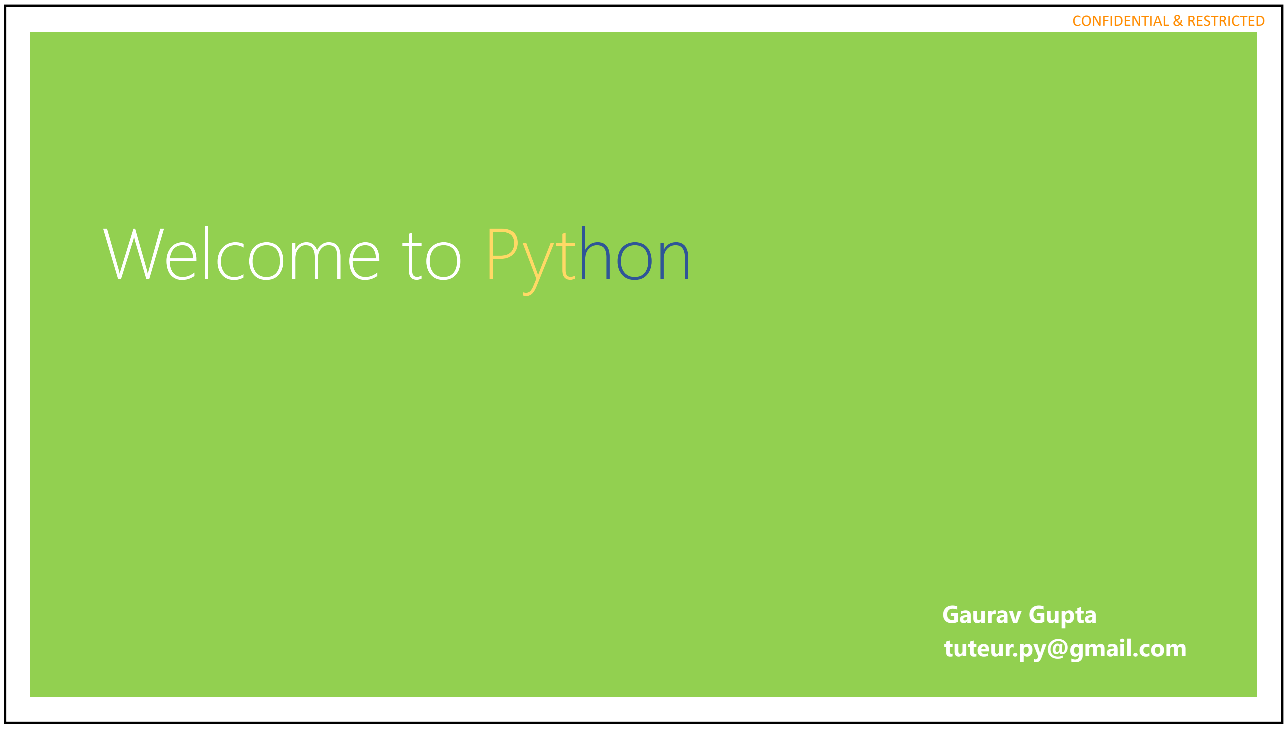
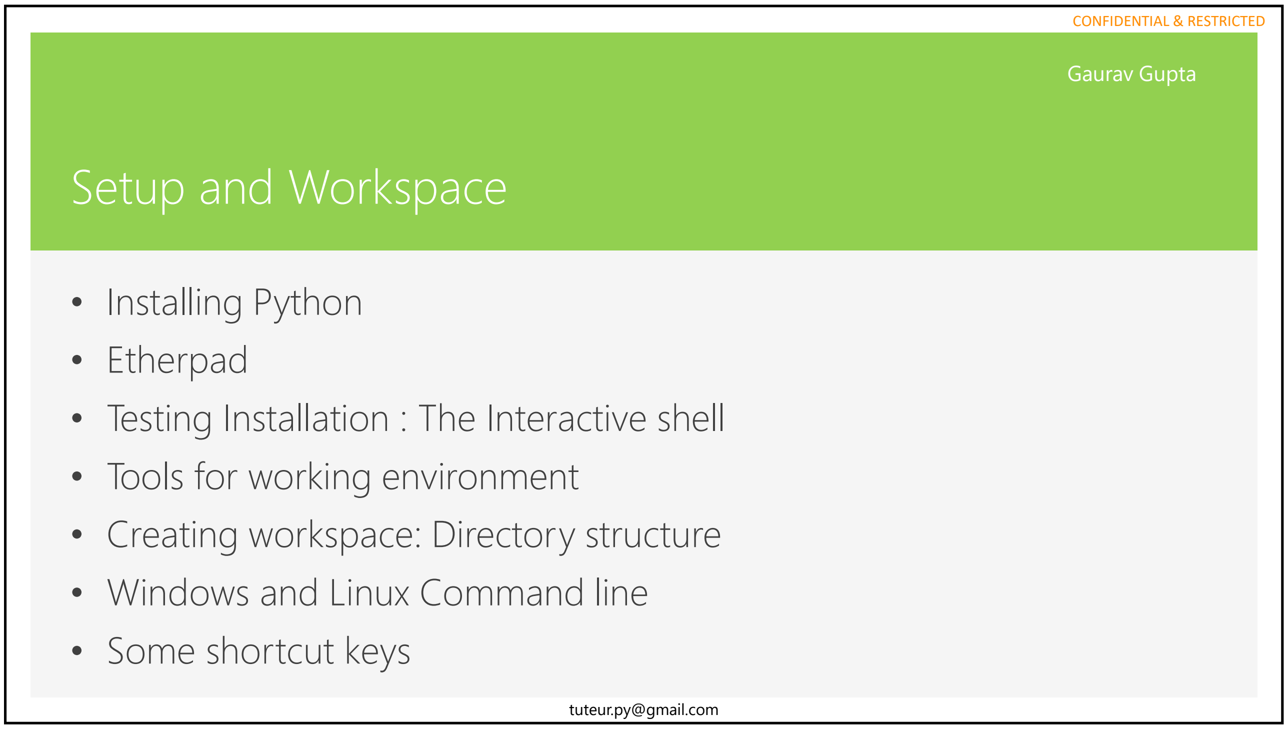


CONFIDENTIAL & RESTRICTED

Welcome to Python

Gaurav Gupta
tuteur.py@gmail.com



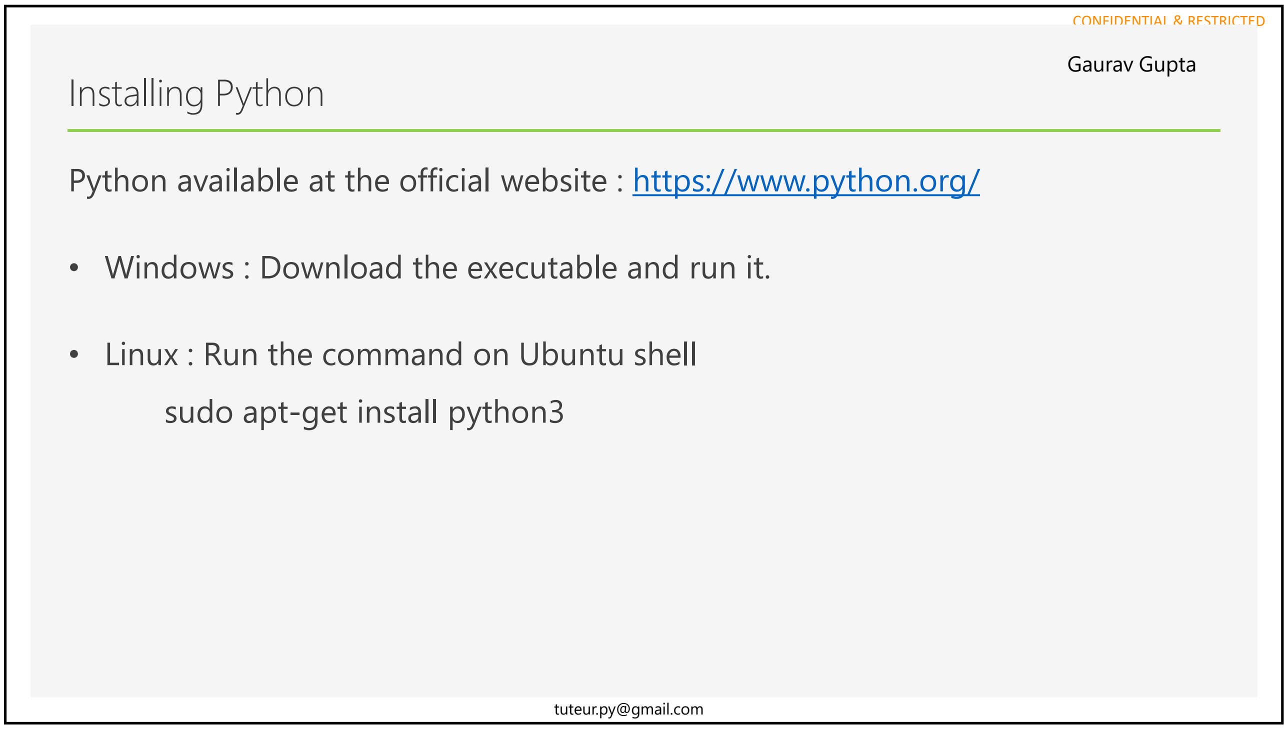
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Setup and Workspace

- Installing Python
- Etherpad
- Testing Installation : The Interactive shell
- Tools for working environment
- Creating workspace: Directory structure
- Windows and Linux Command line
- Some shortcut keys

tuteur.py@gmail.com



Gaurav Gupta

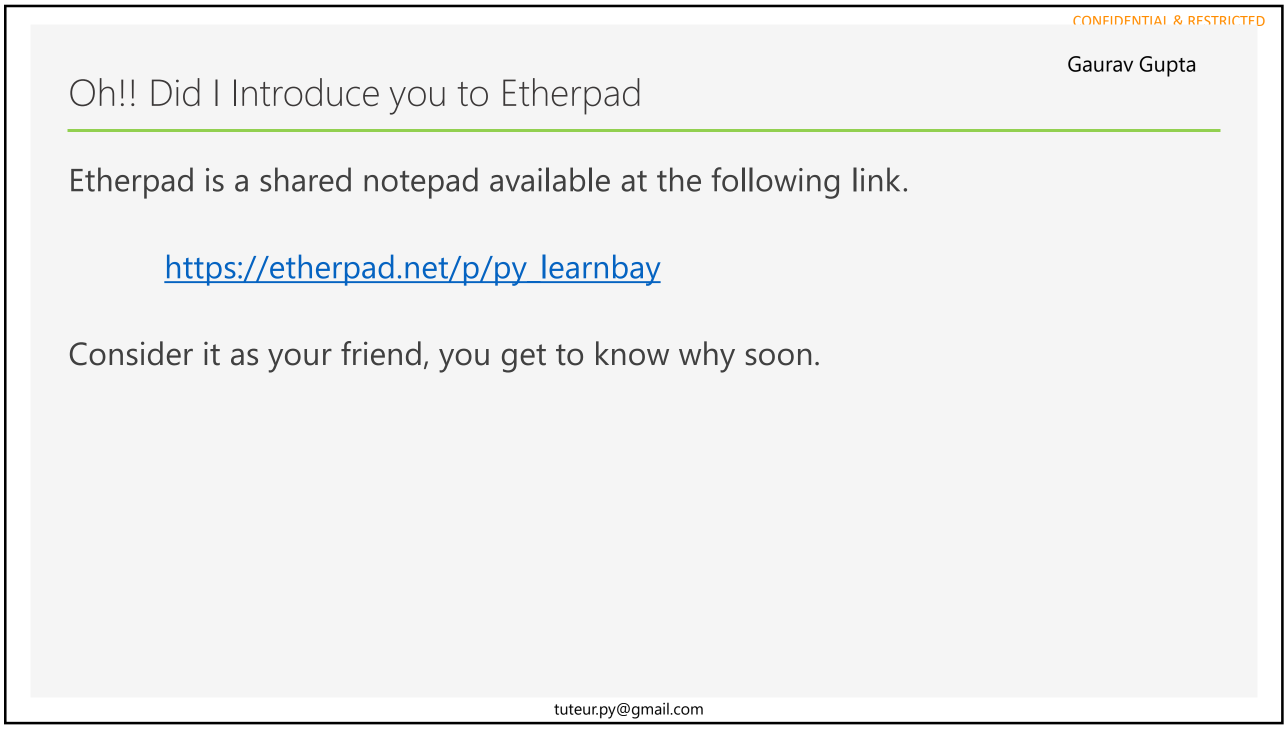
Installing Python

Python available at the official website : https://www.python.org/

- Windows : Download the executable and run it.
- Linux : Run the command on Ubuntu shell

```
sudo apt-get install python3
```

tuteur.py@gmail.com



Gaurav Gupta

Oh!! Did I Introduce you to Etherpad

Etherpad is a shared notepad available at the following link.

https://etherpad.net/p/py_learnbay

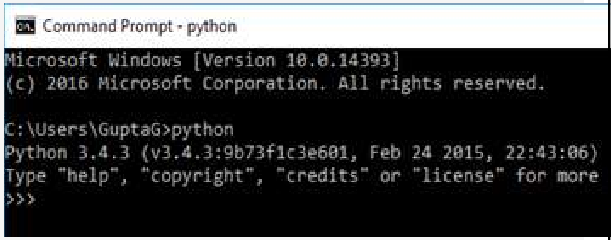
Consider it as your friend, you get to know why soon.

tuteur.py@gmail.com

Gaurav Gupta

Testing Installation : The Interactive shell

- Windows : Press Windows Key and type cmd. On the Terminal type python



```
Command Prompt - python
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\GuptaG>python
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06)
Type "help", "copyright", "credits" or "license" for more
>>>
```

- Linux : Open a terminal (Ubuntu CTRL+ALT+T) and type python.

** if you get error like command not found, add python installation path

tuteur.py@gmail.com

Gaurav Gupta

Tools for working environment

- Use an IDE

  Pycharm IDE with Python 3.x.x

  https://www.jetbrains.com/pycharm/download/

- Use any text editor and Command line (my preferred way)

  Write Scripts using a text editor : Notepad++, vi, vim, Sublime Text..

  Windows or Linux command line for executing.

tuteur.py@gmail.com

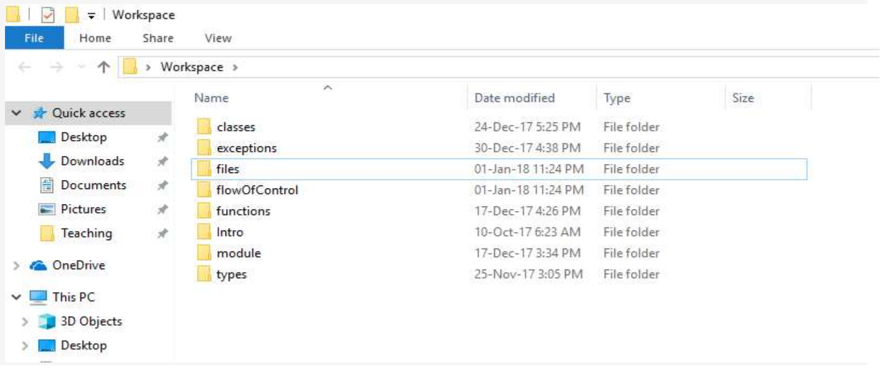
Gaurav Gupta

Creating workspace: Directory structure

- Create a folder ***workspace*** : all our scripts will be in this folder
- Maintain separate folders for each topic in ***workspace*** folder.
- Make sure to name the script files in following convention: f**N_topic**.py

Ex:

f1_ifStatement.py
f2_ifElse.py



Name	Date modified	Type	Size
classes	24-Dec-17 5:25 PM	File folder	
exceptions	30-Dec-17 4:38 PM	File folder	
files	01-Jan-18 11:24 PM	File folder	
flowOfControl	01-Jan-18 11:24 PM	File folder	
functions	17-Dec-17 4:26 PM	File folder	
Intro	10-Oct-17 6:23 AM	File folder	
module	17-Dec-17 3:34 PM	File folder	
types	25-Nov-17 3:05 PM	File folder	

Gaurav Gupta

Windows and Linux Command line

	Windows	Linux
Go to the folder	*cd <folder Name>* *Ex:* *cd Workspace*	*cd <folder Name>* *Ex:* *cd Workspace*
Go to the previous directory	*cd ..*	*cd ..*
List files in current directory	*dir*	*ls* *ls –la*

Use up and down arrow keys to view previous commands in cmd window

Gaurav Gupta

Notepad++ Shortcuts

Ctrl + a	To select everything in current file
Ctrl + s	Save current file
Ctrl + Tab	To switch files
Ctrl + n	To open new file
Ctrl + c	To copy selected text
Ctrl + v	To paste selected text

- Press *Shift* and Arrow keys to make selection of a part of text (you can use Ctrl key while selecting to make selection faster)

tuteur.py@gmail.com

Gaurav Gupta

Windows shortcuts

Open command window in current folder

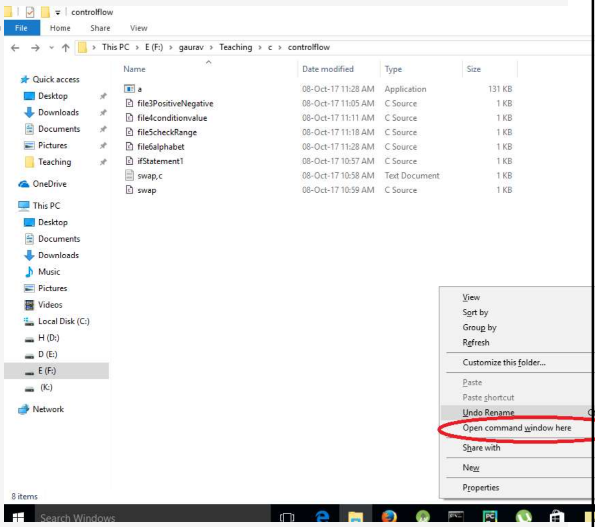
Press the Shift Key and right click

You will see the option :

Open Command Window Here

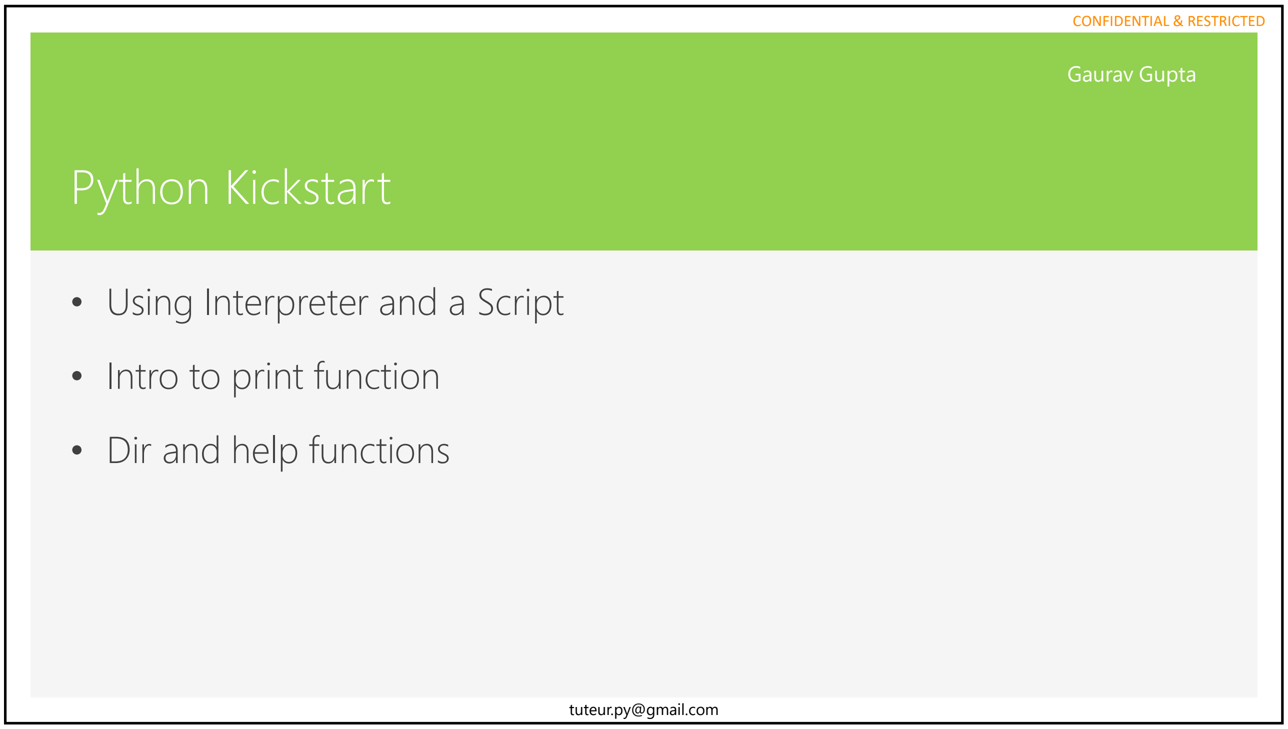
To Switch Tabs/Windows

Alt + Tab and Alt + Shift + Tab



Name	Date modified	Type	Size
a	08-Oct-17 11:28 AM	Application	131 KB
file3PositiveNegative	08-Oct-17 11:05 AM	C Source	1 KB
file4conditionvalue	08-Oct-17 11:11 AM	C Source	1 KB
file5checkRange	08-Oct-17 11:18 AM	C Source	1 KB
file6alphabet	08-Oct-17 11:28 AM	C Source	1 KB
ifStatement1	08-Oct-17 10:57 AM	C Source	1 KB
swap,c	08-Oct-17 10:58 AM	Text Document	1 KB
swap	08-Oct-17 10:59 AM	C Source	1 KB

tuteur.py@gmail.com

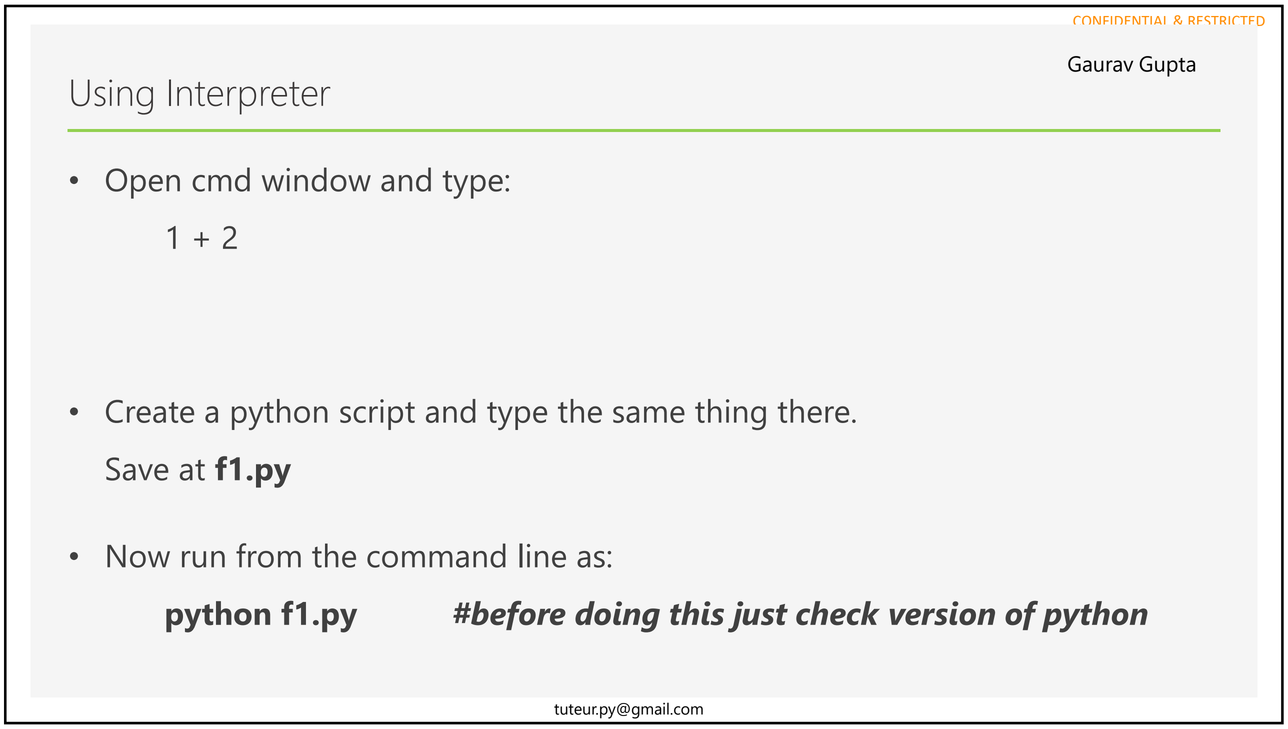


Gaurav Gupta

Python Kickstart

- Using Interpreter and a Script
- Intro to print function
- Dir and help functions

tuteur.py@gmail.com



Gaurav Gupta

Using Interpreter

- Open cmd window and type:

```
1 + 2
```

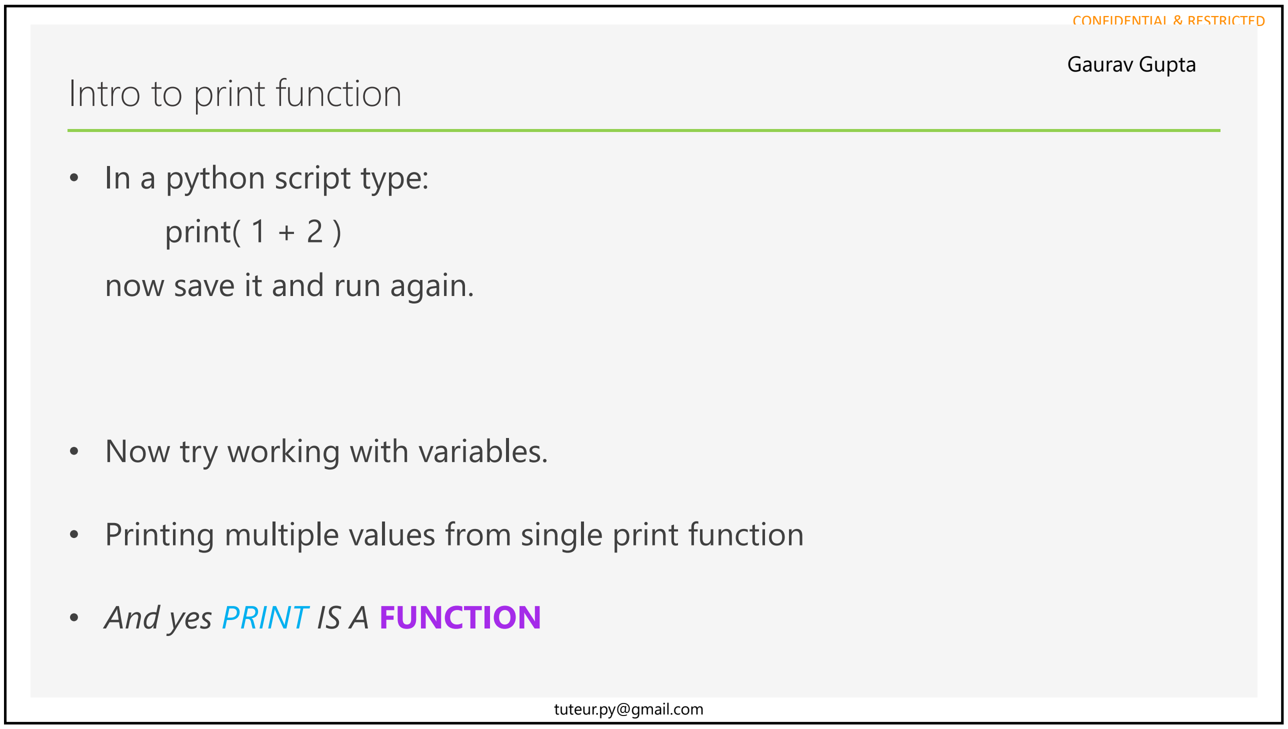
- Create a python script and type the same thing there.

  Save at **f1.py**

- Now run from the command line as:

  python f1.py ***#before doing this just check version of python***

tuteur.py@gmail.com



Gaurav Gupta

Intro to print function

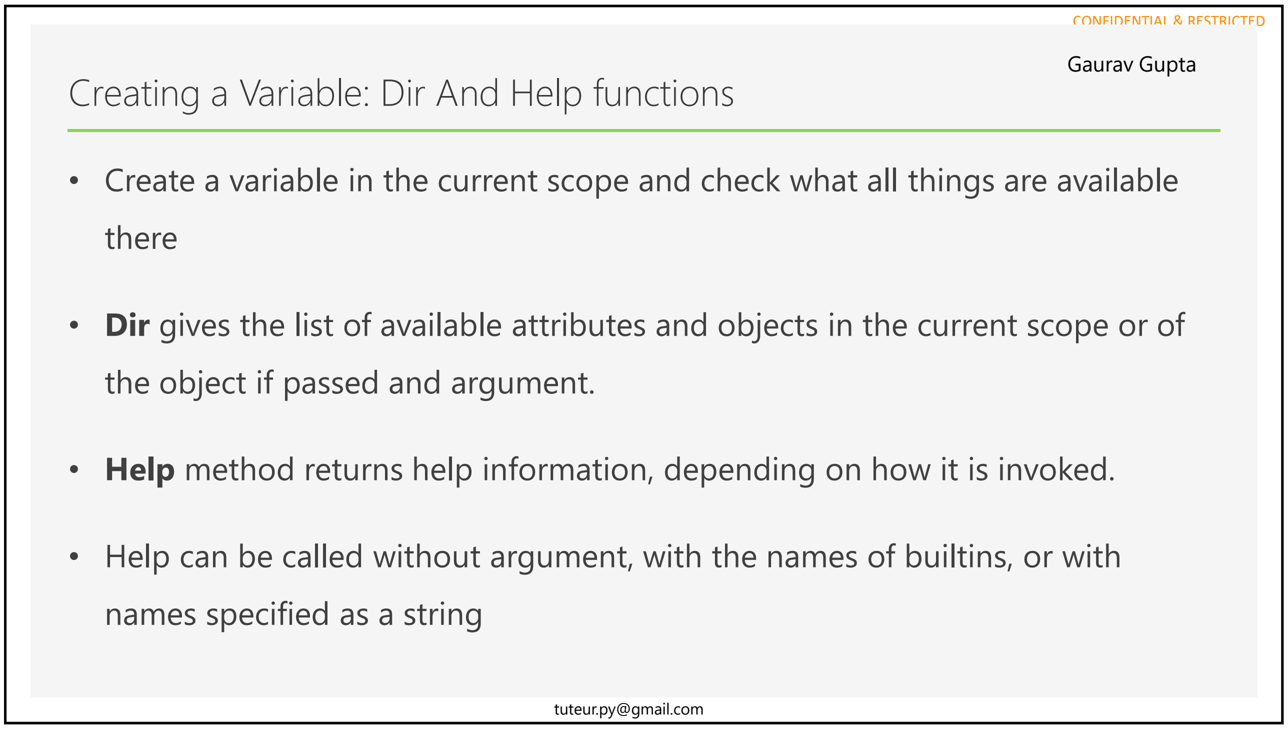
- In a python script type:

```
print( 1 + 2 )
```

  now save it and run again.

- Now try working with variables.
- Printing multiple values from single print function
- *And yes PRINT IS A* **FUNCTION**

tuteur.py@gmail.com

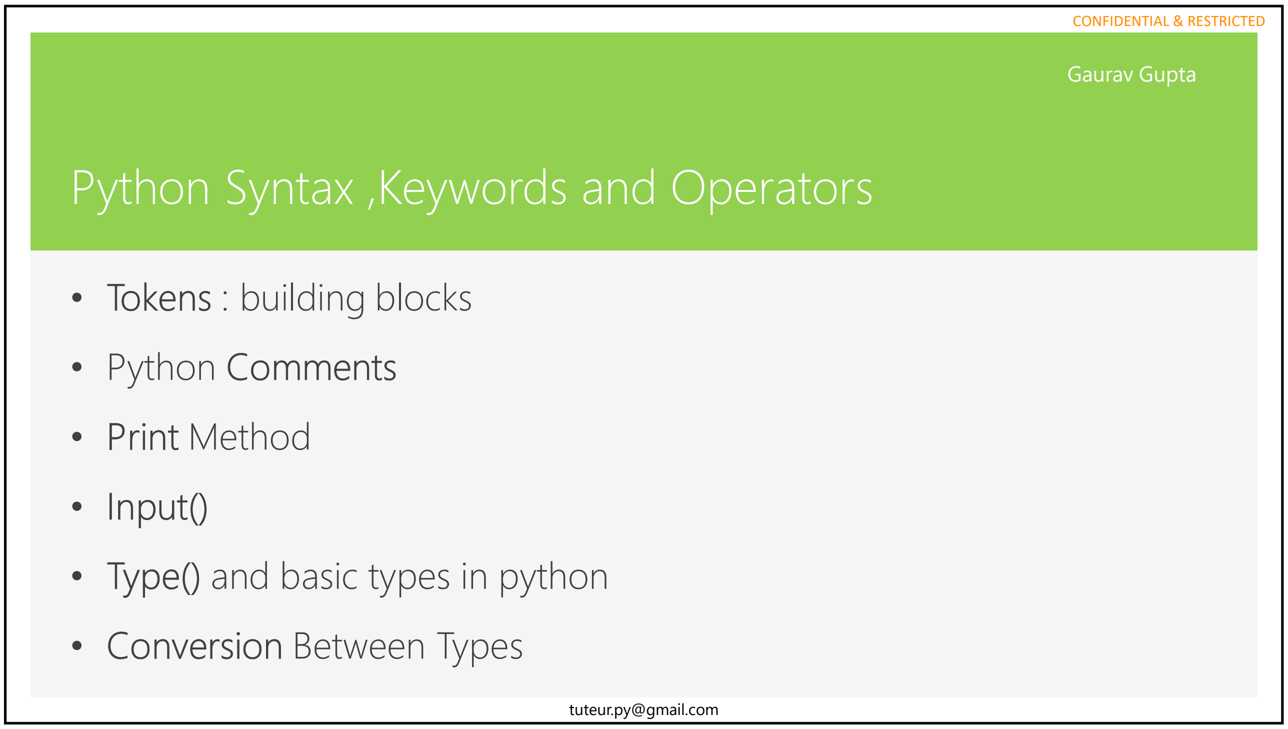


Gaurav Gupta

Creating a Variable: Dir And Help functions

- Create a variable in the current scope and check what all things are available there
- **Dir** gives the list of available attributes and objects in the current scope or of the object if passed and argument.
- **Help** method returns help information, depending on how it is invoked.
- Help can be called without argument, with the names of builtins, or with names specified as a string

tuteur.py@gmail.com

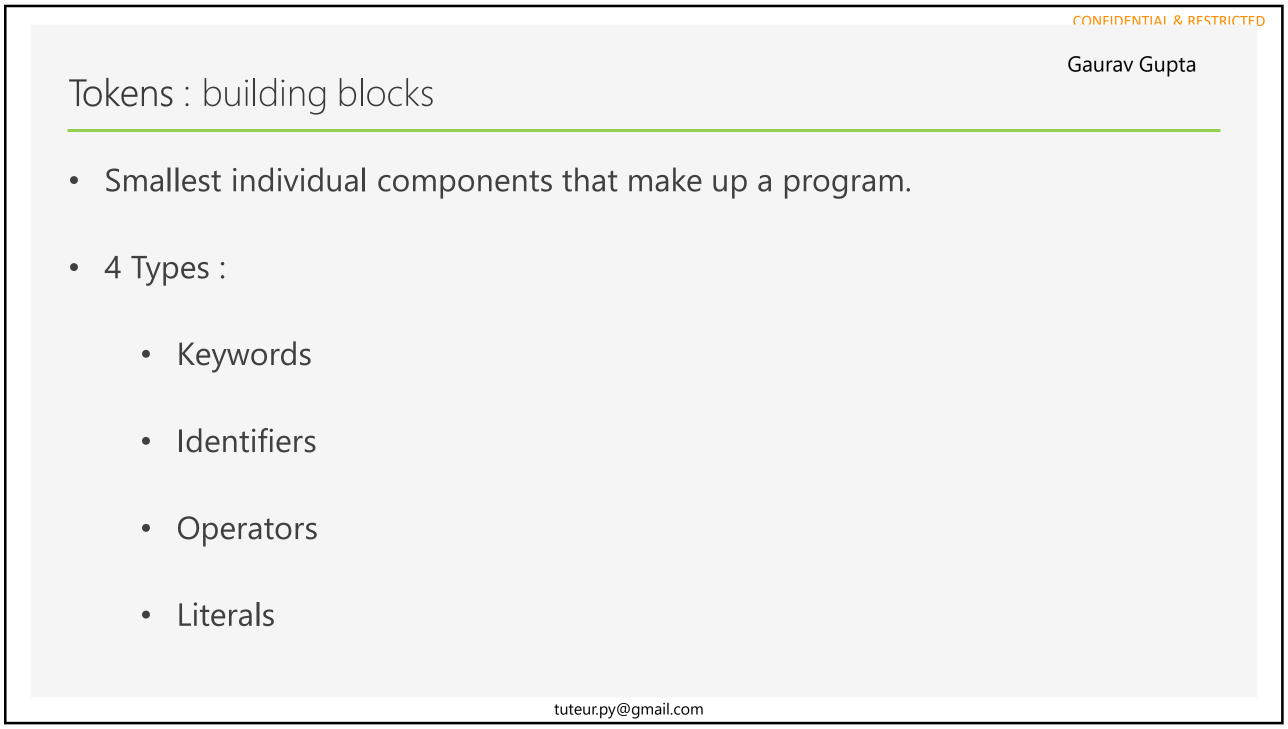


Gaurav Gupta

Python Syntax ,Keywords and Operators

- Tokens : building blocks
- Python Comments
- Print Method
- Input()
- Type() and basic types in python
- Conversion Between Types

tuteur.py@gmail.com



Gaurav Gupta

Tokens : building blocks

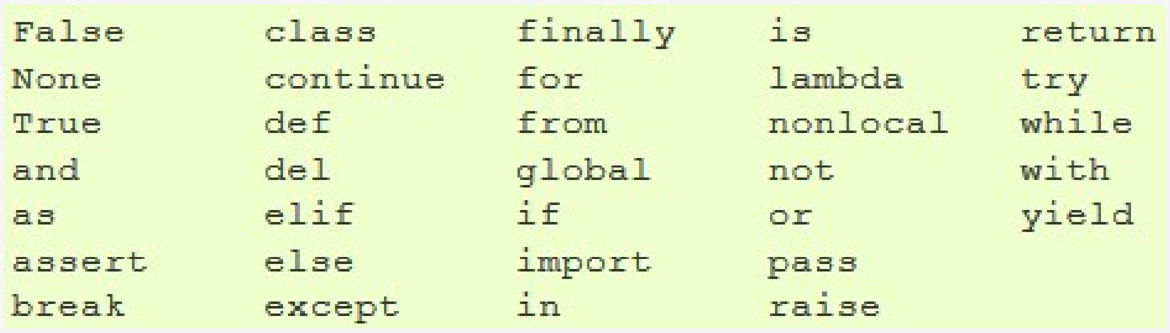
- Smallest individual components that make up a program.
- 4 Types :
  - Keywords
  - Identifiers
  - Operators
  - Literals

tuteur.py@gmail.com

Gaurav Gupta

Keywords

- Special reserved words predefined or reserved by the language.



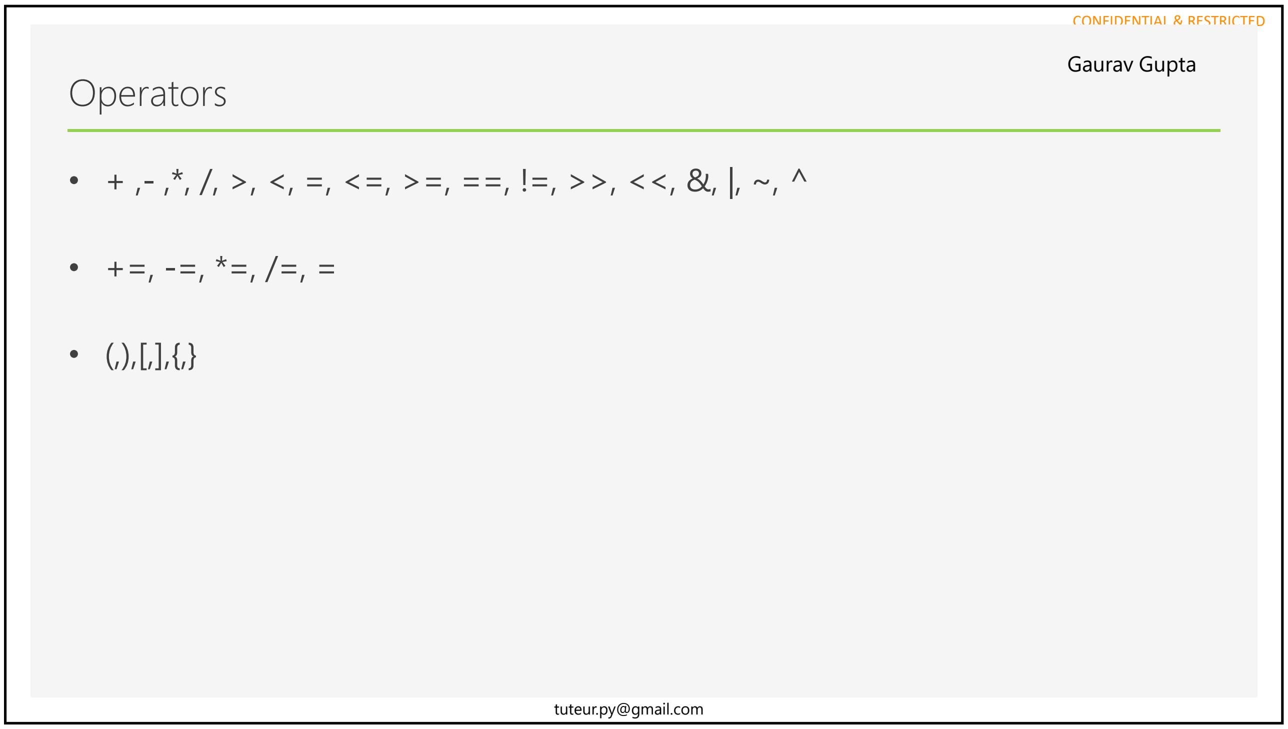
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Gaurav Gupta

Identifiers

- **Identifiers** can be a combination of letters in lowercase (**a to z**) or uppercase (**A to Z**) or digits (**0 to 9**) or an underscore (_)
- Variable names, class names, function names and module names are all identifiers.
- Some special identifiers in Python :

__*__	:	Special Reserved system defined names
__*	:	Used to define private class members

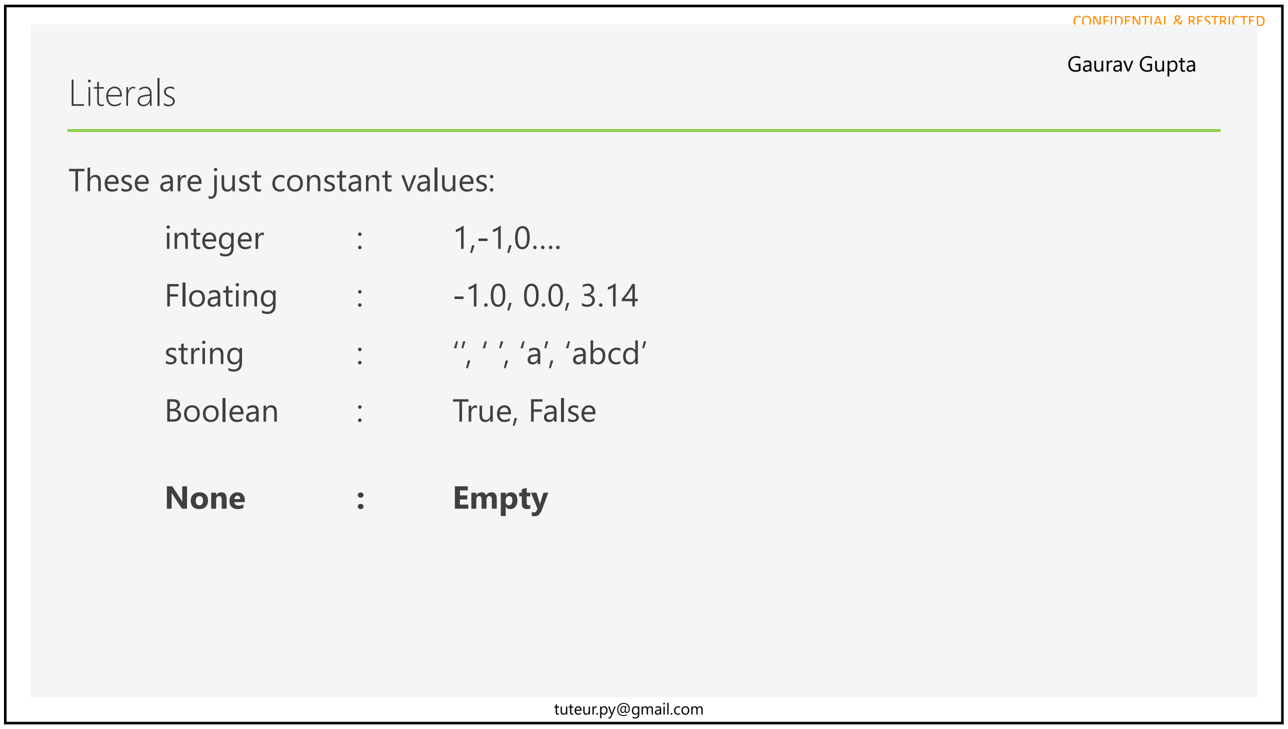


Gaurav Gupta

Operators

- + ,- ,*, /, >, <, =, <=, >=, ==, !=, >>, <<, &, |, ~, ^
- +=, -=, *=, /=, =
- (,),[,],{,}

tuteur.py@gmail.com



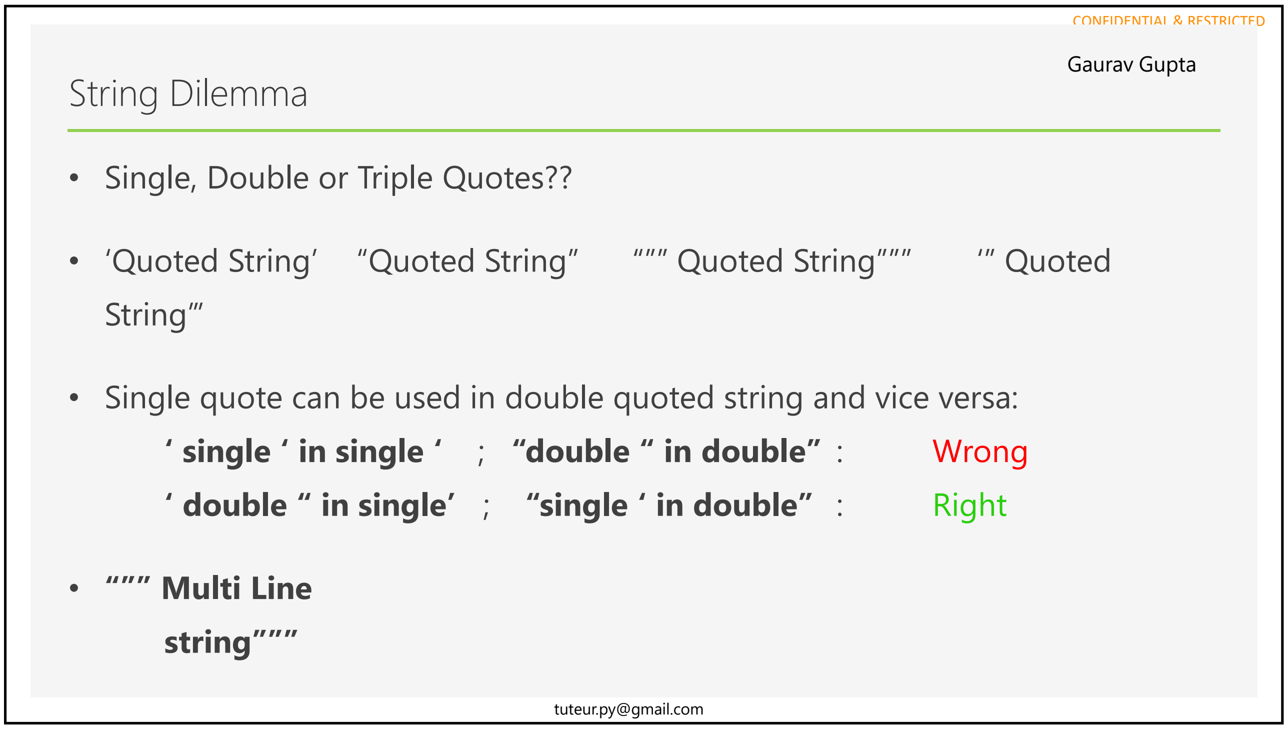
Gaurav Gupta

Literals

These are just constant values:

integer	:	1,-1,0….
Floating	:	-1.0, 0.0, 3.14
string	:	‘’, ‘ ’, ‘a’, ‘abcd’
Boolean	:	True, False
None	**:**	**Empty**

tuteur.py@gmail.com



Gaurav Gupta

String Dilemma

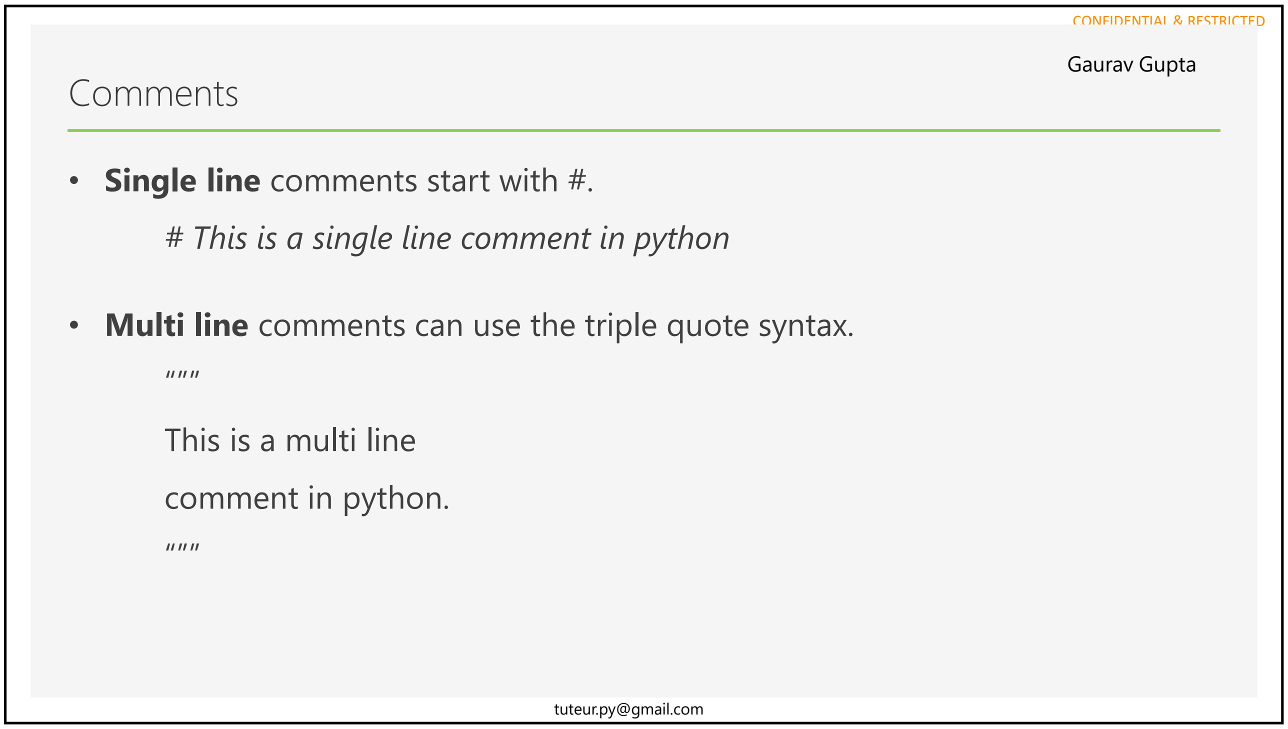
- Single, Double or Triple Quotes??
- ‘Quoted String’ “Quoted String” “”” Quoted String””” ‘” Quoted String’”
- Single quote can be used in double quoted string and vice versa:

  ‘ single ‘ in single ‘ ; **“double “ in double”** : Wrong

  ‘ double “ in single’ ; **“single ‘ in double”** : Right
- **“”” Multi Line**

  string”””

tuteur.py@gmail.com



Gaurav Gupta

Comments

- **Single line** comments start with #.

  # This is a single line comment in python
- **Multi line** comments can use the triple quote syntax.

  """

  This is a multi line

  comment in python.

  """

tuteur.py@gmail.com

Gaurav Gupta

Print Function

- Print method prints to the standard output
- Syntax:

  *print(<var/const>, …, **sep**= '<separator>', **end** = '<delimiter>', **file** = <file object>)*

  sep, file *and* ***end****, arguments are optional and should appear in the end.*
- *Escape Sequences :* ***\n*** *and* ***\t***

tuteur.py@gmail.com

Gaurav Gupta

Type Method

- Syntax:

  type(<object argument>)
- Returns the type of the argument
- Argument might be variables, objects ….
- Some basic types are:

  int, float, string, bool, complex

tuteur.py@gmail.com

Gaurav Gupta

Converting Between types

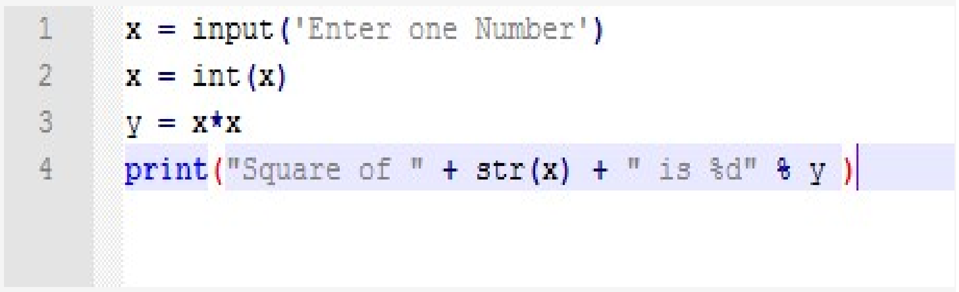
- int(<string>), int(<int>), int(<float>) # converts string containing digits to int
- str(<int/float/….>) # converts any type to its string representation

tuteur.py@gmail.com

Gaurav Gupta

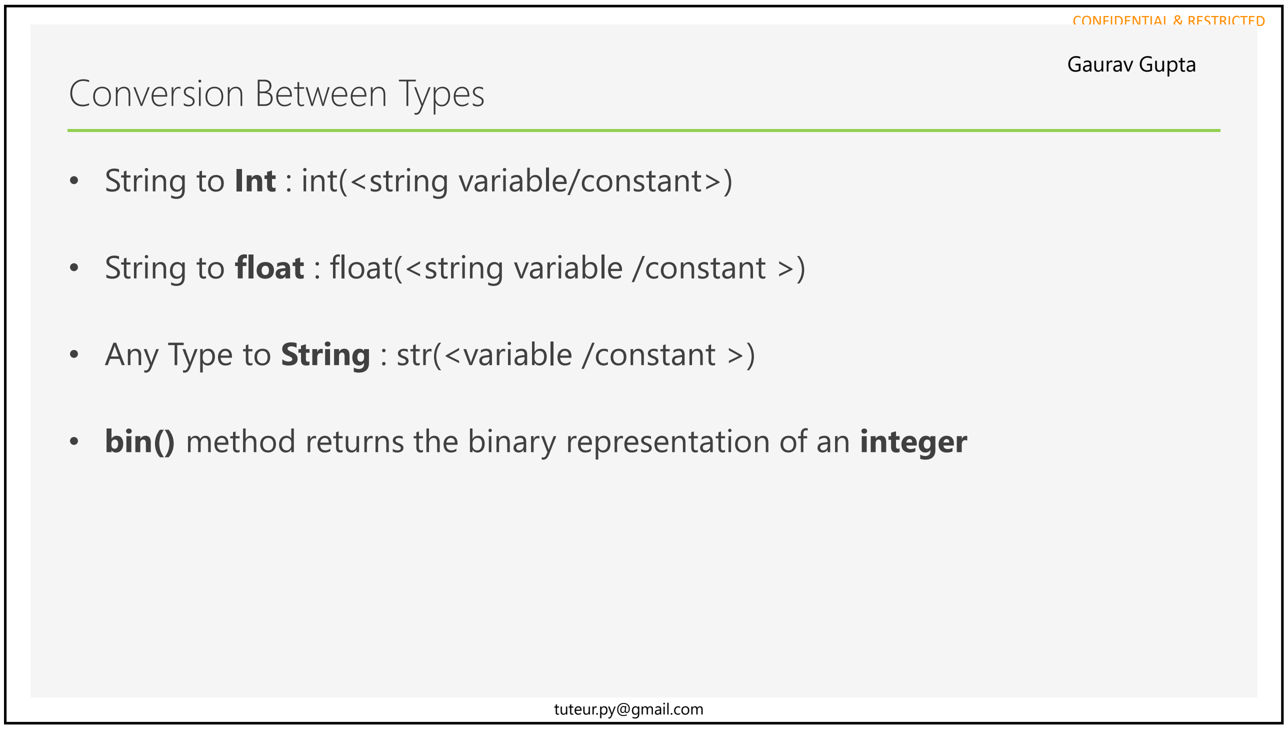
Input()

- The input method returns the value entered by user as a string
- Also allows to specify a string argument for a message to displayed



```
x = input('Enter one Number')
x = int(x)
y = x*x
print("Square of " + str(x) + " is %d" % y )
```

tuteur.py@gmail.com

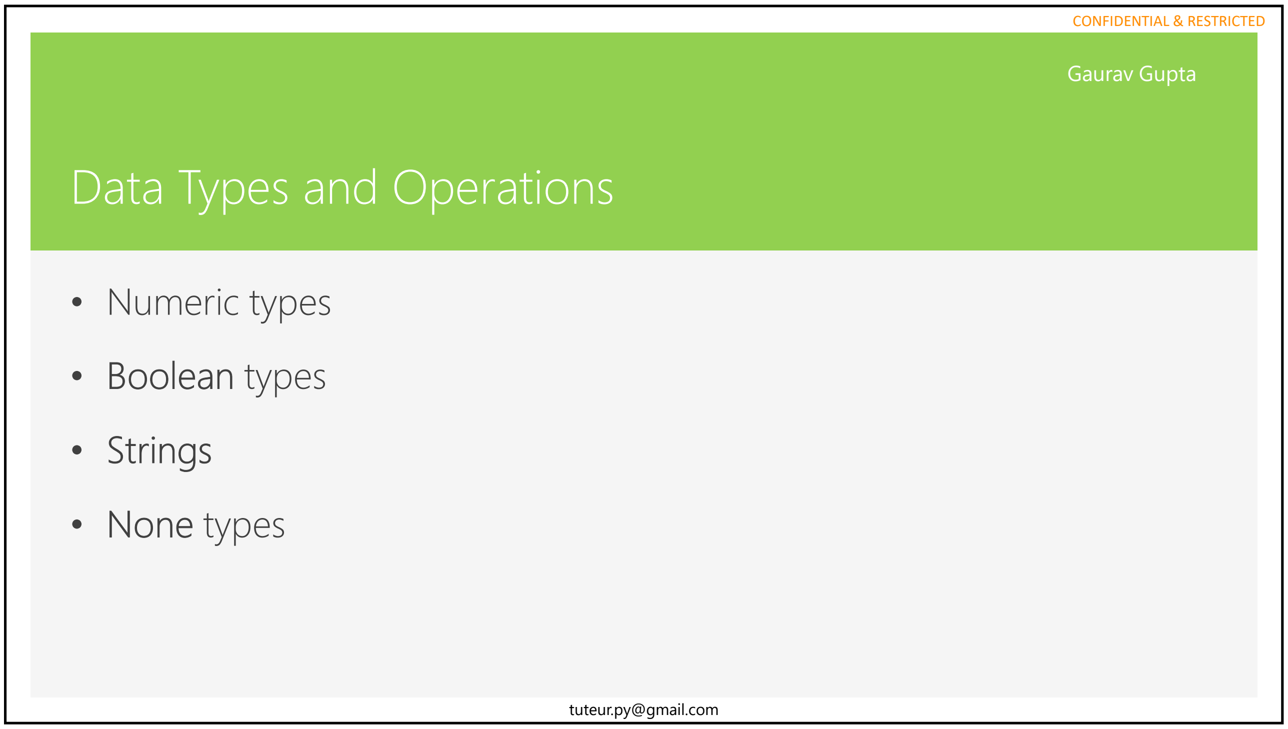


Gaurav Gupta

Conversion Between Types

- String to **Int** : int(<string variable/constant>)
- String to **float** : float(<string variable /constant >)
- Any Type to **String** : str(<variable /constant >)
- **bin()** method returns the binary representation of an **integer**

tuteur.py@gmail.com

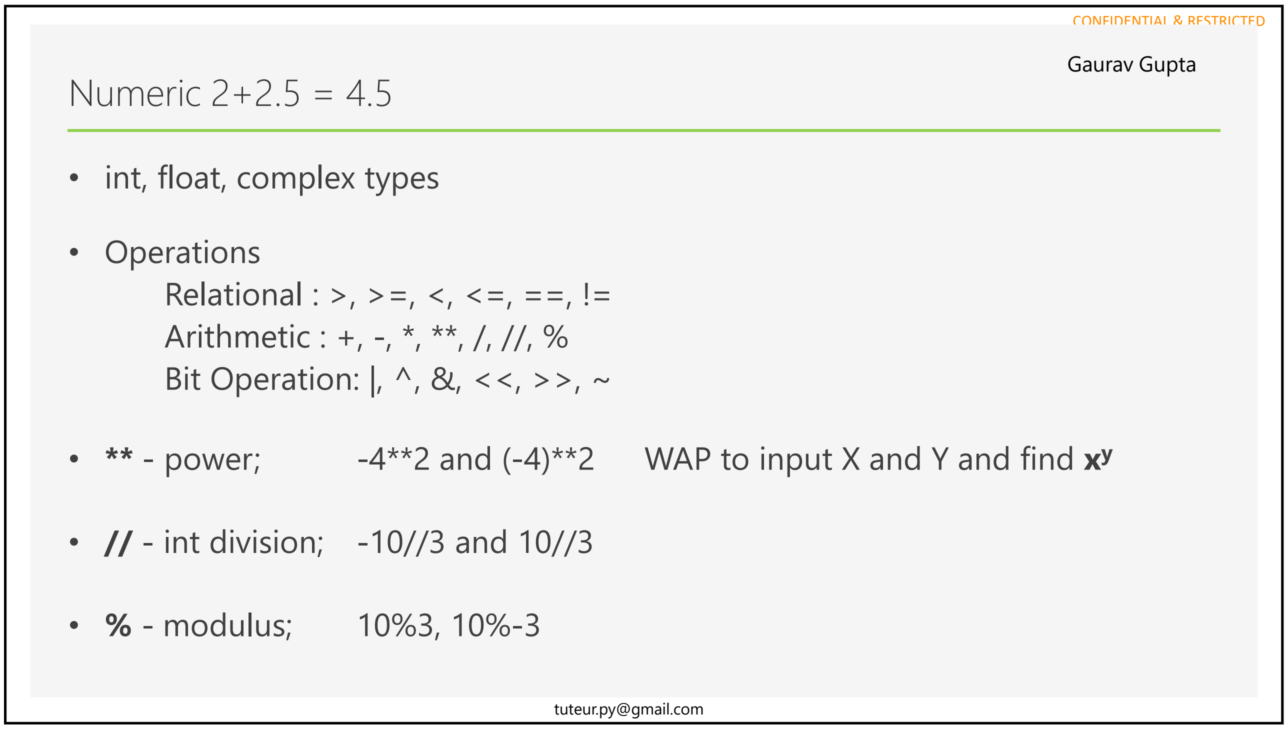


Gaurav Gupta

Data Types and Operations

- Numeric types
- Boolean types
- Strings
- None types

tuteur.py@gmail.com

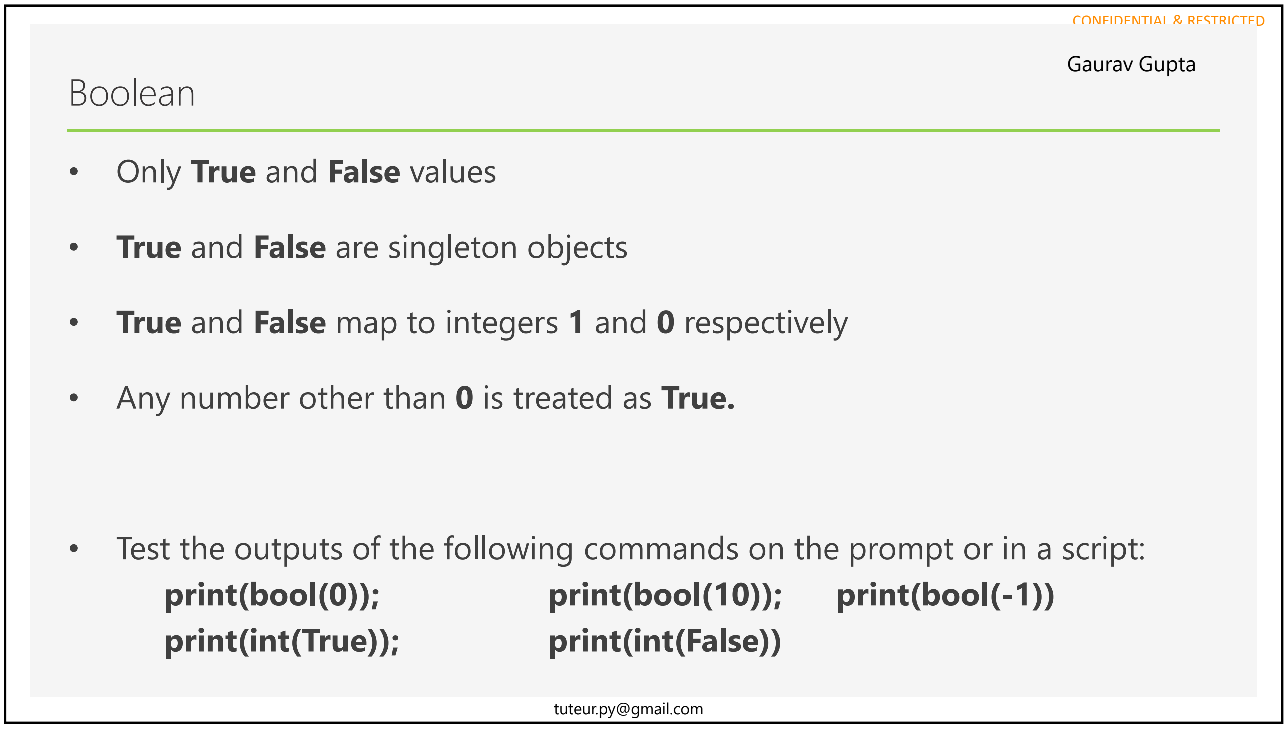


Gaurav Gupta

Numeric 2+2.5 = 4.5

- int, float, complex types
- Operations
  - Relational : >, >=, <, <=, ==, !=
  - Arithmetic : +, -, *, **, /, //, %
  - Bit Operation: |, ^, &, <<, >>, ~
- ****** - power; -4**2 and (-4)**2 WAP to input X and Y and find $\mathbf{x^y}$
- **//** - int division; -10//3 and 10//3
- **%** - modulus; 10%3, 10%-3

tuteur.py@gmail.com



Gaurav Gupta

Boolean

- Only **True** and **False** values
- **True** and **False** are singleton objects
- **True** and **False** map to integers **1** and **0** respectively
- Any number other than **0** is treated as **True.**
- Test the outputs of the following commands on the prompt or in a script:

```
print(bool(0));        print(bool(10));    print(bool(-1))
print(int(True));      print(int(False))
```

tuteur.py@gmail.com

Gaurav Gupta

Str '2'+'2.5' = '22.5'

- Strings are **immutable sequence** of characters
- Ex:

  ' simple string'

  "double quotes"

  """ triple quotes"""

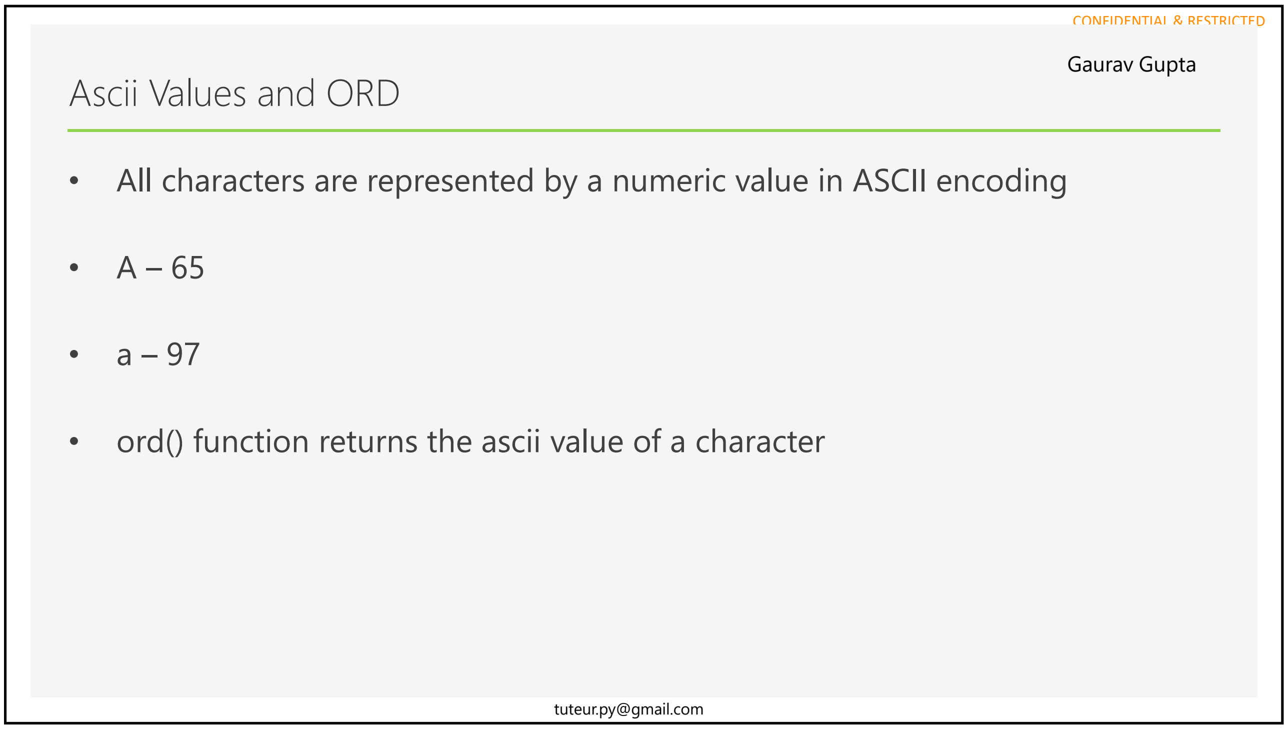
tuteur.py@gmail.com

Gaurav Gupta

None type

- **None** represents null or empty
- Often returned by some methods, to mark no return value.

tuteur.py@gmail.com

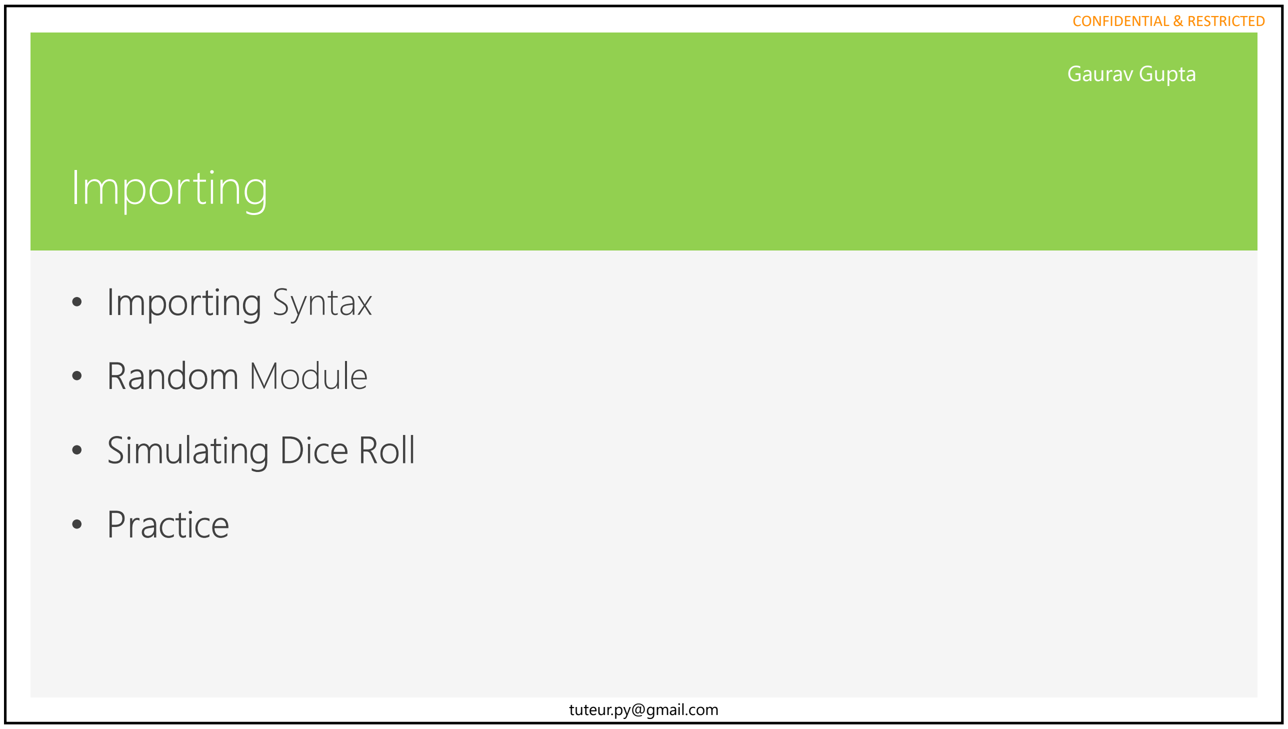


Gaurav Gupta

Ascii Values and ORD

- All characters are represented by a numeric value in ASCII encoding
- A – 65
- a – 97
- ord() function returns the ascii value of a character

tuteur.py@gmail.com



Gaurav Gupta

Importing

- Importing Syntax
- Random Module
- Simulating Dice Roll
- Practice

tuteur.py@gmail.com

Gaurav Gupta

Importing Modules : Import statement

- import <module name> **# import the entire module**

    import cmath

    cmath.sqrt(-1)

- from <module name> import * **# import all components from module**

    *from cmath import **

    sqrt(-1)

- from <module name> import <class/function>**# import selected component from module**

    from cmath import sqrt

    sqrt(-1)

tuteur.py@gmail.com

Gaurav Gupta

Random Library

- import random module using:

    import random

- Random Integers :

    randrange(end) **0 <= N <= end – 1**

    randrange(100)

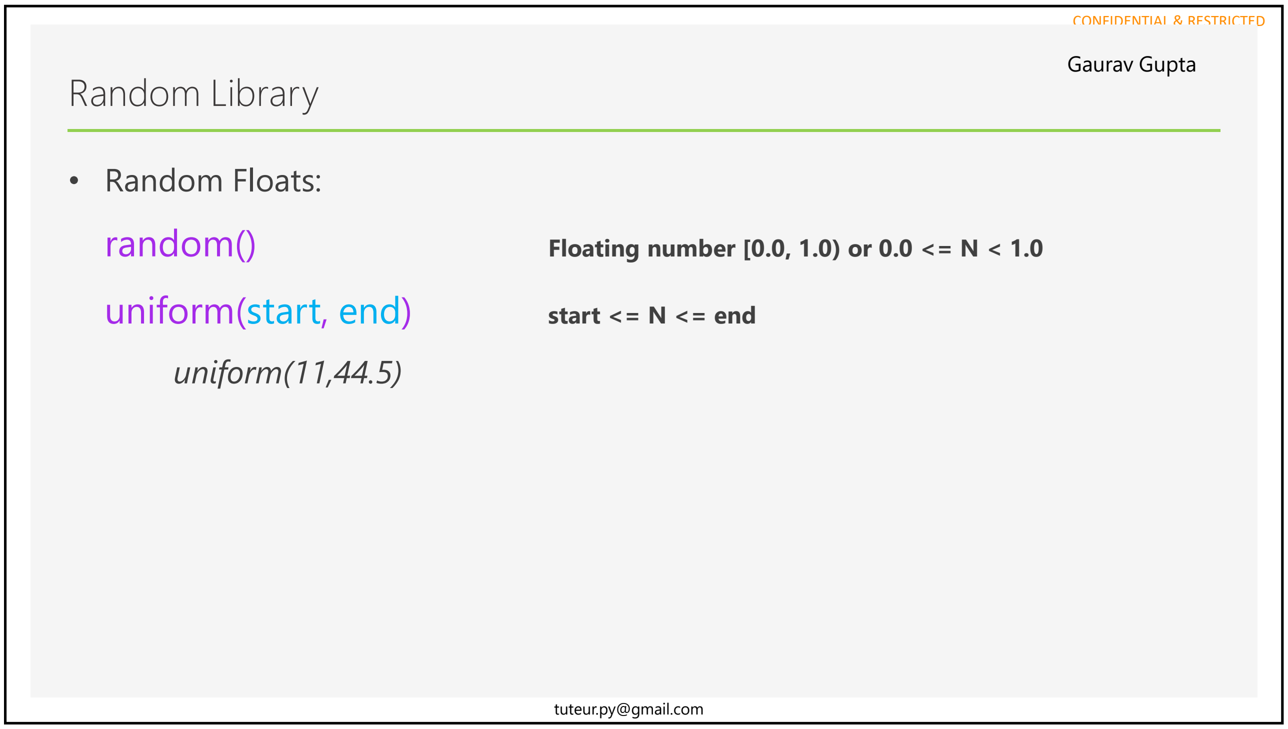
    randrange(start, end, [step]) **one from start, start+step, start + step*2..**

    randrange(10,20,2)

    randint(start, end) **start <= N <= end**

    randint(1,10)

tuteur.py@gmail.com



Gaurav Gupta

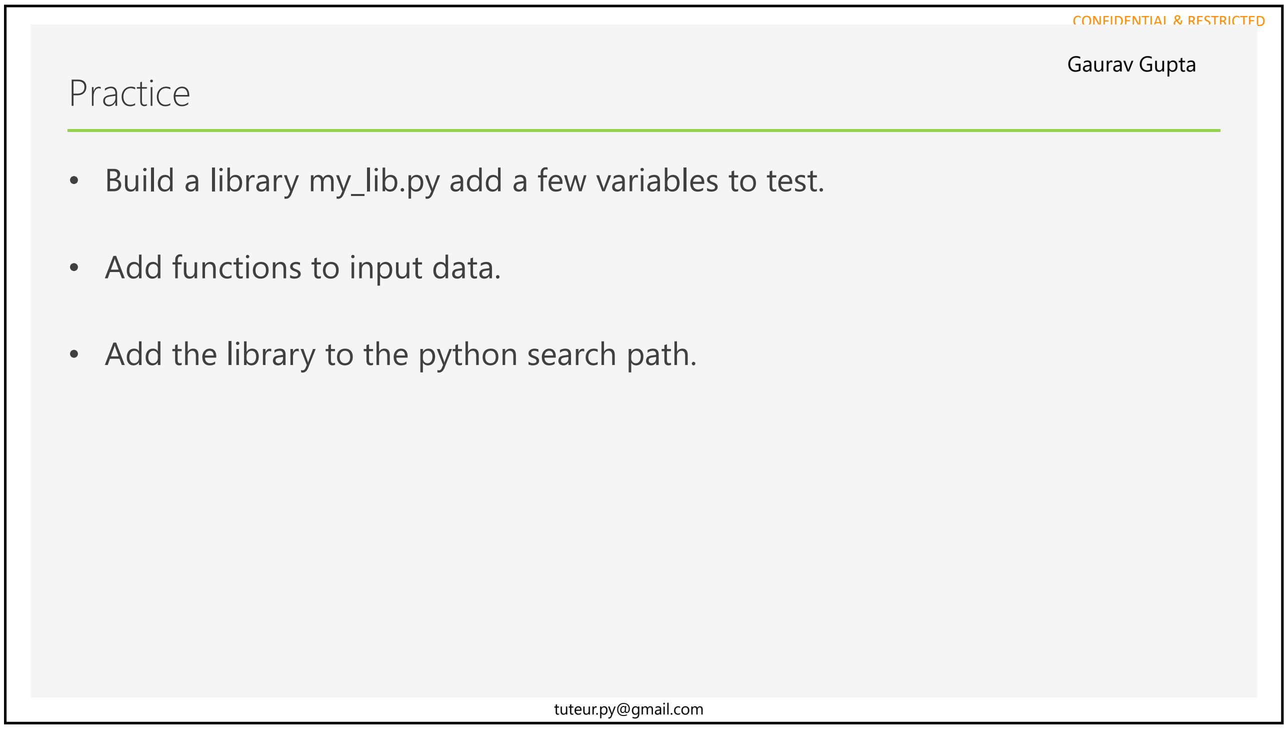
Random Library

- Random Floats:

random()	**Floating number [0.0, 1.0) or 0.0 <= N < 1.0**
uniform(start, end)	**start <= N <= end**

uniform(11,44.5)

tuteur.py@gmail.com

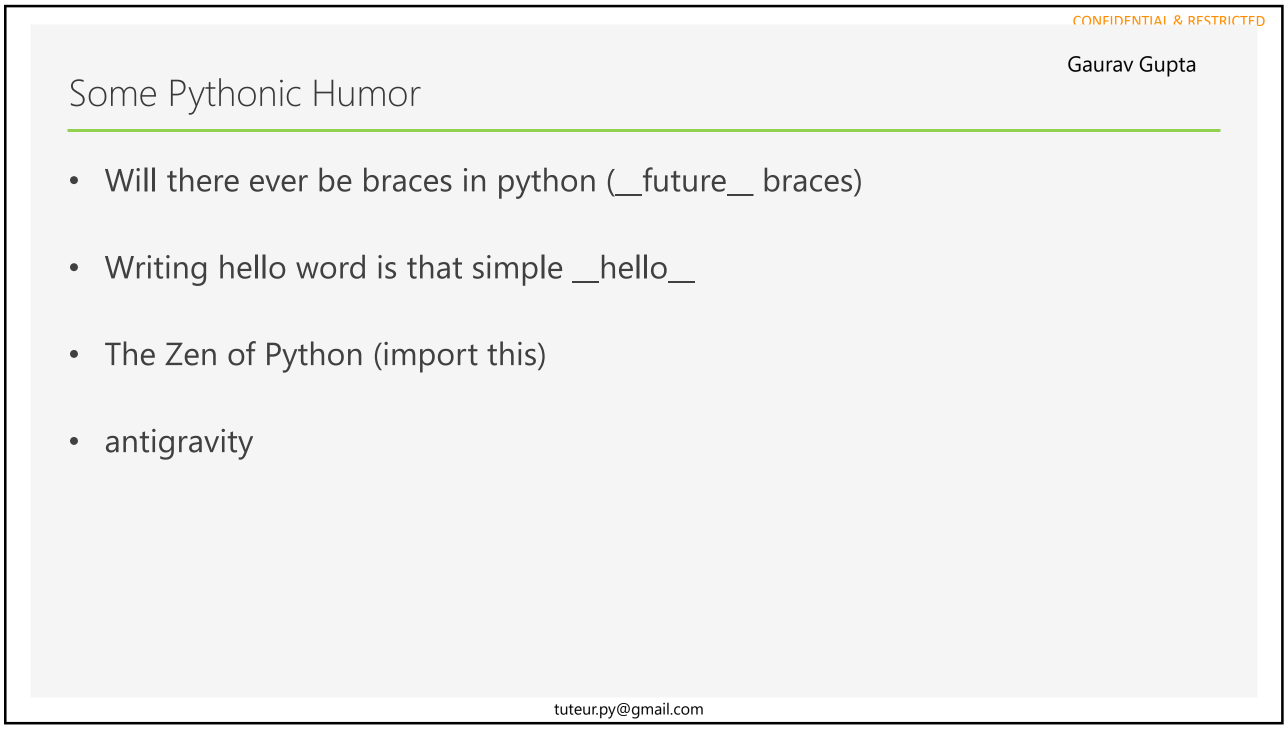


Gaurav Gupta

Practice

- Build a library my_lib.py add a few variables to test.
- Add functions to input data.
- Add the library to the python search path.

tuteur.py@gmail.com

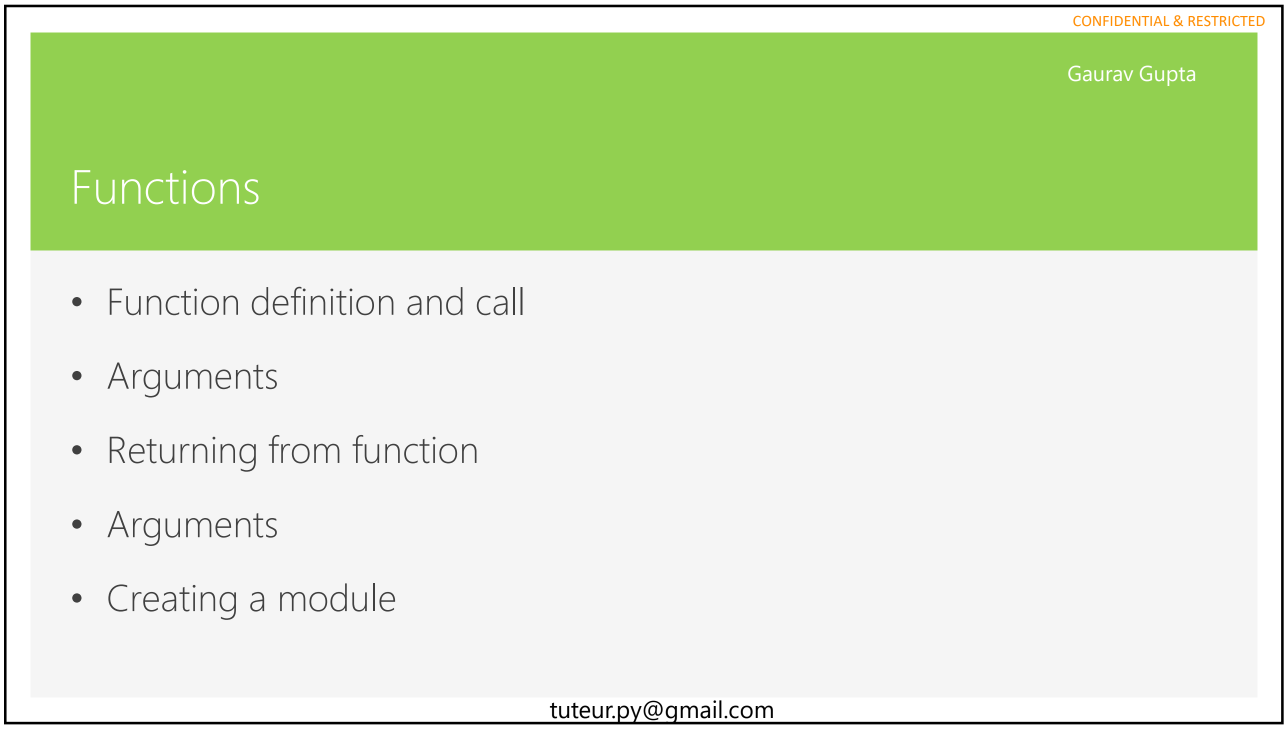


Gaurav Gupta

Some Pythonic Humor

- Will there ever be braces in python (__future__ braces)
- Writing hello word is that simple __hello__
- The Zen of Python (import this)
- antigravity

tuteur.py@gmail.com



Gaurav Gupta

Functions

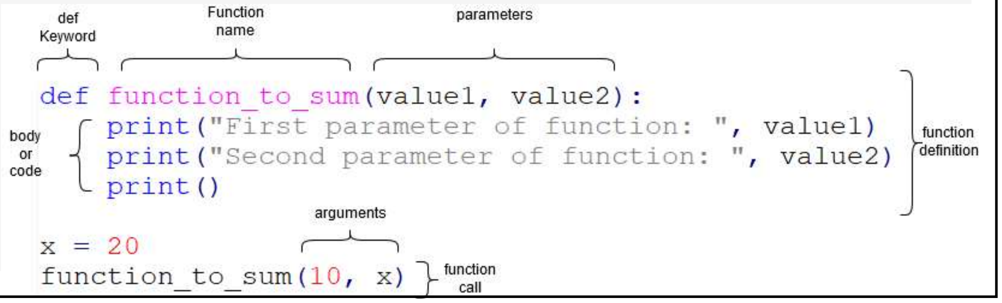
- Function definition and call
- Arguments
- Returning from function
- Arguments
- Creating a module

tuteur.py@gmail.com

Gaurav Gupta

Function Terminology

- **Parameter:** the variables specified in the bracket of a function definition / signature
- **Return value:** the value or variable written after **return** keyword in a function
- **Definition** the code written along with the def statement.
- **Argument** the value passed to a function at *function call*.
- **Function Call** the name of the function along with the arguments if any.



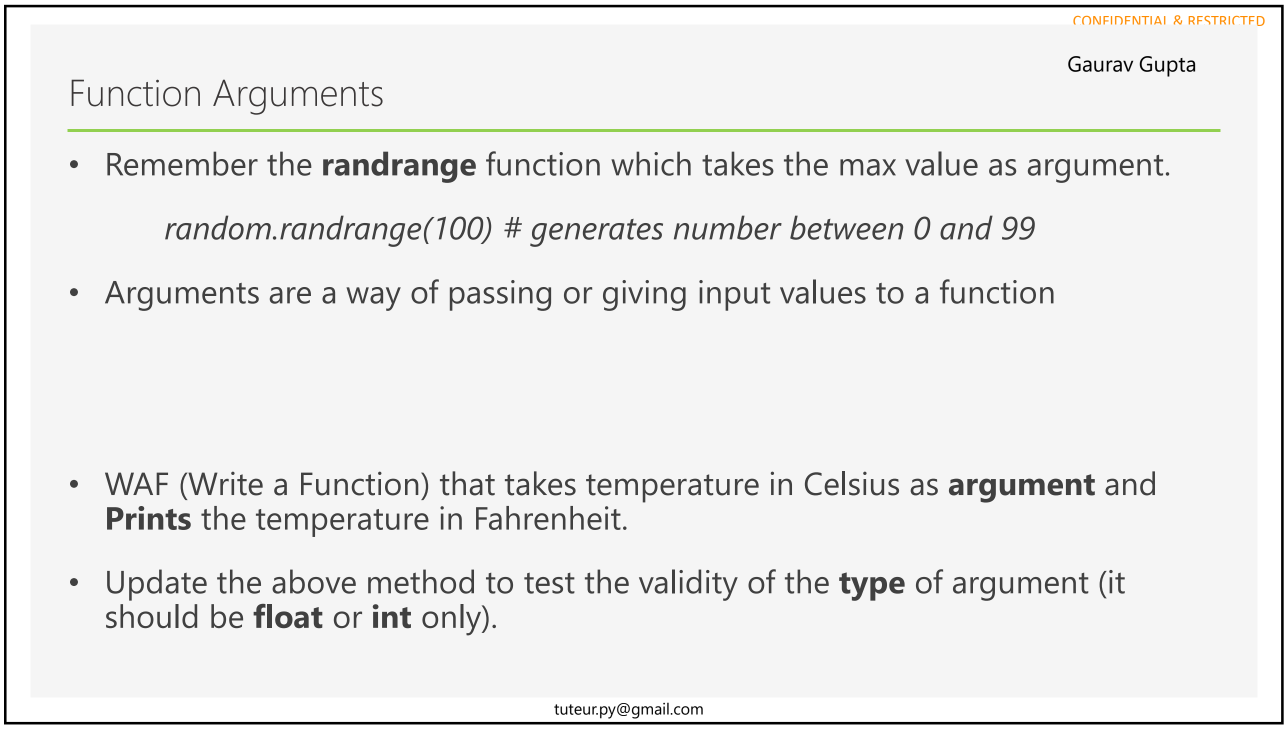
```
def function_to_sum(value1, value2):
    print("First parameter of function: ", value1)
    print("Second parameter of function: ", value2)
    print()

x = 20
function_to_sum(10, x)
```

Gaurav Gupta

Creating Functions

- Syntax:
  def *<function name>(**arguments**):*
  """   optional doc string   """
  # body/logic/code of function
- ***Def*** keyword is used to start a function
- Function may or may not **return** a **value**; depends on the use of ***return*** keyword
- Function gets executed only when it is **called/invoked**
- WAF that **inputs** temperature in Celsius and **Prints** it in Fahrenheit

tuteur.py@gmail.com



Gaurav Gupta

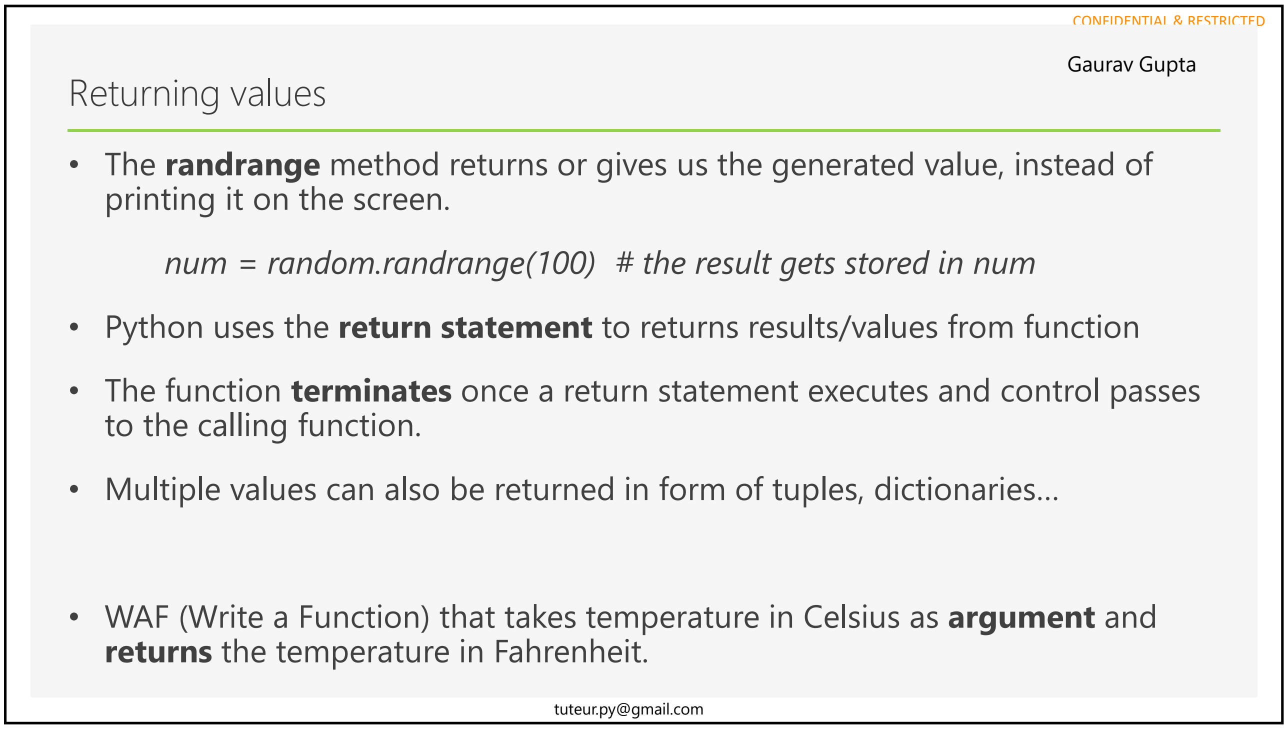
Function Arguments

- Remember the **randrange** function which takes the max value as argument.

  random.randrange(100) # generates number between 0 and 99

- Arguments are a way of passing or giving input values to a function
- WAF (Write a Function) that takes temperature in Celsius as **argument** and **Prints** the temperature in Fahrenheit.
- Update the above method to test the validity of the **type** of argument (it should be **float** or **int** only).

tuteur.py@gmail.com



Gaurav Gupta

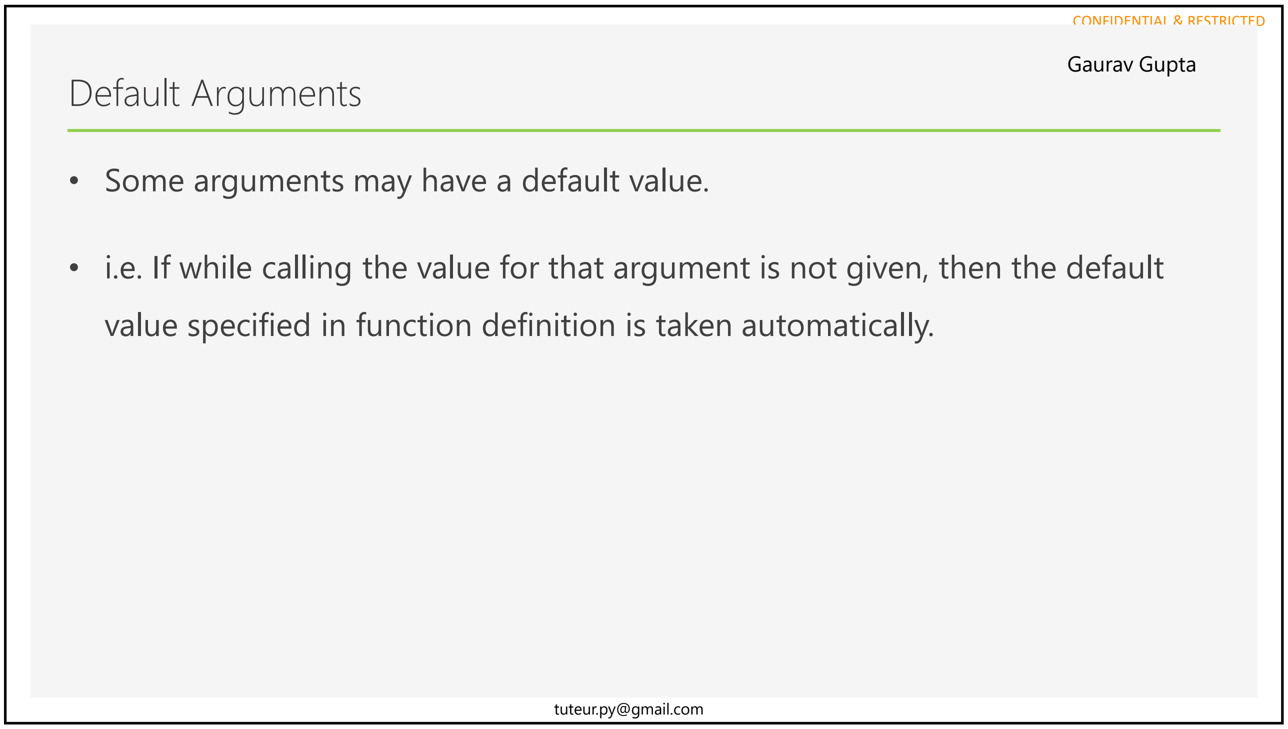
Returning values

- The **randrange** method returns or gives us the generated value, instead of printing it on the screen.

  num = random.randrange(100) # the result gets stored in num

- Python uses the **return statement** to returns results/values from function
- The function **terminates** once a return statement executes and control passes to the calling function.
- Multiple values can also be returned in form of tuples, dictionaries…
- WAF (Write a Function) that takes temperature in Celsius as **argument** and **returns** the temperature in Fahrenheit.

tuteur.py@gmail.com

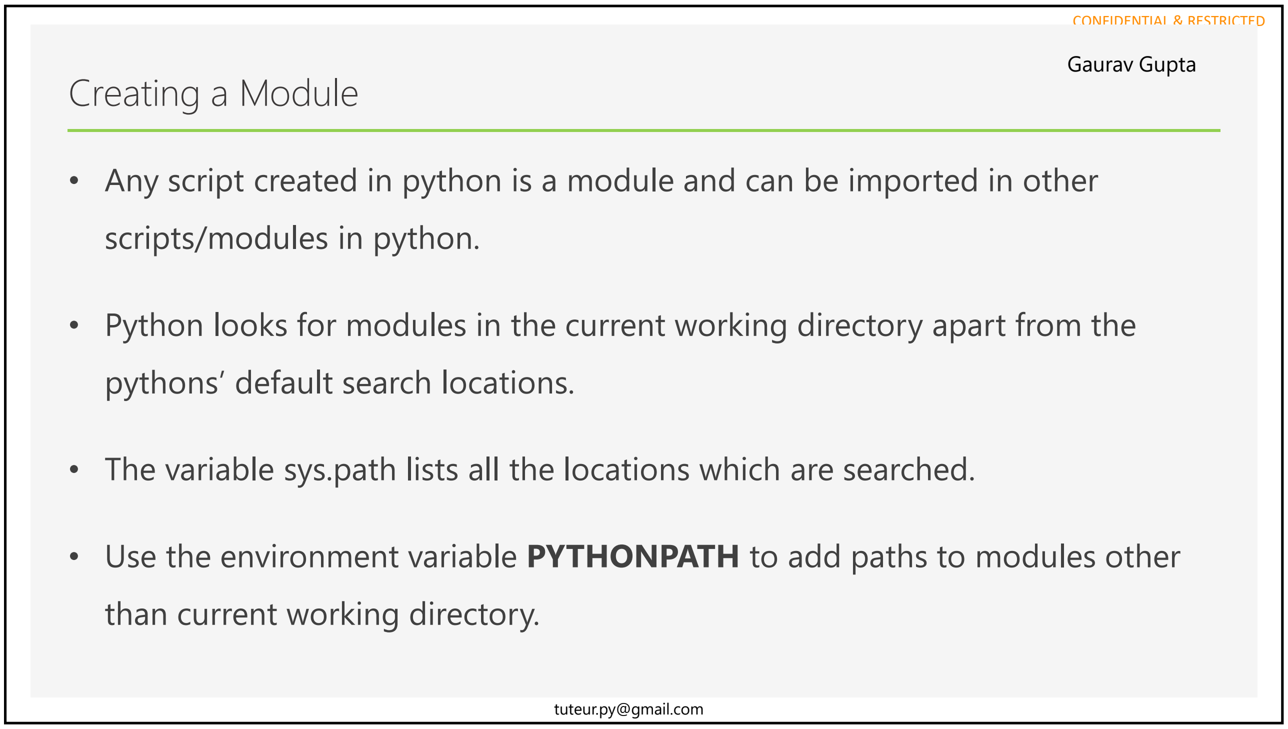


Gaurav Gupta

Default Arguments

- Some arguments may have a default value.
- i.e. If while calling the value for that argument is not given, then the default value specified in function definition is taken automatically.

tuteur.py@gmail.com

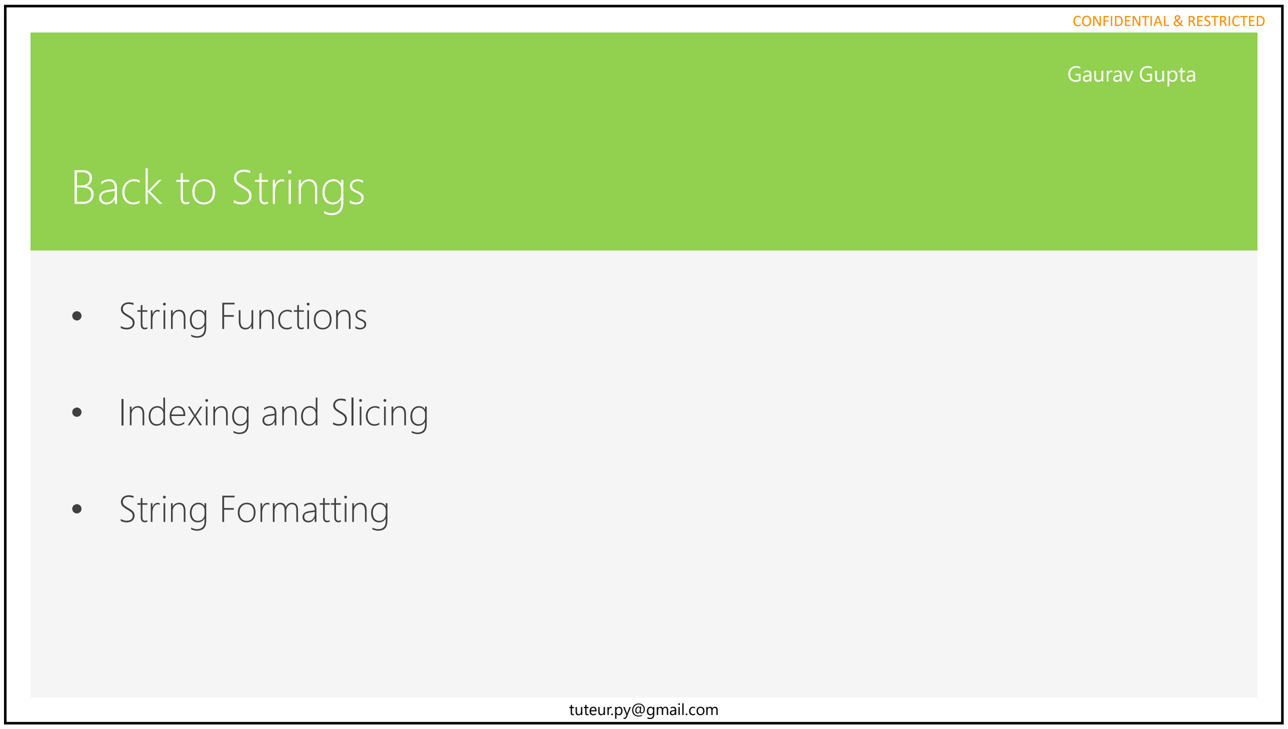


Gaurav Gupta

Creating a Module

- Any script created in python is a module and can be imported in other scripts/modules in python.
- Python looks for modules in the current working directory apart from the pythons' default search locations.
- The variable sys.path lists all the locations which are searched.
- Use the environment variable **PYTHONPATH** to add paths to modules other than current working directory.

tuteur.py@gmail.com

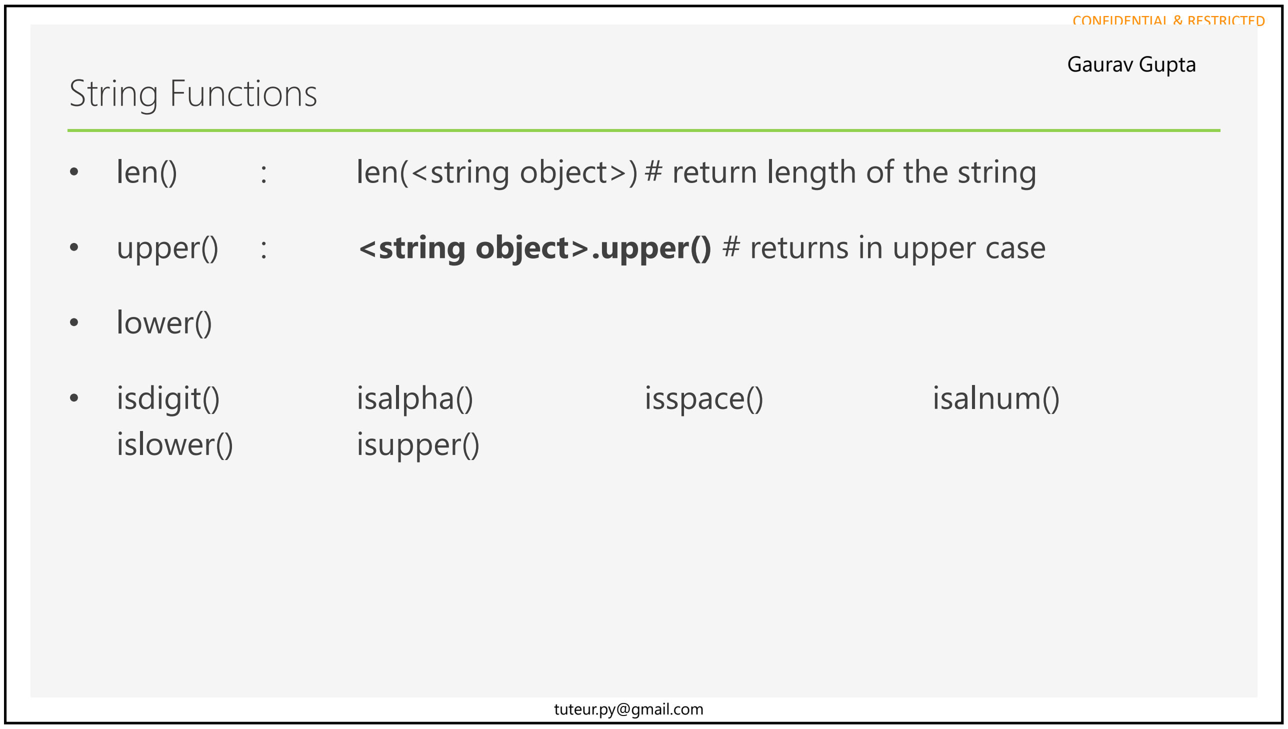


Gaurav Gupta

Back to Strings

- String Functions
- Indexing and Slicing
- String Formatting

tuteur.py@gmail.com

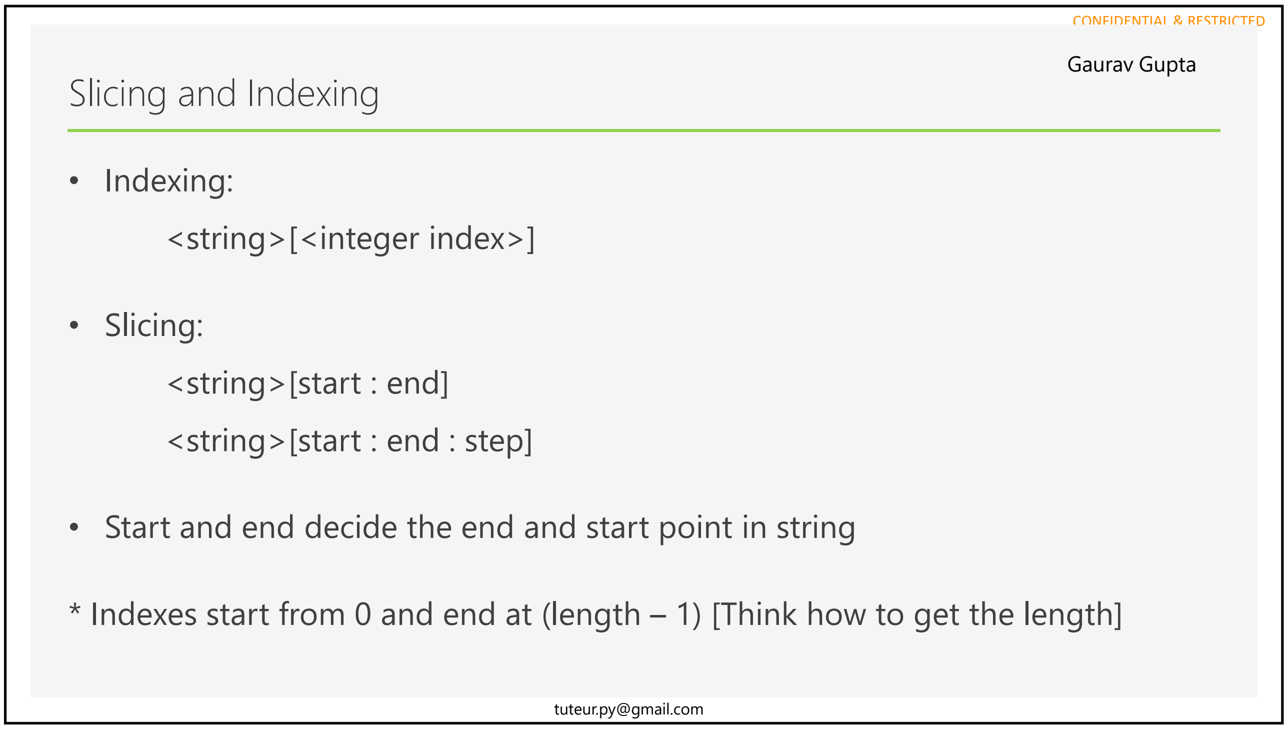


Gaurav Gupta

String Functions

- len() : len(<string object>) # return length of the string
- upper() : **<string object>.upper()** # returns in upper case
- lower()
- isdigit() isalpha() isspace() isalnum() islower() isupper()

tuteur.py@gmail.com



Gaurav Gupta

Slicing and Indexing

- Indexing:

    <string>[<integer index>]

- Slicing:

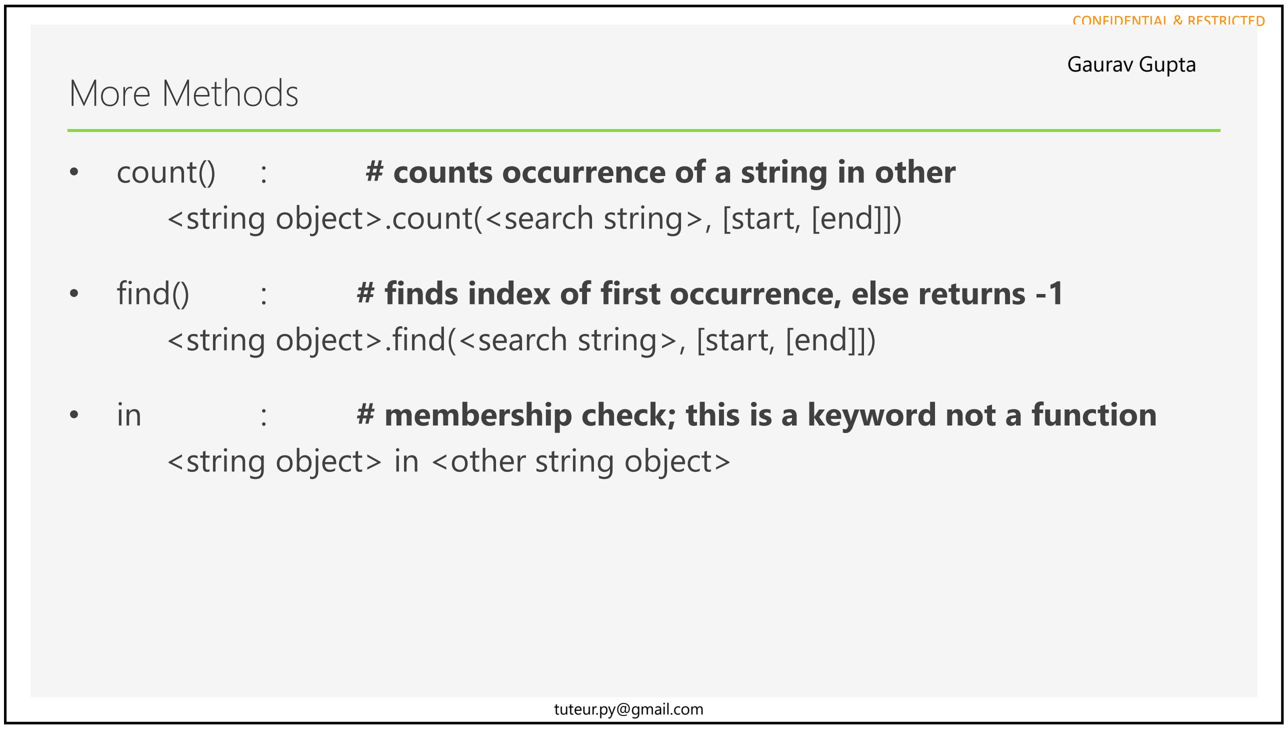
    <string>[start : end]

    <string>[start : end : step]

- Start and end decide the end and start point in string

* Indexes start from 0 and end at (length – 1) [Think how to get the length]

tuteur.py@gmail.com



Gaurav Gupta

More Methods

- count() : **# counts occurrence of a string in other**

    <string object>.count(<search string>, [start, [end]])

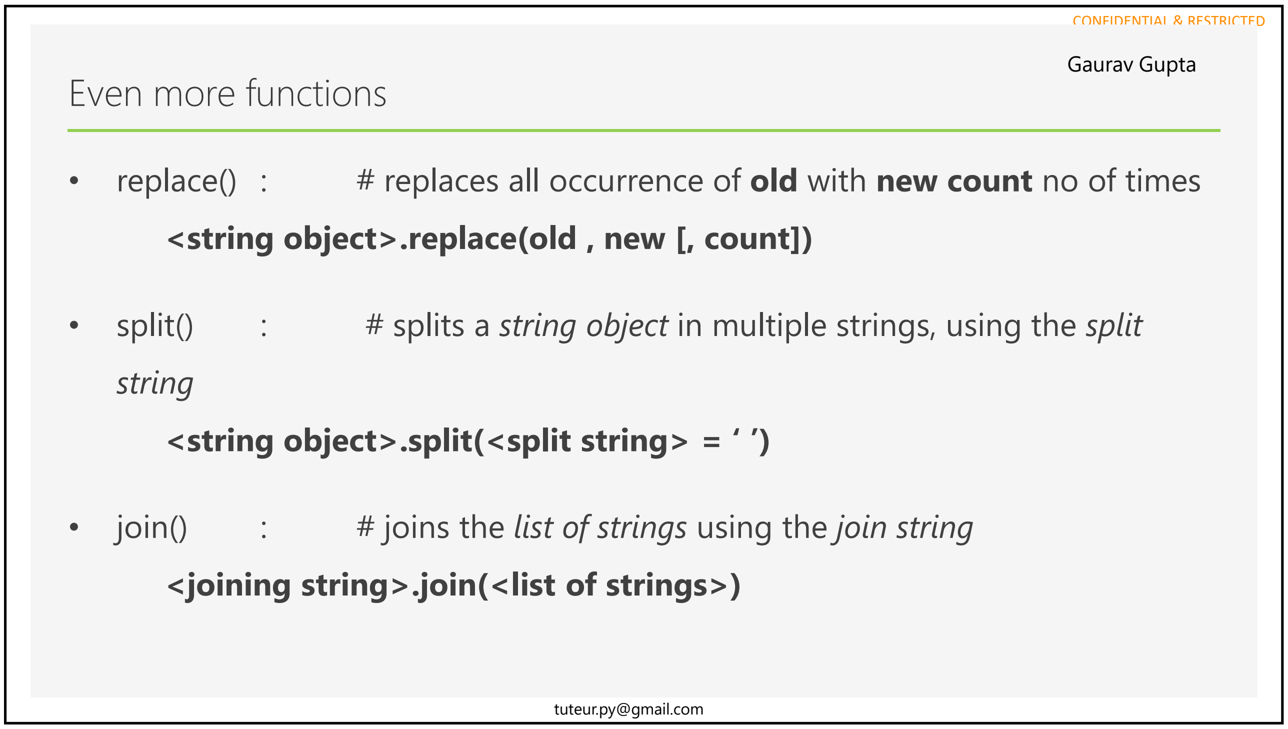
- find() : **# finds index of first occurrence, else returns -1**

    <string object>.find(<search string>, [start, [end]])

- in : **# membership check; this is a keyword not a function**

    <string object> in <other string object>

tuteur.py@gmail.com



Gaurav Gupta

Even more functions

- replace() : # replaces all occurrence of **old** with **new count** no of times

  <string object>.replace(old , new [, count])

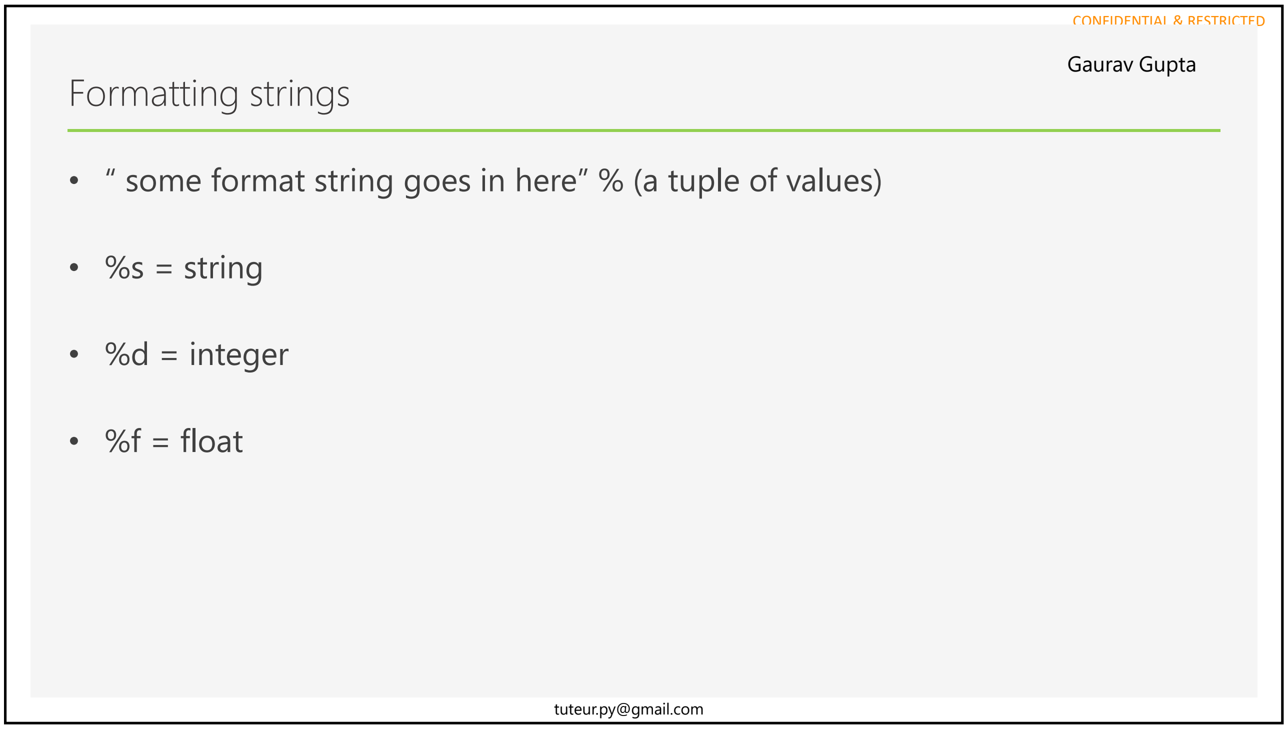
- split() : # splits a *string object* in multiple strings, using the *split string*

  <string object>.split(<split string> = ‘ ’)

- join() : # joins the *list of strings* using the *join string*

  <joining string>.join(<list of strings>)

tuteur.py@gmail.com



Gaurav Gupta

Formatting strings

- “ some format string goes in here” % (a tuple of values)
- %s = string
- %d = integer
- %f = float

tuteur.py@gmail.com