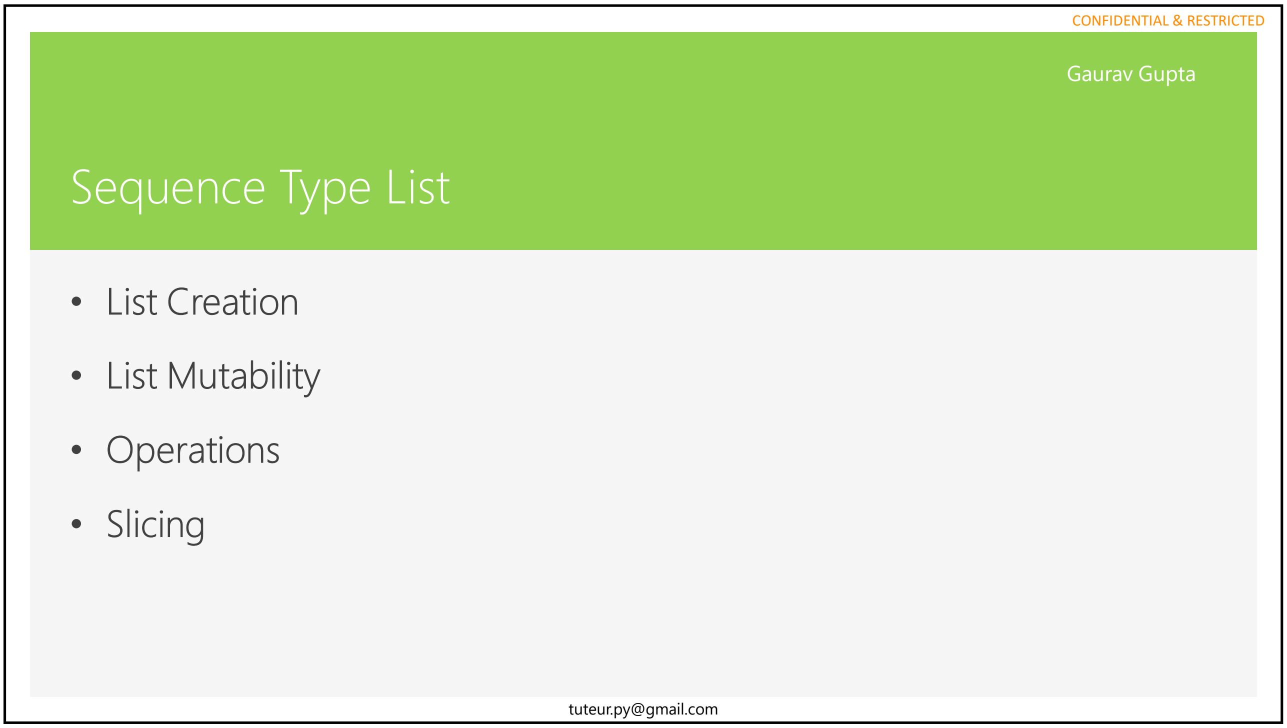


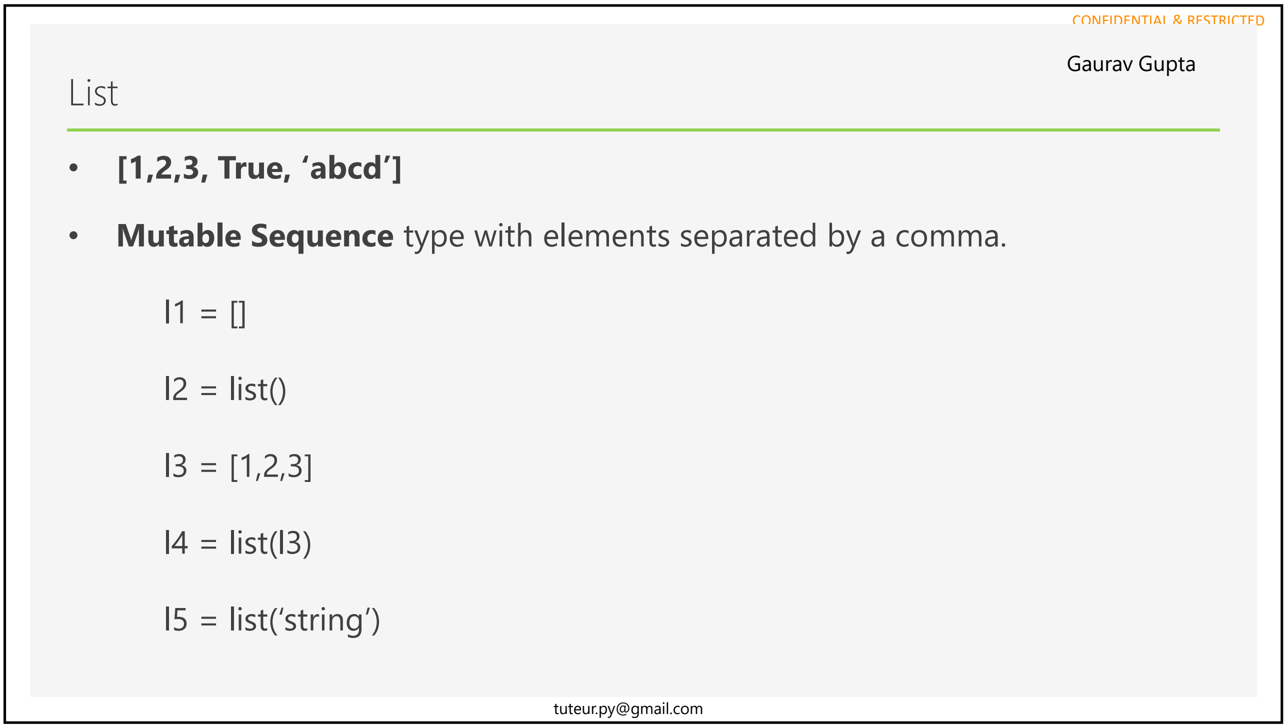
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Sequence Type List

- List Creation
- List Mutability
- Operations
- Slicing

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

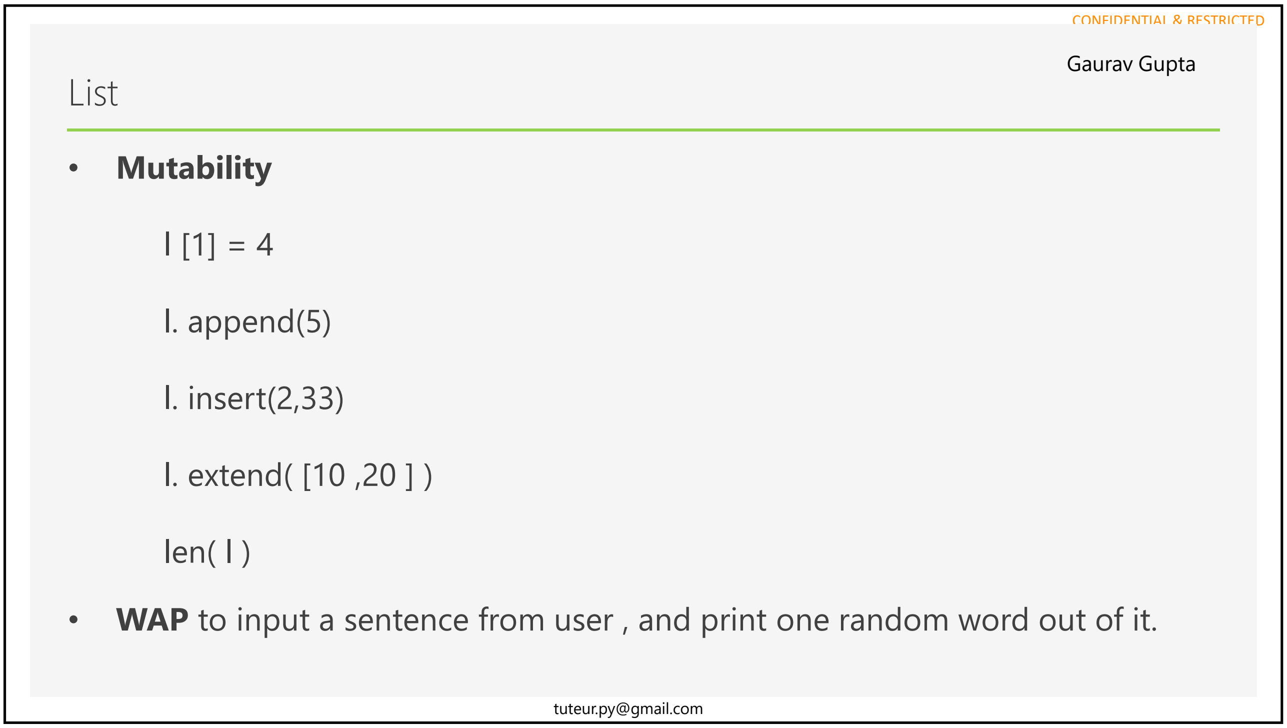
Gaurav Gupta

List

- **[1,2,3, True, ‘abcd’]**
- **Mutable Sequence** type with elements separated by a comma.

```
l1 = []

l2 = list()

l3 = [1,2,3]

l4 = list(l3)

l5 = list(‘string’)
```

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

List

- **Mutability**

  l [1] = 4

  l. append(5)

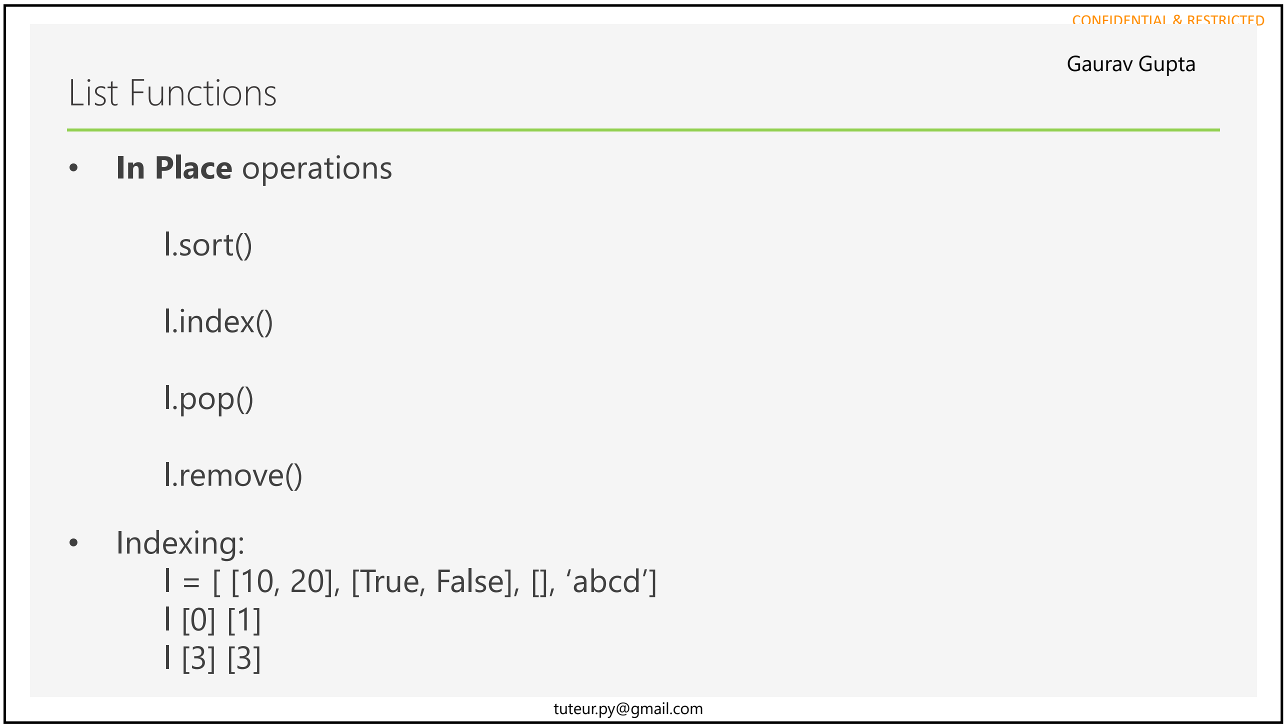
  l. insert(2,33)

  l. extend([10 ,20])

  len(l)

- **WAP** to input a sentence from user , and print one random word out of it.

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

List Functions

- **In Place** operations

  l.sort()

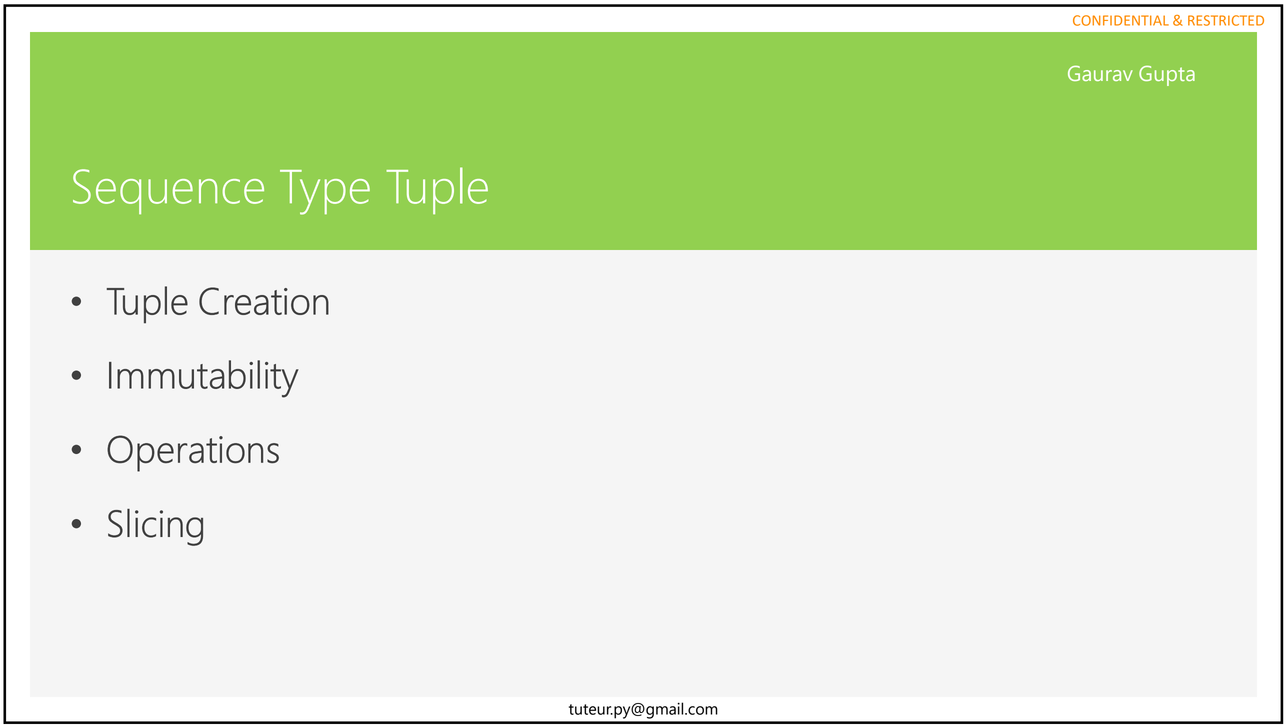
  l.index()

  l.pop()

  l.remove()

- Indexing:
  l = [[10, 20], [True, False], [], ‘abcd’]
  l [0] [1]
  l [3] [3]

tuteur.py@gmail.com



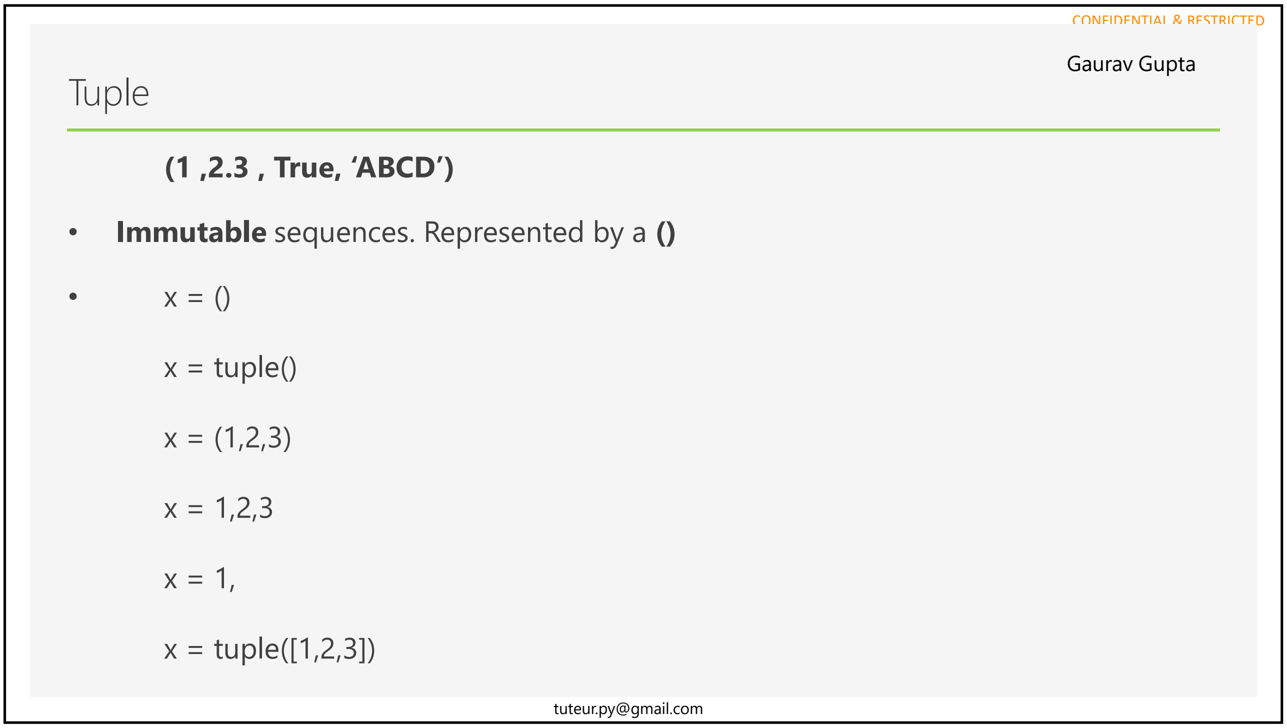
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Sequence Type Tuple

- Tuple Creation
- Immutability
- Operations
- Slicing

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

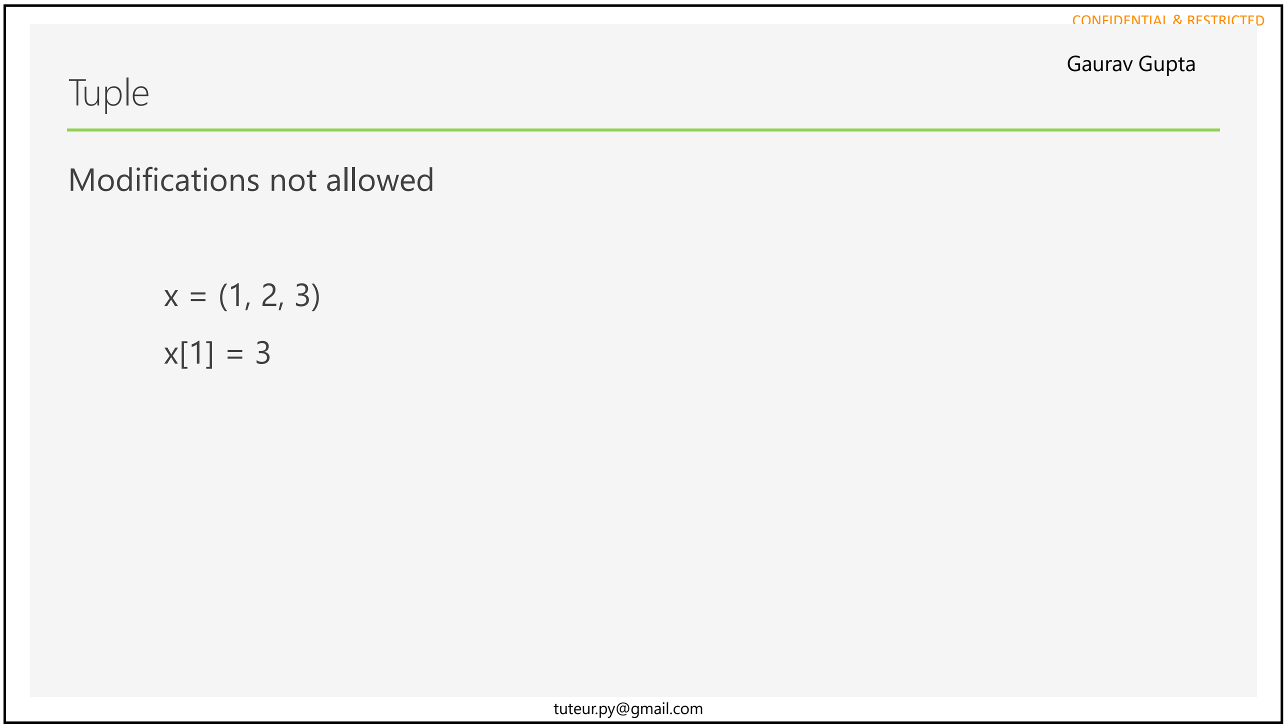
Tuple

(1 ,2.3 , True, 'ABCD')

- **Immutable** sequences. Represented by a **()**
-

```
x = ()
x = tuple()
x = (1,2,3)
x = 1,2,3
x = 1,
x = tuple([1,2,3])
```

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

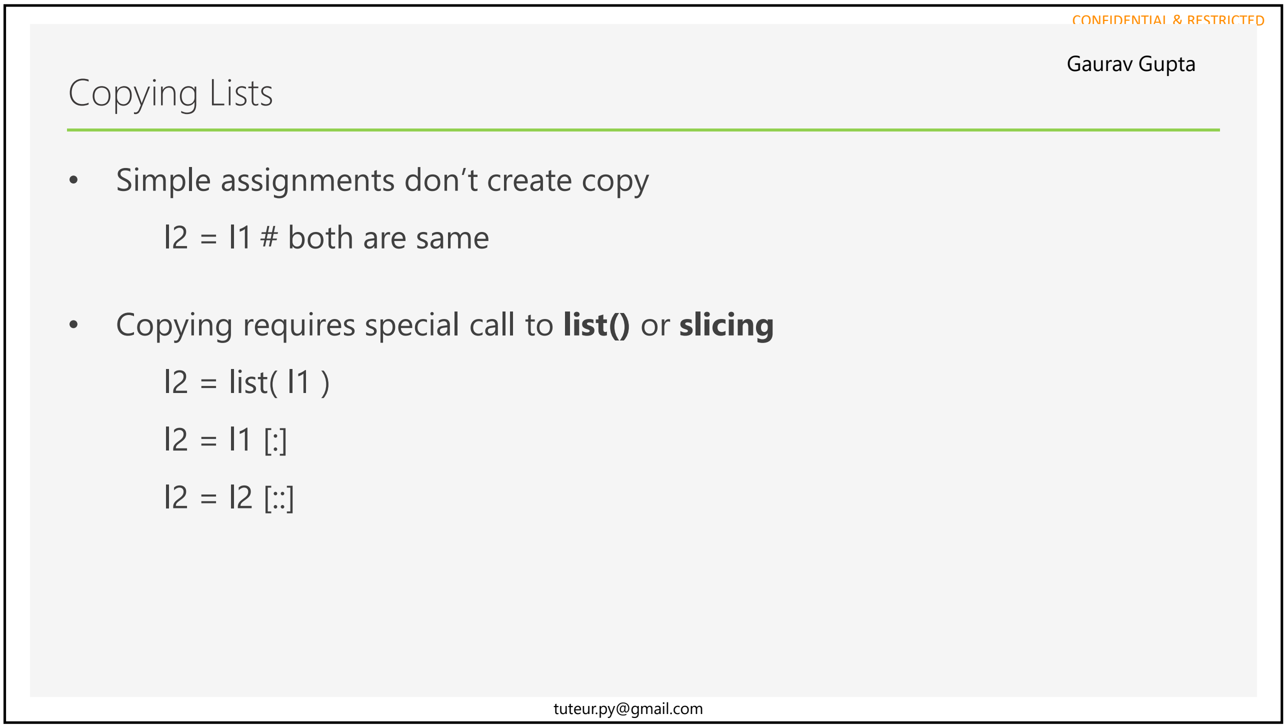
Gaurav Gupta

Tuple

Modifications not allowed

```
x = (1, 2, 3)
x[1] = 3
```

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Copying Lists

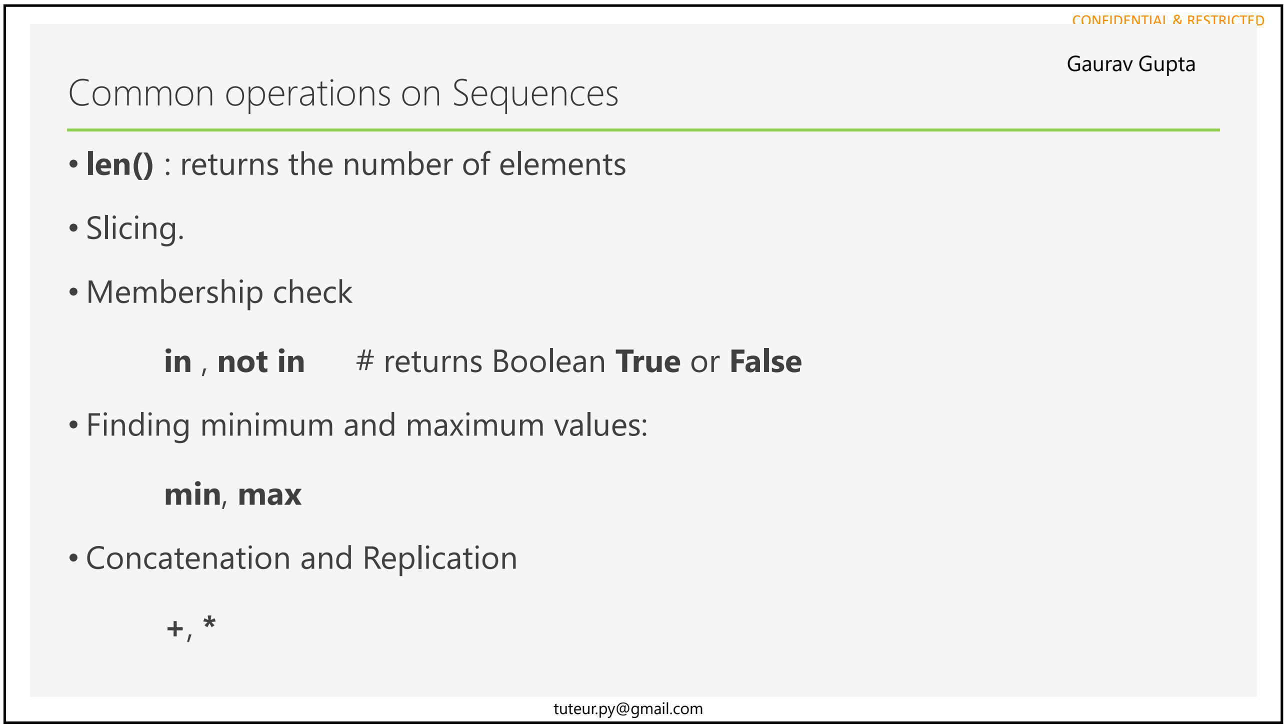
- Simple assignments don’t create copy

```
l2 = l1 # both are same
```

- Copying requires special call to **list()** or **slicing**

```
l2 = list( l1 )
l2 = l1 [:]
l2 = l2 [::]
```

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Common operations on Sequences

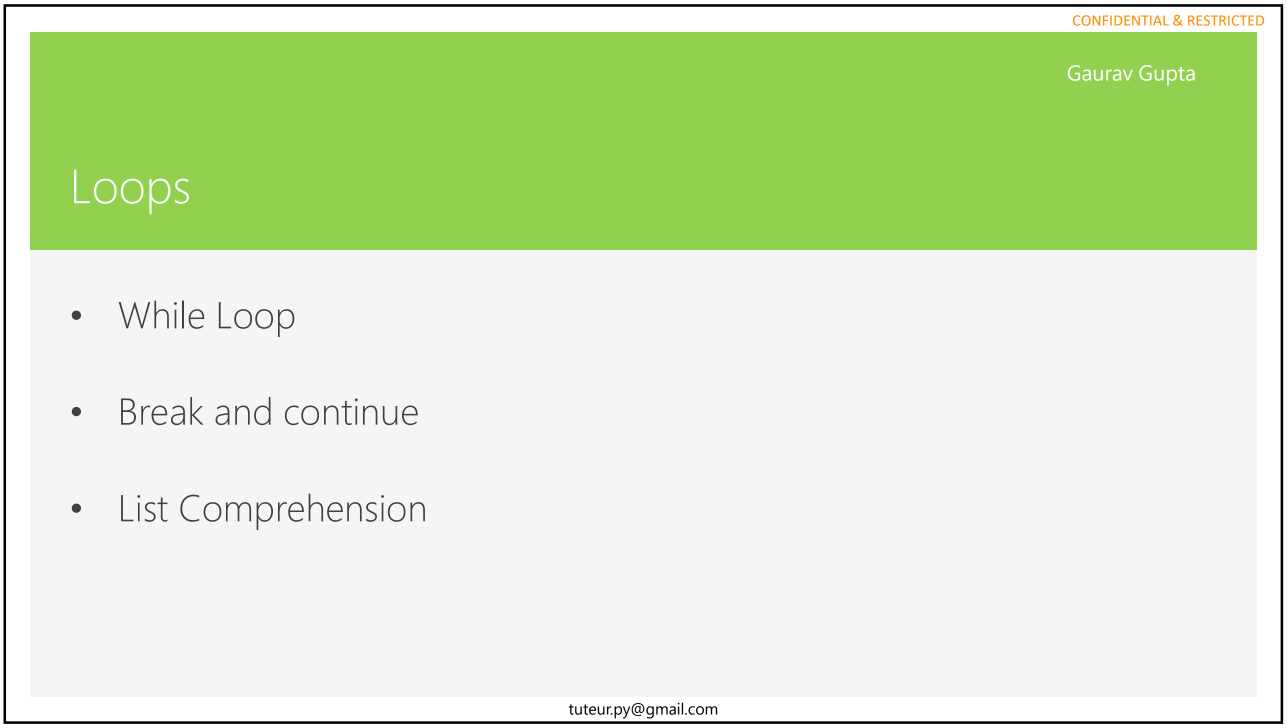
- **len()** : returns the number of elements
- Slicing.
- Membership check

  in , **not in**   # returns Boolean **True** or **False**
- Finding minimum and maximum values:

  min, **max**
- Concatenation and Replication

  +, *****

tuteur.py@gmail.com



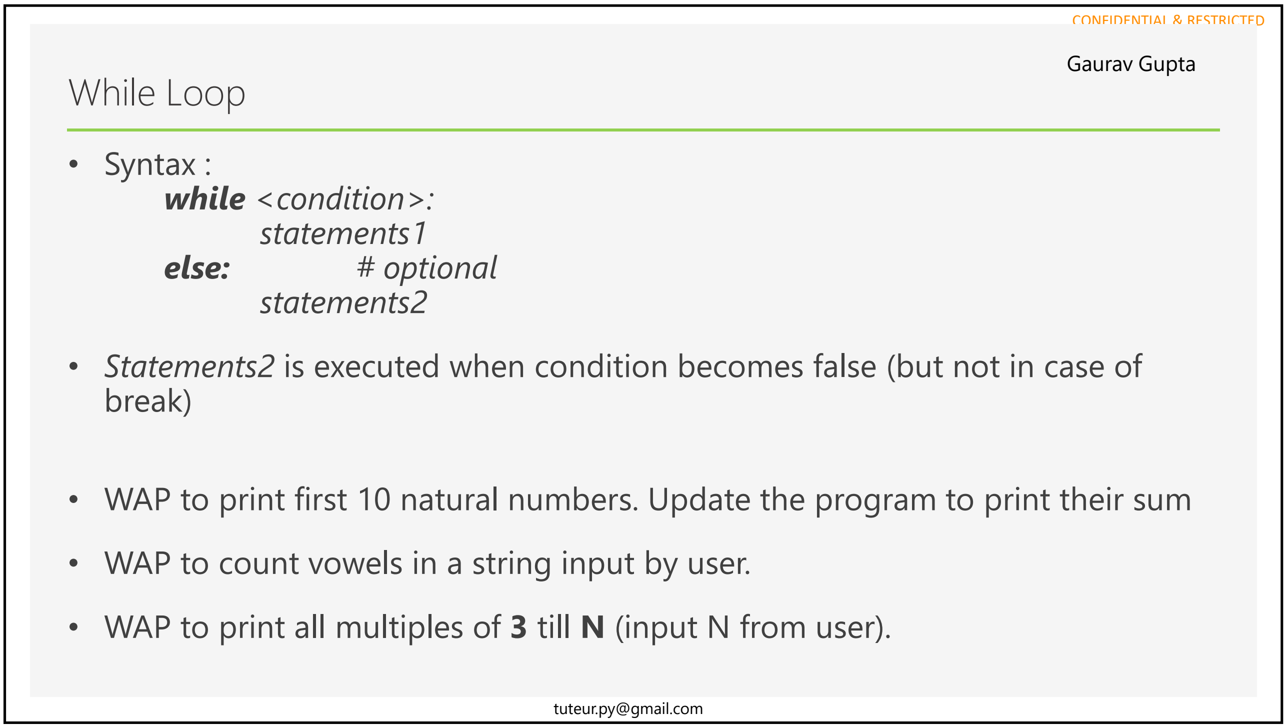
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Loops

- While Loop
- Break and continue
- List Comprehension

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

While Loop

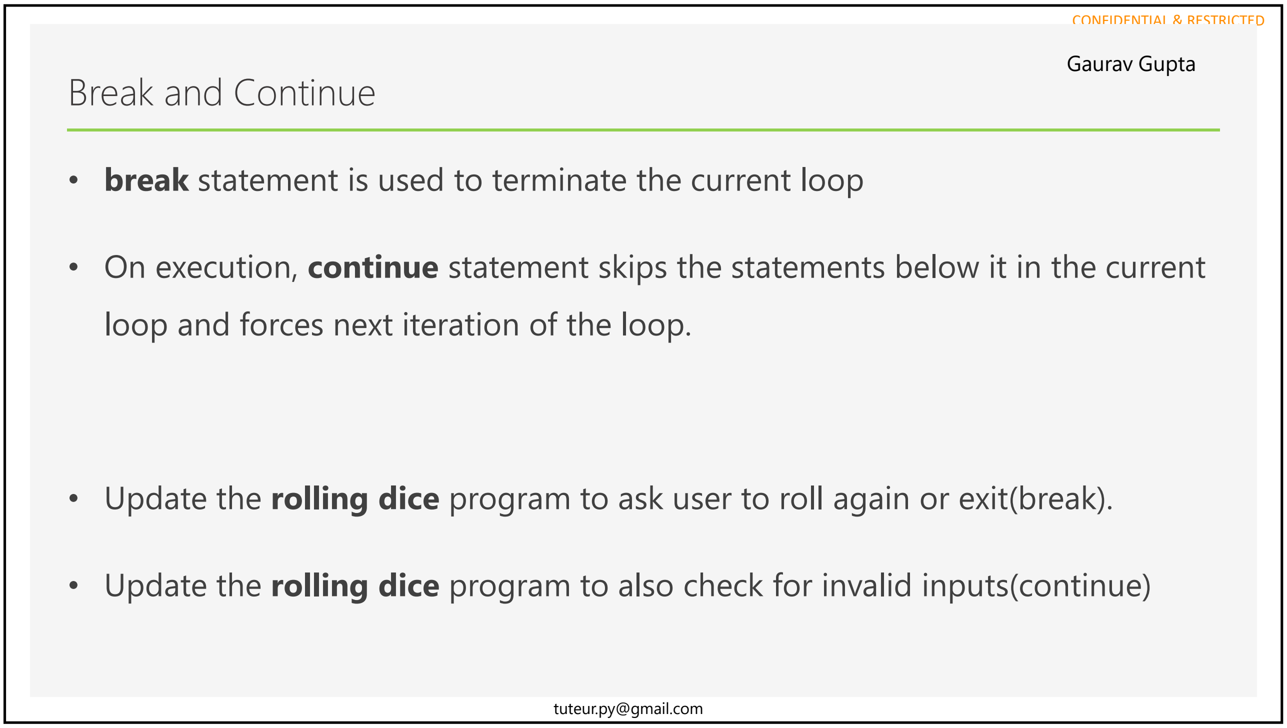
- Syntax :

```
    while <condition>:
          statements1
    else:           # optional
          statements2
```

- *Statements2* is executed when condition becomes false (but not in case of break)

- WAP to print first 10 natural numbers. Update the program to print their sum
- WAP to count vowels in a string input by user.
- WAP to print all multiples of **3** till **N** (input N from user).

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

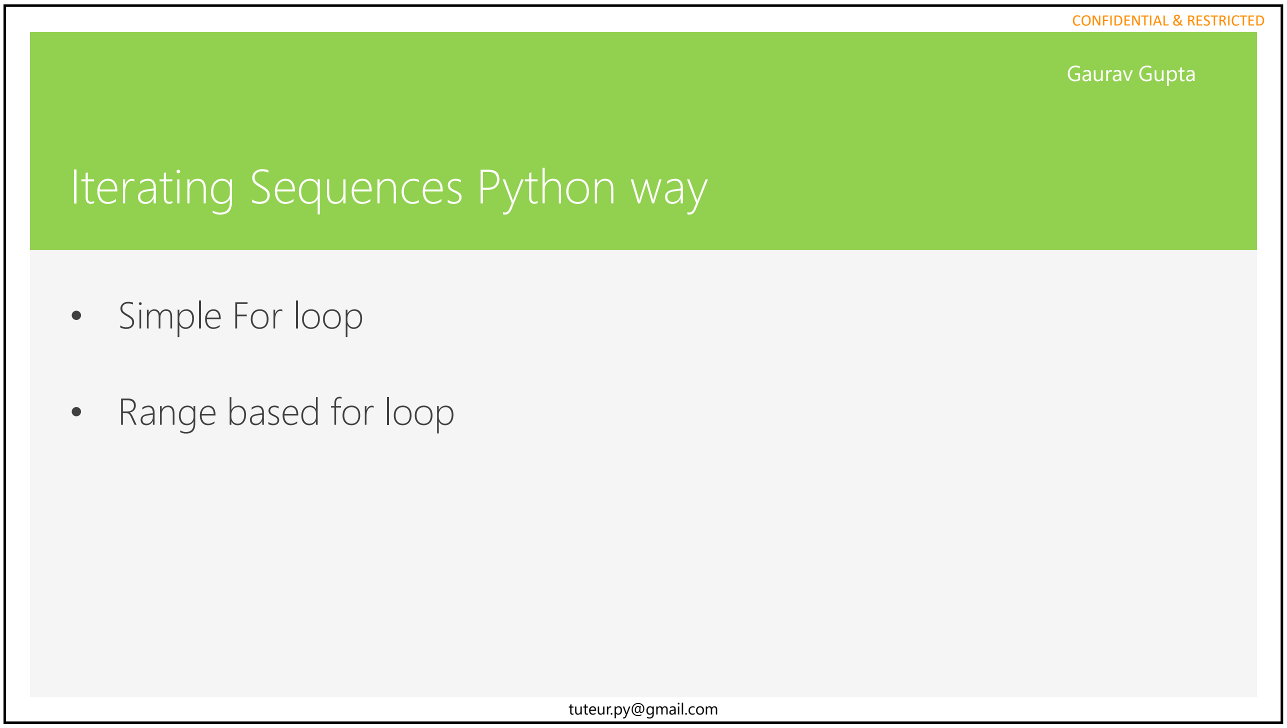
Gaurav Gupta

Break and Continue

- **break** statement is used to terminate the current loop
- On execution, **continue** statement skips the statements below it in the current loop and forces next iteration of the loop.

- Update the **rolling dice** program to ask user to roll again or exit(break).
- Update the **rolling dice** program to also check for invalid inputs(continue)

tuteur.py@gmail.com



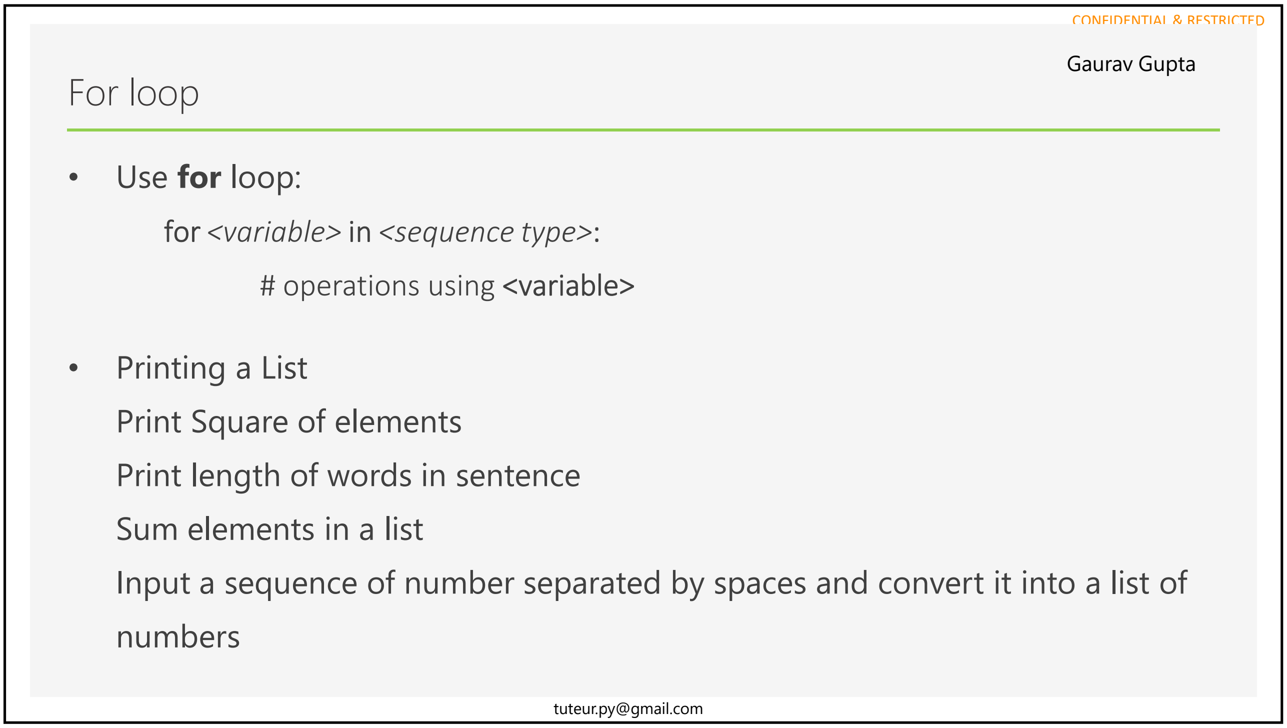
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Iterating Sequences Python way

- Simple For loop
- Range based for loop

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

For loop

- Use **for** loop:

```
for <variable> in <sequence type>:
        # operations using <variable>
```

- Printing a List

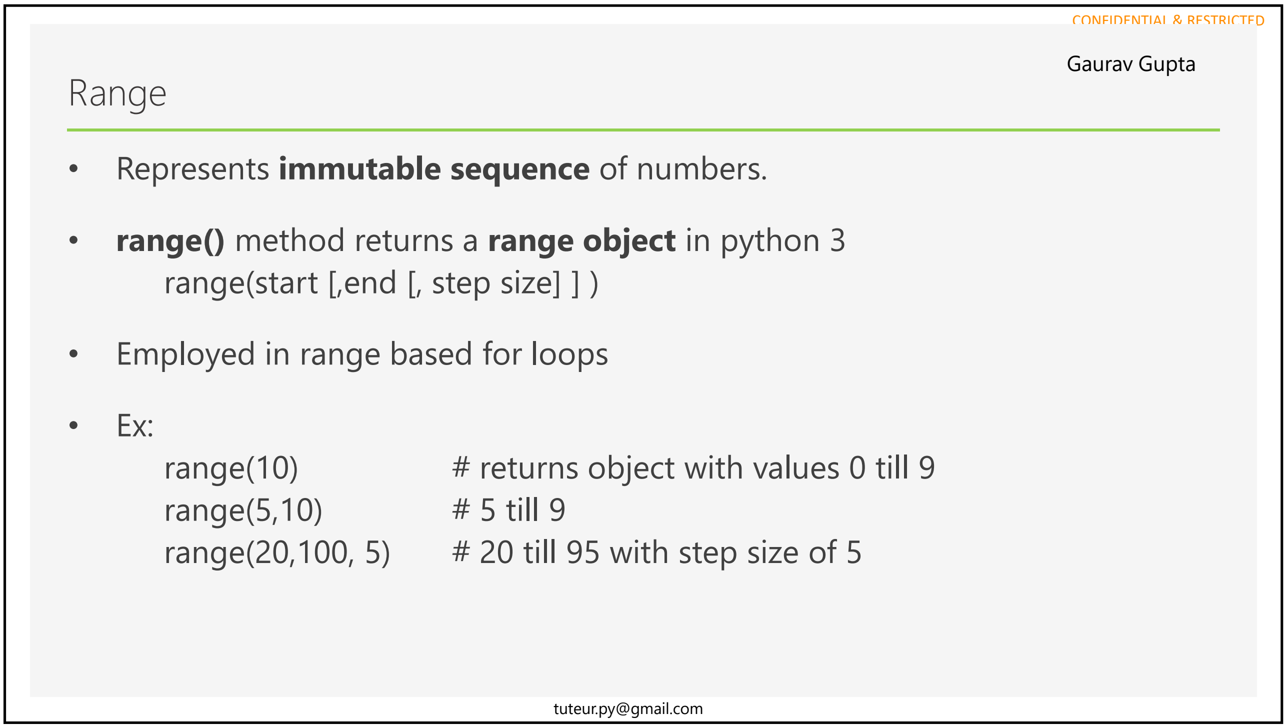
  Print Square of elements

  Print length of words in sentence

  Sum elements in a list

  Input a sequence of number separated by spaces and convert it into a list of numbers

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Range

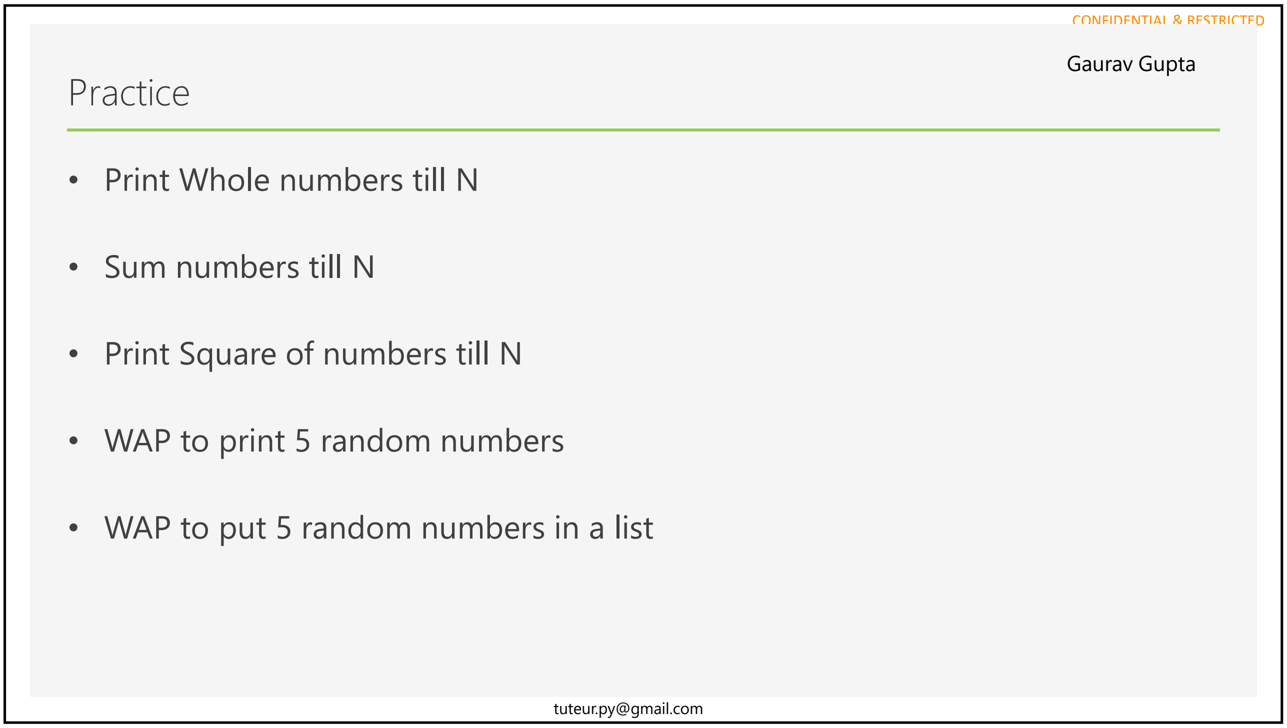
- Represents **immutable sequence** of numbers.
- **range()** method returns a **range object** in python 3

```
range(start [,end [, step size] ] )
```

- Employed in range based for loops
- Ex:

```
range(10)          # returns object with values 0 till 9
range(5,10)        # 5 till 9
range(20,100, 5)   # 20 till 95 with step size of 5
```

tuteur.py@gmail.com



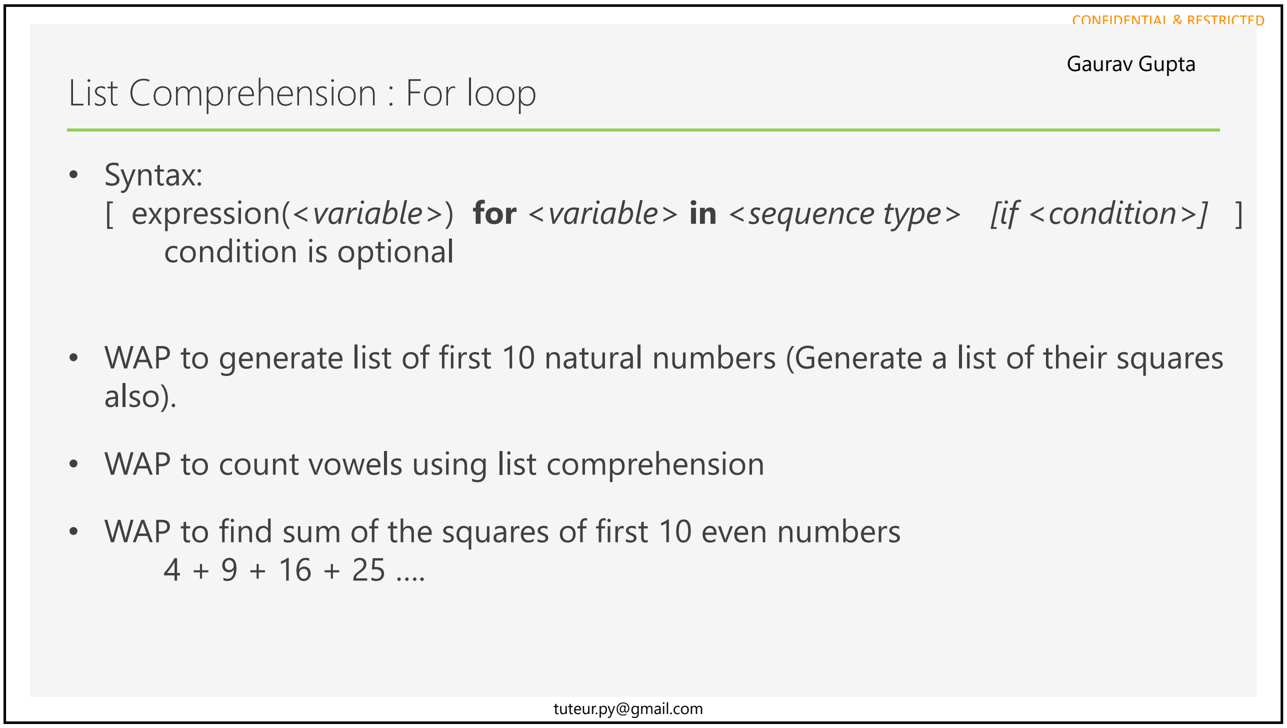
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Practice

- Print Whole numbers till N
- Sum numbers till N
- Print Square of numbers till N
- WAP to print 5 random numbers
- WAP to put 5 random numbers in a list

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

List Comprehension : For loop

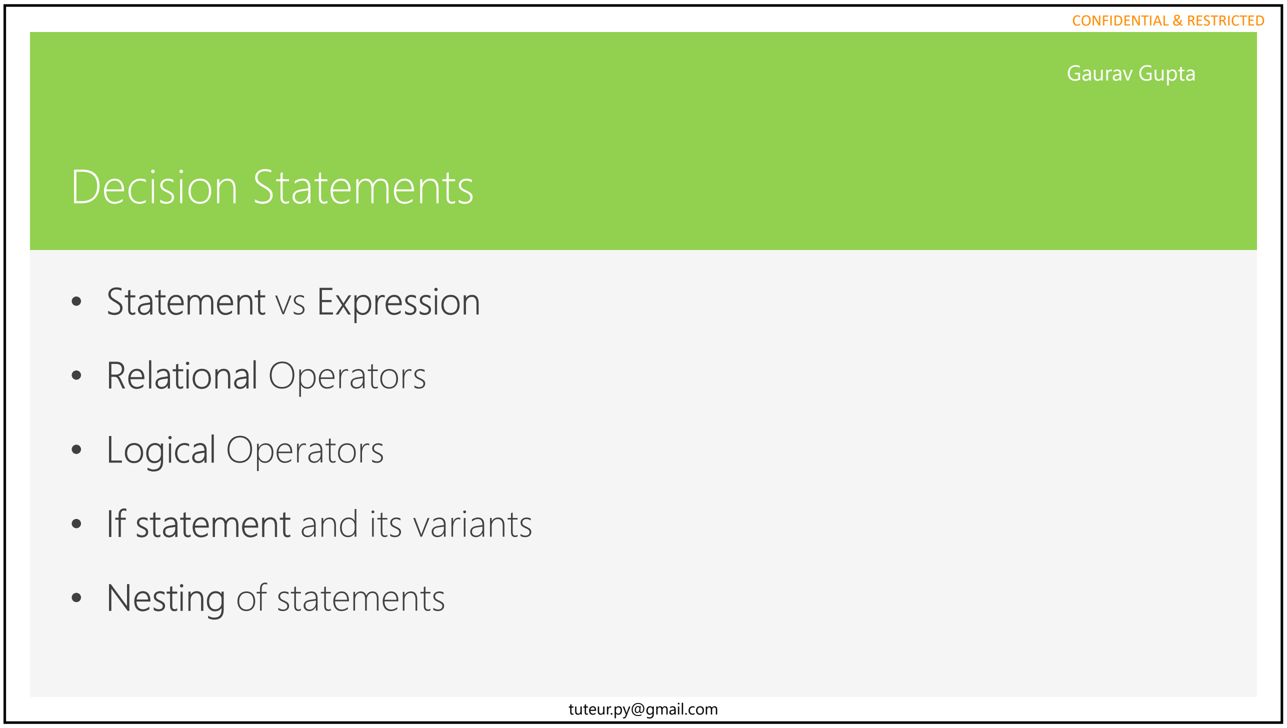
- Syntax:
  [expression(<*variable*>) **for** <*variable*> **in** <*sequence type*> *[if <condition>]*]
  condition is optional

- WAP to generate list of first 10 natural numbers (Generate a list of their squares also).

- WAP to count vowels using list comprehension

- WAP to find sum of the squares of first 10 even numbers
  4 + 9 + 16 + 25 ….

tuteur.py@gmail.com



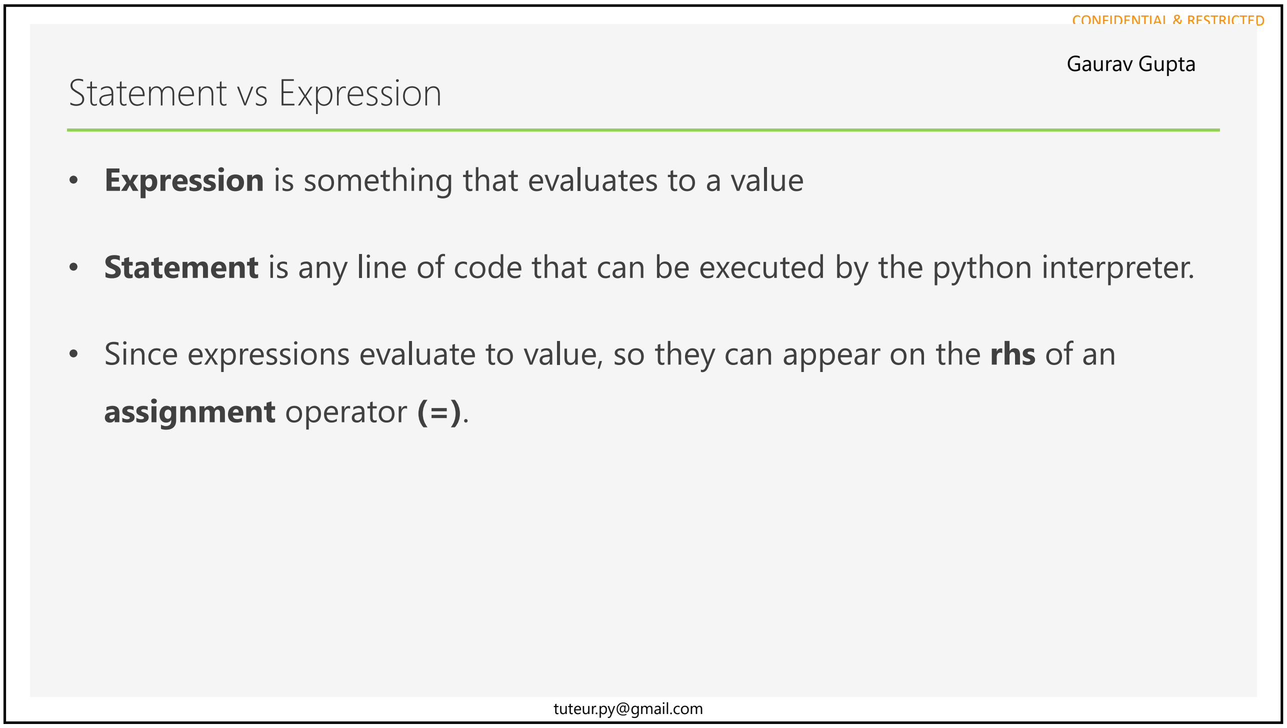
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Decision Statements

- Statement vs Expression
- Relational Operators
- Logical Operators
- If statement and its variants
- Nesting of statements

tuteur.py@gmail.com



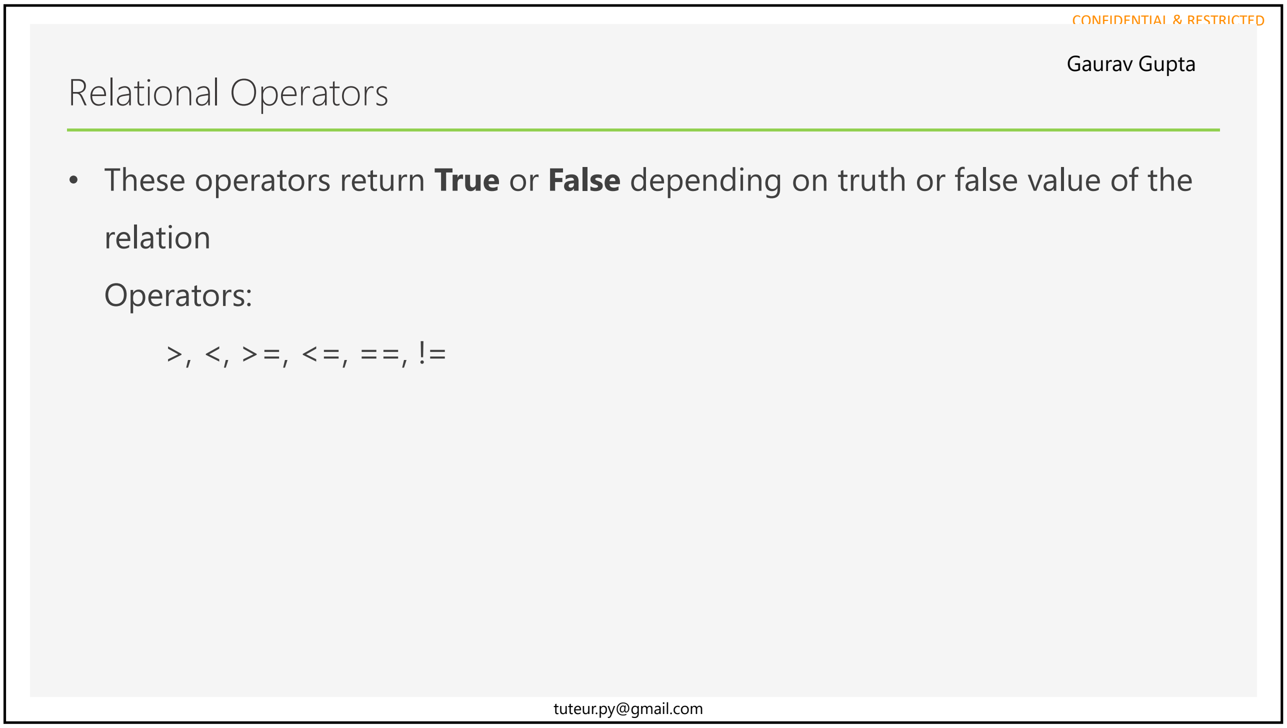
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Statement vs Expression

- **Expression** is something that evaluates to a value
- **Statement** is any line of code that can be executed by the python interpreter.
- Since expressions evaluate to value, so they can appear on the **rhs** of an **assignment** operator **(=)**.

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

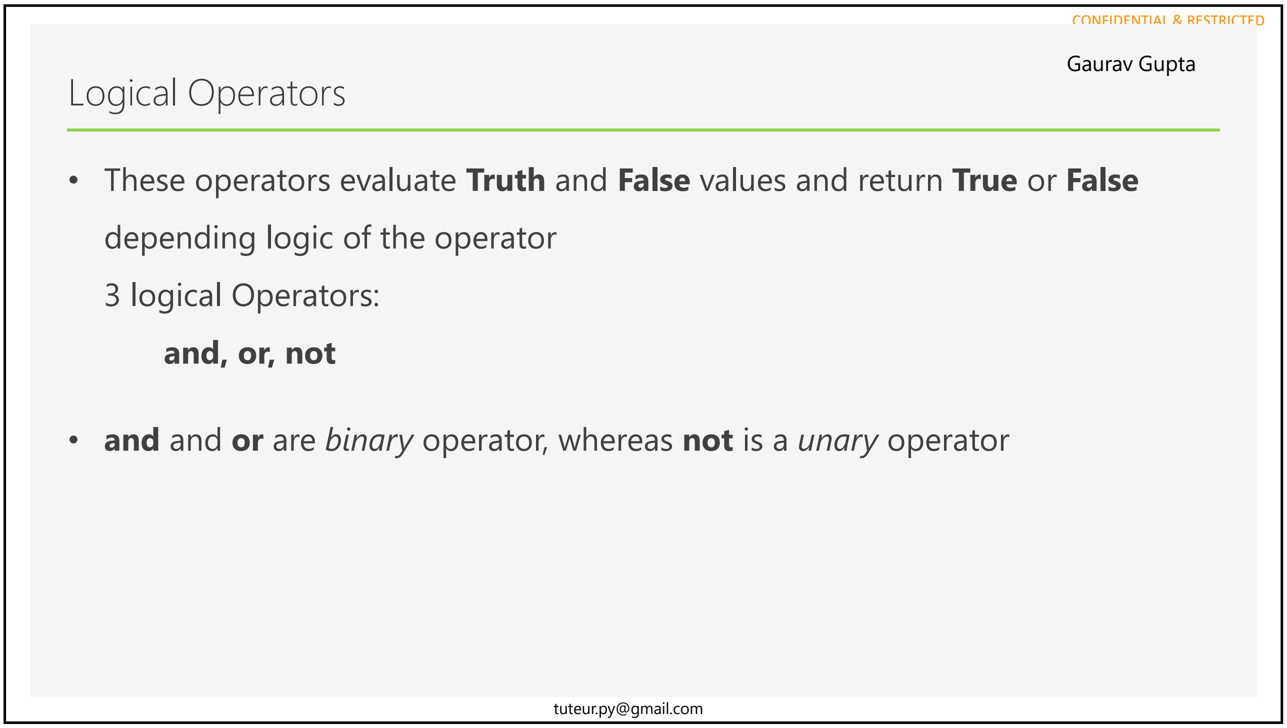
Relational Operators

- These operators return **True** or **False** depending on truth or false value of the relation

  Operators:

  >, <, >=, <=, ==, !=

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Logical Operators

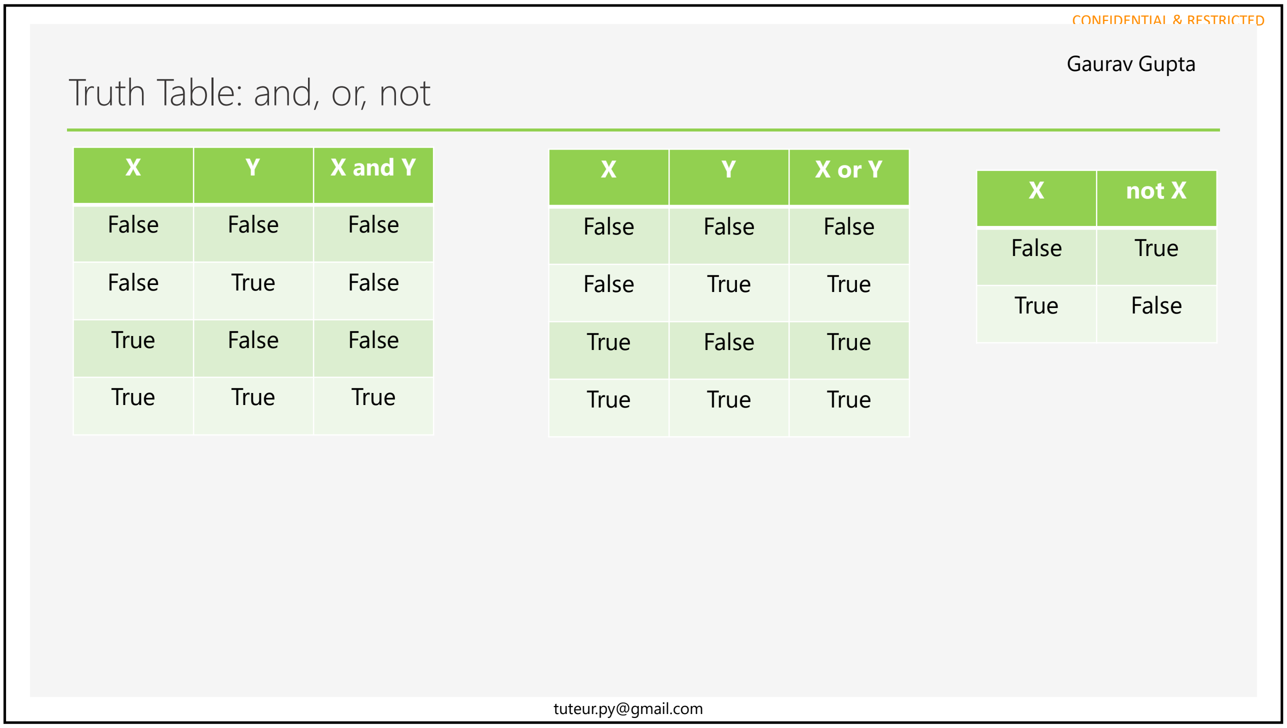
- These operators evaluate **Truth** and **False** values and return **True** or **False** depending logic of the operator

  3 logical Operators:

  and, or, not

- **and** and **or** are *binary* operator, whereas **not** is a *unary* operator

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

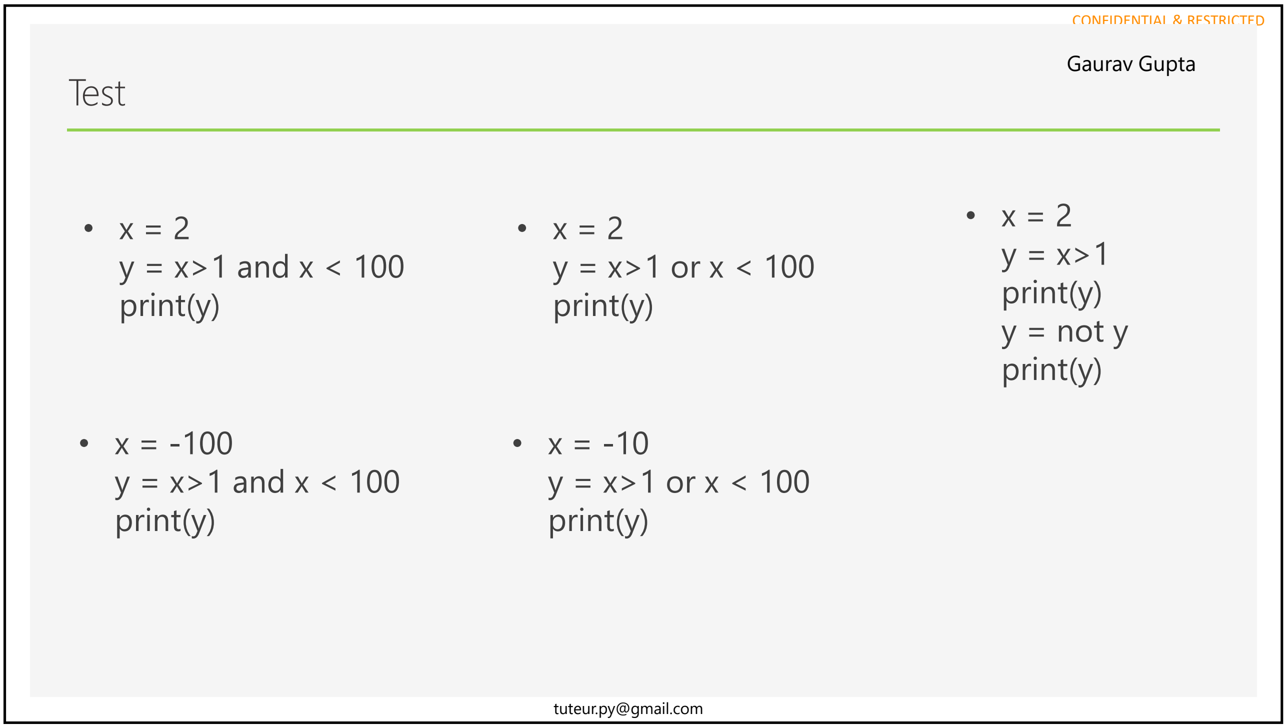
Truth Table: and, or, not

X	Y	X and Y
False	False	False
False	True	False
True	False	False
True	True	True

X	Y	X or Y
False	False	False
False	True	True
True	False	True
True	True	True

X	not X
False	True
True	False

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Test

-
  ```
  x = 2
  y = x>1 and x < 100
  print(y)
  ```

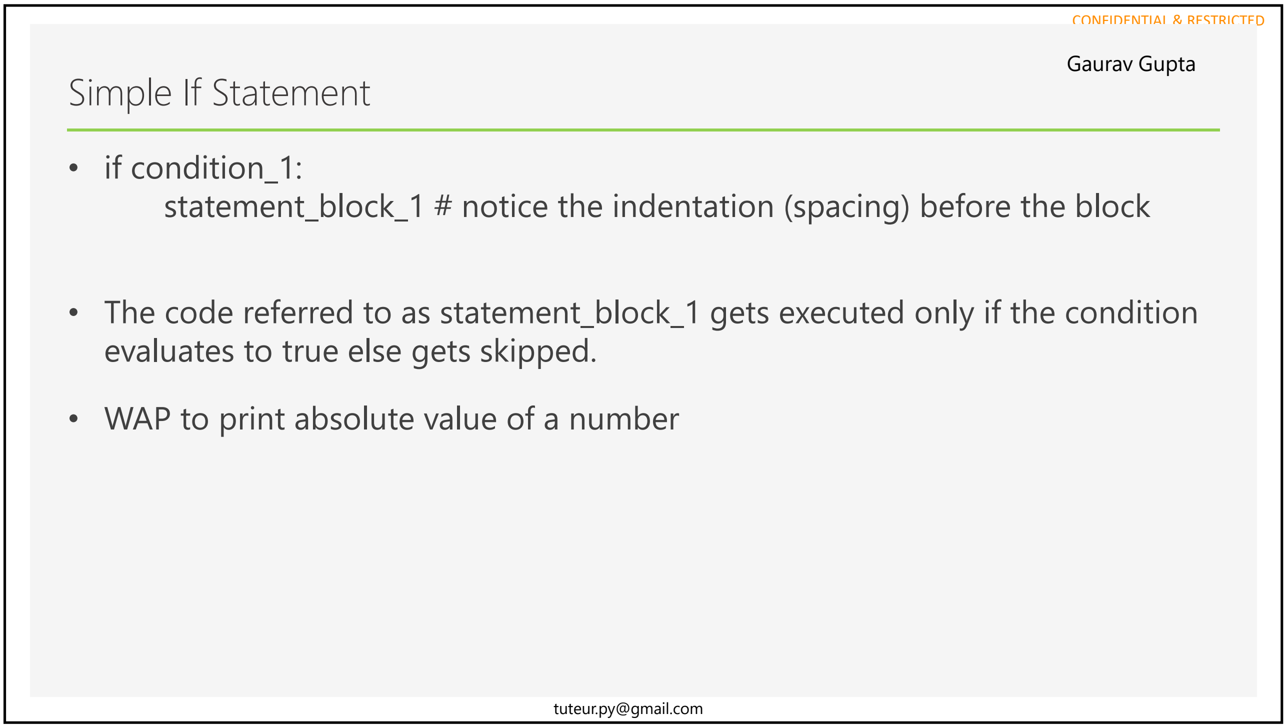
-
  ```
  x = 2
  y = x>1 or x < 100
  print(y)
  ```

-
  ```
  x = 2
  y = x>1
  print(y)
  y = not y
  print(y)
  ```

-
  ```
  x = -100
  y = x>1 and x < 100
  print(y)
  ```

-
  ```
  x = -10
  y = x>1 or x < 100
  print(y)
  ```

tuteur.py@gmail.com



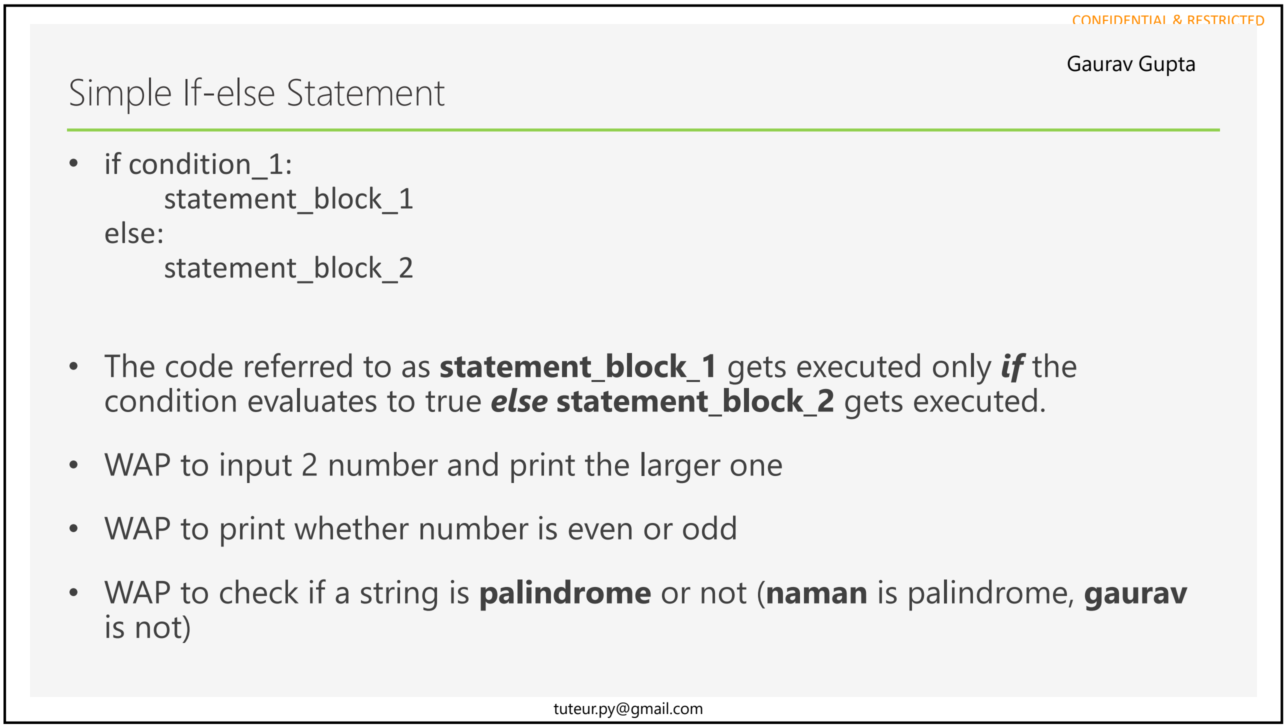
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Simple If Statement

-
  ```
  if condition_1:
        statement_block_1 # notice the indentation (spacing) before the block
  ```
- The code referred to as statement_block_1 gets executed only if the condition evaluates to true else gets skipped.
- WAP to print absolute value of a number

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

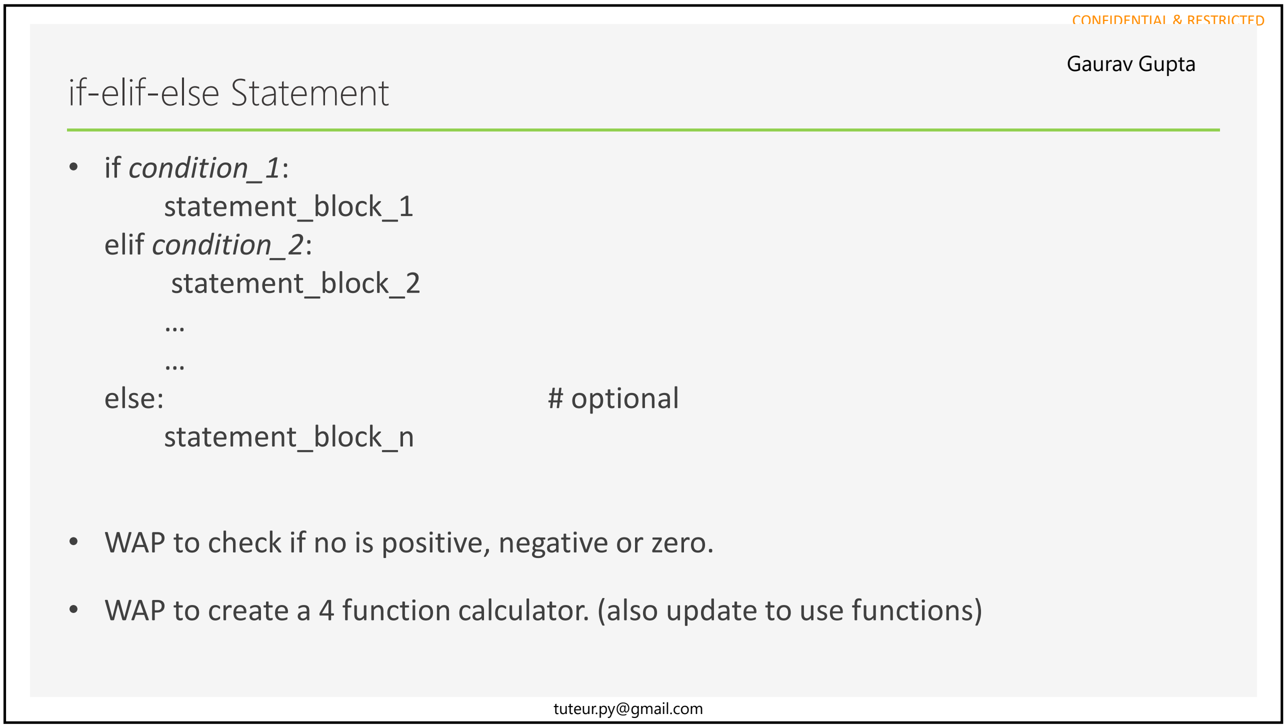
Gaurav Gupta

Simple If-else Statement

-
```
if condition_1:
        statement_block_1
else:
        statement_block_2
```

- The code referred to as **statement_block_1** gets executed only ***if*** the condition evaluates to true ***else*** **statement_block_2** gets executed.
- WAP to input 2 number and print the larger one
- WAP to print whether number is even or odd
- WAP to check if a string is **palindrome** or not (**naman** is palindrome, **gaurav** is not)

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

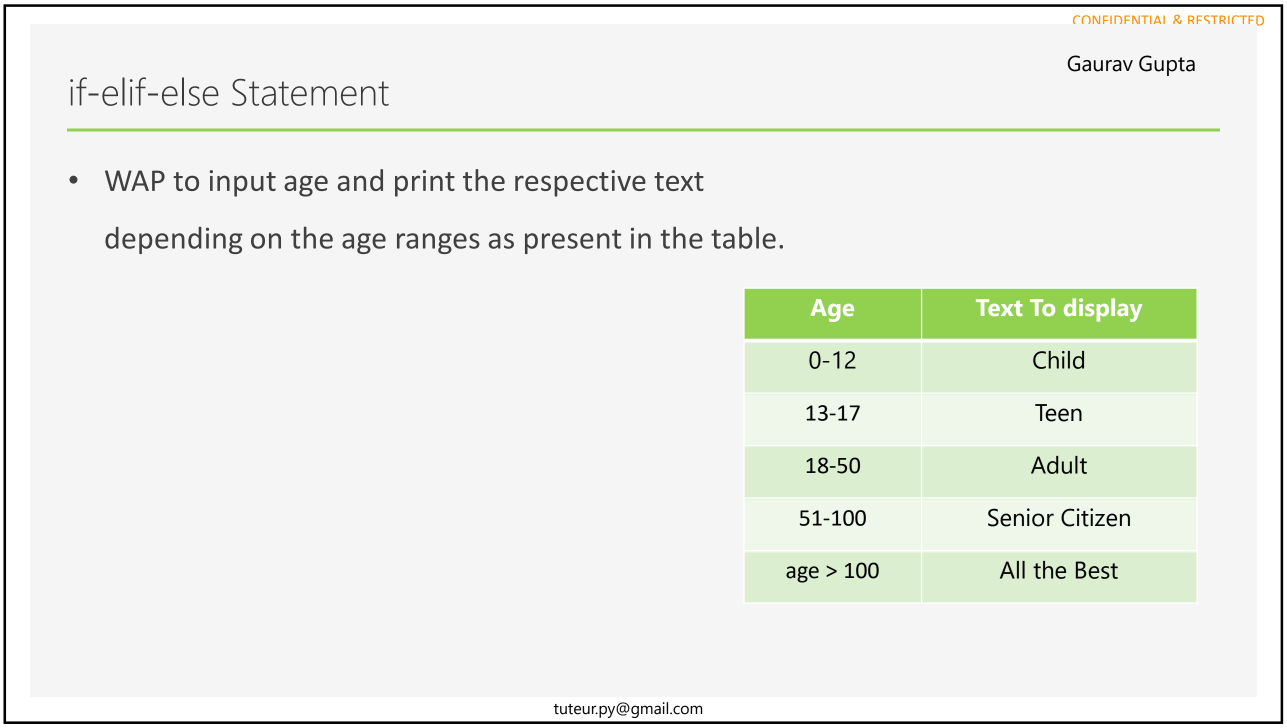
Gaurav Gupta

if-elif-else Statement

-
```
if condition_1:
        statement_block_1
elif condition_2:
         statement_block_2
        …
        …
else:                              # optional
        statement_block_n
```

- WAP to check if no is positive, negative or zero.
- WAP to create a 4 function calculator. (also update to use functions)

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

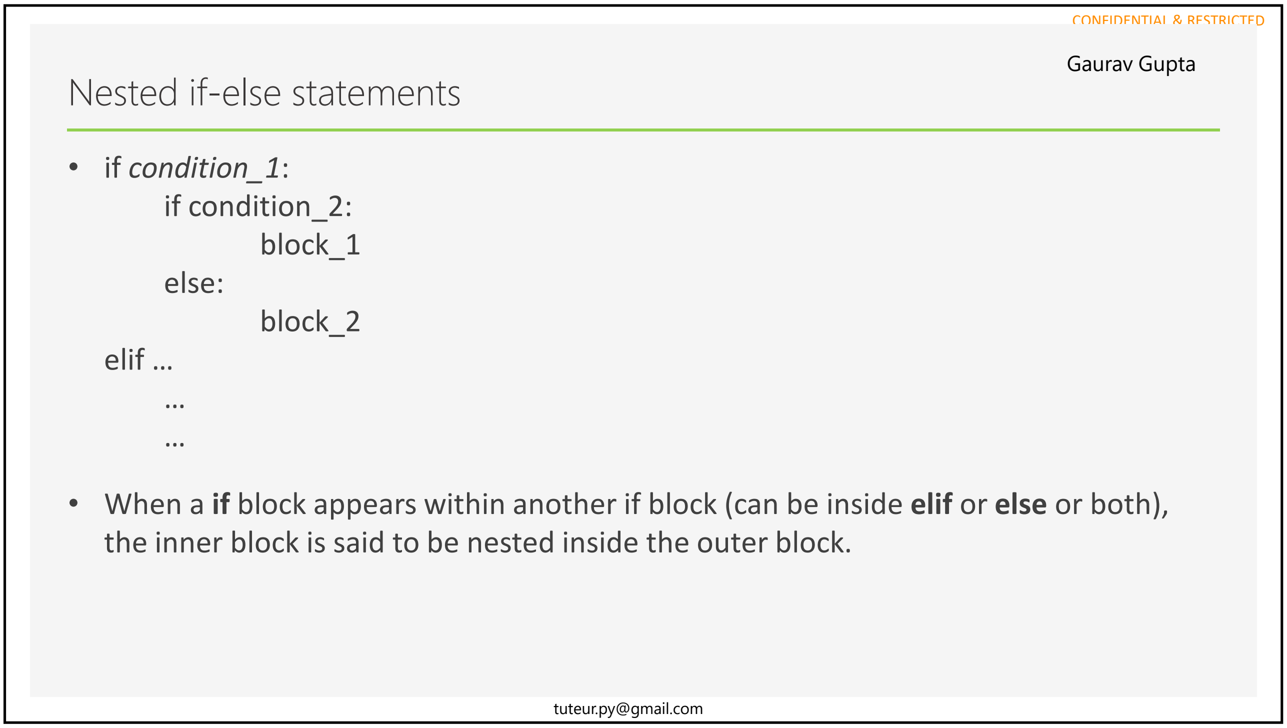
if-elif-else Statement

- WAP to input age and print the respective text

  depending on the age ranges as present in the table.

Age	Text To display
0-12	Child
13-17	Teen
18-50	Adult
51-100	Senior Citizen
age > 100	All the Best

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Nested if-else statements

```
• if condition_1:
      if condition_2:
              block_1
      else:
              block_2
  elif …
      …
      …
```

- When a **if** block appears within another if block (can be inside **elif** or **else** or both), the inner block is said to be nested inside the outer block.

tuteur.py@gmail.com

CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Test

- WAP to input 2 numbers. And do operation depending on the following:
    1. if any of the numbers is negative:
        a. if both are odd, add them
        b. otherwise, subtract them
    2. otherwise:
        a. if both are odd, multiply
        b. if one of them is odd, divide
        c. otherwise, find remainder
- WAP to input 2 numbers and check whether the first is divisible by the second and print true or false depending on the divisibility.
- WAP to print the value of the largest of 3 numbers taken as input from the user.

tuteur.py@gmail.com

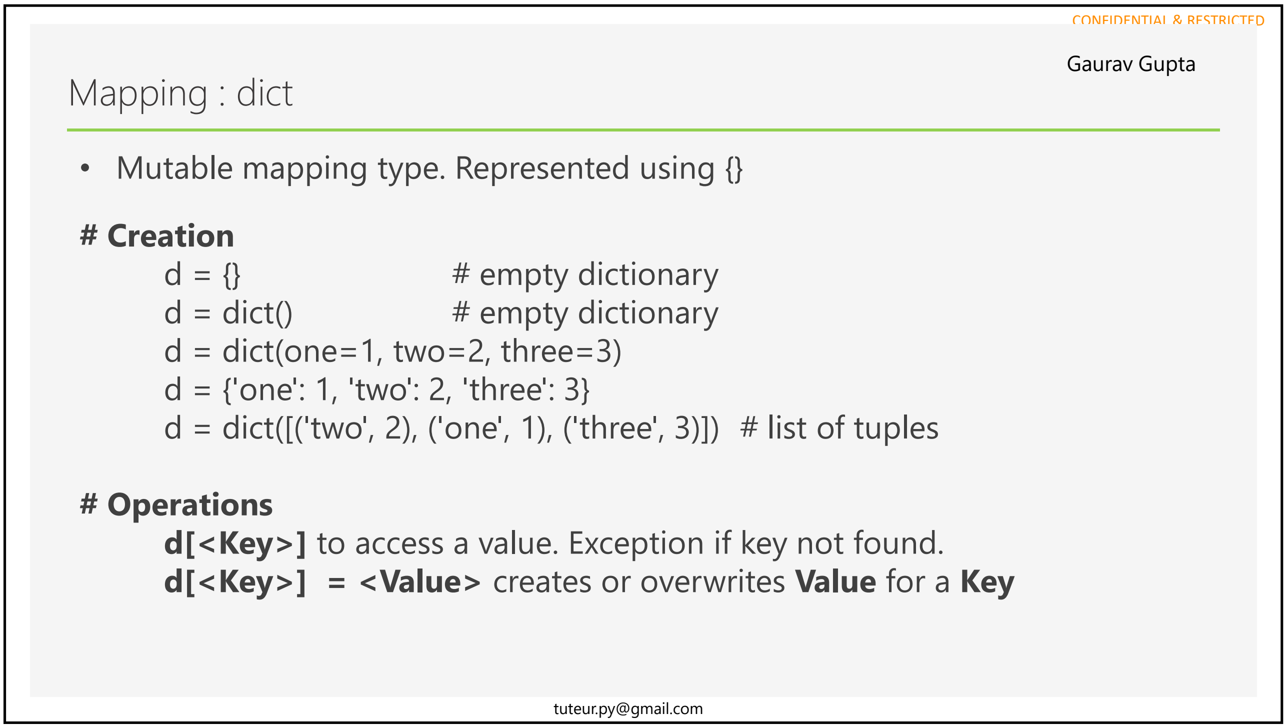
CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Mapping Type : Dict

- Dictionary
- Operations
- Programs

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Mapping : dict

- Mutable mapping type. Represented using {}

# Creation

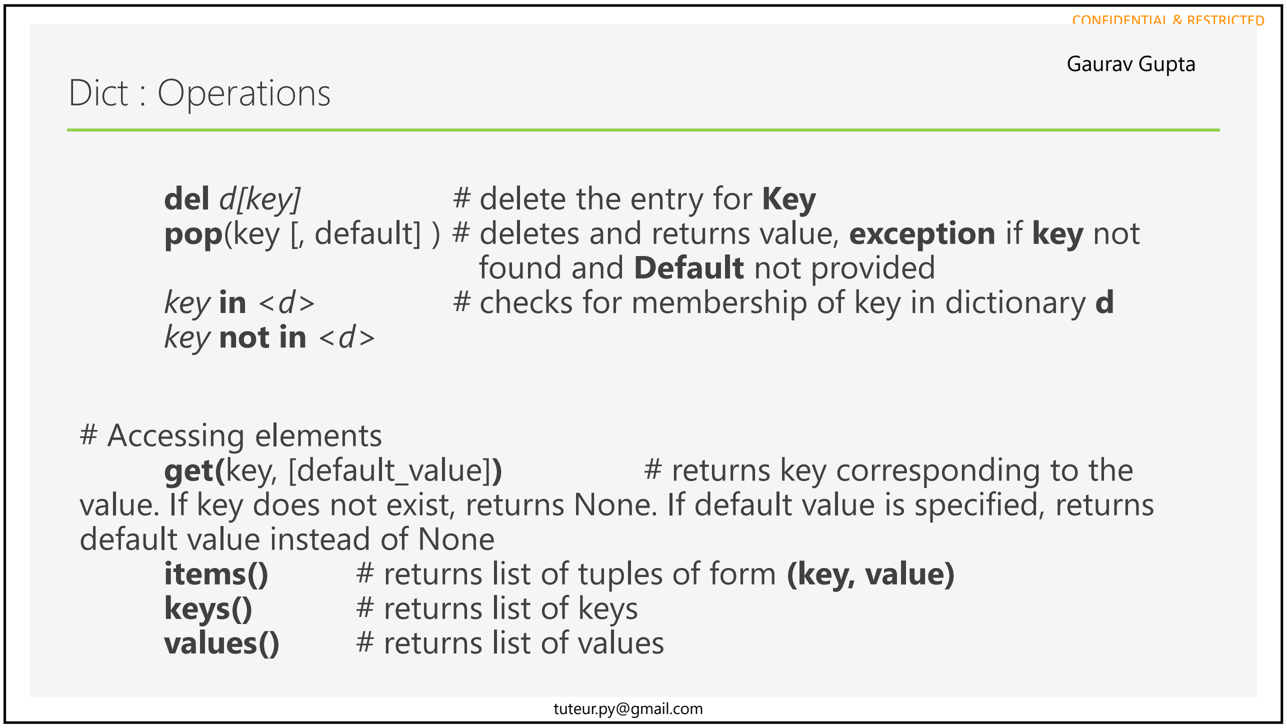
```
d = {}                    # empty dictionary
d = dict()                # empty dictionary
d = dict(one=1, two=2, three=3)
d = {'one': 1, 'two': 2, 'three': 3}
d = dict([('two', 2), ('one', 1), ('three', 3)])  # list of tuples
```

# Operations

d[<Key>] to access a value. Exception if key not found.

d[<Key>] = <Value> creates or overwrites **Value** for a **Key**

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Dict : Operations

del *d[key]* # delete the entry for **Key**

pop(key [, default]) # deletes and returns value, **exception** if **key** not found and **Default** not provided

key **in** <*d*> # checks for membership of key in dictionary **d**

key **not in** <*d*>

# Accessing elements

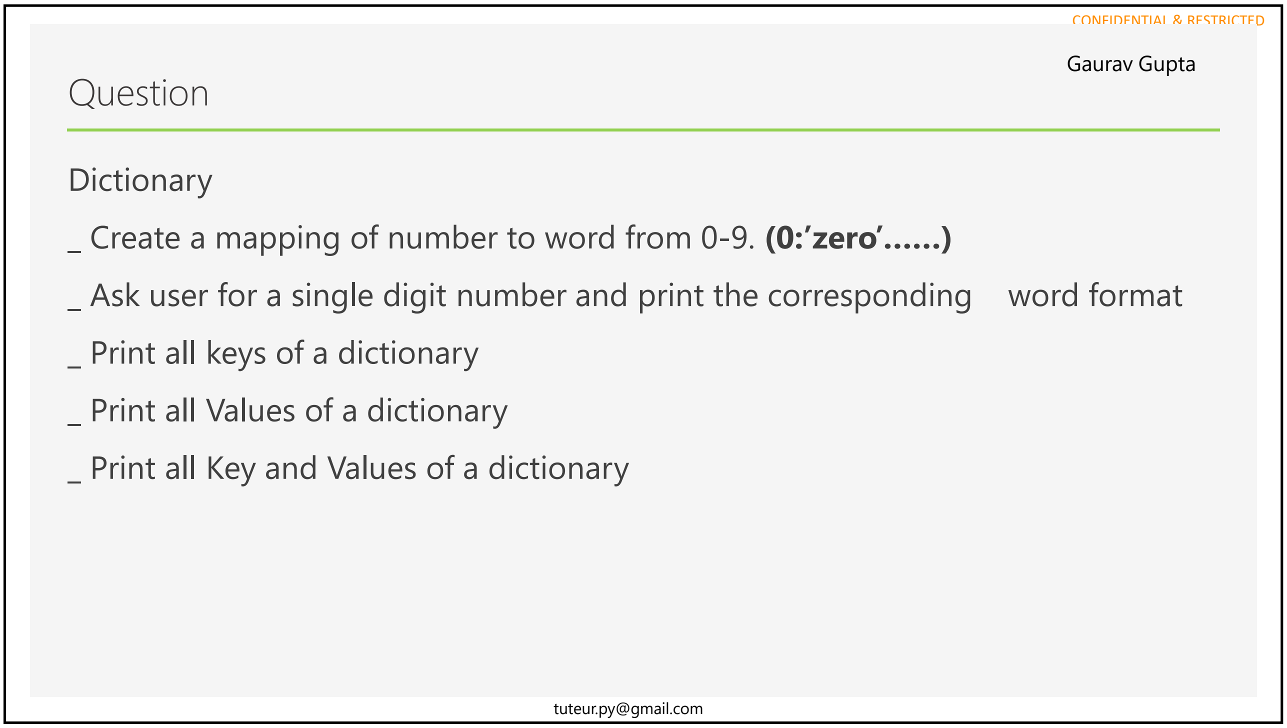
get(key, [default_value]**)** # returns key corresponding to the value. If key does not exist, returns None. If default value is specified, returns default value instead of None

items() # returns list of tuples of form **(key, value)**

keys() # returns list of keys

values() # returns list of values

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Question

Dictionary

_ Create a mapping of number to word from 0-9. **(0:'zero'……)**

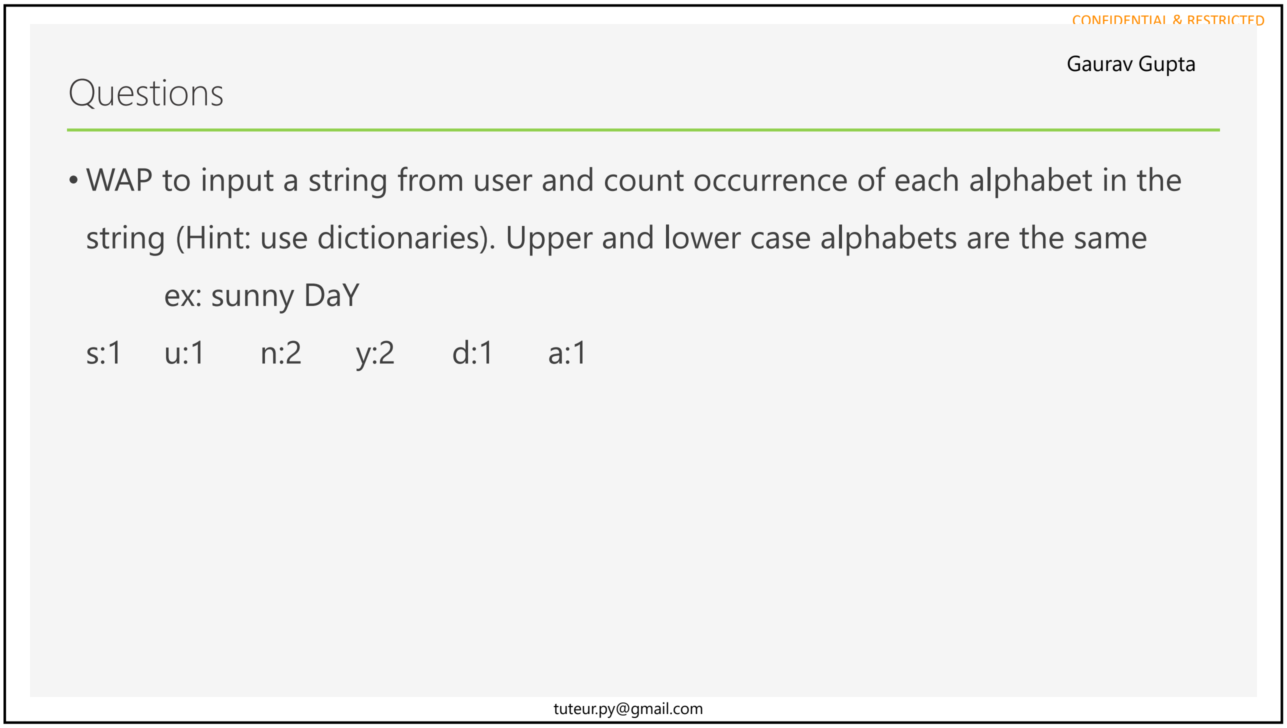
_ Ask user for a single digit number and print the corresponding word format

_ Print all keys of a dictionary

_ Print all Values of a dictionary

_ Print all Key and Values of a dictionary

tuteur.py@gmail.com



CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Questions

- WAP to input a string from user and count occurrence of each alphabet in the string (Hint: use dictionaries). Upper and lower case alphabets are the same

ex: sunny DaY

s:1 u:1 n:2 y:2 d:1 a:1

tuteur.py@gmail.com