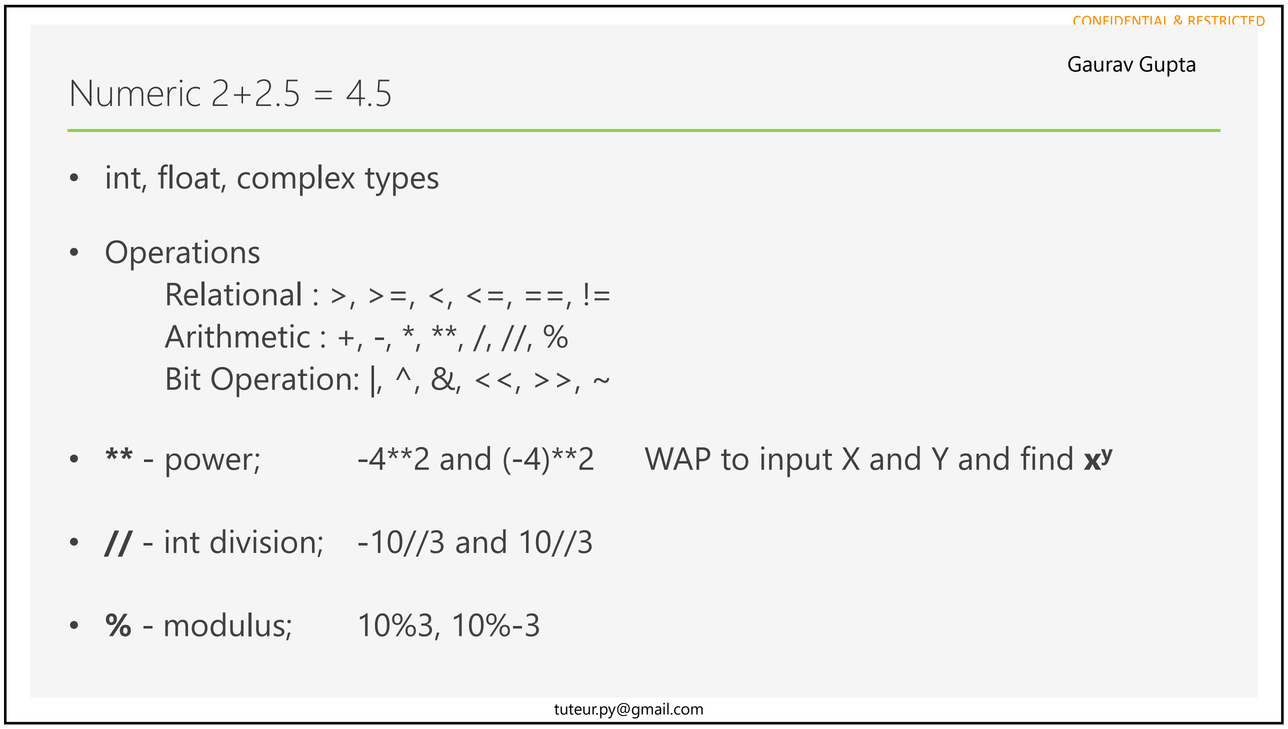
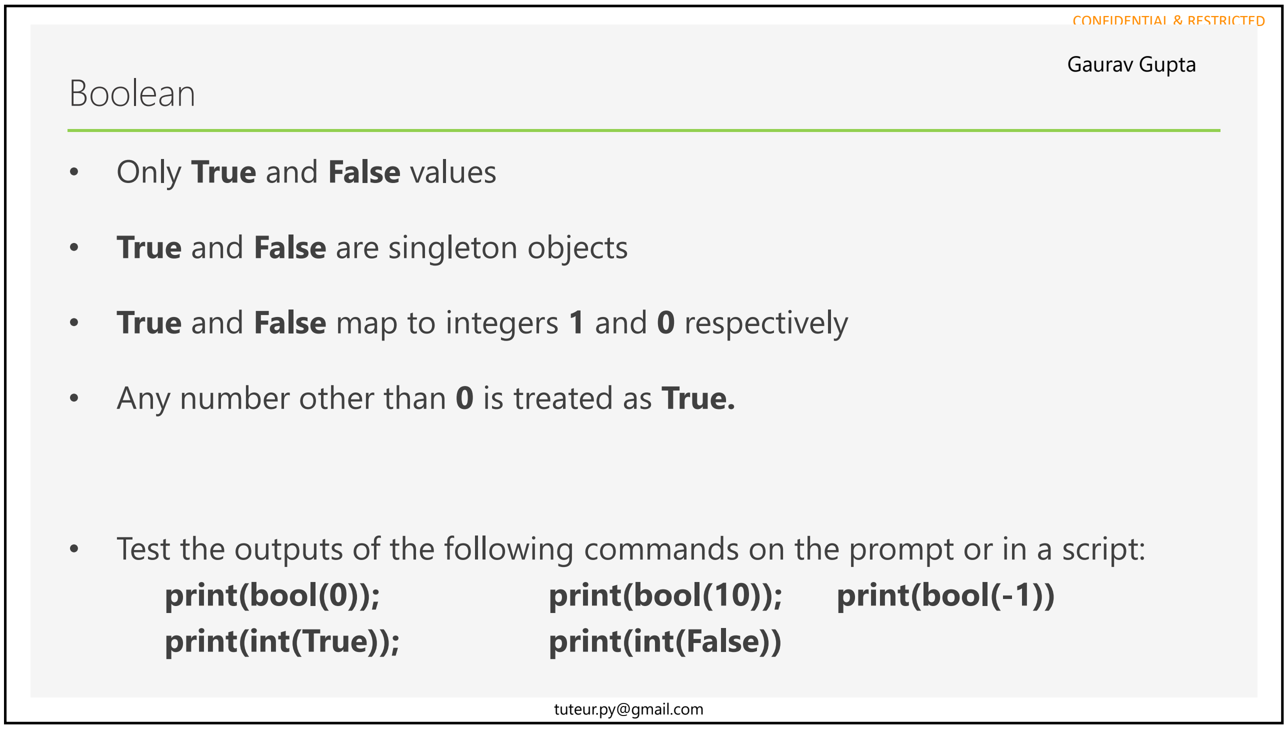


Gaurav Gupta

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 **x**$^{\mathbf{y}}$
- **//** - int division; -10//3 and 10//3
- **%** - modulus; 10%3, 10%-3

tuteur.py@gmail.com



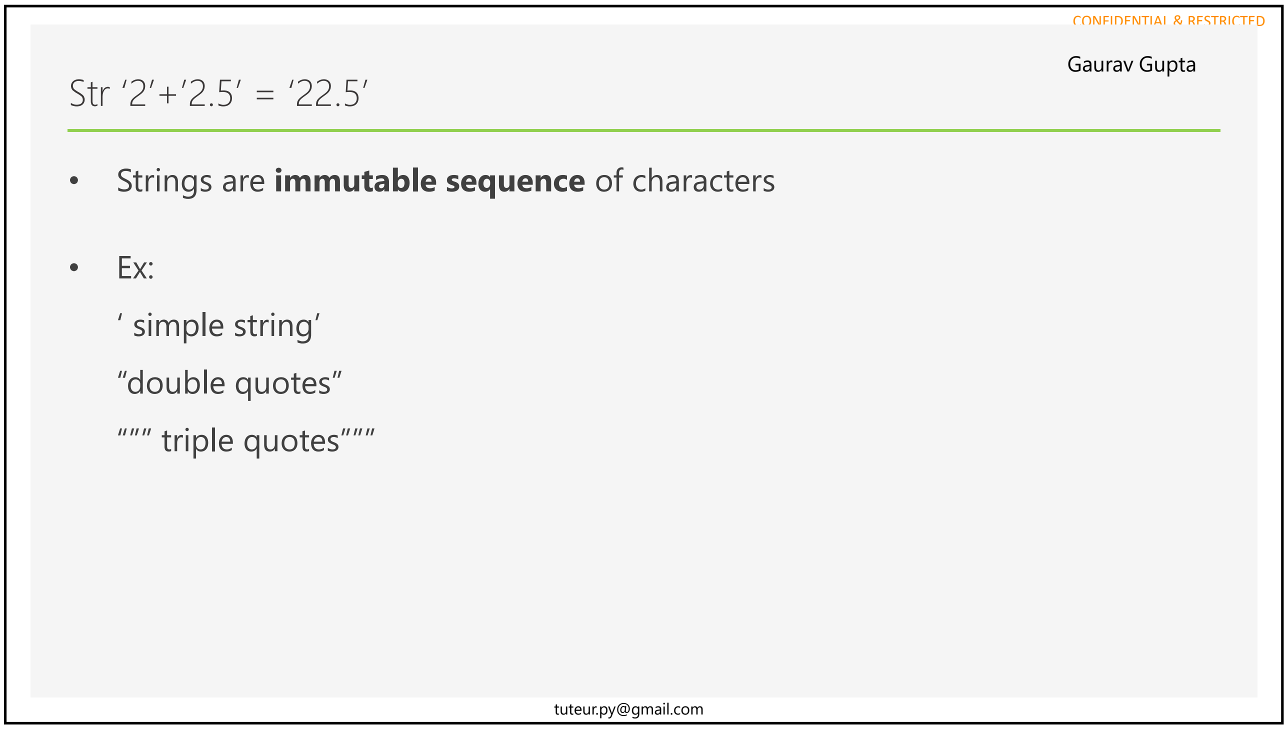
Gaurav Gupta

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'

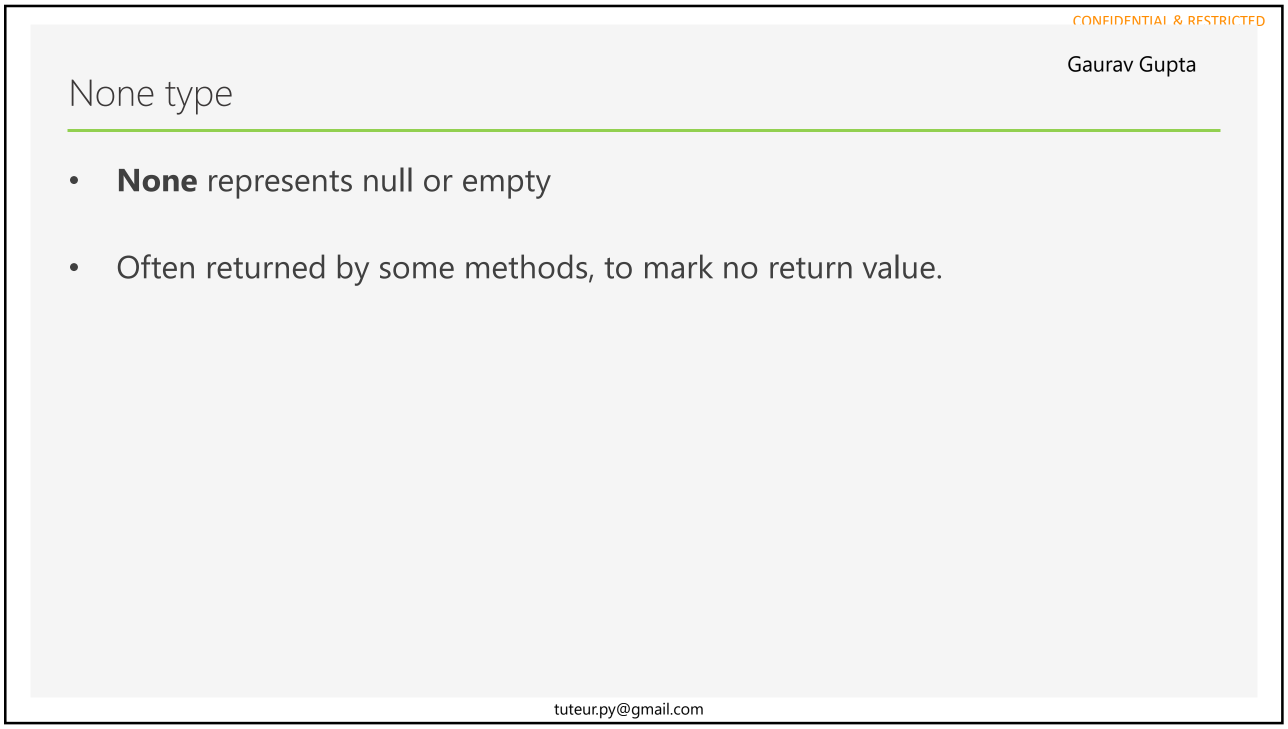
- Strings are **immutable sequence** of characters
- Ex:

  ' 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

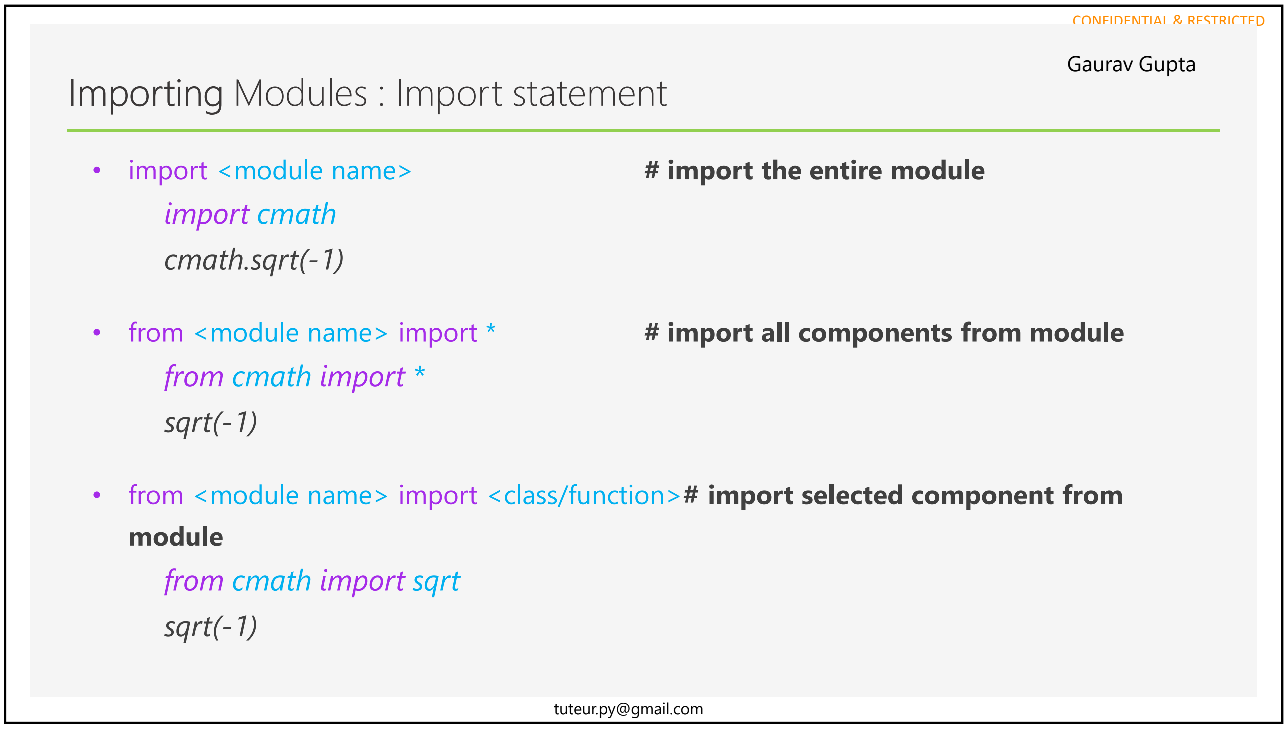
tuteur.py@gmail.com

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 **

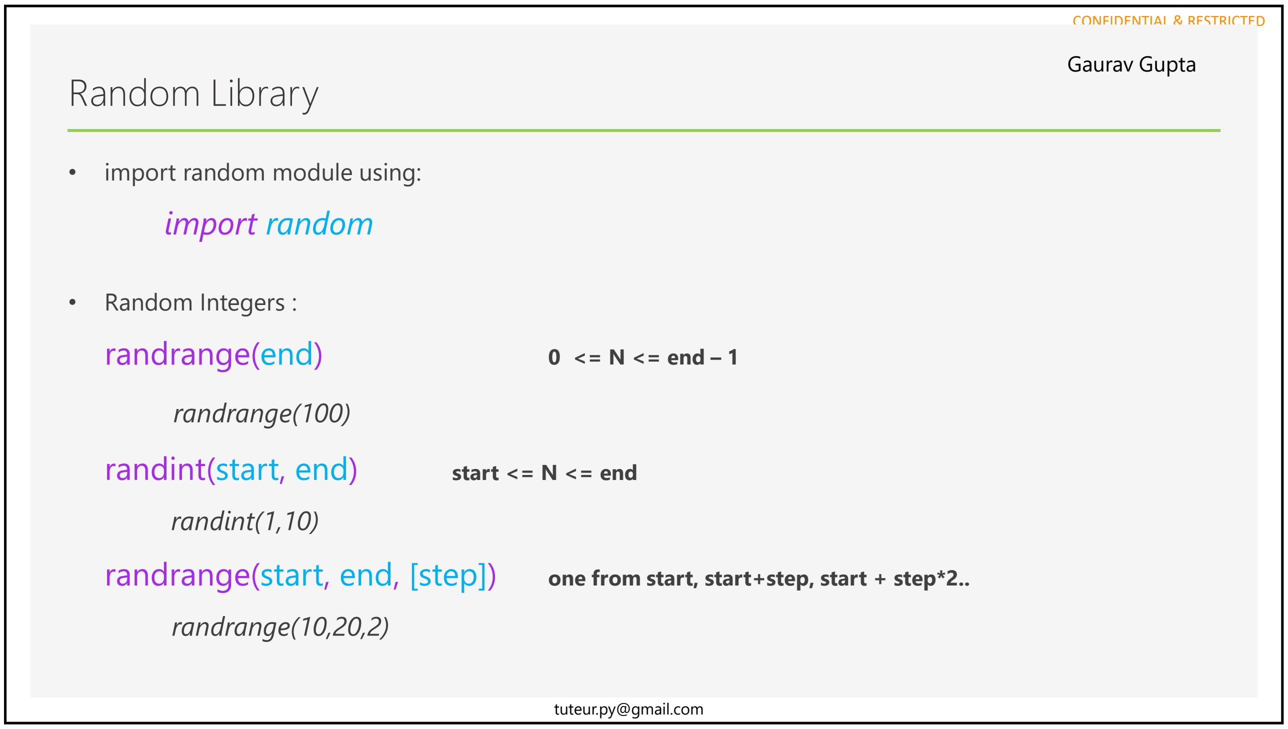
  sqrt(-1)

- 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)

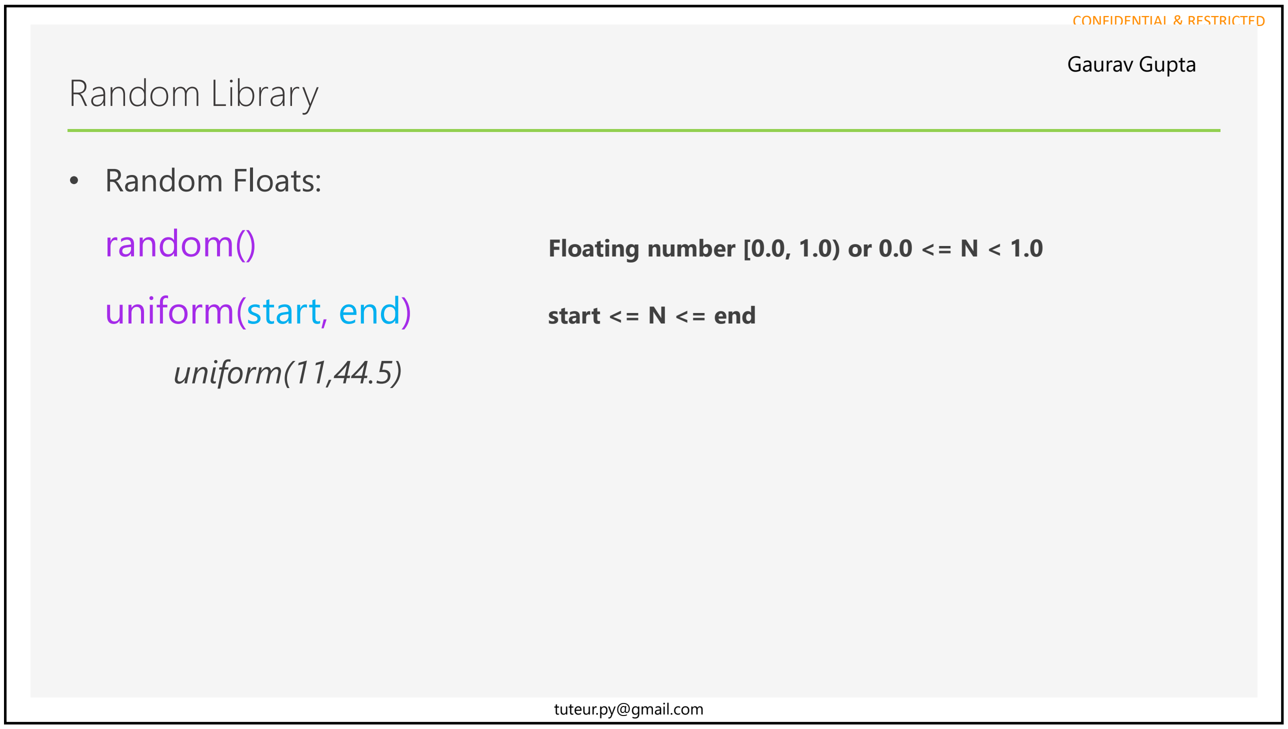
  randint(start, end) **start <= N <= end**

  randint(1,10)

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

  randrange(10,20,2)

tuteur.py@gmail.com



Gaurav Gupta

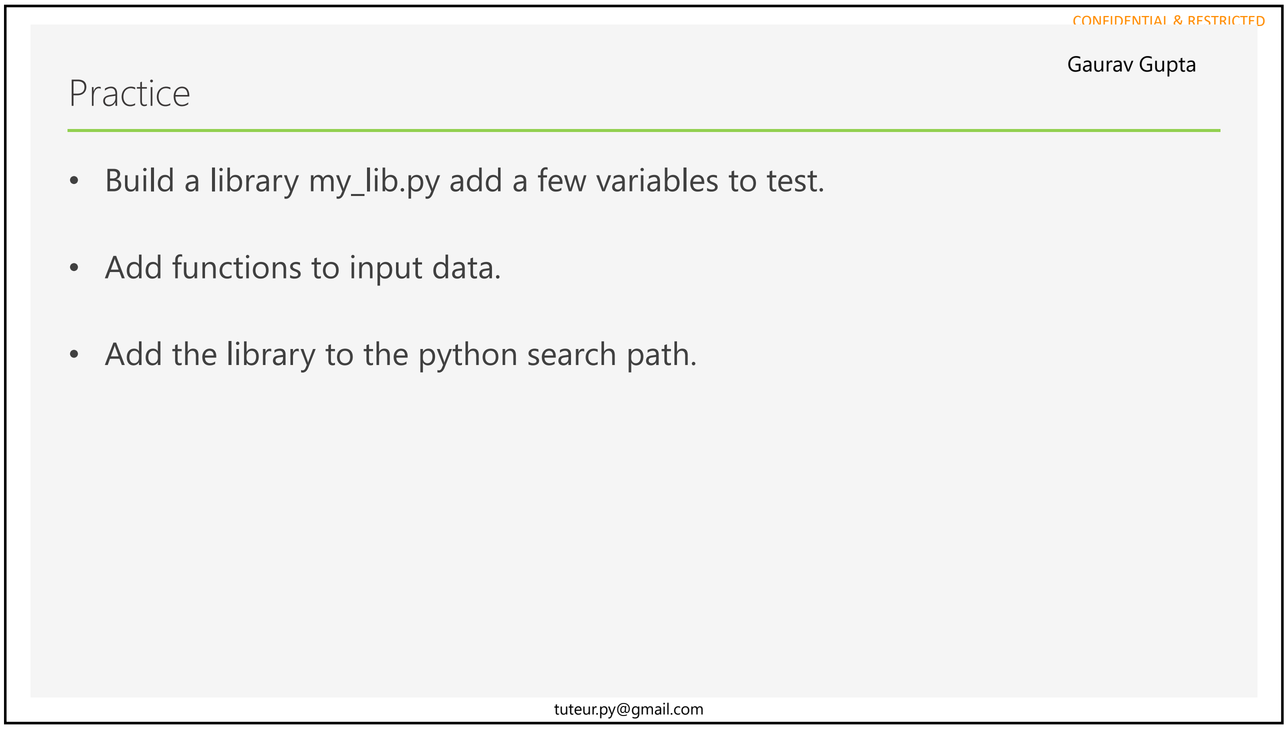
Random Library

- 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

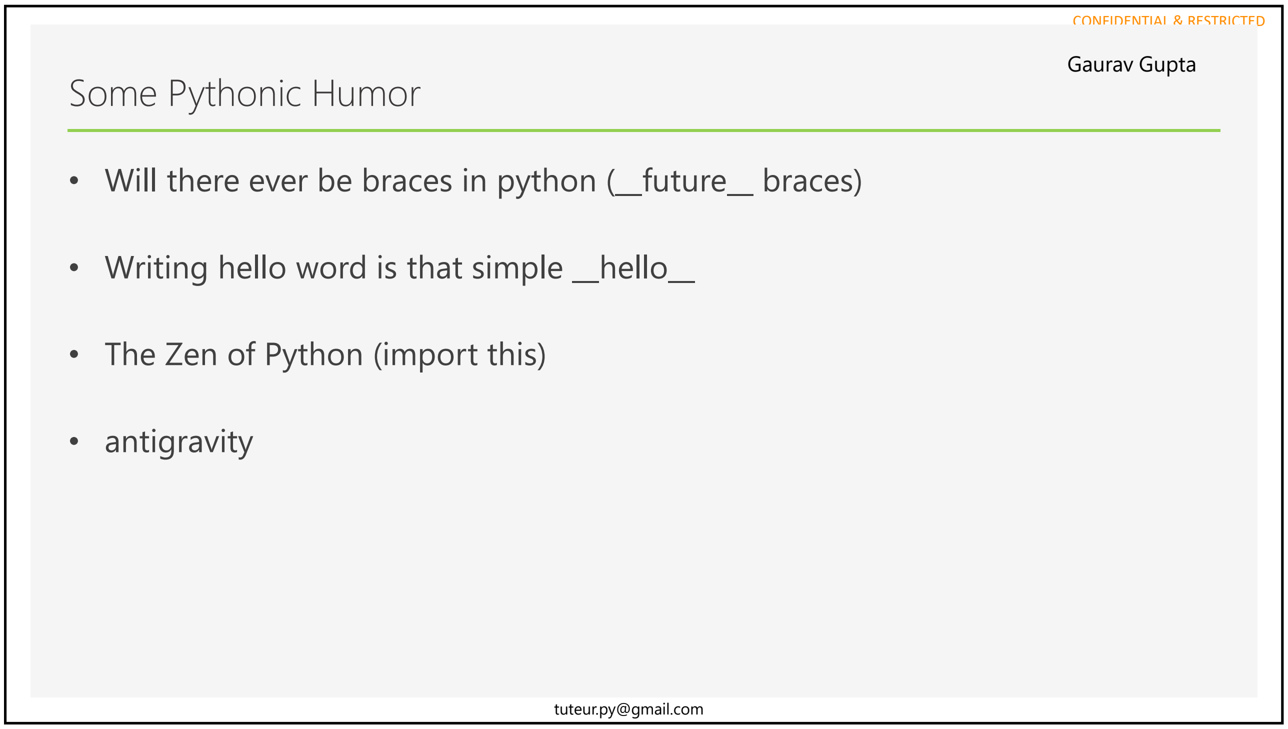


Gaurav Gupta

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



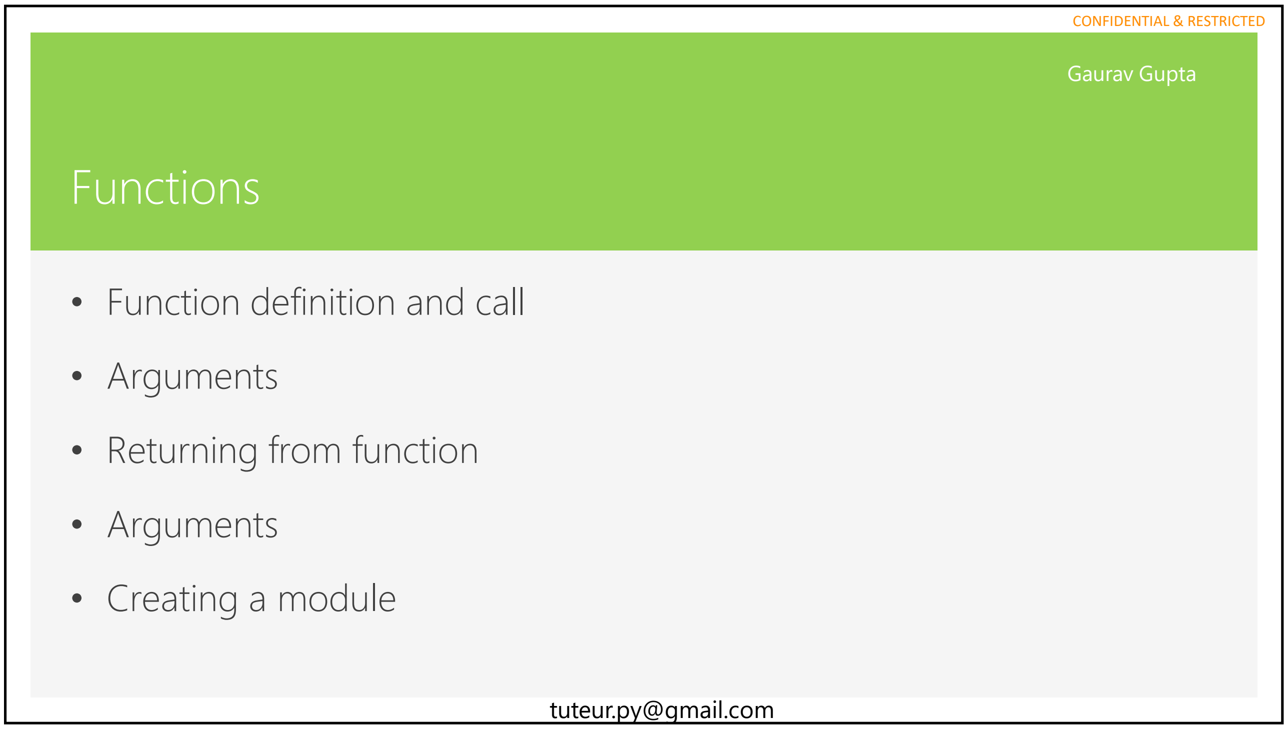
CONFIDENTIAL & RESTRICTED

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



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Functions

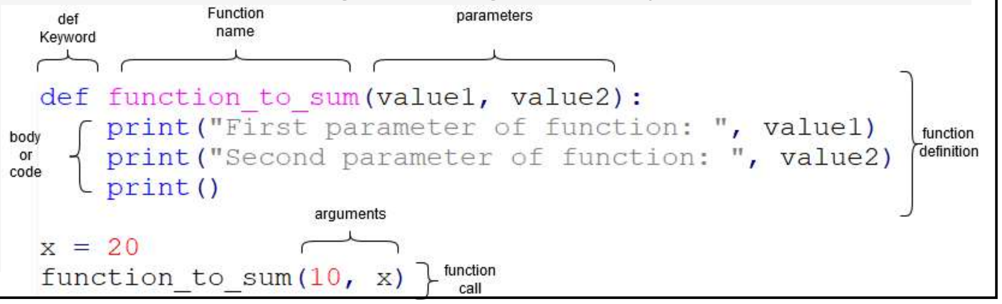
- Function definition and call
- Arguments
- Returning from function
- Arguments
- Creating a module

tuteur.py@gmail.com

Gaurav Gupta

Function Terminology

- **Parameter:** the variables specified in the bracket of a function definition
- **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

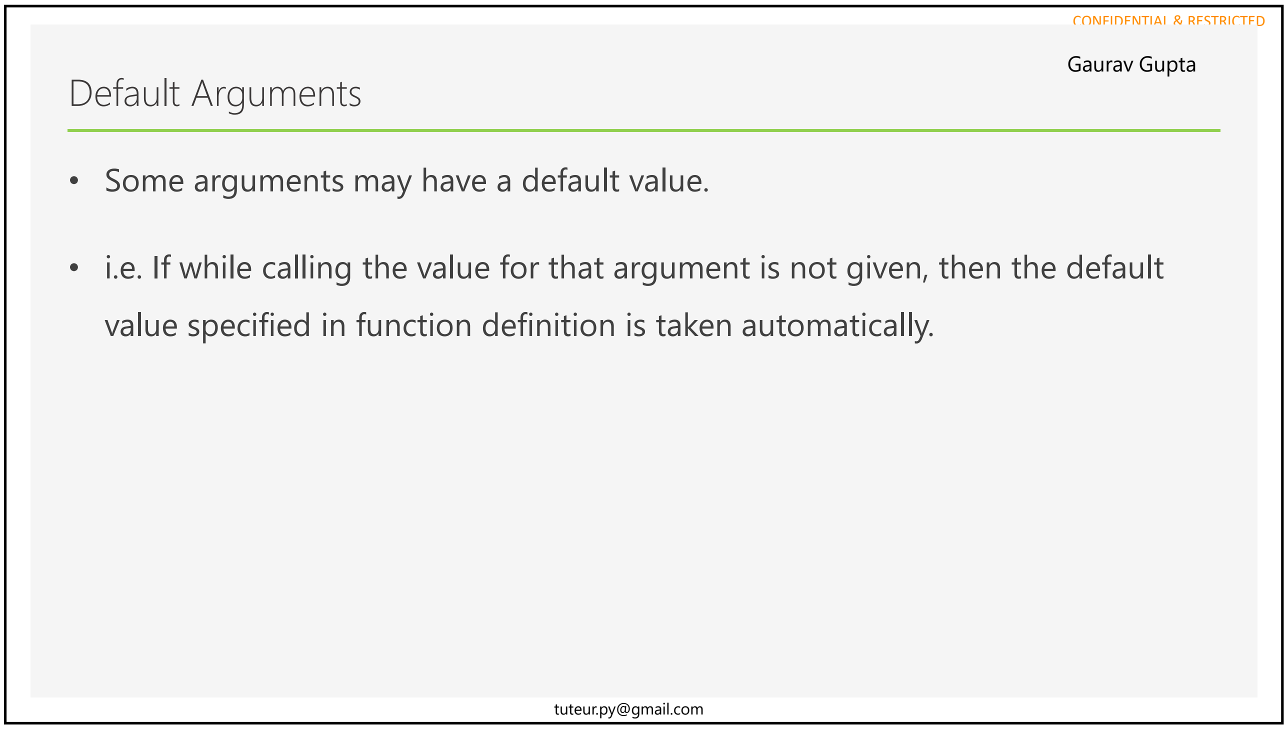
- The **randrange** method returns or gives us the generated value, instead of printing it on the screen.

  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.

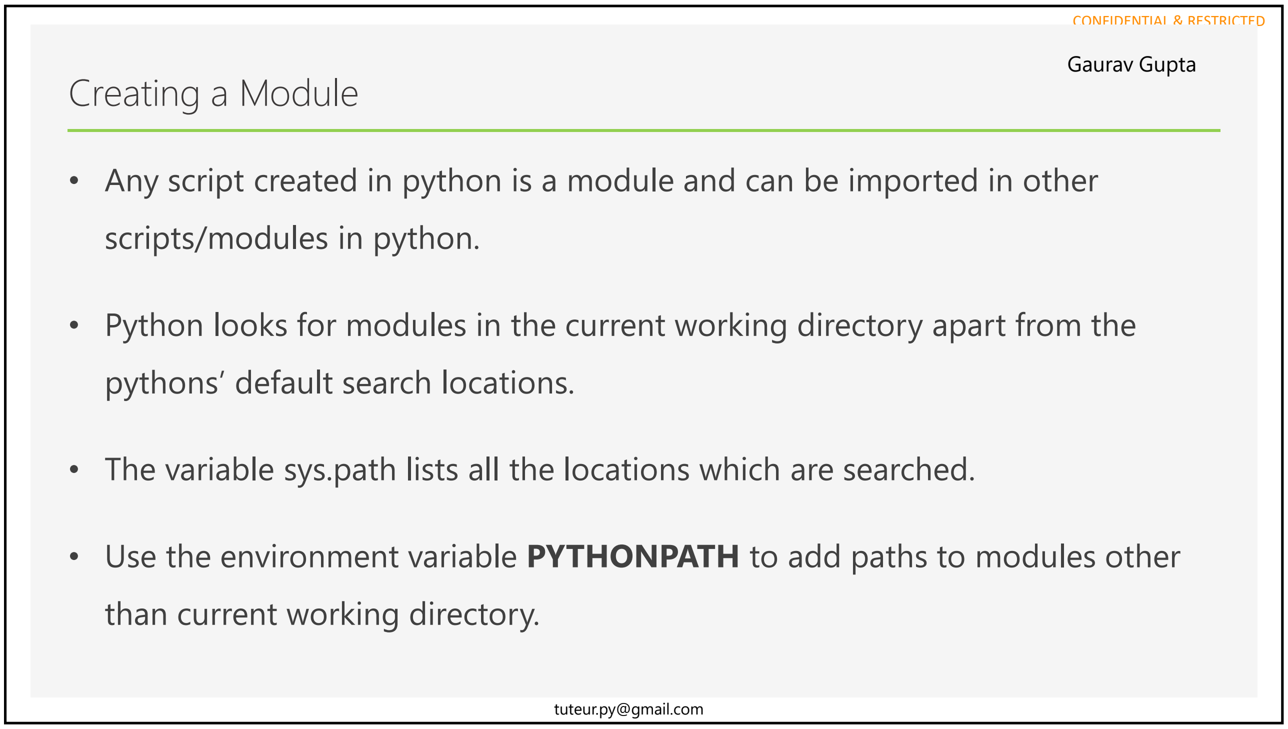
tuteur.py@gmail.com



Gaurav Gupta

Default Arguments

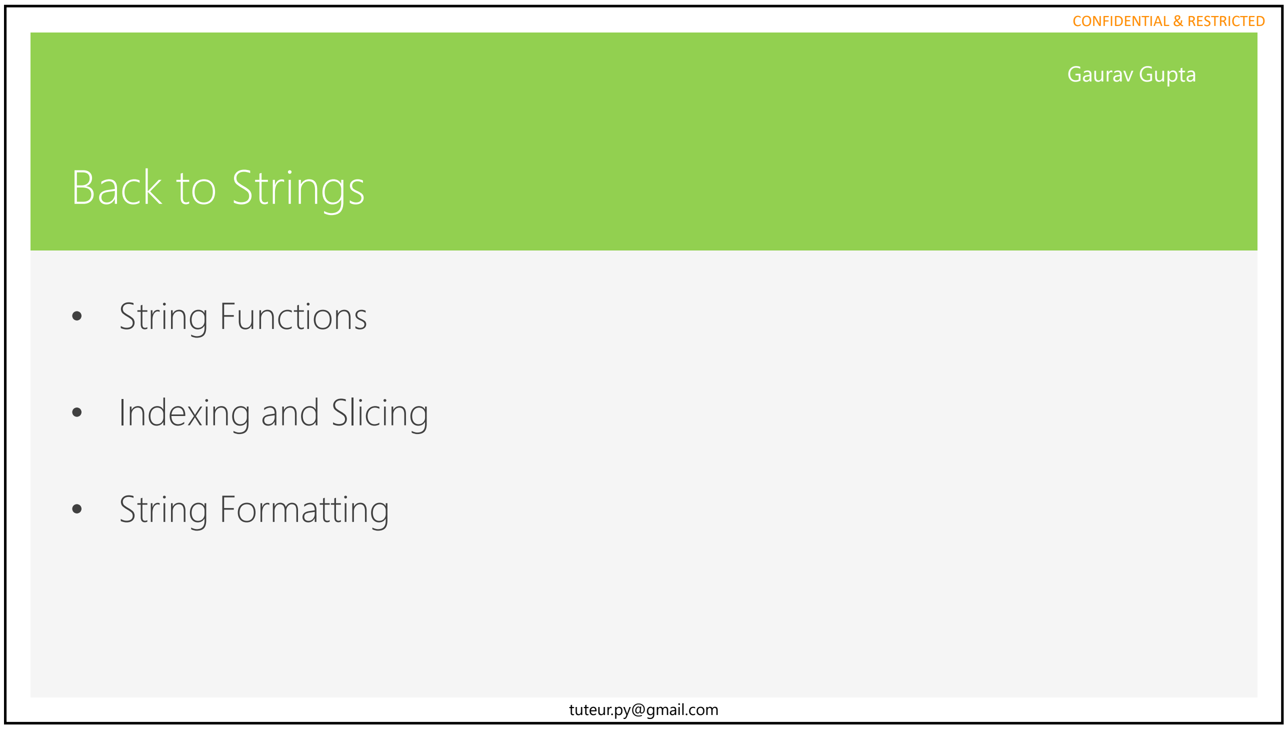
- Some arguments may have a default value.
- i.e. If while calling the value for that argument is not given, then the default value specified in function definition is taken automatically.



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.

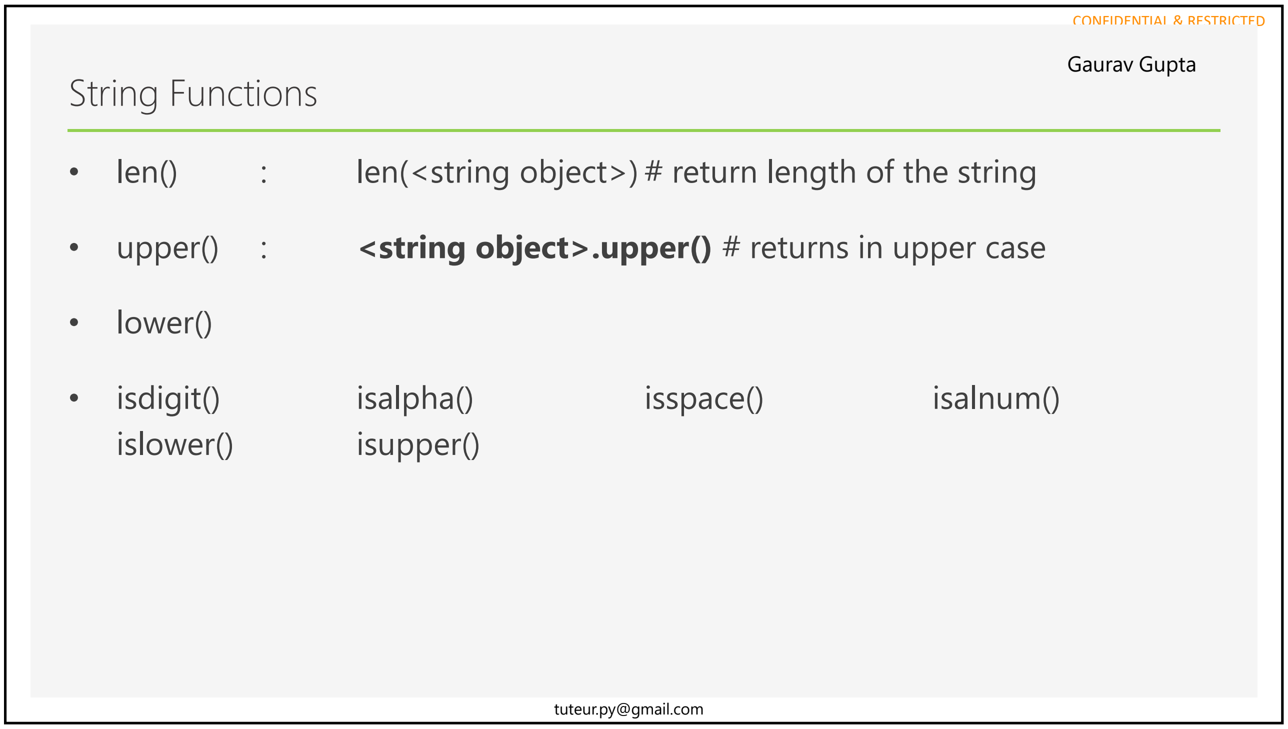


Gaurav Gupta

Back to Strings

- String Functions
- Indexing and Slicing
- String Formatting

tuteur.py@gmail.com

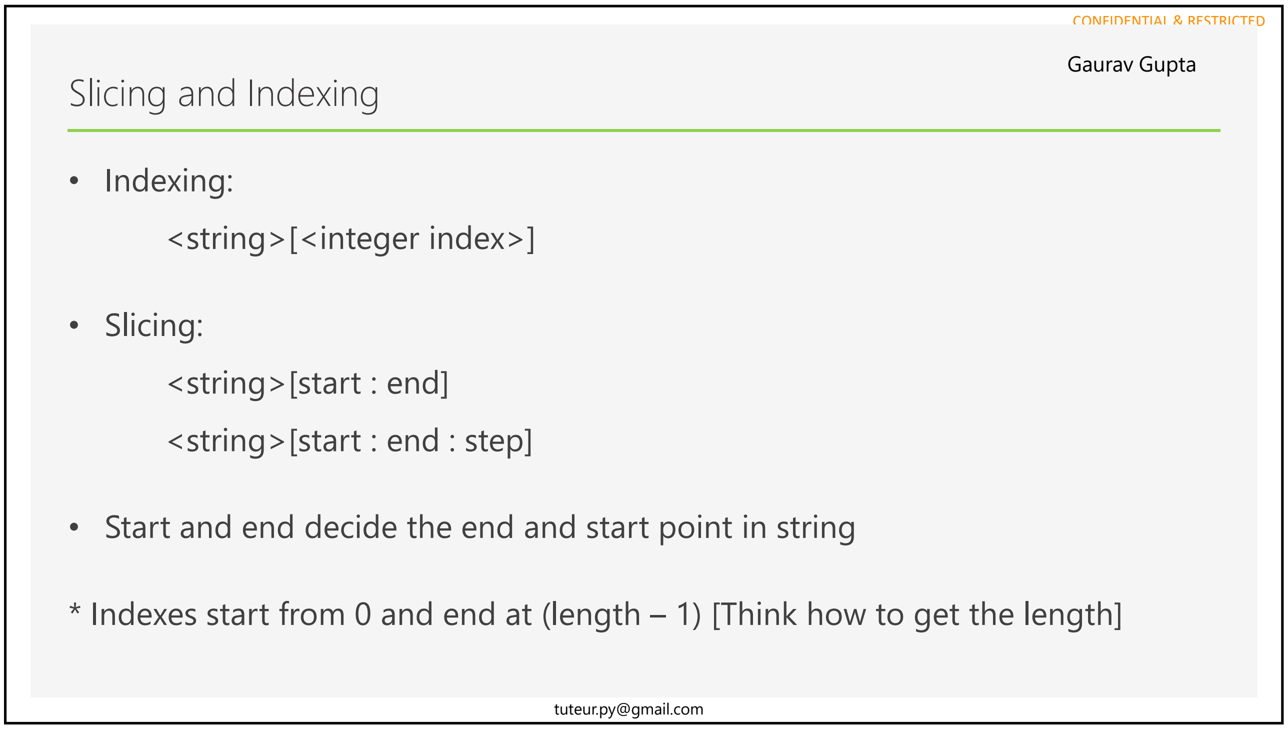


Gaurav Gupta

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:

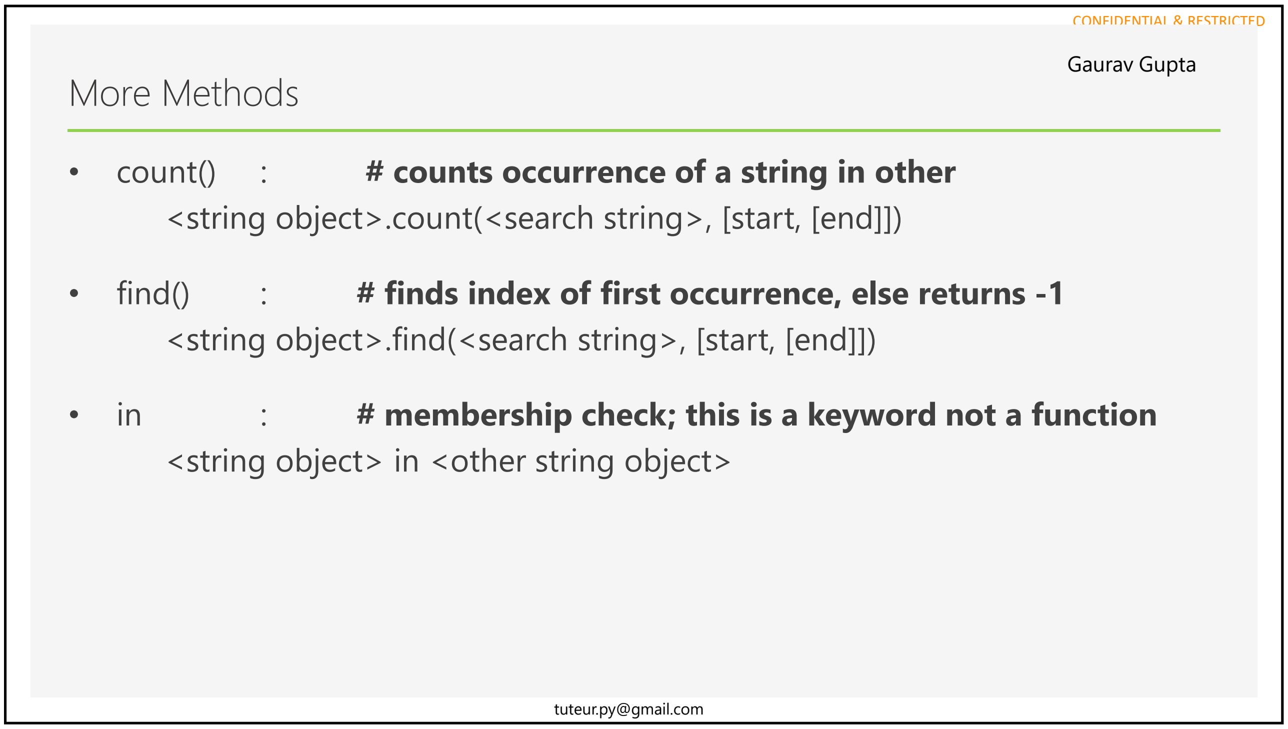
    <string>[start : end]

    <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]])

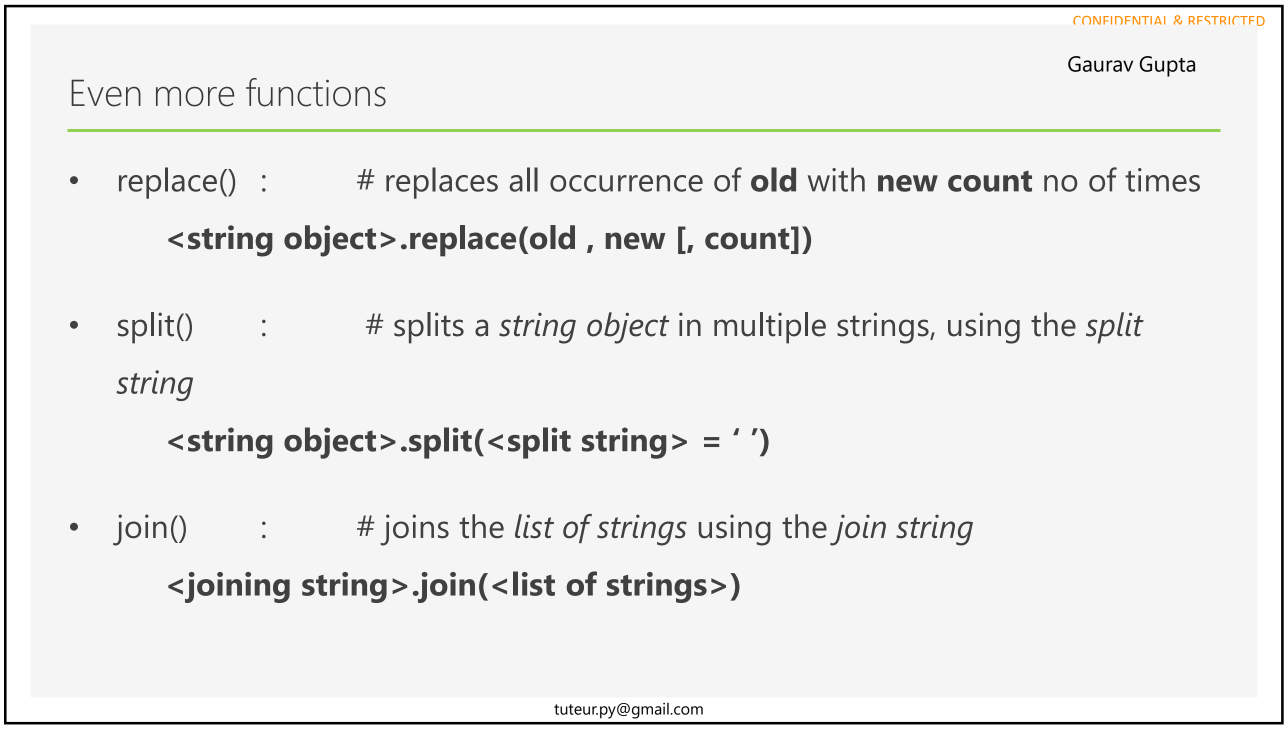
- find() : **# finds index of first occurrence, else returns -1**

    <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])

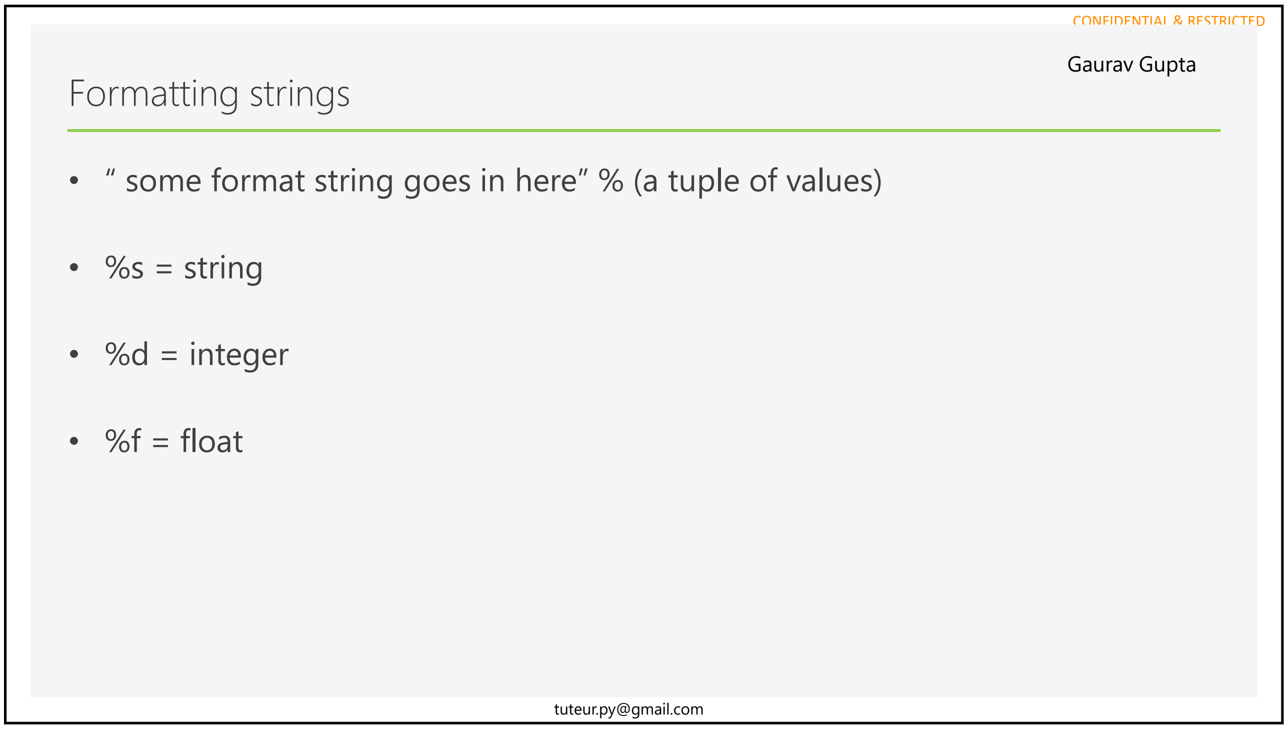
- split() : # splits a *string object* in multiple strings, using the *split string*

  <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