

CONFIDENTIAL &amp; RESTRICTED

# Welcome to Python

Gaurav Gupta  
tuteur.py@gmail.com

1

CONFIDENTIAL &amp; 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

2

CONFIDENTIAL &amp; RESTRICTED

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

3

CONFIDENTIAL &amp; RESTRICTED

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](https://etherpad.net/p/py_learnbay)

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

tuteur.py@gmail.com

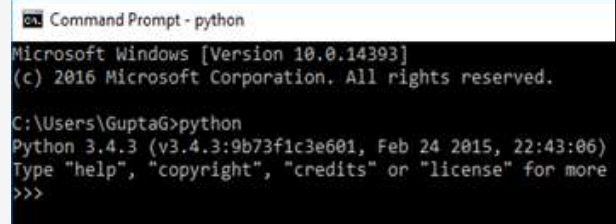
4

CONFIDENTIAL &amp; RESTRICTED

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

5

CONFIDENTIAL &amp; RESTRICTED

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

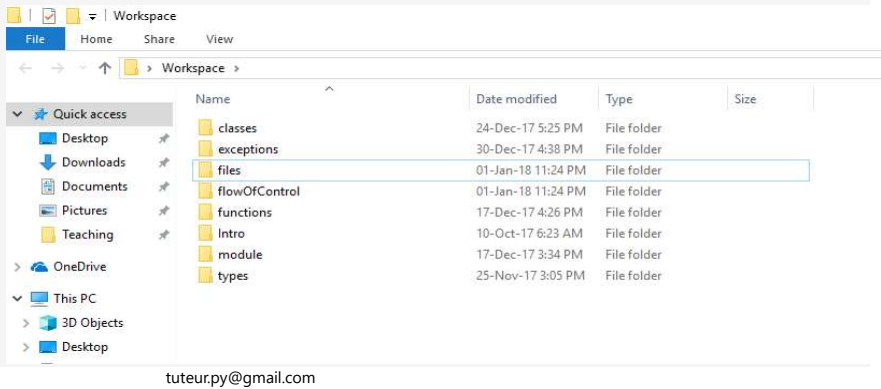
6

### 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: **fN\_topic.py**

Ex:

f1\_ifStatement.py  
f2\_ifElse.py



7

### Windows and Linux Command line

	Windows	Linux
Go to the folder	<code>cd &lt;folder Name&gt;</code> Ex: <code>cd Workspace</code>	<code>cd &lt;folder Name&gt;</code> Ex: <code>cd Workspace</code>
Go to the previous directory	<code>cd ..</code>	<code>cd ..</code>
List files in current directory	<code>dir</code>	<code>ls</code> <code>ls -la</code>

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

8

CONFIDENTIAL &amp; RESTRICTED

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

9

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Windows shortcuts

### Open command window in current folder

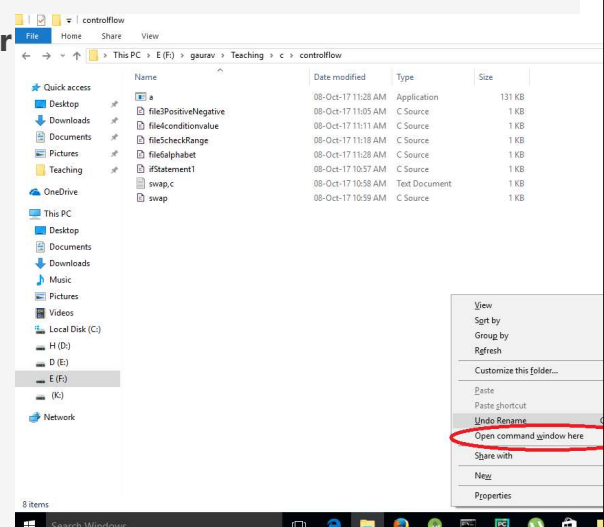
Press the Shift Key and right click

You will see the option :

*Open Command Window Here*

### To Switch Tabs/Windows

Alt + Tab      and      Alt + Shift + Tab



tuteur.py@gmail.com

10

CONFIDENTIAL & RESTRICTED

Gaurav Gupta

## Python Kickstart

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

tuteur.py@gmail.com

11

CONFIDENTIAL & RESTRICTED

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

12

CONFIDENTIAL &amp; RESTRICTED

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

13

CONFIDENTIAL &amp; RESTRICTED

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

14

CONFIDENTIAL &amp; RESTRICTED

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

15

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

### Tokens : building blocks

---

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

tuteur.py@gmail.com

16



CONFIDENTIAL &amp; RESTRICTED

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	

tuteur.py@gmail.com

17

CONFIDENTIAL &amp; RESTRICTED

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 :

<code>__*</code>	:	Special Reserved system defined names
<code>_*</code>	:	Used to define private class members

tuteur.py@gmail.com

18

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Operators

---

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

tuteur.py@gmail.com

19

CONFIDENTIAL &amp; RESTRICTED

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
<b>None</b>	:	<b>Empty</b>

tuteur.py@gmail.com

20

CONFIDENTIAL &amp; RESTRICTED

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

21

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Comments

- **Single line** comments start with #.
- **Multi line** comments can use the triple quote syntax.

"""

This is a multi line  
comment in python.

"""

tuteur.py@gmail.com

22

CONFIDENTIAL &amp; RESTRICTED

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

23

CONFIDENTIAL &amp; RESTRICTED

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

24

CONFIDENTIAL &amp; RESTRICTED

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

25

CONFIDENTIAL &amp; RESTRICTED

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

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

tuteur.py@gmail.com

26

CONFIDENTIAL &amp; RESTRICTED

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**
- **hex()** method returns the binary representation of an **integer**

tuteur.py@gmail.com

27

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Data Types and Operations

- Numeric types
- Boolean types
- Strings
- None types

tuteur.py@gmail.com

28

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

Numeric  $2+2.5 = 4.5$

---

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

tuteur.py@gmail.com

29

CONFIDENTIAL &amp; RESTRICTED

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:  

<b>print(bool(0));</b>	<b>print(bool(10));</b>	<b>print(bool(-1))</b>
<b>print(int(True));</b>	<b>print(int(False))</b>	

tuteur.py@gmail.com

30

CONFIDENTIAL &amp; RESTRICTED

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

31

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

None type

---

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

tuteur.py@gmail.com

32



CONFIDENTIAL &amp; RESTRICTED

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
- chr() is used to convert Numeric to Character

tuteur.py@gmail.com

33

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Importing

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

tuteur.py@gmail.com

34

CONFIDENTIAL &amp; RESTRICTED

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

35

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Random Library

- import random module using:  
`import random`
- Random Integers :  
`randrange(end)`  **$0 \leq N \leq \text{end} - 1$**   
`randrange(100)`  
`randrange(start, end, [step])` **one from start, start+step, start + step\*2..**  
`randrange(10,20,2)`  
`randint(start, end)`  **$\text{start} \leq N \leq \text{end}$**   
`randint(1,10)`

tuteur.py@gmail.com

36

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Random Library

---

- Random Floats:

`random()`

**Floating number [0.0, 1.0) or  $0.0 \leq N < 1.0$**

`uniform(start, end)`

**`start <= N <= end`**

*`uniform(11,44.5)`*

tuteur.py@gmail.com

37

CONFIDENTIAL &amp; RESTRICTED

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

38

CONFIDENTIAL &amp; 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

39

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Functions

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

tuteur.py@gmail.com

40

CONFIDENTIAL &amp; RESTRICTED

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)
  
```

The diagram labels the components of the code above:

- def Keyword**: points to the `def` keyword.
- Function name**: points to `function_to_sum`.
- parameters**: points to the parentheses and their contents `(value1, value2)`.
- body or code**: points to the indented lines of code inside the function.
- function definition**: a bracket on the right side grouping the entire function definition.
- arguments**: points to the values `10` and `x` passed in the function call.
- function call**: a bracket on the right side grouping the entire function call line.

41

CONFIDENTIAL &amp; RESTRICTED

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

42

CONFIDENTIAL &amp; RESTRICTED

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

43

CONFIDENTIAL &amp; RESTRICTED

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 return 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

44

CONFIDENTIAL &amp; RESTRICTED

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

45

CONFIDENTIAL &amp; RESTRICTED

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 python's 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

46

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Back to Strings

- String Functions
- Indexing and Slicing
- String Formatting

tuteur.py@gmail.com

47

CONFIDENTIAL &amp; RESTRICTED

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

48



CONFIDENTIAL &amp; RESTRICTED

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

49

CONFIDENTIAL &amp; RESTRICTED

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

50

CONFIDENTIAL &amp; RESTRICTED

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

51

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Formatting strings

---

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

tuteur.py@gmail.com

52

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Sequence Type List

- List Creation
- List Mutability
- Operations
- Slicing

tuteur.py@gmail.com

53

CONFIDENTIAL &amp; 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

54

CONFIDENTIAL &amp; 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

55

CONFIDENTIAL &amp; 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

56

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Sequence Type Tuple

- Tuple Creation
- Immutability
- Operations
- Slicing

tuteur.py@gmail.com

57

CONFIDENTIAL &amp; 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

58

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Tuple

---

Modifications not allowed

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

```
x[1] = 3
```

tuteur.py@gmail.com

59

CONFIDENTIAL &amp; 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

60

CONFIDENTIAL &amp; 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

61

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Loops

- While Loop
- Break and continue
- List Comprehension

tuteur.py@gmail.com

62

CONFIDENTIAL &amp; 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

63

CONFIDENTIAL &amp; 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

64



CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Iterating Sequences Python way

- Simple For loop
- Range based for loop

tuteur.py@gmail.com

65

CONFIDENTIAL &amp; 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

66

CONFIDENTIAL &amp; 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:  

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

tuteur.py@gmail.com

67

CONFIDENTIAL &amp; 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

68

CONFIDENTIAL &amp; 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

69

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Decision Statements

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

tuteur.py@gmail.com

70

CONFIDENTIAL &amp; 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

71

CONFIDENTIAL &amp; 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

72

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

73

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

74

CONFIDENTIAL &amp; 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

75

CONFIDENTIAL &amp; 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

76

CONFIDENTIAL &amp; 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

77

CONFIDENTIAL &amp; 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

78

CONFIDENTIAL &amp; 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

79

CONFIDENTIAL &amp; 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

80



CONFIDENTIAL &amp; 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.

tuteurpy@gmail.com