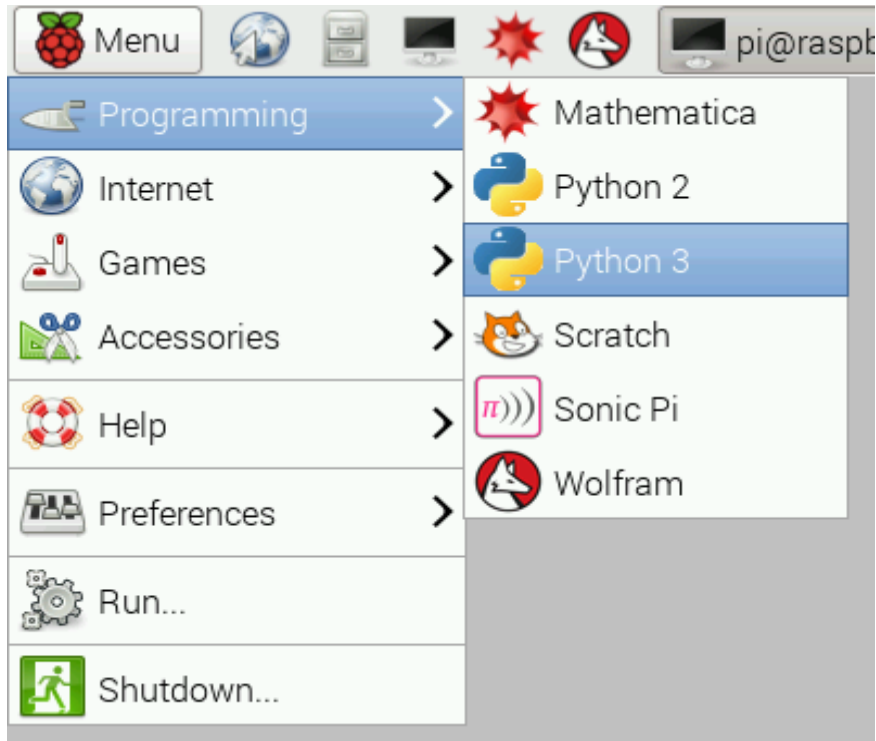
3. IDLE Development Environment

- Shell for interactive evaluation.
- Text editor with color-coding and smart indenting for creating python files.
- Menu commands for changing system settings & running files.
- <https://www.python.org/>



IDLE

- A Python development environment.



Lines and Indentation

- One of the most prominent features of Python is the fact that there are no braces to **indicate blocks of code** for class and function definitions or flow control. Blocks of code are denoted by line indentation.
- The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. Example:

```
if True:
    print "True"
else:
    print "False"
```



Lines:

Quotations in Python: Python accepts single ('), double (") and triple (''' or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes can be used to span the string across multiple lines.

For example, all the following are legal:

```
word = 'word'
```

```
sentence = "This is a sentence."
```

```
paragraph = """This is a paragraph. It is made up of multiple  
lines and sentences."""
```

Whitespace

Whitespace is meaningful in Python: especially indentation and placement of newlines

- Use a newline to end a line of code

Use `\` when must go to next line prematurely

- No braces `{ }` to mark blocks of code, use *consistent* indentation instead
 - First line with *less* indentation is outside of the block
 - First line with *more* indentation starts a nested block
- Colons start of a new block in many constructs, e.g. function definitions, then clauses



Comments:

- A hash sign (#) that is not inside a string literal begins a comment.
- All characters after the # and up to the physical line end are part of the comment, and the Python interpreter ignores them.

First comment

print("Hello, Python!") # second comment

Basic Datatypes

- Integers (default for numbers)
`z = 5 / 2` # Answer 2, integer division
- Floats
`x = 3.456`
- Strings
 - Can use `""` or `' '` to specify with `"abc" == 'abc'`
 - Unmatched can occur within the string: `"matt's"`
 - Use triple double-quotes for multi-line strings or strings than contain both `'` and `"` inside of them:
`"""a'b"c"""`

Variables and Data Types

- Python variables do not have to be explicitly declared to reserve memory space.
- The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

For example:

```
counter = 100          # An integer assignment
miles   = 1000.0        # A floating point
name    = "John"        # A string
```

First Program:

// C Program

```
# include <stdio.h>
void main(void)
{
printf("Hello world!");
}
```

//Java program

```
public class abc(){
public static void main(String args[])
{
    System.out.println("Hello world!");
}
}
```

Python program

```
print ("Hello world!")
```

Example 1: command prompt

```
>>> a=10
>>> b=20
>>> c = a+ b
>>> print(c)
30
>>>
```

```
>>>print ("My name is %s and weight is %d kg!" % ('William', 21))
```

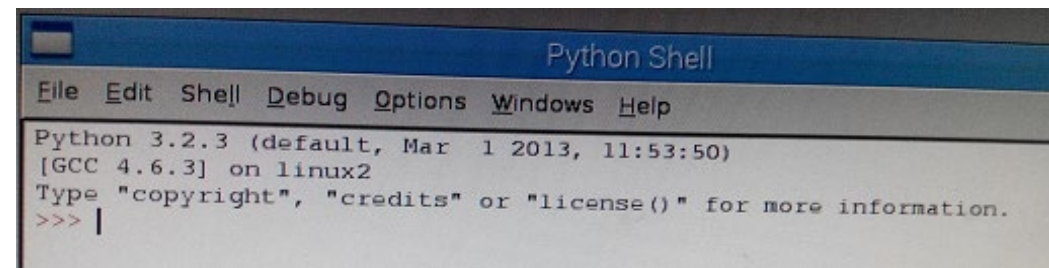
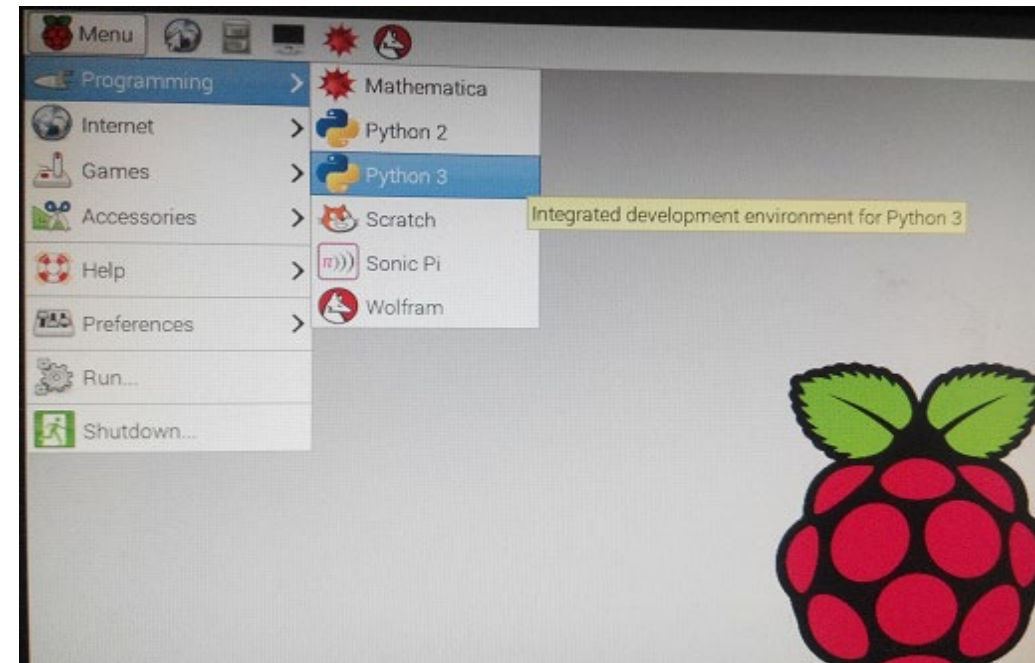
This will produce the following result:

My name is William and weight is 21 kg!

```
>>>>
```

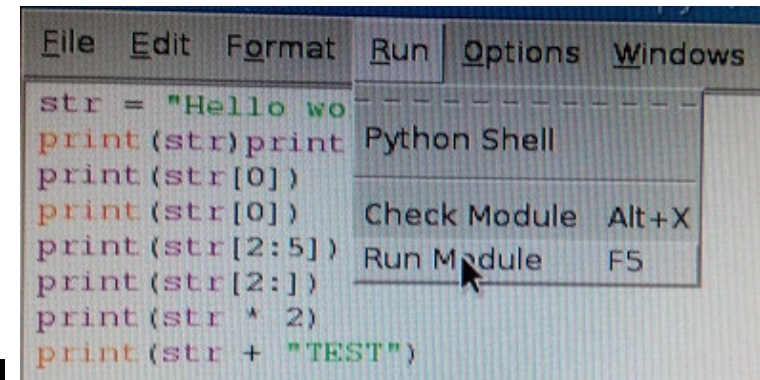

Example 2: Using IDLE 3

- Launch **Python 3**
- Go to **File > New Window**
- Type out the code for this example (next slide)
- **File > Save As**
- And then type the file name and .py (example **abc.py**) and click **Save**
- By default file is stored in the /home/pi

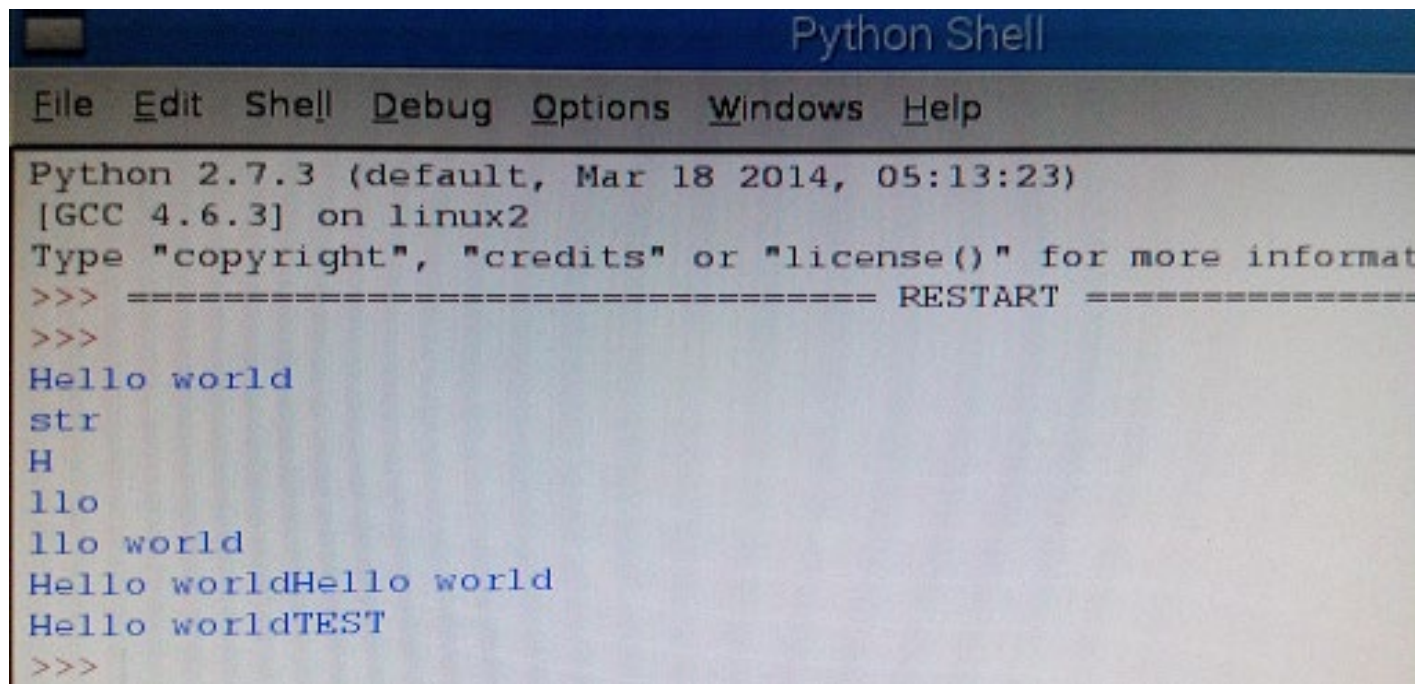


Example 2: Using IDLE 3 continued

- Click Run > Run Module



- The output is displayed in the new shell like

A screenshot of the Python Shell window. The title bar says 'Python Shell'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Windows', and 'Help'. The shell displays the following text:

```
Python 2.7.3 (default, Mar 18 2014, 05:13:23)
[GCC 4.6.3] on linux2
Type "copyright", "credits" or "license()" for more informat
>>> ===== RESTART =====
>>>
Hello world
str
H
llo
llo world
Hello worldHello world
Hello worldTEST
>>>
```

Example 3: Output

```
str = 'Hello World!'
print (str)          # Prints complete string
print('str')
print (str[0])       # Prints first character of the string
print (str[2:5])     # Prints characters starting from 3rd to 6th
print (str[2:])      # Prints string starting from 3rd character
print (str * 2)      # Prints string two times
print (str + "TEST") # Prints concatenated string
```

The above code will produce following output:

```
Hello World!
Str
H
llo
llo World!
Hello World!Hello World!
Hello World!TEST
```

Example 4: Input

```
age = input("How old are you? ")  
print ("Your age is", age)  
print ("You have", (62 - int(age)), "years until retirement")
```

The above code will produce following output:

```
How old are you? 20  
Your age is 20  
You have 42 years until retirement  
>>>
```

Example 5: Variables

```
#Variables demonstrated  
print ("This program is a demo of variables.")  
v = 1  
print("The value of v is now", v)  
v = v + 1  
print("v now equals itself plus one, making it worth", v)
```

Example 6: Calculate the square root

```
# Python Program to calculate the square root
# change this value for a different result

num = 8

# uncomment to take the input from the user
#num = float(input('Enter a number: '))

num_sqrt = num ** 0.5
print('The square root of %0.3f is %0.3f'%(num ,num_sqrt))
```

The square root of 8.000 is 2.828

Example 7: strings

#Giving variables text, and adding text.

```
word1 = "Good"
```

```
word2 = "morning"
```

```
word3 = "to you too!"
```

```
print(word1, word2)
```

```
sentence = word1 + " " + word2 + " " + word3
```

```
print(sentence)
```

Example 8: if-else

```
grade = 60
if grade >= 70:
    print ("Congratulations!")
else:
    print ("You could do so much better.")
    print ("Try harder")
```

The above code will produce following output:

You could do so much better.
Try harder.

Example 9: elif

```
z = 4
if z > 70:
    print("Something is very wrong")
elif z < 7:
    print("This is normal")
else:
    print("Everything is wrong")
```

The above code will produce following output:

Example 10: Find the largest number

```
# change the values of num1, num2 and num3
# for a different result
num1 = 10
num2 = 14
num3 = 12
# uncomment following lines to take three numbers from user
#num1 = float(input("Enter first number: "))
#num2 = float(input("Enter second number: "))
#num3 = float(input("Enter third number: "))
if (num1 > num2) and (num1 > num3):
    largest = num1
elif (num2 > num1) and (num2 > num3):
    largest = num2
else:
    largest = num3
print("The largest number between",num1,",",num2,"and",num3,"is",largest)
```

The largest number between 10 , 14 and 12 is 14

Example 11: for

```
for n in range(1, 4):  
    print ("This is the number", n)  
fruits = ['banana', 'apple', 'mango']  
for fruit in fruits:      # Second Example  
    print ('Current fruit :', fruit)
```

The above code will produce following output:

```
This is the number 1  
This is the number 2  
This is the number 3  
Current fruit : banana  
Current fruit : apple  
Current fruit : mango  
>>>
```

Example 12: for

Enter the following at the command prompt:

```
>>> a = [1, 2, 3, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1]
>>> b = [' ' * 2 * (7 - i) + 'very' * i for i in a]
>>> for line in b: print (line)
```

The expected output is:

```
      very
    veryvery
  veryveryvery
veryveryveryvery
veryveryveryveryvery
veryveryveryveryveryvery
veryveryveryveryveryveryvery
veryveryveryveryveryvery
  veryveryveryvery
    veryveryvery
      veryvery
        very
```

Example 13: for Loop with list

Enter the following at the command prompt:

```
num_list = [2, 5, 9, 1, 4]
total = 0
for num in num_list:
    total += num
print ("Total: ", total)
```

The expected output is:

```
'Total: 21'
```

Example 14: while

```
count = 0
while (count < 9):
    print ('The count is:', count)
    count = count + 1
print ("Good bye!")
```

The above code will produce following output:

```
The count is: 0
The count is: 1
The count is: 2
The count is: 3
The count is: 4
The count is: 5
The count is: 6
The count is: 7
The count is: 8
Good bye!
>>>
```

Example 15: while

Type this in and see what it does.

```
x = 10
while x != 0:
    print(x)
    x = x - 1
    print("Wow, we've counted x down, and now it equals", x)
print ("And now the loop has ended.")
```

Example 16: Guessing Game

```
import random
n = random.randint(1, 99)
guess = int(raw_input("Enter an integer from 1 to 99: "))
while n != "guess":
    print
    if guess < n:
        print "guess is low"
        guess = int(raw_input("Enter an integer from 1 to 99: "))
    elif guess > n:
        print "guess is high"
        guess = int(raw_input("Enter an integer from 1 to 99: "))
    else:
        print "you guessed it!"
        break
    print
```


Example 17: guessing number game

```
# This is a guessing number game.
import random

guessesTaken = 0

print('Hello! What is your name?')
myName = raw_input()

number = random.randint(1, 20)
print('Well, ' + myName + ', I am thinking of a number between 1 and 20.')

while guessesTaken < 6:
    print('Take a guess.')
    guess = input()
    guess = int(guess)
```

.....continued to the next page

Example 17: continued

```
guessesTaken = guessesTaken + 1
```

```
if guess < number:
```

```
    print('Your guess is too low.')
```

```
if guess > number:
```

```
    print('Your guess is too high.')
```

```
if guess == number:
```

```
    break
```

```
if guess == number:
```

```
    guessesTaken = str(guessesTaken)
```

```
    print('Good job, ' + myName + '! You guessed my number in ' +  
        guessesTaken + ' guesses!')
```

```
if guess != number:
```

```
    number = str(number)
```

```
    print('Nope. The number I was thinking of was ' + number)
```