

CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Importing

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

tuteur.py@gmail.com

CONFIDENTIAL & 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

CONFIDENTIAL & RESTRICTED

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

CONFIDENTIAL & RESTRICTED

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

CONFIDENTIAL & 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

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

CONFIDENTIAL & 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)
```

Diagram labels:

- def**: Keyword
- function_to_sum**: Function name
- value1, value2**: parameters
- print**: body or code
- function_to_sum(10, x)**: function call
- 10, x**: arguments
- def function_to_sum(value1, value2): ...**: function definition

CONFIDENTIAL & 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

CONFIDENTIAL & 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

CONFIDENTIAL & 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

CONFIDENTIAL & 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

CONFIDENTIAL & 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

CONFIDENTIAL & RESTRICTED

Gaurav Gupta

Back to Strings

- String Functions
- Indexing and Slicing
- String Formatting

tuteur.py@gmail.com

CONFIDENTIAL & 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

CONFIDENTIAL & 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

CONFIDENTIAL & 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

CONFIDENTIAL & 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

CONFIDENTIAL & 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

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

- x = 2
y = x>1 or x < 100
print(y)
- x = -10
y = x>1 or x < 100
print(y)

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