

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:

| | |
|-------------------------------|--|
| <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

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 &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.

tuteur.py@gmail.com

CONFIDENTIAL &amp; RESTRICTED

Gaurav Gupta

## Mapping Type : Dict

- Dictionary
- Operations
- Programs

tuteur.py@gmail.com

CONFIDENTIAL &amp; 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 &amp; 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 &amp; 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 &amp; 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