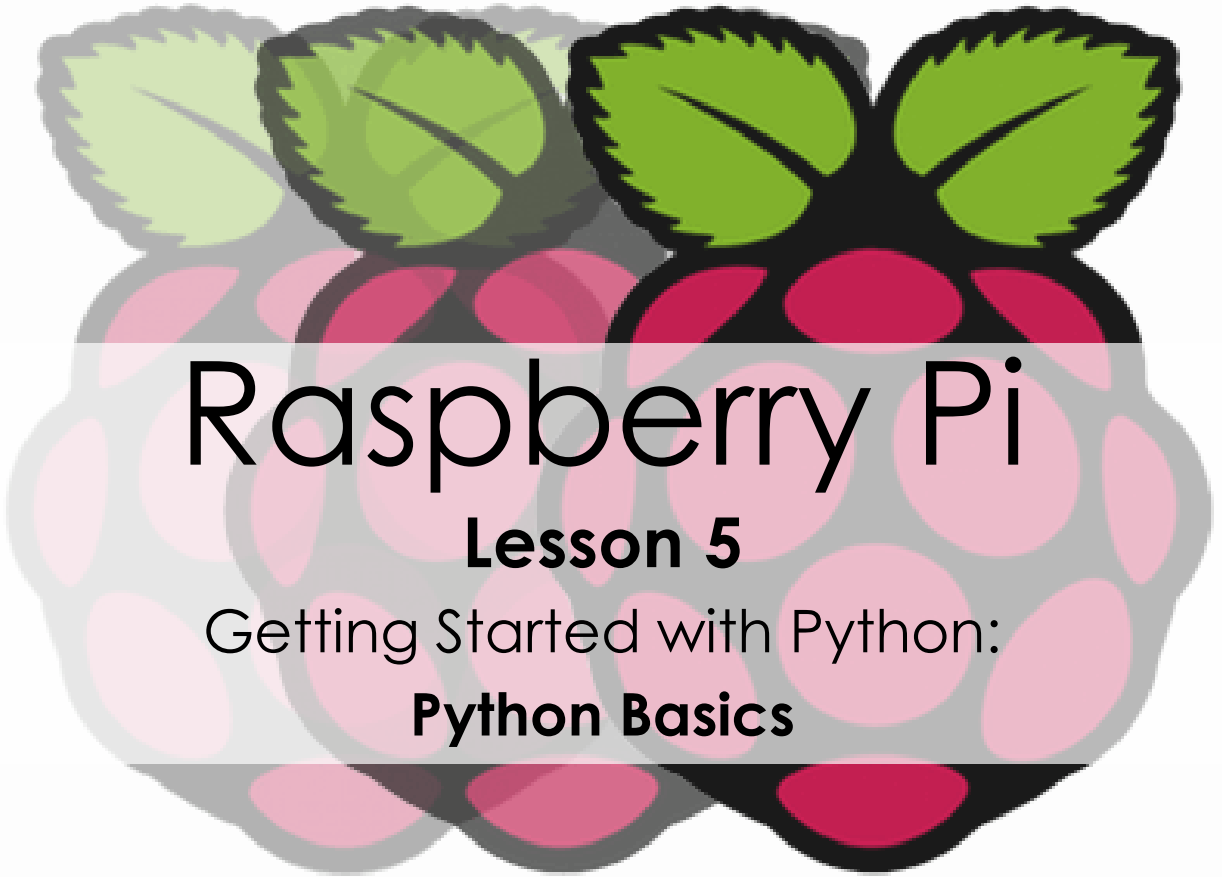


Raspberry Pi

Lesson 5

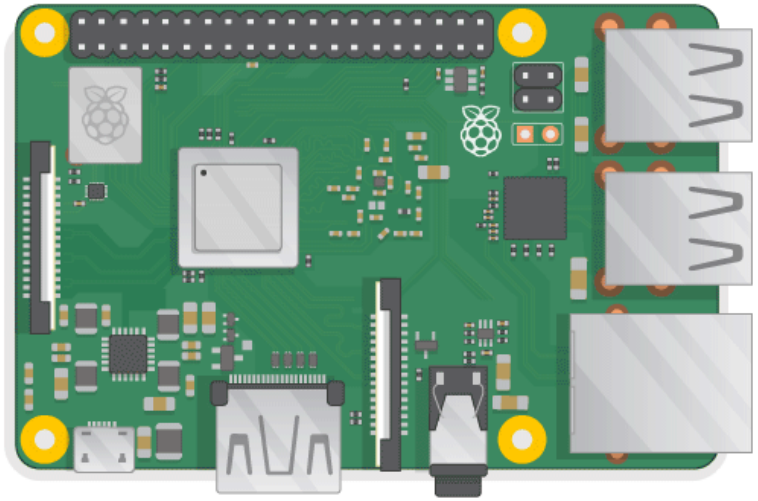
Getting Started with Python:

Python Basics

MAKERHOUSE
EMPOWERING MAKERS

Contents

Required Parts

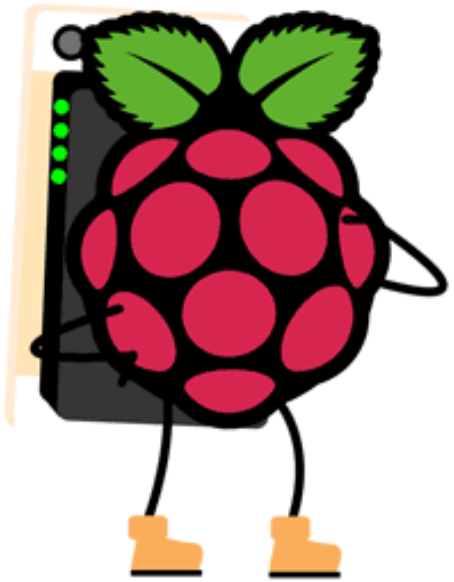


Raspberry Pi Computer

Others: Your Creativity

Introduction

Why Python?



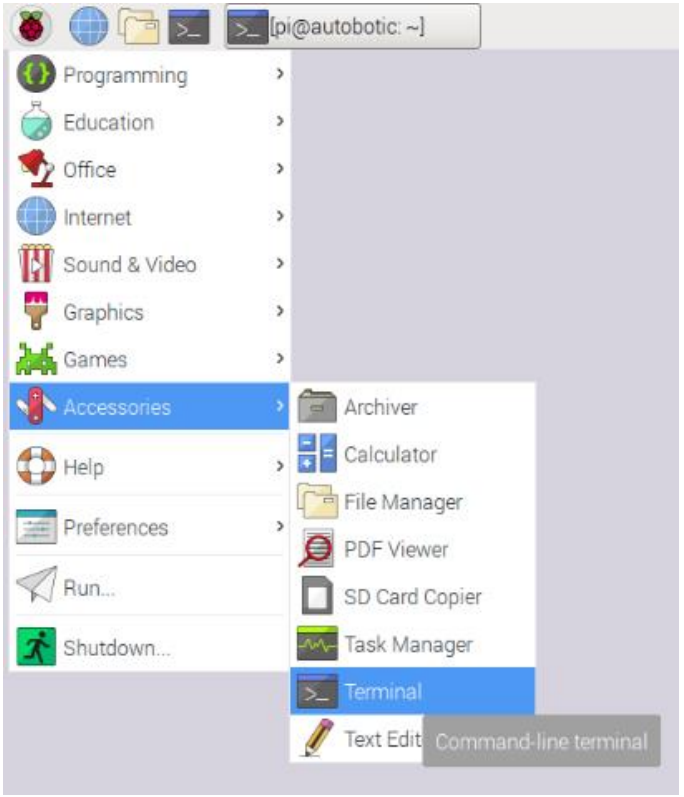
Did you know?

Fact: Pi in Raspberry Pi is inspired by the word Python

MAKERHOUSE
EMPOWERING MAKERS

Getting Started

Open Terminal

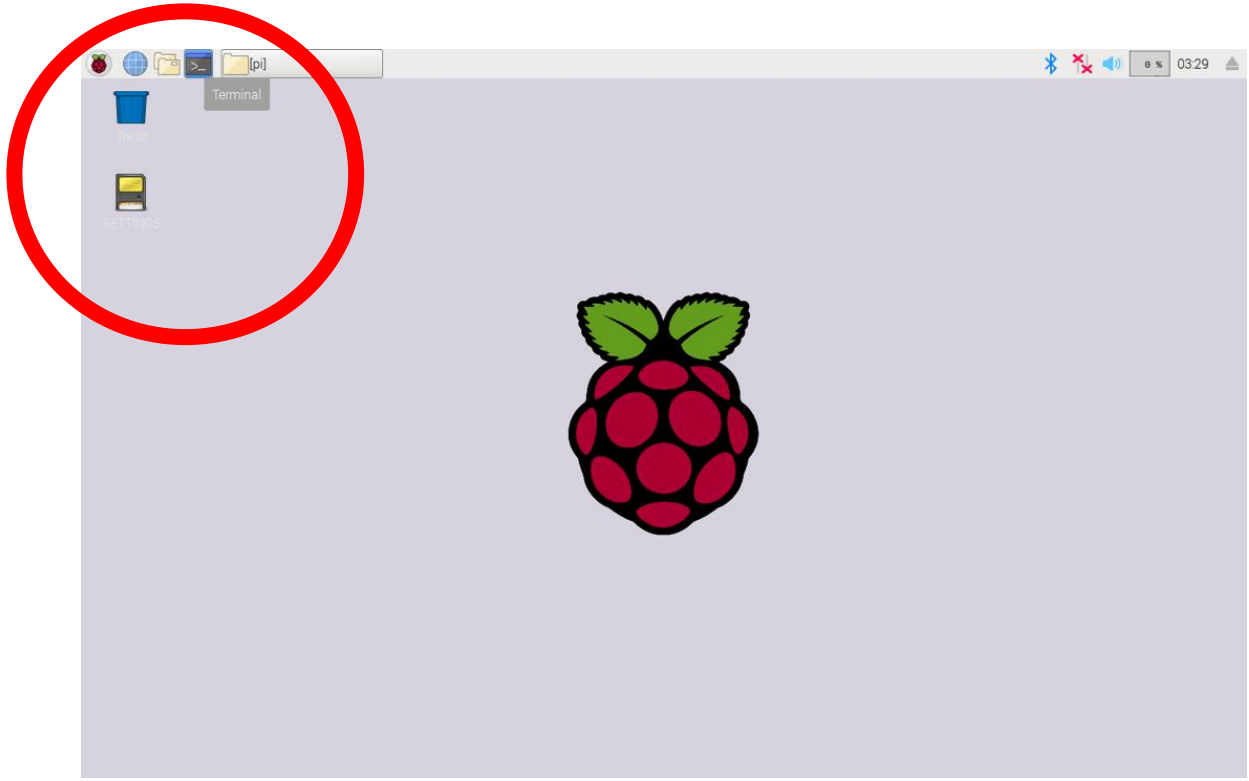


Using GUI

Main menu > Accessories > Terminal

Getting Started

Open Terminal

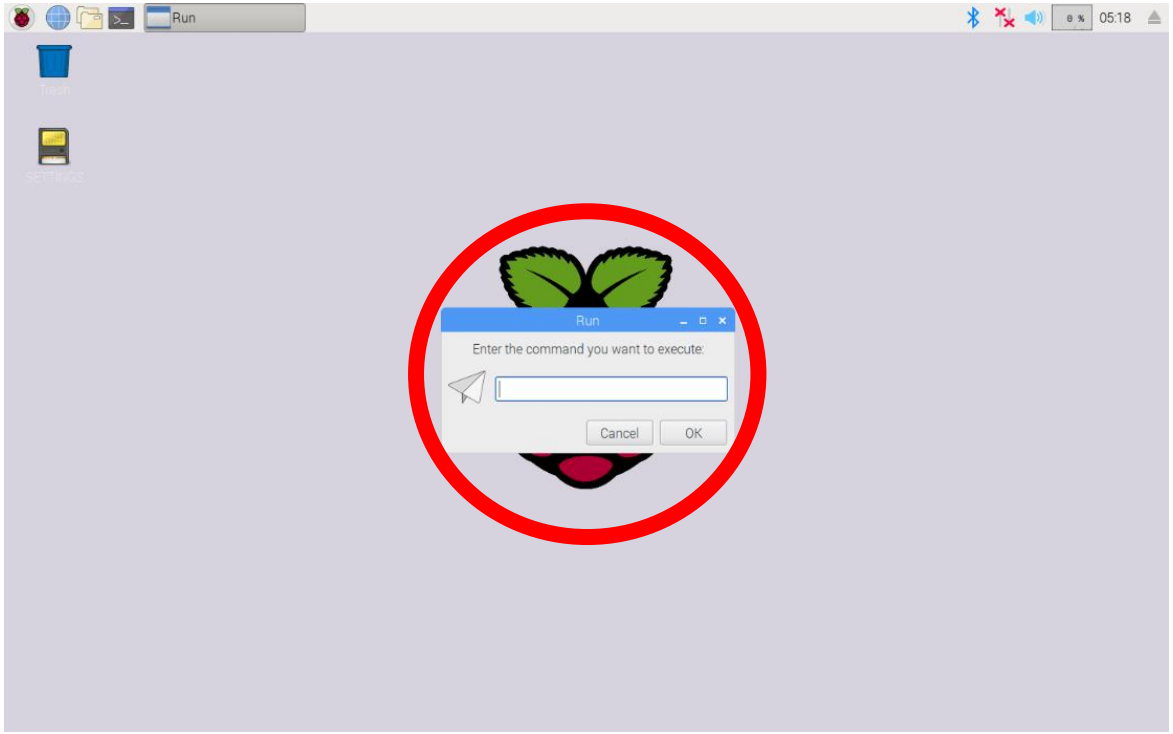


Using GUI

Shortcuts Icon

Getting Started

Open Terminal



Using GUI

Alt + F7 > ltxterminal

MAKERHOUSE
EMPOWERING MAKERS

Python Basics

Which Python?

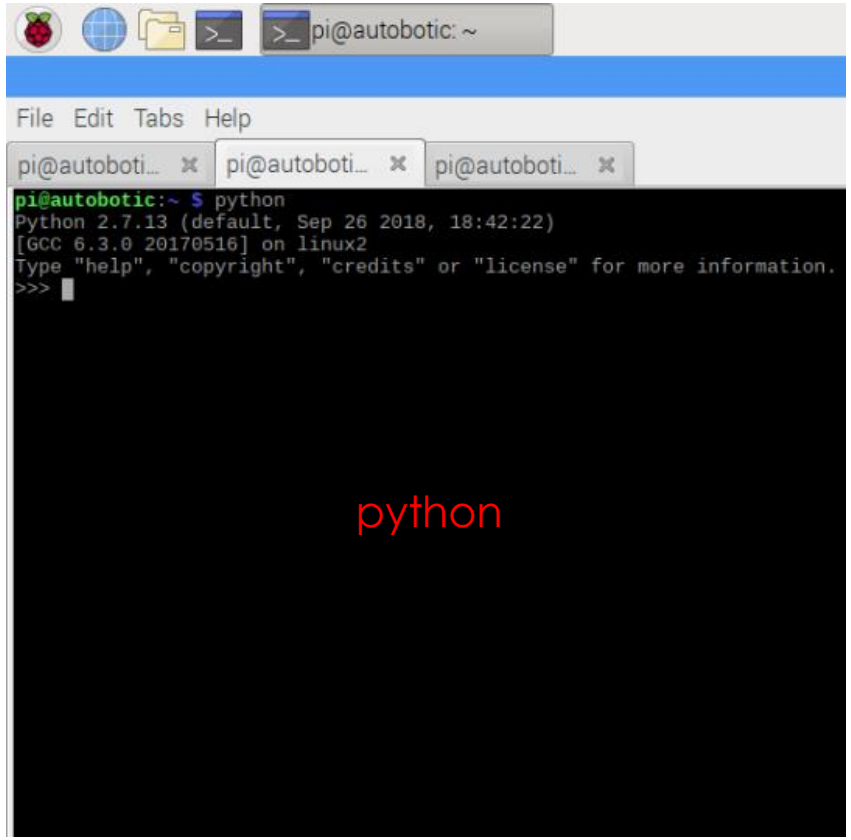
Python 2.x is legacy,
Python 3.x is the present and future of the language

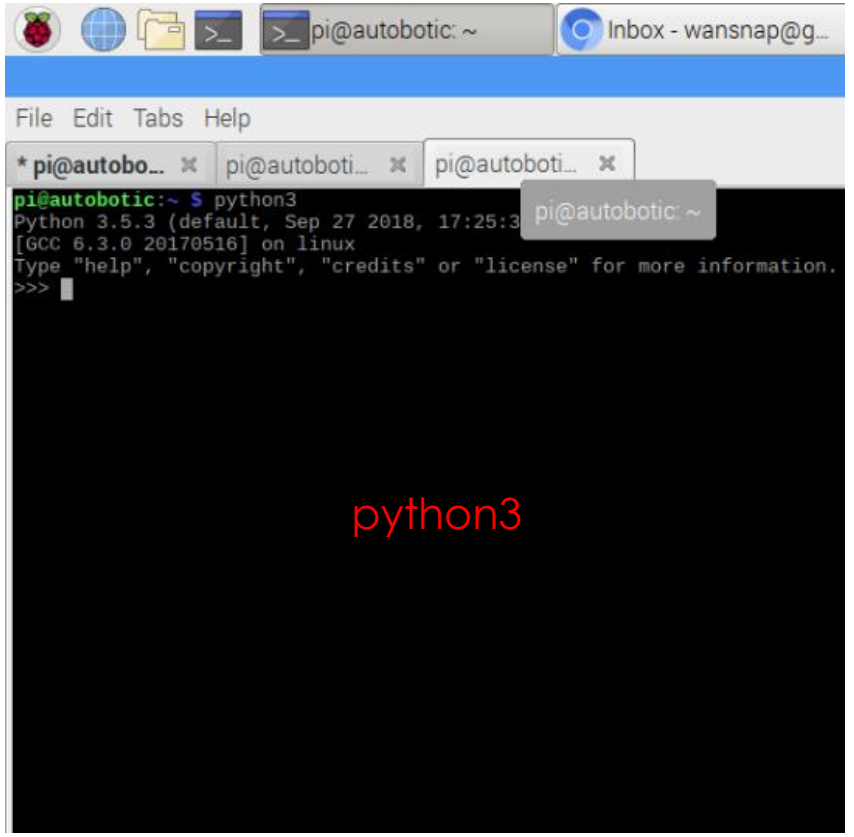
Use Python 3 until you face a problem that is best solved by reverting to version 2

MAKERHOUSE
EMPOWERING MAKERS

Python Basics

Python Console





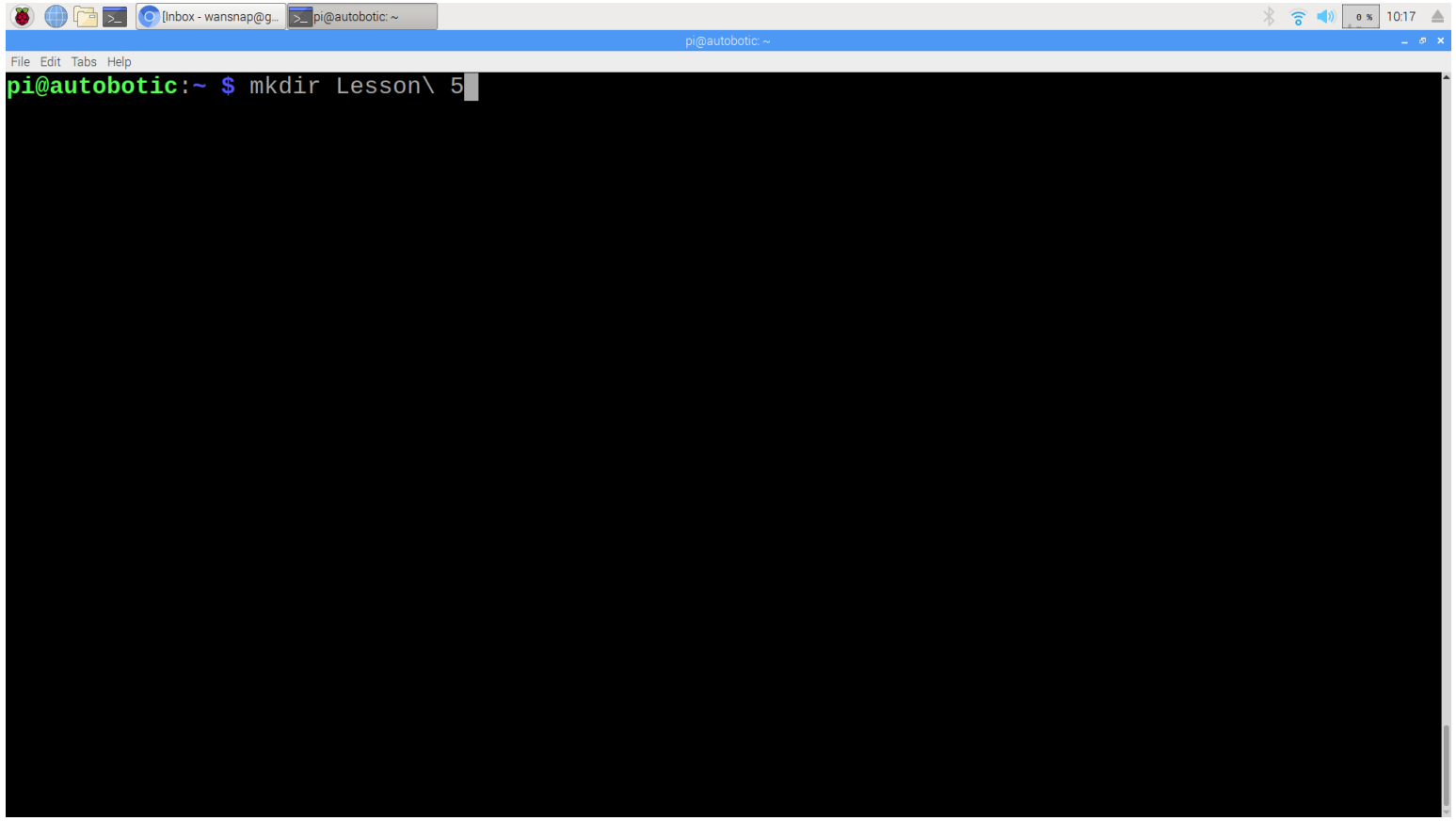
Python 2.x VS Python 3.x

Python Console – handy, used to write a short Python command -- troubleshooting

MAKERHOUSE
EMPOWERING MAKERS

Python Basics

Create a Folder and File



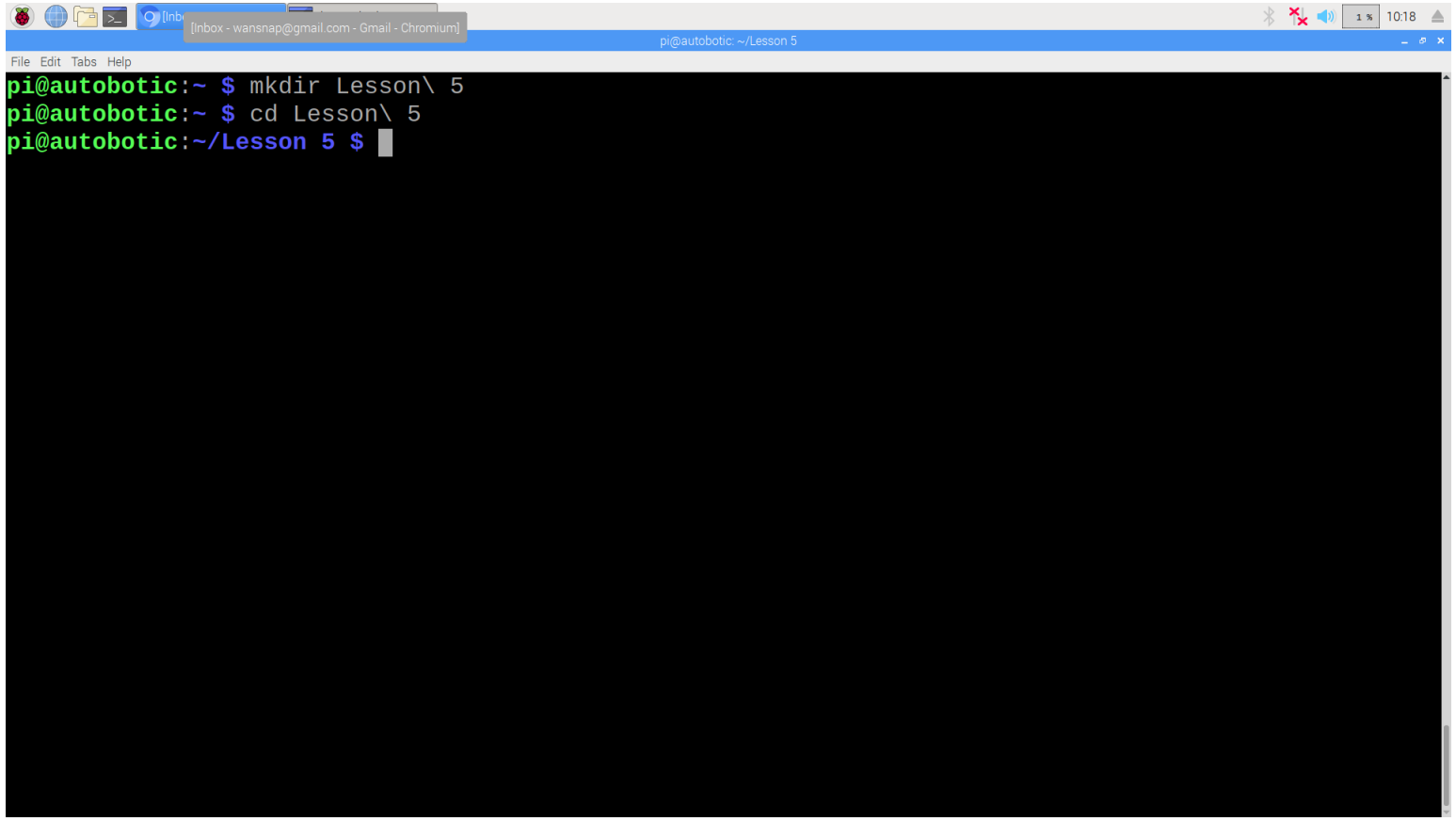
mkdir [File...] → mkdir Lesson\ 5

Hint: Remember last lesson

MAKERHOUSE
EMPOWERING MAKERS

Python Basics

Create a Folder and File



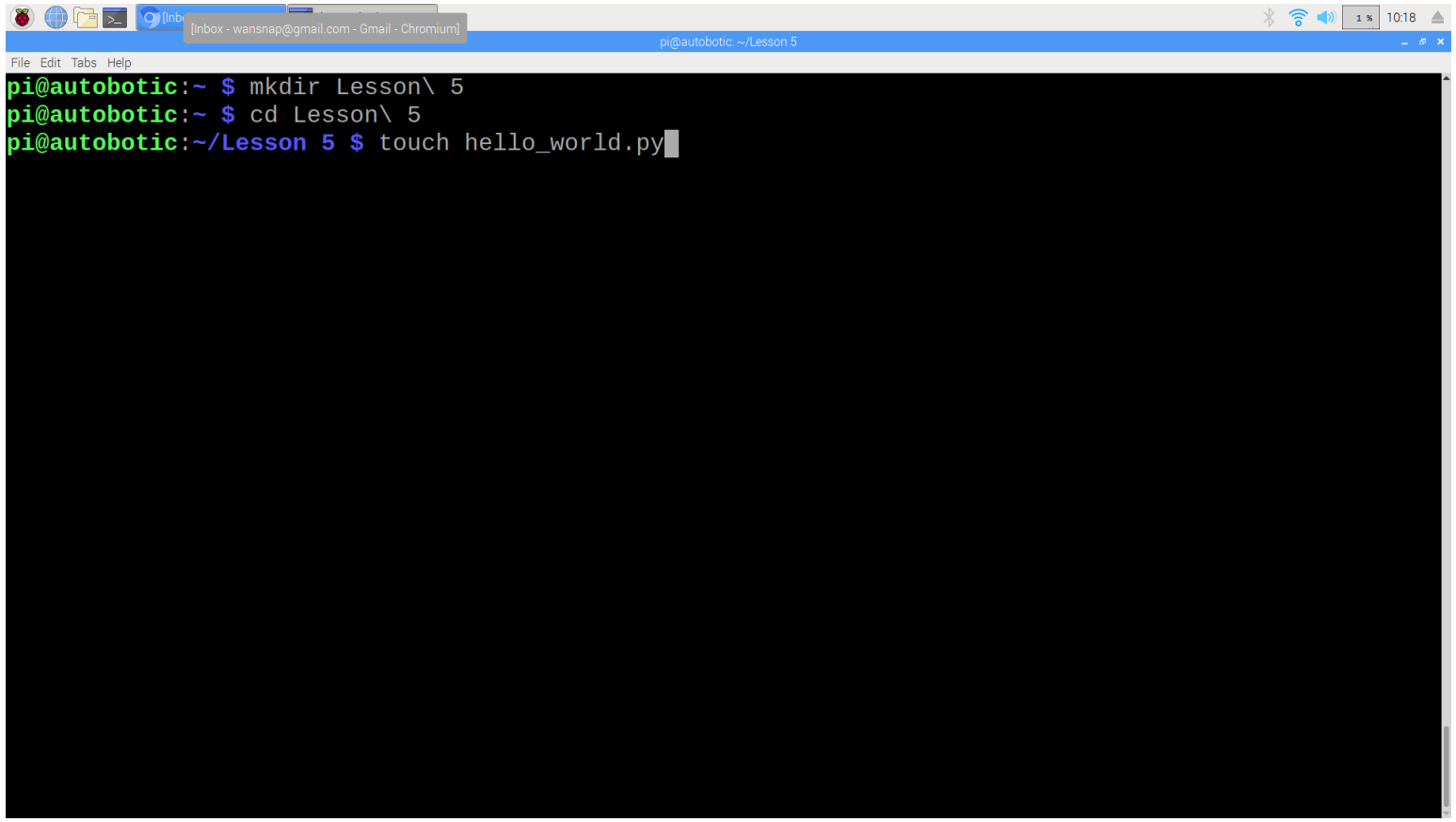
cd [File...] → cd Lesson\ 5

Hint: Remember last lesson

MAKERHOUSE
EMPOWERING MAKERS

Python Basics

Create a Folder and File

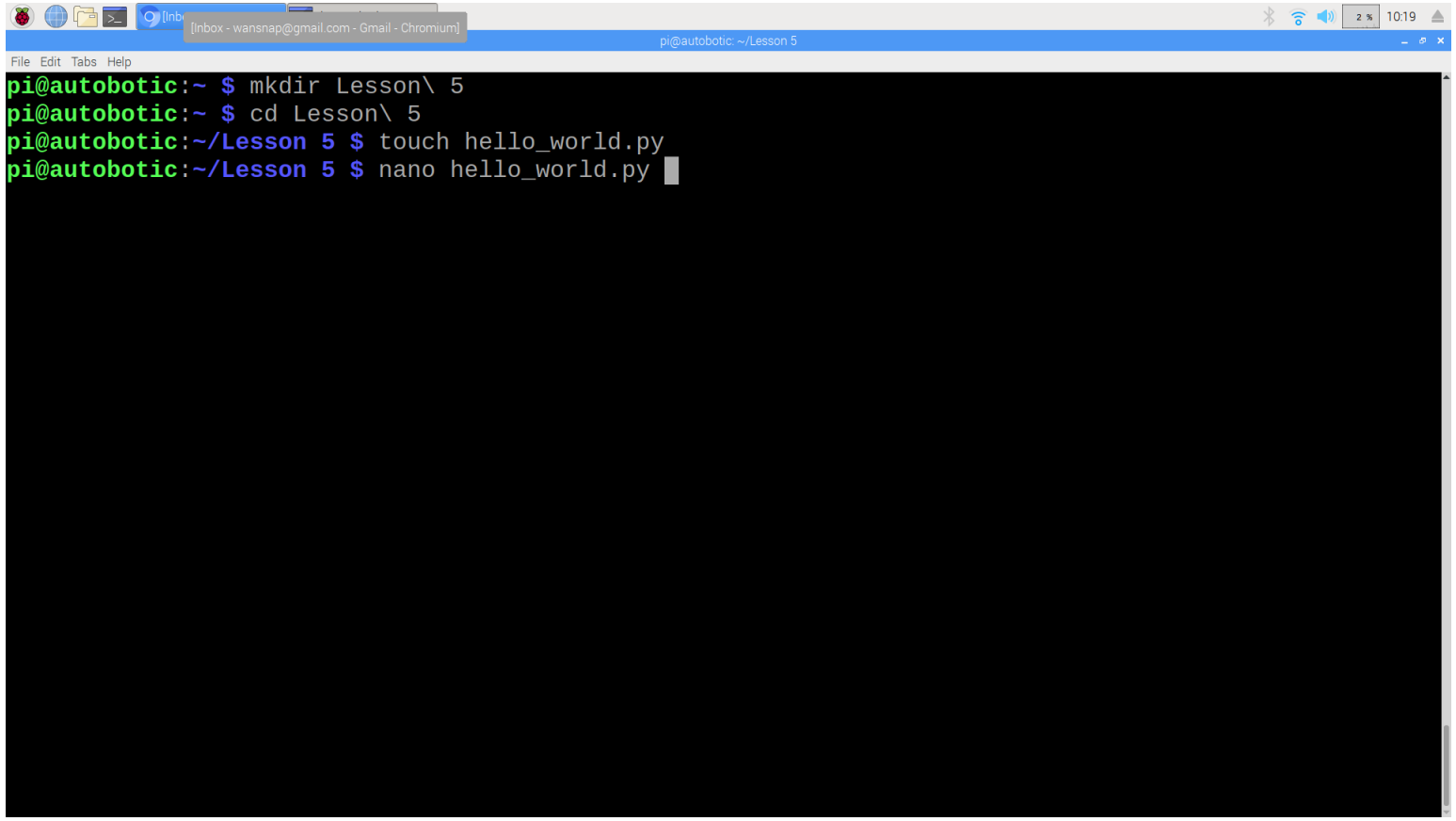


touch [File...] → touch hello_world.py

New: Create an empty file

Python Basics

Create a Folder and File

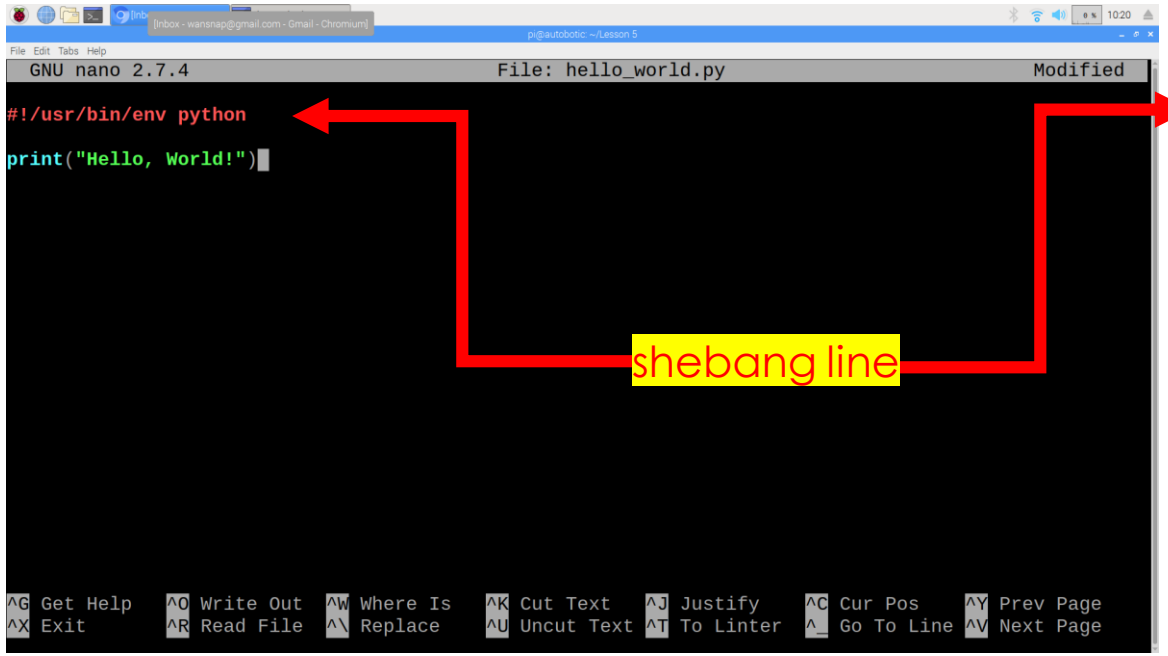


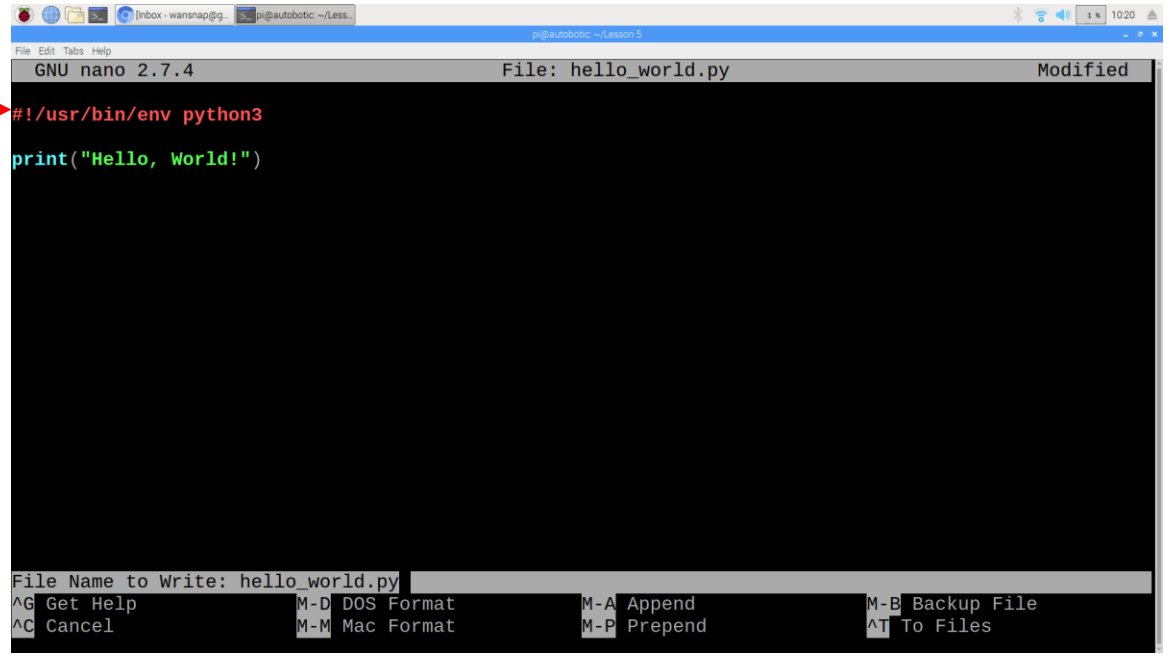
nano [File...] → nano hello_world.py

Hint: Remember last lesson

Python Basics

Create a Folder and File



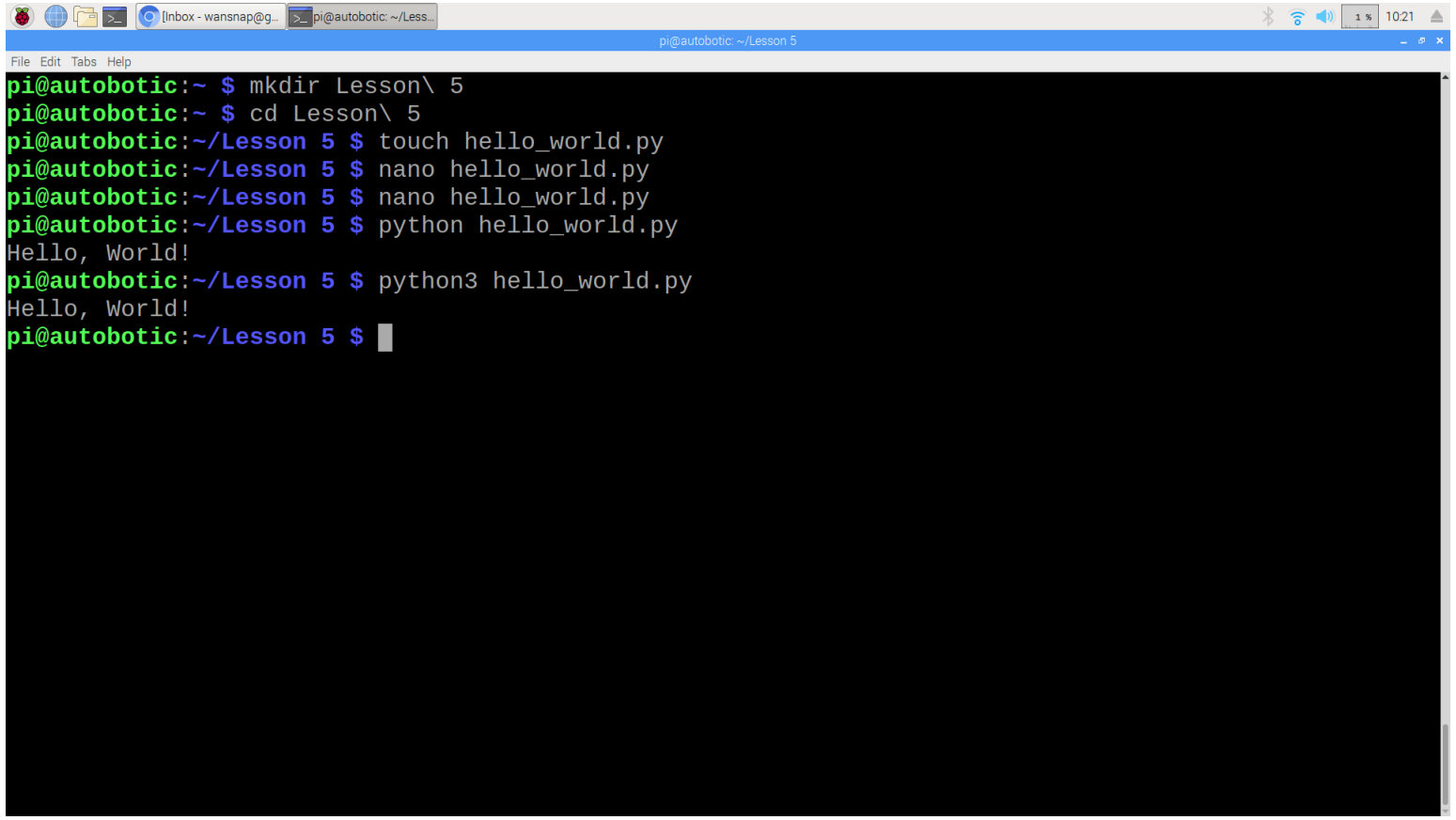


ctrl + x > y > Enter (Return)

Hint: Save the File

Python Basics

Execute Python Scripts



python [File...] or python3 [File...]

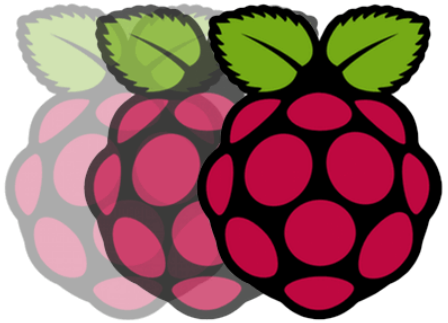
Hint: Save the File

MAKERHOUSE
EMPOWERING MAKERS

Python Basics

Variables

a	=	123
b	=	12.34
c	=	"Hello"
d	=	'Hello'
e	=	True

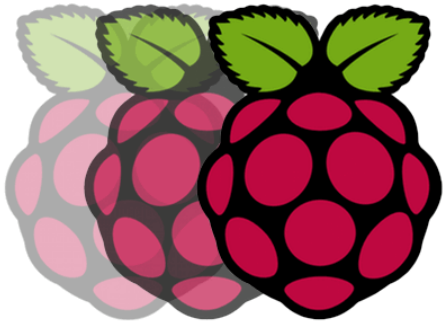


Python Basics

Displaying Output

```
#!/usr/bin/env python3

print("Hello, World!")
```

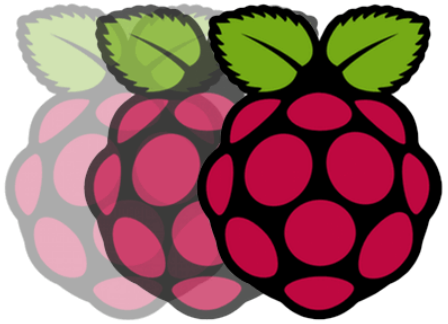

MAKERHOUSE
EMPOWERING MAKERS



Python Basics

Reading User Input

```
#!/usr/bin/env python3

x = input("Enter Value: ")

print(x)
```



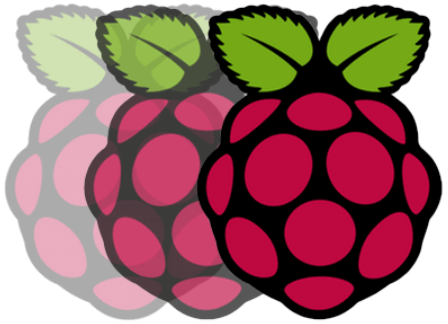
Python Basics

Arithmetic

```
#!/usr/bin/env python3

tempC = input("Enter temp in C: ")

tempF = (int(tempC) * 9) / 5 + 32

print(tempF)
```

addition (+) , subtraction (-), multiplication (*) , division (/), modulus (%), power (**)

MAKERHOUSE
EMPOWERING MAKERS



Python Basics

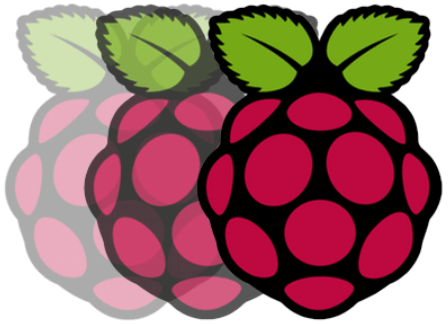
Strings

```
#!/usr/bin/env python3

s = "abc def“

print(s)
```

Create a string

Hint: double/single quotes marks

MAKERHOUSE
EMPOWERING MAKERS



Python Basics

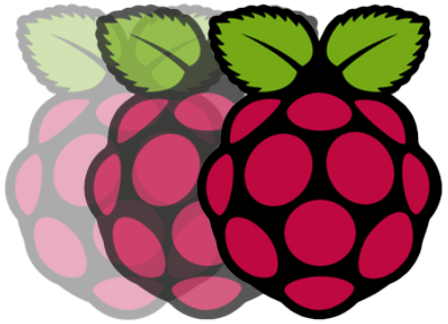
Strings

```
#!/usr/bin/env python3

s = "abc" + "def"

print(s)
```

Join a strings

Hint: Simple "+" addition

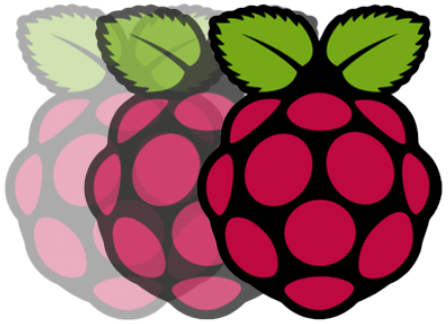
MAKERHOUSE
EMPOWERING MAKERS



Python Basics

Number to Strings

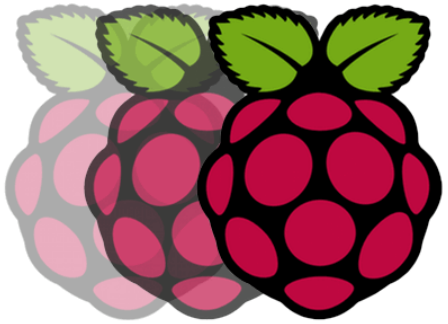
```
#!/usr/bin/env python3

s = str(123)

print(s)
```



Python Basics

Strings to Number

```
#!/usr/bin/env python3

s = int("-123")

print(s)
```

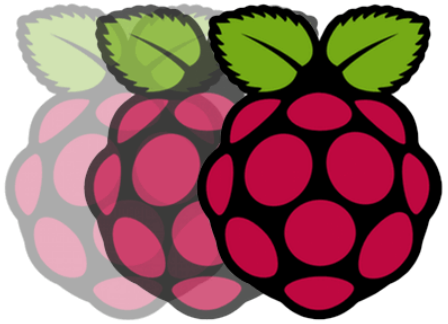


Python Basics

Strings Length

```
#!/usr/bin/env python3

s = "abc def“

print(len(s))
```

len(string) function

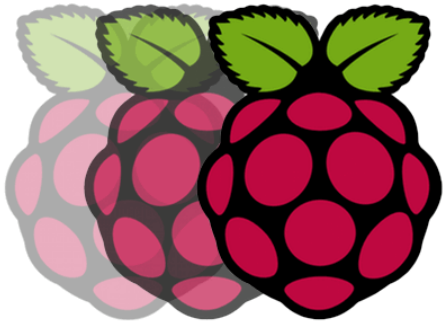


Python Basics

Find String in String

```
#!/usr/bin/env python3

s = "abcdefghi“

print(s.find("def"))
```

string.find(string) function

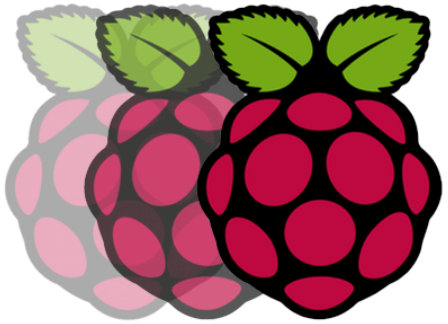


Python Basics

Extract Part in String

```
#!/usr/bin/env python3

s = "abcdefghi“

print(s[1:5])
```

[:] notation

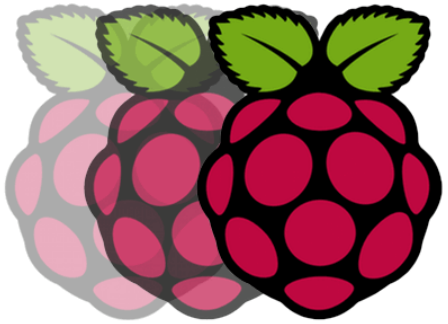


Python Basics

Replace String in String

```
#!/usr/bin/env python3

s = "It was the best of X. It was the worst of X“

s.replace("X", "times")

print(s)
```

string.replace(“old”, “new”)



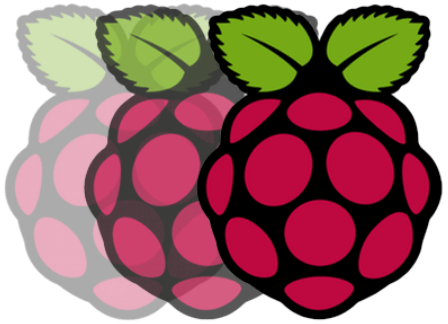
Python Basics

Uppercase and Lowercase

```
#!/usr/bin/env python3

s = "aBcDe“

print(s.upper())
```

string.upper() or string.lower()

MAKERHOUSE
EMPOWERING MAKERS



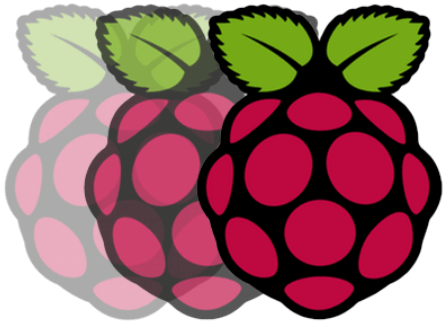
Python Basics

Uppercase and Lowercase

```
#!/usr/bin/env python3

s = "aBcDe“

print(s.upper())
```

string.upper() or string.lower()

MAKERHOUSE
EMPOWERING MAKERS

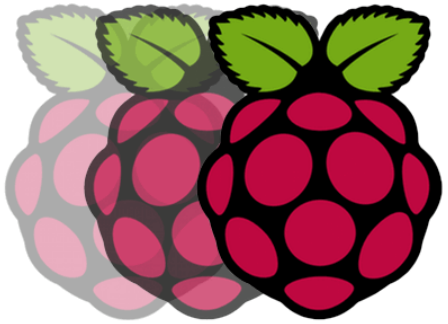


Python Basics

Conditional Command

```
#!/usr/bin/env python3

x = 101

if x > 100:

print("x is big")
```

if conditional

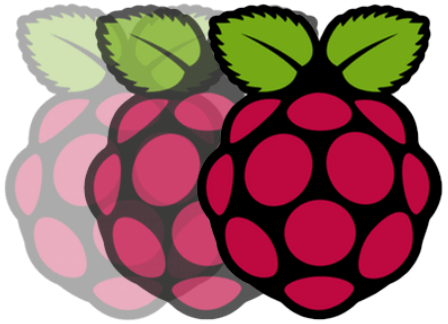


Python Basics

Conditional Command

```
#!/usr/bin/env python3

x = 101

if x > 100:
        print("x is big")
else:
        print("x is small")

print("This will always print")
```

if…else conditional



Python Basics

Conditional Command

```
#!/usr/bin/env python3

x = 90

if x > 100:
        print("x is big")

elif x < 10:
        print("x is small")

else:
        print("x is medium")
```

If…elif…else conditional

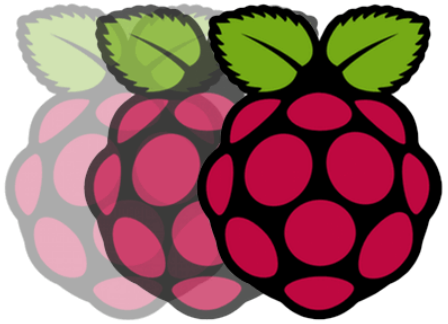
MAKERHOUSE
EMPOWERING MAKERS

Python Basics

Comparing Values

```
#!/usr/bin/env python3

print(1 != 3)

print("aa" > "ab")
```

less than (<) , greater than (>), equal to (==) , less than or equal to (<=), greater than or equal to (>=), not equal (!=)



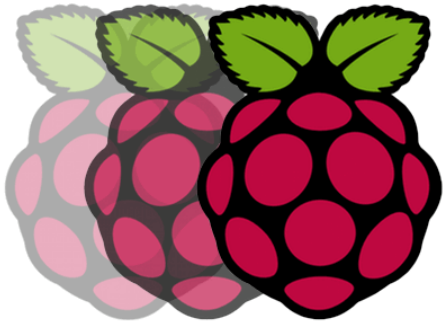
Python Basics

Logical Operator

```
#!/usr/bin/env python3

x = 17

if x >= 10 and x <= 20:
    print("x is in the middle")
```

and, or, not

MAKERHOUSE
EMPOWERING MAKERS

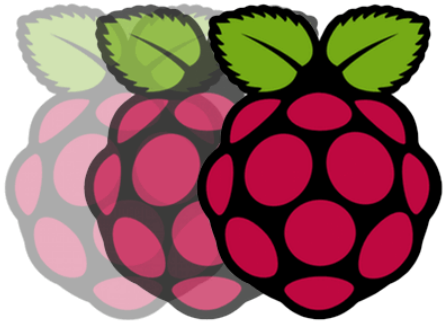


Python Basics

Repeating Instruction an Exact Number of Times

```
#!/usr/bin/env python3

for i in range (1, 11):

    print(i)
```

for loops function



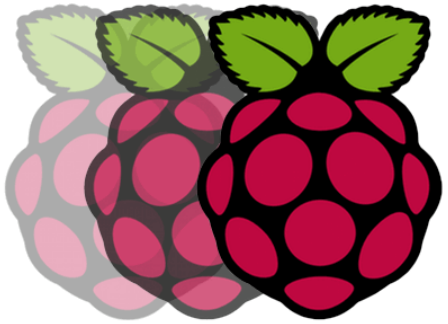
Python Basics

Repeating Instruction an Exact Number of Times

```
#!/usr/bin/env python3

answer = " "

while answer != "X":
        answer = input("Enter command: ")
```

while loops function

MAKERHOUSE
EMPOWERING MAKERS

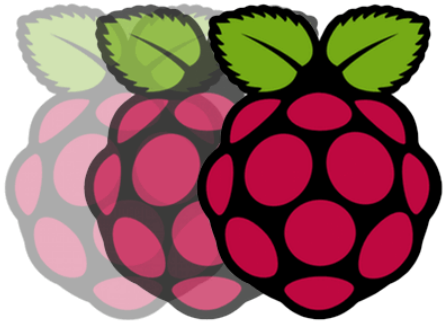


Python Basics

Breaking out of loops

```
#!/usr/bin/env python3

while True:

        answer = input("Enter command: ")

        if answer == "X":

                break
```

break function



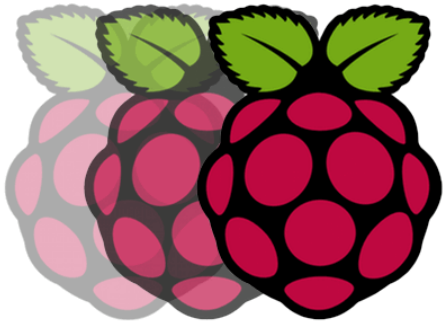
Python Basics

Functions

```
#!/usr/bin/env python3

def count_to_10 ():

        for i in range (1, 11):

                print(i)

count_to_10 ()
```

custom function

no parameters



Python Basics

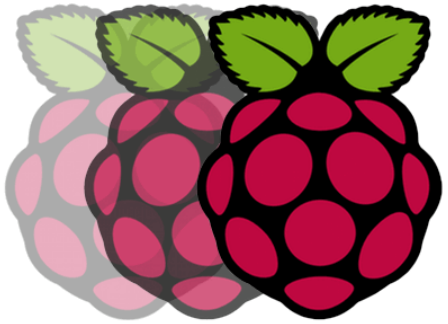
Functions

```
#!/usr/bin/env python3

def count_to_n (n):

        for i in range (1, n + 11):

                print(i)

count_to_n (10)
```

custom function

with single parameters -- required

MAKERHOUSE
EMPOWERING MAKERS



Python Basics

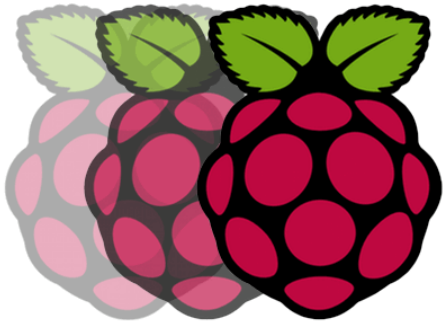
Functions

```
#!/usr/bin/env python3

def count_to_n (n = 10):

        for i in range (1, n + 11):

            print(i)

count_to_n ()
```

custom function

with single parameters -- optional

MAKERHOUSE
EMPOWERING MAKERS



Python Basics

Functions

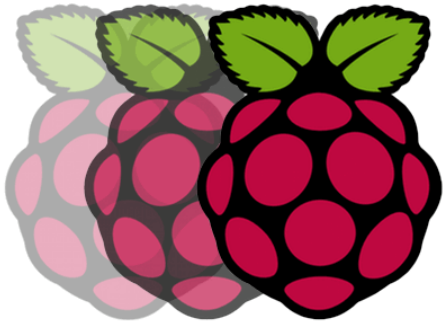
```
#!/usr/bin/env python3

def count_to_n (from_num = 1, to_num = 10):

        for i in range (from_num, to_num + 1):

            print(i)
```

```
count_to_n ()
count_to_n (2)
count_to_n (2, 8)
```

custom function

with multiple parameters

MAKERHOUSE
EMPOWERING MAKERS



Python Basics

Functions

```
#!/usr/bin/env python3

def make_polite (sentence):

        return sentence + “please”

print(make_polite(“Pass the cheese”))
```

custom function

with string parameters

MAKERHOUSE
EMPOWERING MAKERS