



**Silver Oak College of Engineering and Technology**

GUJARAT TECHNOLOGICAL UNIVERSITY BACHELOR OF ENGINEERING

PYTHON PROGRAMMING (2180711)

8th SEMESTER

Laboratory Manual

## DEPARTMENT OF COMPUTER ENGINEERING

**VISION**

To create competent professionals in the field of Computer Engineering and promote research with a motive to serve as a valuable resource for the IT industry and society.

**MISSION**

1. To produce technically competent and ethically sound Computer Engineering professionals by imparting quality education, training, hands on experience and value based education.
2. To inculcate ethical attitude, sense of responsibility towards society and leadership ability required for a responsible professional computer engineer.
3. To pursue creative research, adapt to rapidly changing technologies and promote self-learning approach in Computer Engineering and across disciplines to serve the dynamic needs of industry, government and society.

## Program Educational Objectives (PEO):

**PEO1:** To provide the fundamentals of science, mathematics, electronics and computer science and engineering and skills necessary for a successful IT professional.

**PEO2:** To provide scope to learn, apply skills, techniques and competency to use modern engineering tools to solve computational problems.

**PEO3:** To enable young graduates to adapt to the challenges of evolving career opportunities in their chosen fields of career including higher studies, research avenues, entrepreneurial activities etc.

**PEO4:** To inculcate life-long learning aptitude, leadership qualities and teamwork ability with sense of ethics for a successful professional career in their chosen field.

**PROGRAM OUTCOMES (POs)**

**Engineering Graduates will be able to:**

1. **Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life- long learning in the broadest context of technological change.

## PYTHON PROGRAMMING PRACTICAL BOOK DEPARTMENT OF COMPUTER ENGINEERING PREFACE

It gives us immense pleasure to present the second edition of Python Programming Practical Book for the B.E. 4th year students of Silver Oak College of Engineering and Technology.

The theory and laboratory course of Python Programming, at Silver Oak College of Engineering and Technology, Ahmedabad, is designed in such a manner that students can develop the basic understanding of the subject during theory classes and gain the hands-on practical experience during their laboratory sessions. Python Programming being one of the most prominent languages opens the door to much application development involving Functions, Scoping, Abstractions, Mutability, Testing, Assertions, Simple Data Structures and much more.

The Laboratory Manual presented here to you takes you onto learning journey of Python Programming, both at the basic and advanced levels. In this you will be exploring the wide range of topics from Functions, Scoping, Abstractions, Mutability, Testing, Assertions, Simple Data Structures and much more.

Lab Manual Revised by: Prof. Mansi Dave, Silver Oak College of Engineering and Technology

Lab Manual Revision No.: SOCET\_2180711\_LM\_2020\_1

## INSTRUCTIONS TO STUDENTS

* 1. Be prompt in arriving to the laboratory and always come well prepared for the experiment.
  2. Students need to maintain a proper decorum in the computer lab. Students must use the equipment with care. Any damage is caused is punishable.
  3. Students are instructed to come to lab in formal dresses only.
  4. Students are supposed to occupy the systems allotted to them and are not supposed to talk or make noise in the lab.
  5. Students are required to carry their observation book and lab records with completed exercises while entering the lab.
  6. Lab records need to be submitted every week.
  7. Students are not supposed to use pen drives in the lab.
  8. The grades for the PYTHON PROGRAMMING using PYTHON Practical course work will be awarded based on your performance in the laboratory, regularity, recording of experiments in the PYTHON PROGRAMMING using PYTHON practical Final Notebook, lab quiz, regular viva-voce and end-term examination.
  9. **Students needs to perform the experiments and paste the output or image in given space for output, while pasting the output make sure that student’s enrollment number is visible. (For that student’s needs to make folder of their enrollment number and set path up to that folder and then compile and run the program.)**
  10. Find the answers of all the questions mentioned under the section ‘Find the Answers’ at the end of each experiment in the PYTHON PROGRAMMING using PYTHON Practical Book.

**CERTIFICATE**

# This is to certify that Mr./Ms.......................................................................................................................

*with enrolment no. ................................................................from Semester*

*………Div has successfully completed his/her laboratory experiments in the*

***PYTHON PROGRAMMING (2180711)*** *from the department of*

# ...................................................................during the academic year ............... -........

Date of Submission: ......................... Staff Incharge: ...........................

Head of Department: ...........................................

**TABLE OF CONTENT**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr. No** | **Experiment Title** | **Page No** | | **Date of Start** | **Date of Completion** | **Sign** | **Marks (out of 10)** |
| **To** | **From** |
| 1 | Write a program that asks the user to input 10 integers, and then prints the largest odd number that was entered. If no odd number was entered, it should print a message to that effect. | 1 | 3 |  |  |  |  |
| 2 | Write a program that asks the user to enter an integer and prints two integers, root and pwr, such that 0  <pwr< 6 and root\*\*pwr is equal  to the integer entered by the user. If no such pair of integers exists, it should print a message to that  effect. | 4 | 6 |  |  |  |  |
| 3 | Determine the greatest common  divisor and least common multiple of a pair of integers | 7 | 10 |  |  |  |  |
| 4 | Write a program to find maximum and minimum number from  a given list | 11 | 12 |  |  |  |  |
| 5 | Write a program that reads the words in words.txt and stores them as keys in a dictionary and count the frequency of it as a value and print 10 records with highest frequency. | 13 | 14 |  |  |  |  |
| 6 | Write a program that create two tuples and add them into third tuple. | 15 | 16 |  |  |  |  |
| 7 | Write a python script to find Fibonacci recursion. | 17 | 18 |  |  |  |  |
| 8 | Write a program to open the file test.txt and read it line by line. For  each line, split the line into a list of | 19 | 20 |  |  |  |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | words using the split function. For each word, check to see if the word is already in a list. If the word is not in the list, add it to the list. When the program completes, sort and print  the resulting words in alphabetical order. |  |  |  |  |  |  |
| 9 | Write a python program to find whether the entered string is  Palindrome or not. | 21 | 22 |  |  |  |  |
| 10 | Write a python program to show exception handling. | 23 | 24 |  |  |  |  |
| 11 | Write a python script to implement Hash Table. | 25 | 26 |  |  |  |  |
| 12 | Write a program for assertions in python. | 27 | 28 |  |  |  |  |
| 13 | Write a program to implement selection sort in python. | 29 | 30 |  |  |  |  |
| 14 | Write a program to implement bubble sort in python. | 31 | 33 |  |  |  |  |
| 15 | Write a program to implement  quick sort in python. | 34 | 36 |  |  |  |  |
| 16 | Write a program to learn  regular expression in python. | 37 | 39 |  |  |  |  |
| 17 | Write a program to implement chat application in python. | 40 | 42 |  |  |  |  |
| 18 | * 1. Write a program to implement Ceasar cipher in python.   2. Write a program to Implement Transposition cipher in python. | 43 | 46 |  |  |  |  |
| 19 | Write a program to draw Graphics python. Using Turtle  in python | 47 | 48 |  |  |  |  |
| 20 | Write a program to learn GUI  programming using Tkinter in python. | 49 | 50 |  |  |  |  |
| 21 | Write a program to implement  Merge sort in python. | 51 | 56 |  |  |  |  |

## PRACTICAL SET – 1

**Description:**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables. The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.

## References:

[1] <https://www.sanfoundry.com/python-questions-answers-variable-names>

**AIM:** Write a program that asks the user to input 10 integers, and then prints the largest odd number that was entered. If no odd number was entered, it should print a message to that effect.

## Code:

n=int(input("Enter the number of elements to be in the list:"))

b=[]

for i in range(0,n):

a=int(input("Element: "))

b.append(a)

c=[]

d=[]

for i in b:

if(i%2==0):

c.append(i)

else:

d.append(i)

c.sort()

d.sort()

count1=0

count2=0

for k in c:

count1=count1+1

for j in d:

count2=count2+1

print("Largest even number:",c[count1-1])

print("Largest odd number",d[count2-1])

## Output:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## Post Practical Questions:

1. Is Python case sensitive when dealing with identifiers?
2. **yes**
3. no
4. machine dependent
5. none of the mentioned
6. What is the maximum possible length of an identifier?
7. 31 characters
8. 63 characters
9. **79 characters**
10. none of the mentioned
11. Why are local variable names beginning with an underscore discouraged?
12. **they are used to indicate a private variables of a class**
13. they confuse the interpreter
14. they are used to indicate global variables
15. they slow down execution
16. Which of the following is not a keyword?
17. eval
18. assert
19. **nonlocal**
20. pass
21. All keywords in Python are in
22. lower case
23. UPPER CASE
24. Capitalized
25. **None of the mentioned**
26. Which of the following is true for variable names in Python?
27. **unlimited length**
28. all private members must have leading and trailing underscores
29. underscore and ampersand are the only two special characters allowed
30. none of the mentioned

## Conclusion:

We learned about some variable and identifiers of the python and it’s characteristics .

**PRACTICAL SET – 2**

**Description:**

The pow() method takes three parameters:

x - number which is to be powered

y - number which is to be powered with x

z (Optional) - number which is to be used for modulus operation. i.e. pow(x, y) % z.

The range() function is used to generate a sequence of numbers over time. At its simplest, it accepts an integer and returns a range object (a type of iterable). In Python 2, the range() returns a list which is not very efficient to handle large data.

## References:

1. https:/[/www](http://www.tutorialspoint.com/python/number_pow.htm).[tutorialspoint.com/python/number\_pow.htm](http://www.tutorialspoint.com/python/number_pow.htm)
2. <https://www.sanfoundry.com/python-questions-answers-variable-names>

**AIM:** Write a program that asks the user to enter an integer and prints two integers, root and pwr, such that 0 <pwr< 6 and root\*\*pwr is equal to the integer entered by the user. If no such pair of integers exists, it should print a message to that effect.

## Code:

x = int(input("Enter an integer: "))

pwr = 1

while pwr < 6 :

if x < 0:

root = x

else:

root = 0

while root \*\* pwr <= x:

if root \*\* pwr == x:

print(root, " \*\* ", pwr, "=", x)

else:

if x \*\* 1 != x:

print(x, "No such integer pair exists.")

root += 1

pwr += 1

## Output:

## 

## Post Practical Questions:

Write down the output of following code:

1. print(pow(2, 2))

4

1. print(pow(2, -2))

0.25

1. print(pow(-2, 2))

4

4) print(pow(-2, -2))

0.25

5) print (pow(3,4,10))

1

## Conclusion:

We learned about some power function how it’s working in python.

**PRACTICAL SET – 3**

**Description:**

Decision making is required when we want to execute a code only if a certain condition is satisfied.

## Syntax of if...else:

if test expression:

Body of if else:

Body of else

## While Loop:

In python, while loop is used to execute a block of statements repeatedly until a given a condition is satisfied. And when the condition becomes false, the line immediately after the loop in program is executed.

## Syntax :

while expression: statement(s)

## References:

[1] <https://www.sanfoundry.com/python-questions-answers-variable-names>

**AIM:** Determine the greatest common divisor and least common multiple of a pair of integers.

## Code:

def gcd(a,b):

if a == 0:

return b

return gcd(b % a, a)

# Function to return LCM of two numbers

def lcm(a,b):

return (a / gcd(a,b))\* b

# Driver program to test above function

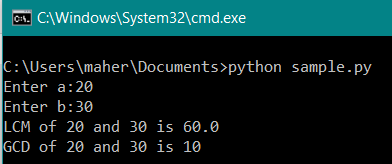
a=int(input("Enter a:"))

b = int(input("Enter b:"))

print('LCM of', a, 'and', b, 'is', lcm(a, b))

print('GCD of', a, 'and', b, 'is', gcd(a, b))

**Output:**



## Post Practical Questions:

1. What is the output of the following? x = ['ab', 'cd']

for i in x: i.upper() print(x)

a) **[‘ab’, ‘cd’].**

b) [‘AB’, ‘CD’].

1. [None, None].
2. none of the mentioned
3. What is the output of the following? i = 1

while True:

if i%3 == 0: break print(i) i + = 1

a) 1 2

b) 1 2 3

1. **error**
2. none of the mentioned
3. What is the output of the following? i = 1

while False:

if i%2 == 0:

break print(i) i += 2

a) 1

**b) 1 3 5 7 …**

c) 1 2 3 4 …

d) none of the mentioned

1. What is the output of the following? True =

False while True: print(True)

break

1. **True**
2. False
3. None
4. none of the mentioned
5. What is the output of the following? i = 2

while True: if i%3 == 0: break print(i)

i += 2

a) 2 4 6 8 10 …

1. 2 4
2. 2 3
3. **error**

## Conclusion:

We learned about some function of python and GCD and LCM, True & Falase etc…..

**PRACTICAL SET – 4**

**Description:**

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming language, and works more like an iterator method as found in other object-oriented programming languages.

With the for loop we can execute a set of statements, once for each item in a list, tuple, set etc.

## References:

[1] <https://github.com/tudelft3d/Solar3Dcity/blob/master/polygon3dmodule.py>

**AIM:** Write a program to find maximum and minimum number position from a given list.

## Code:

## def maxminposition(A, n):

## # inbuilt function to find the position of minimum

## minposition = A.index(min(A))

## # inbuilt function to find the position of maximum

## maxposition = A.index(max(A))

## print ("The maximum is at position::", maxposition + 1)

## print ("The minimum is at position::", minposition + 1)

## # Driver code

## A=list()

## n=int(input("Enter the size of the List ::"))

## print("Enter the Element ::")

## for i in range(int(n)):

## k=int(input(""))

## A.append(k)

## maxminposition(A,n)

## Output:

## 

## Post Practical Questions:

1. What will to following code segment print? if (10 < 0) and (0 < -10): print("A")

elif (10 > 0) or False: print("B") else:

print("C")

1. A
2. **B**
3. C
4. B and C
5. The output of the snippet of code shown below? bool(‘False’)

bool()

1. True True
2. **False True**
3. False False
4. True False
5. What is the output of the code shown below? if (9 < 0) and (0 < -9):

print("hello")

elif (9 > 0) or False:

print("good") else:

print("bad")

1. error
2. hello
3. **good**
4. bad
5. What is the output of the code shown below? l=[1, 0, 2, 0, 'hello', '', []]

list(filter(bool, l))

a) **Error**

b) [1, 0, 2, 0, ‘hello’, ”, []]

c) [1, 0, 2, ‘hello’, ”, []]

d) [1, 2, ‘hello’]

## Conclusion:

We learned about some confitional function like if, else, and elseif(elif).

**PRACTICAL SET – 5**

**Description:**

The key function for working with files in Python is the open() function. The open() function takes two parameters; filename, and mode.

"r" - Read - Default value. Opens a file for reading, error if the file does not exist "a" -

Append - Opens a file for appending, creates the file if it does not exist

"w" - Write - Opens a file for writing, creates the file if it does not exist "x" - Create - Creates the specified file, returns an error if the file exists.

## References:

[1] https:/[/www](http://www.sanfoundry.com/python-questions-answers-variable-names).[sanfoundry.com/python-questions-answers-variable-names](http://www.sanfoundry.com/python-questions-answers-variable-names)

**AIM:** Write a program that reads the words in words.txt and stores them as keys in a dictionary and count the frequency of it as a value and print 10 records with highest frequency.

## Code:

import operator

def countWords(lines):

wordDict = {}

wordList = lines.split(' ') for word in wordList:

if word in wordDict: wordDict[word] += 1 else: wordDict[word] = 1

sorted\_wordDict = sorted(wordDict.items(), key=operator.itemgetter(1),reverse=True) return sorted\_wordDict

def main(): fp=open("words.txt") lines=fp.read() sorted\_wordDict=countWords(lines)

for i in range(1,11): print sorted\_wordDict[i] main()

## Output:

(‘is’,’4’)

(‘hi’,’3’)

(‘big’,’3’)

(‘a’,’2’)

(‘text’,’2’)

(‘this’,’1’)

(‘about’,’1’)

(‘for’,’1’)

(‘I’,’1’)

(‘ ‘,’1’)

## Post Practical Questions:

1. To open a file c:\scores.txt for reading, we use
2. **infile = open(“c:\scores.txt”, “r”)**
3. infile = open(“c:\\scores.txt”, “r”)
4. infile = open(file = “c:\scores.txt”, “r”)
5. infile = open(file = “c:\\scores.txt”, “r”)
6. To open a file c:\scores.txt for writing, we use
7. outfile = open(“c:\scores.txt”, “w”)
8. outfile = open(“c:\\scores.txt”, “w”)
9. **outfile = open(file = “c:\scores.txt”, “w”)**
10. outfile = open(file = “c:\\scores.txt”, “w”)
11. To open a file c:\scores.txt for appending data, we use
12. **outfile = open(“c:\\scores.txt”, “a”)**
13. outfile = open(“c:\\scores.txt”, “rw”)
14. outfile = open(file = “c:\scores.txt”, “w”)
15. outfile = open(file = “c:\\scores.txt”, “w”)
16. Which of the following statements are true?
17. When you open a file for reading, if the file does not exist, an error occurs
18. When you open a file for writing, if the file does not exist, a new file is created
19. When you open a file for writing, if the file exists, the existing file is overwritten with the new file
20. **All of the mentioned**
21. To read two characters from a file object in file, we use
22. **infile.read(2)**
23. infile.read()
24. infile.readline()
25. infile.readlines()
26. What is the output? f = None

for i in range (5):

with open("data.txt", "w") as f: if i > 2: break print(f.closed)

1. True
2. False
3. **None**
4. Error

## Conclusion:

## We learned about file management and it’s basic concept using python.

**PRACTICAL SET – 6**

**Description:**

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5 );

tup3 = "a", "b", "c", "d";

The empty tuple is written as two parentheses containing nothing − tup1 = ();

To write a tuple containing a single value you have to include a comma, even though there is only one value −

tup1 = (50,);

## References:

[1] https:/[/www](http://www.sanfoundry.com/python-questions-answers-variable-names).[sanfoundry.com/python-questions-answers-variable-names](http://www.sanfoundry.com/python-questions-answers-variable-names)

**AIM:** Write a program that create two tuples and add them into third tuple.

## Code:

tup1 = (12, 34.56);

tup2 = ('abc', 'xyz');

# Following action is not valid for tuples # tup1[0] = 100; # So let's create a new tuple as follows

tup3 = tup1 + tup2;

print(tup3)

## Output:

(12, 34.56, 'abc', 'xyz')

## Post Practical Questions:

1. Which of the following is a Python tuple? a) [1, 2, 3].

b) **(1, 2, 3**)

c) {1, 2, 3}

d) {}

1. Suppose t = (1, 2, 4, 3), which of the following is incorrect?

a) print(t[3])

b) **t[3] = 45**

1. print(max(t))
2. print(len(t))
3. What is the data type of (1)?
4. Tuple
5. Integer
6. List
7. **Both tuple and integer**

4. What type of data is: a=[(1,1),(2,4),(3,9)]?

1. Array of tuples
2. **List of tuples**
3. Tuples of lists
4. Invalid type

5. What is the output of the following code?

>>> a=(1,2,(4,5))

>>> b=(1,2,(3,4))

>>> a<b

1. **False**
2. True
3. Error, < operator is not valid for tuples
4. Error, < operator is valid for tuples but not if there are sub-tuples

## Conclusion:

## We learned about tuples and it’s function like tuple type and tuple length etc….

**PRACTICAL SET -7**

**Description:**

A Fibonacci sequence is the integer sequence of 0, 1, 1, 2, 3, 5, 8....

The first two terms are 0 and 1. All other terms are obtained by adding the preceding two terms. This means to say the nth term is the sum of (n-1)th and (n-2)th term.

In Python, a [function](https://www.programiz.com/python-programming/function) can call other functions. It is even possible for the function to call itself. These type of construct are termed as recursive functions.

A recursive function is used to calculate the nth term of the sequence.

## References:

[1] h[ttp://www.programiz.com/python-programming/examples/fibonacci-sequence](https://www.programiz.com/python-programming/examples/fibonacci-sequence)

**AIM:** Write a python script to find Fibonacci sequence using recursion.

## Code:

## def fibonacci(n):

## if(n <= 1):

## return n

## else:

## return(fibonacci(n-1) + fibonacci(n-2))

## n = int(input("Enter number of terms:"))

## print("Fibonacci sequence:")

## for i in range(n):

## print(fibonacci(i))

## Output: Enter Number of term: 7

## Finonacci series is 0 1 1 2 3 5 8

## Post Practical Questions:

1. Suppose the first fibonnaci number is 0 and the second is 1. What is the sixth fibonnaci number?
2. **5**
3. 6
4. 7
5. 8
6. Which of the following is not a fibonnaci number?
7. 8
8. 21
9. 55
10. **14**
11. Which of the following methods can be used to find the nth fibonnaci number?
12. Dynamic programming
13. Recursion
14. Iteration
15. **All of the mentioned**
16. Which of the following recurrence relations can be used to find the nth fibonacci number? a) F(n) = F(n) + F(n – 1)

b) F(n) = F(n) + F(n + 1)

c) F(n) = F(n – 1)

d) **F(n) = F(n – 1) + F(n – 2)**

1. What is the space complexity of the above recursive implementation to find the nth fibonacci number?
2. **O(1)**
3. O(2\*n)



1. O(n2)
2. O(2n)

## Conclusion:

We Found Fibonacci series by python code.

**PRACTICAL SET -8**

**Description:**

Split() method returns a list of strings after breaking the given string by the specified separator. At some point, you may need to break a large string down into smaller chunks, or strings. This is the opposite of concatenation which merges or combines strings into one.

To do this, you use the split function. What it does is split or breakup a string and add the data to a string array using a defined separator.

If no separator is defined when you call upon the function, whitespace will be used by default. In simpler terms, the separator is a defined character that will be placed between each variable.

## References:

[1] https:/[/www](http://www.pythonforbeginners.com/dictionary/python-split).[pythonforbeginners.com/dictionary/python-split](http://www.pythonforbeginners.com/dictionary/python-split)

**AIM:** Write a program to open the file test.txt and read it line by line. For each line, split the line into a list of words using the split function. For each word, check to see if the word is already in a list. If the word is not in the list, add it to the list. When the program completes, sort and print the resulting words in alphabetical order.

## Code:

lst=[]

from sys import argv script,filename=argv fp=open(filename) txt=fp.read() word=txt.split(' ') for e in word:

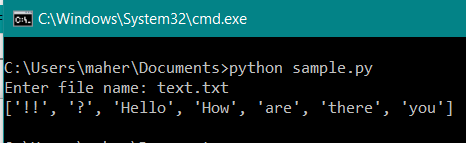
if e in lst:

continue else:

lst+=[e] lst.sort()

print lst

## Output:



## Post Practical Questions:

1. Which module in Python supports regular expressions?
2. **re**
3. regex
4. pyregex
5. none of the mentioned
6. Which of the following creates a pattern object?
7. re.create(str)
8. re.regex(str)
9. **re.compile(str)**
10. re.assemble(str)
11. What does the function re.match do?
12. **matches a pattern at the start of the string**
13. matches a pattern at any position in the string
14. such a function does not exist
15. none of the mentioned
16. What does the function re.search do?
17. matches a pattern at the start of the string
18. **matches a pattern at any position in the string**
19. such a function does not exist
20. none of the mentioned
21. The character Dot (that is, ‘.’) in the default mode, matches any character other than
22. **caret**
23. ampersand
24. percentage symbol
25. newline

## Conclusion:

We learned to use list and for loop in python.

**PRACTICAL SET -9**

**Description:**

A palindrome is a string which is same read forward or backwards.

For example: "dad" is the same in forward or reverse direction. Another example is "aibohphobia" which literally means, an irritable fear of palindromes.

The program takes a string and checks if a string is a palindrome or not. Problem Solution. Take a string from the user and store it in a variable.

Program/Source Code. Here is source code of the Python Program to check if a string is a palindrome or not.

## References:

[1] <http://www.codequiz.in/python-program-to-check-whether-the-no-is-palindrome-or-not>

**AIM:** Write a python program to find whether the entered string is Palindrome or not.

## Code:

my\_str = 'naman'

rev\_str = reversed(my\_str)

if list(my\_str) == list(rev\_str):

print("It is palindrome")

else:

print("It is not palindrome")

## Output:

## Post Practical Questions:

1. Compare Java and Python:

Java is a compile language when python is a interpreted language. Java is a statically typed when python is dynamically typed. Java complex learning curve when python is easy to learn. Java programs runs slowly than python programs. Multiple inheritances is partially done with interface when python offers both single and multiple interface. Java has more lines programs compare to python.

1. What is Python?

Python is a interpreted, high-level and general purpose programming language.

1. What is the purpose of PYTHONPATH environment variable?

PYTHONPATH is an environment variable which you can set to add additional directories where python will look for modules and packages.

1. What are the supported data types in python?

Integer, float-point Number and complex numbers and strings are the supported data types in python.

1. What is the difference between list and tuples?

One of the most important differences between list and tuple is that list is mutable, whereas a tuple is immutable. This means that lists can be changed, and tuples cannot be changed.

## Conclusion:

We done palindrome program and some basic about python and difference between java and python.

**PRACTICAL SET -10**

**Description:**

Python has many built-in exceptions which forces your program to output an error when something in it goes wrong.

When these exceptions occur, it causes the current process to stop and passes it to the calling process until it is handled. If not handled, our program will crash.

In Python, exceptions can be handled using a try statement.

A critical operation which can raise exception is placed inside the try clause and the code that handles exception is written in except clause.

## References:

[1] https:/[/www](http://www.sanfoundry.com/python-questions-answers-exception-handling).[sanfoundry.com/python-questions-answers-exception-handling](http://www.sanfoundry.com/python-questions-answers-exception-handling)

**AIM:** Write a python program to show exception handling.

## Code:

import sys

randomList = ['a', 0, 2]

for entry in randomList:

try:

print("The entry is", entry)

r = 1/int(entry)

break

except:

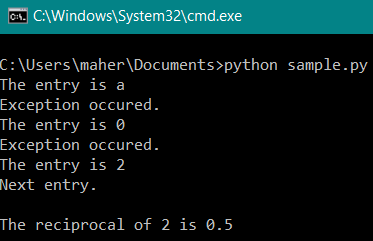
print("Exception occured.")

print("Next entry.")

print()

print("The reciprocal of",entry,"is",r)

## Output:



## Post Practical Questions:

1. How many except statements can a try-except block have?
2. zero
3. **one**
4. more than one
5. more than zero
6. When will the else part of try-except-else be executed?
7. always
8. when an exception occurs
9. **when no exception occurs**
10. when an exception occurs in to except block
11. Can one block of except statements handle multiple exception?
12. **yes, like except TypeError, SyntaxError [,…].**
13. yes, like except [TypeError, SyntaxError].
14. no
15. none of the mentioned
16. When is the finally block executed?
17. when there is no exception
18. when there is an exception
19. only if some condition that has been specified is satisfied
20. **always**
21. What happens when ‘1’ == 1 is executed?
22. we get a True
23. **we get a False**
24. an TypeError occurs
25. a ValueError occurs

## Conclusion:

We learned about exception handling.

**PRACTICAL SET -11**

**Description:**

Hash Table is a data structure where data are stored in an associative manner (in key, value format). The key/index is unique. This kind of storage makes it easier to find the data later on.

Hash Table stores data into an array format. It uses a hashing function that generates a slot or an index to store/insert any element or value.

This deals with generating slot or index to any “key” value. Perfect hashing or perfect hash function is the one which assigns a unique slot for every key value. Sometimes, there can be cases where the hash function generates the same index for multiple key values. The size of the hash table can be increased to improve the perfection of the hash function.

## References:

[1] https:/[/www](http://www.sanfoundry.com/data-structure-questions-answers-hash-tables/).[sanfoundry.com/data-structure-questions-answers-hash-tables/](http://www.sanfoundry.com/data-structure-questions-answers-hash-tables/)

**AIM:** Write a python script to implement Hash Table.

## Code:

#For this demonstration we use a simple hash function, using the modulus operator such that #h(x)=x mod k where k is the table size.

def hash\_function(x):

return x % 10

def insert(table,input,value):

table[hash\_function(input)] = value insert(table,41,'apple') insert(table,93,'banana') insert(table,78,'guava')

print table

## Output:

[0, 'apple', 0, 'banana', 0, 0, 0, 0, 'guava', 0]

## Post Practical Questions:

1. What is a hash table?
2. A structure that maps values to keys
3. **A structure that maps keys to values**
4. A structure used for storage
5. A structure used to implement stack and queue
6. If several elements are competing for the same bucket in the hash table, what is it called?
7. Diffusion
8. Replication
9. **Collision**
10. None of the mentioned
11. What is direct addressing?
12. **Distinct array position for every possible key**
13. Fewer array positions than keys
14. Fewer keys than array positions
15. None of the mentioned
16. What is the search complexity in direct addressing?
17. O(n)
18. O(logn)
19. O(nlogn)
20. **O(1)**
21. What is a hash function?
22. A function has allocated memory to keys
23. **A function that computes the location of the key in the array**
24. A function that creates an array
25. None of the mentioned

## Conclusion:

We learned to create hash table.

**PRACTICAL SET -12**

**Description:**

Assertions are statements that assert or state a fact confidently in your program. For example, while writing a division function, you're confident the divisor shouldn't be zero, you assert divisor is not equal to zero.

Assertions are simply boolean expressions that checks if the conditions return true or not. If it is true, the program does nothing and move to the next line of code. However, if it's false, the program stops and throws an error.

It is also a debugging tool as it brings the program on halt as soon as any error is occurred and shows on which point of the program error has occurred.

## References:

[1] https:/[/www](http://www.programiz.com/python-programming/assert-statement).[programiz.com/python-programming/assert-statement](http://www.programiz.com/python-programming/assert-statement)

**AIM:** Write a program for assertions in python.

## Code:

def avg(marks):

assert len(marks) != 0,"List is empty."

return sum(marks)/len(marks)

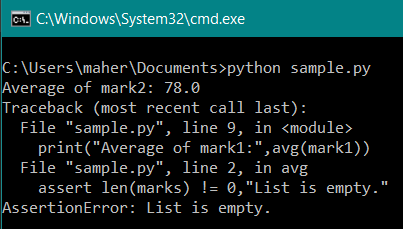
mark2 = [55,88,78,90,79]

print("Average of mark2:",avg(mark2))

mark1 = []

print("Average of mark1:",avg(mark1))

## Output:



## Post Practical Questions:

1. What is assertion in python?

The assert keyword is used when debugging code. The **assert** keyword lets you test if a condition in your code returns True, if not, the program will raise an AssertionError.

1. Syntax of assert statement in python

Assert <condition>, <error message>.

1. What is assertion condition in python

Condition in the assertion is that is always supposed to be the truth in the code.

1. What is AssertionError?

When such an assert statement fails (i.e. returns a False-y value), an AssertionError is raised.

1. Can we use assert without error message.

No.

## Conclusion

We learned use of assertion command in python.

**PRACTICAL SET - 13**

**Description:**

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison- based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of Ο(n2), where n is the number of items.

For Eg.

arr[] = 64 25 12 22 11

Find the minimum element in arr[0...4] and place it at beginning 11 25 12 22 64

Find the minimum element in arr[1...4] and place it at beginning of arr[1. 4] 11 12 25 22 64

Find the minimum element in arr[2...4] and place it at beginning of arr[2. 4] 11 12 22 25 64

Find the minimum element in arr[3...4] and place it at beginning of arr[3. 4] 11 12 22 25 64

**References:** [1]https:/[/www](http://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm).[tutorialspoint.com/data\_structures\_algorithms/selection\_sort\_algorithm.htm](http://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm) [2]<https://www.geeksforgeeks.org/selection-sort/>

**AIM:** Write a program to implement Selection Sort in python.

## Code:

# Selection sort in Python

def selectionSort(array, size):

for step in range(size):

min\_idx = step

for i in range(step + 1, size)

# to sort in descending order, change > to < in this line

# select the minimum element in each loop

if array[i] < array[min\_idx]:

min\_idx = i

# put min at the correct position

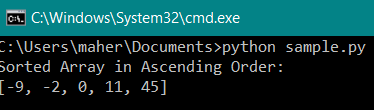
(array[step], array[min\_idx]) = (array[min\_idx], array[step])

data = [-2, 45, 0, 11, -9]

size = len(data)

selectionSort(data, size)

print('Sorted Array in Ascending Order:')

print(data)

## Output:

## Post Practical Questions:

1. In the following scenarios, when will you use selection sort?
2. The input is already sorted
3. A large file has to be sorted
4. **Large values need to be sorted with small keys**
5. Small values need to be sorted with large keys
6. What is the worst case complexity of selection sort?
7. O(nlogn)
8. O(logn)
9. O(n)
10. **O(n2)**
11. What is the advantage of selection sort over other sorting techniques?
12. **It requires no additional storage space**
13. It is scalable
14. It works best for inputs which are already sorted
15. It is faster than any other sorting technique
16. What is the average case complexity of selection sort?
17. O(nlogn)
18. O(logn)
19. O(n)
20. **O(n2)**
21. What is the disadvantage of selection sort?
22. It requires auxiliary memory
23. **It is not scalable**
24. It can be used for small keys
25. None of the mentioned

## Conclusion:

We learned to implement of selection sort in python.

**PRACTICAL SET – 14**

**Description:**

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.

For Eg.

## First Pass:

( 5 1 4 2 8 ) –> ( 1 5 4 2 8 ), Here, algorithm compares the first two elements, swaps since 5 >

1.

( 1 5 4 2 8 ) –> ( 1 4 5 2 8 ), Swap since 5 > 4

( 1 4 5 2 8 ) –> ( 1 4 2 5 8 ), Swap since 5 > 2

( 1 4 2 5 8 ) –> ( 1 4 2 5 8 ), Now, since these elements are already in order (8 > 5), algorithm does not swap them.

## Second Pass:

( 1 4 2 5 8 ) –> ( 1 4 2 5 8 )

( 1 4 2 5 8 ) –> ( 1 2 4 5 8 ), Swap since 4 > 2

( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )

( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )

Now, the array is already sorted, but our algorithm does not know if it is completed. The algorithm needs one whole pass without any swap to know it is sorted.

## Third Pass:

( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )

( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )

( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )

( 1 2 4 5 8 ) –> ( 1 2 4 5 8 )

**References:** [1]https:/[/www](http://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm).[tutorialspoint.com/data\_structures\_algorithms/selection\_sort\_algorithm.htm](http://www.tutorialspoint.com/data_structures_algorithms/selection_sort_algorithm.htm) [2]https:/[/www](http://www.geeksforgeeks.org/bubble-sort/).[geeksforgeeks.org/bubble-sort/](http://www.geeksforgeeks.org/bubble-sort/)

**AIM:** Write a program to implement Bubble Sort in python.

## Code:

# Creating a bubble sort function

def bubble\_sort(list1):

    # Outer loop for traverse the entire list

    for i in range(0,len(list1)-1):

        for j in range(len(list1)-1):

            if(list1[j]**>**list1[j+1]):

                temp = list1[j]

                list1[j] = list1[j+1]

                list1[j+1] = temp

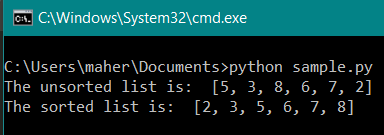
    return list1

list1 = [5, 3, 8, 6, 7, 2]

print("The unsorted list is: ", list1)

print("The sorted list is: ", bubble\_sort(list1))

## Output:



## Post Practical Questions:

1. What is the worst case complexity of bubble sort
2. O(nlogn)
3. O(logn)
4. O(n)
5. **O(n2)**
6. What is the advantage of bubble sort over other sorting techniques?
7. It is faster
8. Consumes less memory
9. **Detects whether the input is already sorted**
10. All of the mentioned
11. The given array is arr = {1,2,4,3}. Bubble sort is used to sort the array elements. How many iterations will be done to sort the array with improvised version?
12. **4**
13. 2
14. 1
15. 0
16. What is the average case complexity of bubble sort?
17. O(nlogn)
18. O(logn)
19. O(n)
20. **O(n2)**
21. What is the best case efficiency of bubble sort in the improvised version?
22. O(nlogn)
23. O(logn)
24. **O(n)**
25. O(n2)

## Conclusion:

We learned to implement of bubble short algorithm using python.

**PRACTICAL SET – 15**

**Description:**

Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value.

Quick sort partitions an array and then calls itself recursively twice to sort the two resulting subarrays. This algorithm is quite efficient for large-sized data sets as its average and worst case complexity are of Ο(n2), where n is the number of items.

It is also called partition-exchange sort. This algorithm divides the list into three main parts:

1. Elements less than the Pivot element
2. Pivot element(Central element)
3. Elements greater than the pivot element

Pivot element can be any element from the array, it can be the first element, the last element or any random element. In this tutorial, we will take the rightmost element or the last element as pivot

For example: In the array {52, 37, 63, 14, 17, 8, 6, 25}, we take 25 as pivot. So after the first pass, the list will be changed like this.

{6 8 17 14 25 63 37 52}

Hence after the first pass, pivot will be set at its position, with all the elements smaller to it on its left and all the elements larger than to its right. Now 6 8 17 14 and 63 37 52 are considered as two separate sunarrays, and same recursive logic will be applied on them, and we will keep doing this until the complete array is sorted.

## References:

1. https:/[/www](http://www.tutorialspoint.com/data_structures_algorithms/quick_sort_algorithm.htm).[tutorialspoint.com/data\_structures\_algorithms/quick\_sort\_algorithm.htm](http://www.tutorialspoint.com/data_structures_algorithms/quick_sort_algorithm.htm)
2. https:/[/www](http://www.studytonight.com/data-structures/quick-sort).[studytonight.com/data-structures/quick-sort](http://www.studytonight.com/data-structures/quick-sort)

**AIM:** Write a program to implement Quick Sort in python.

## Code:

# divide function

def partition(arr,low,high):

i = ( low-1 )

pivot = arr[high] # pivot element

for j in range(low , high):

# If current element is smaller

if arr[j] <= pivot:

# increment

i = i+1

arr[i],arr[j] = arr[j],arr[i]

arr[i+1],arr[high] = arr[high],arr[i+1]

return ( i+1 )

# sort

def quickSort(arr,low,high):

if low < high:

# index

pi = partition(arr,low,high)

# sort the partitions

quickSort(arr, low, pi-1)

quickSort(arr, pi+1, high)

# main

arr = [2,5,3,8,6,5,4,7]

n = len(arr)

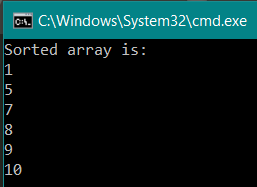
quickSort(arr,0,n-1)

print ("Sorted array is:")

for i in range(n):

print (arr[i],end=" ")

## Output:



## Post Practical Questions:

1. What is the Worst case time complexity in Quick Sort?
2. O(nlogn)
3. O(logn)
4. O(n)
5. **O(n2)**
6. Which of the following is not a stable sorting algorithm in its typical implementation.
7. Insertion Sort
8. **Quick Sort**
9. Bubble Sort
10. Merge Sort
11. Suppose we are sorting an array of eight integers using quicksort, and we have just finished the first partitioning with the array looking like this:

2 5 1 7 9 12 11 10

1. **The Pivot can be either 7 or 9.**
2. The pivot could be the 7, but it is not the 9
3. The pivot is not the 7, but it could be the 9
4. Neither the 7 nor the 9 is the pivot.
5. You have to sort 1 GB of data with only 100 MB of available main memory. Which sorting technique will be most appropriate?
6. Insertion Sort
7. Quick Sort
8. Bubble Sort
9. **Merge Sort**
10. n quick sort, for sorting n elements, the (n/4)th smallest element is selected as pivot using an O(n) time algorithm. What is the worst case time complexity of the quick sort?
11. O(nlogn)
12. O(logn)
13. **O(n)**
14. O(n2)

## Conclusion:

We learned to implement of quick sort using python.

**PRACTICAL SET – 16**

**Description:**

A regular expression is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern. Regular expressions are widely used in UNIX world. The module re provides full support for Perl-like regular expressions in Python. The re module raises the exception re.error if an error occurs while compiling or using a regular expression.

There are various characters, which would have special meaning when they are used in regular expression. To avoid any confusion while dealing with regular expressions, we would use Raw Strings as expression'.

The match Function

This function attempts to match RE pattern to string with optional flags. Here is the syntax for this function − re.match(pattern, string, flags=0) Here is the description of the parameters−

|  |  |
| --- | --- |
| Sr.No  . | Parameter & Description |
| 1 | pattern - This is the regular expression to be matched. |
| 2 | string - This is the string, which would be searched to match the  pattern at the beginning of string. |
| 3 | flags - You can specify different flags using bitwise OR (|). These are  modifiers, which are listed in the table below. |

The re.match function returns a match object on success, None on failure. We usegroup(num) or groups() function of match object to get matched expression.

|  |  |
| --- | --- |
| Sr.No  . | Match Object Method & Description |
| 1 | group(num=0) - This method returns entire match (or specific subgroup num) |
| 2 | groups() - This method returns all matching subgroups in a tuple  (empty if there weren't any) |

## References:

[1] https:/[/www](http://www.tutorialspoint.com/python/python_reg_expressions.htm).[tutorialspoint.com/python/python\_reg\_expressions.htm](http://www.tutorialspoint.com/python/python_reg_expressions.htm)

**AIM:** Write a program to learn regular expression in python.

## Code:

import re

pattern = '^a...s$'

test\_string = 'abyss'

result = re.match(pattern, test\_string)

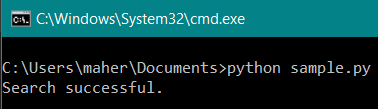
if result:

print("Search successful.")

else:

print("Search unsuccessful.")

## Output:



## Post Practical Questions:

1. Which module in Python supports regular expressions?
2. **re**
3. regex
4. pyregex
5. none of the mentioned
6. Which of the following creates a pattern object?
7. **re.create(str)**
8. re.regex(str
9. re.compile(str)
10. re.assemble(str)
11. What does the function re.match do?
12. matches a pattern at the start of the string
13. **matches a pattern at any position in the string**
14. such a function does not exist
15. none of the mentioned
16. What does the function re.search do?
17. matches a pattern at the start of the string
18. **matches a pattern at any position in the string**
19. such a function does not exist
20. none of the mentioned
21. What is the output of the following?

sentence matched = 'we are humans' sentence)

= re.match(r'(.\*) (.\*?) (.\*)', print(matched.groups())

1. **(‘we’, ‘are’, ‘humans’)**
2. (we, are, humans)
3. (‘we’, ‘humans’)
4. ‘we are humans’

## Conclusion:

We learn to use of regular expression in python.

**PRACTICAL SET - 17**

**Description:**

A Chat Application code uses the concept of Sockets and Threading.

## Socket programming

Sockets can be thought of as endpoints in a communication channel that is bi-directional, and establishes communication between a server and one or more clients. Here, we set up a socket on each end and allow a client to interact with other clients via the server. The socket on the server side associates itself with some hardware port on the server side. Any client that has a socket associated with the same port can communicate with the server socket.

## Multi-Threading

A thread is sub process that runs a set of commands individually of any other thread. So, every time a user connects to the server, a separate thread is created for that user and communication from server to client takes place along individual threads based on socket objects created for the sake of identity of each client.

We will require two scripts to establish this chat room. One to keep the serving running, and another that every client should run in order to connect to the server.

## References:

[1] https:/[/www](http://www.geeksforgeeks.org/simple-chat-room-using-python/).[geeksforgeeks.org/simple-chat-room-using-python/](http://www.geeksforgeeks.org/simple-chat-room-using-python/)

## AIM: Write a program to implement chat application in python.

**Code:**

**Server Side Code:**

import time, socket, sys

print('Setup Server...')

time.sleep(1)

#Get the hostname, IP Address from socket and set Port

soc = socket.socket()

host\_name = socket.gethostname()

ip = socket.gethostbyname(host\_name)

port = 1234

soc.bind((host\_name, port))

print(host\_name, '({})'.format(ip))

name = input('Enter name: ')

soc.listen(1) #Try to locate using socket

print('Waiting for incoming connections...')

connection, addr = soc.accept()

print("Received connection from ", addr[0], "(", addr[1], ")\n")

print('Connection Established. Connected From: {}, ({})'.format(addr[0], addr[0]))

#get a connection from client side

client\_name = connection.recv(1024)

client\_name = client\_name.decode()

print(client\_name + ' has connected.')

print('Press [bye] to leave the chat room')

connection.send(name.encode())

while(True):

message = input('Me > ')

if message == '[bye]':

message = 'Good Night...'

connection.send(message.encode())

print("\n")

breakconnection.send(message.encode())

message = connection.recv(1024)

message = message.decode()

print(client\_name, '>', message)

**Client Side Code:**

import time, socket, sys

print('Client Server...')

time.sleep(1)

#Get the hostname, IP Address from socket and set Port

soc = socket.socket()

shost = socket.gethostname()

ip = socket.gethostbyname(shost)

#get information to connect with the server

print(shost, '({})'.format(ip))

server\_host = input('Enter server\'s IP address:')

name = input('Enter Client\'s name: ')

port = 1234

print('Trying to connect to the server: {}, ({})'.format(server\_host, port))

time.sleep(1)

soc.connect((server\_host, port))

print("Connected...\n")

soc.send(name.encode())

server\_name = soc.recv(1024)

server\_name = server\_name.decode()

print('{} has joined...'.format(server\_name))

print('Enter [bye] to exit.')

while(True):

message = soc.recv(1024)

message = message.decode()

print(server\_name, ">", message)

message = input(str("Me > "))

if message == "[bye]":

message = "Leaving the Chat room"

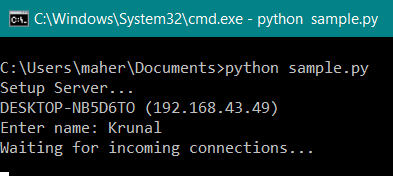
soc.send(message.encode())

print("\n")

break

soc.send(message.encode())

## Output:



**Post Practical Questions:**

1. Which methods are commonly used in Server Socket class?
2. Public Output Stream get Output Stream ()
3. Public Socket accept ()
4. Public synchronized void close ()
5. None of the mentioned
6. Which constructor of Datagram Socket class is used to create a datagram socket and binds it with the given Port Number?
7. Datagram Socket(int port)
8. Datagram Socket(int port, Int Address address)
9. Datagram Socket()
10. None of the mentioned
11. The client in socket programming must know which information?
12. IP address of Server
13. Port number
14. Both IP address of Server & Port number
15. None of the mentioned
16. What is multithreaded programming?
17. It’s a process in which two different processes run simultaneously
18. It’s a process in which two or more parts of same process run simultaneously
19. It’s a process in which many different process are able to access same information
20. It’s a process in which a single process can access information from manysources
21. Which of these are types of multitasking?
22. Process based
23. Thread based
24. Process and Thread based
25. None of the mentioned

## Conclusion:

We learned to built chat application in python.

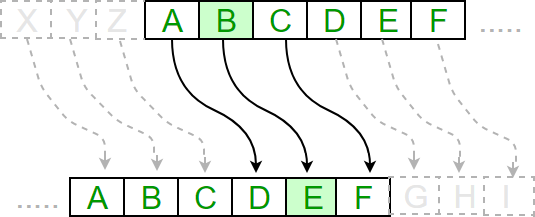
**PRACTICAL SET-18**

**Description:**

The Caesar Cipher technique is one of the earliest and simplest method of encryption technique. It’s simply a type of substitution cipher, i.e., each letter of a given text is replaced by a letter some fixed number of positions down the alphabet. For example with a shift of 1, A would be replaced by B, B would become C, and so on. The method is apparently named after Julius Caesar, who apparently used it to communicate with his officials.Thus to cipher a given text we need an integer value, known as shift which indicates the number of position each letter of the text has been moved down.

The encryption can be represented using modular arithmetic by first transforming the letters into numbers, according to the scheme, A = 0, B = 1,…, Z = 25. Encryption of a letter by a shift n can be described mathematically as.

En(x) = (x+n)mod26 (Encryption Phase with shift n) Dn(x)= (x-n)mod26(Decryption Phase with shift n)



## References :

[1]https:/[/www](http://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_ca).[tutorialspoint.com/cryptography\_with\_python/cryptography\_with\_python\_ca](http://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_ca) esa r\_cipher.htm

**AIM 18.1. :** Write a program to implement Caesar cipher in python.

## Code:

def encrypt(text,s):

result = ""

# transverse the plain text

for i in range(len(text)):

char = text[i]

# Encrypt uppercase characters in plain text

if (char.isupper()):

result += chr((ord(char) + s-65) % 26 + 65)

# Encrypt lowercase characters in plain text

else:

result += chr((ord(char) + s - 97) % 26 + 97)

return result

#check the above function

text = "CEASER CIPHER DEMO"

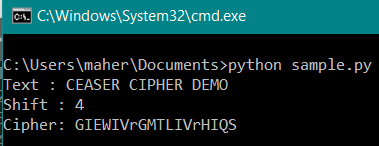
s = 4

print( "Plain Text : " + text)

print ("Shift pattern : " + str(s))

print ("Cipher: " + encrypt(text,s))

**Output:**



## TRANSPOSITION CYPHER

Transposition Cipher is a cryptographic algorithm where the order of alphabets in the plaintext is rearranged to form a cipher text. In this process, the actual plain text alphabets are not included.

## Example :

A simple example for a transposition cipher is columnar transposition cipher where each character in the plain text is written horizontally with specified alphabet width. The cipher is written vertically, which creates an entirely different cipher text.

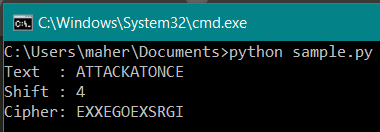
Consider the plain text hello world, and let us apply the simple columnar transposition technique as shown below

The plain text characters are placed horizontally and the cipher text is created with vertical format as : holewdlo lr. Now, the receiver has to use the same table to decrypt the cipher text to plain text.

**References:** [1][https://www.tutorialspoint.com/cryptography\_with\_python/cryptography\_with\_python\_tra](https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_trans%20position_cipher.htm) [ns position\_cipher.htm](https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_trans%20position_cipher.htm)

**AIM 18.2** Write a program to implement Transposition cipher in python

## Code:



**Post Practical Questions:**

1. The is the message after transformation.
2. **ciphertext**
3. plaintext
4. secret-text
5. none of the above
6. The is the message after transformation.
7. **ciphertext**
8. plaintext
9. secret-text
10. none of the above
11. The attack can endanger the security of the Diffie-Hellman method if two parties are not authenticated to each other.
12. **man-in-the-middle**
13. ciphertext attack
14. plaintext attack
15. none of the above
16. A combination of an encryption algorithm and a decryption algorithm is called a .
17. **cipher**
18. secret
19. key
20. none of the above
21. In an asymmetric-key cipher, the receiver uses the key.
22. **private**
23. public
24. either a or b
25. neither (a) nor (b)

## Conclusion:

We learned to implement ceaser cipher and transposition cipher using python.

**PRACTICAL SET - 19**

**Description:**

TURTLE IN PYTHON

* “Turtle” is a Python feature like a drawing board, which lets us command a turtle to draw all over it! We can use functions like turtle.forward(…) and turtle.right(…) which can move the turtle around.

**AIM :** Write a program to draw graphics using Turtle in python.

## Code :

import turtle turtle.forward(100) turtle.left(90) turtle.forward(50) turtle.left(90) turtle.forward(100) turtle.left(90) turtle.forward(50) turtle.left(90)

## Output :

**Post Practical Work:**

1. What is the output shape of the code shown? import turtle

t=turtle.Pen()

for i in range(0,4):

t.forward(100) t.left(120)

1. square
2. rectangle
3. **triangle**
4. kite
5. The command which helps us to reset the pen (turtle):
6. **turtle.reset**
7. turtle.penreset
8. turtle.penreset()
9. turtle.reset()
10. What will be the output shape of the code shown below? import turtle

t=turtle.Pen()

for i in range(1,4):

t.forward(60) t.left(90)

1. Rectangle
2. **Trapezium**
3. Triangle
4. Square
5. Which of the following functions can be used to make the arrow black?
6. turtle.color(0,1,0)
7. turtle.color(1,0,0)
8. turtle.color(0,0,1)
9. **turtle.color(0,0,0)**
10. In which direction is the turtle pointed by default?
11. **North**
12. South
13. East
14. West
15. The command used to set only the x coordinate of the turtle at 45 units is:
16. reset(45)
17. **setx(45)**
18. xset(45)
19. xreset(45)
20. Which of the following functions returns a value in degrees, counterclockwise from the horizontal right?
21. **heading()**
22. degrees()
23. position()
24. window\_height()

## Conclusion:

We learned turtle in python.

**PRACTICAL SET - 20**

**Description:**

**TKINTER PROGRAMMING**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the Tkinter module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.

## References :

[1] https:/[/www](http://www.tutorialspoint.com/python/python_gui_programming.htm).[tutorialspoint.com/python/python\_gui\_programming.htm](http://www.tutorialspoint.com/python/python_gui_programming.htm)

**AIM :** Write a program to learn GUI programming using Tkinter in python.

## Code :

## import tkinter

## top = tkinter.Tk()

## C = tkinter.Canvas(top, bg="blue", height=250, width=300)

## coord = 10, 50, 240, 210

## arc = C.create\_arc(coord, start=0, extent=150, fill="red")

## C.pack()

## top.mainloop()

## Output:

**Post Practical Questions:**

1. How do you create a window?
2. window = newWindow()
3. window = Window()
4. window = Frame()
5. **window = Tk()**
6. How do you create a frame?
7. window = newWindow()
8. window = Window()
9. **window = Frame()**
10. window = Tk()
11. How do you create an event loop?
12. window.loop()
13. window.main()
14. window.mainloop()
15. **window.eventloop()**
16. To create a label under parent window, use .
17. label = Label(text = "Welcome to Python")
18. **label = Label(window, text = "Welcome to Python")**
19. label = Label(text = "Welcome to Python", fg = " red")
20. label = Label(text = "Welcome to Python", fg = " red", bg = "white")
21. To create a button under parent window with command processButton, use .
22. **Button(text = "OK", fg = "red", command = processButton)**
23. Button(window, text = "OK", fg = "red")
24. Button(window, text = "OK", fg = "red")
25. Button(window, text = "OK", command = processButton)

## Conclusion:

We learned to implement GUI programming using tkinter using python.

**PRACTICAL SET - 21**

**Description:**

Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. The merge() function is used for merging two halves. The merge(arr, l, m, r) is key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one.

## References:

[1] <http://interactivepython.org/courselib/static/pythonds/SortSearch/TheMergeSort.html>

**AIM:** Write a program to implement Merge sort in python.

## Code:

# funtion to divide the lists in the two sublists

def merge\_sort(list1, left\_index, right\_index):

if left\_index >= right\_index:

return

middle = (left\_index + right\_index)//2

merge\_sort(list1, left\_index, middle)

merge\_sort(list1, middle + 1, right\_index)

merge(list1, left\_index, right\_index, middle)

# Defining a function for merge the list

def merge(list1, left\_index, right\_index, middle):

# Creating subparts of a lists

left\_sublist = list1[left\_index:middle + 1]

right\_sublist = list1[middle+1:right\_index+1]

# Initial values for variables that we use to keep

# track of where we are in each list1

left\_sublist\_index = 0

right\_sublist\_index = 0

sorted\_index = left\_index

# traverse both copies until we get run out one element

while left\_sublist\_index < len(left\_sublist) and right\_sublist\_index < len(right\_sublist):

# If our left\_sublist has the smaller element, put it in the sorted

# part and then move forward in left\_sublist (by increasing the pointer)

if left\_sublist[left\_sublist\_index] <= right\_sublist[right\_sublist\_index]:

list1[sorted\_index] = left\_sublist[left\_sublist\_index]

left\_sublist\_index = left\_sublist\_index + 1

# Otherwise add it into the right sublist

else:

list1[sorted\_index] = right\_sublist[right\_sublist\_index]

right\_sublist\_index = right\_sublist\_index + 1

# move forward in the sorted part

sorted\_index = sorted\_index + 1

# we will go through the remaining elements and add them

while left\_sublist\_index < len(left\_sublist):

list1[sorted\_index] = left\_sublist[left\_sublist\_index]

left\_sublist\_index = left\_sublist\_index + 1

sorted\_index = sorted\_index + 1

while right\_sublist\_index < len(right\_sublist):

list1[sorted\_index] = right\_sublist[right\_sublist\_index]

right\_sublist\_index = right\_sublist\_index + 1

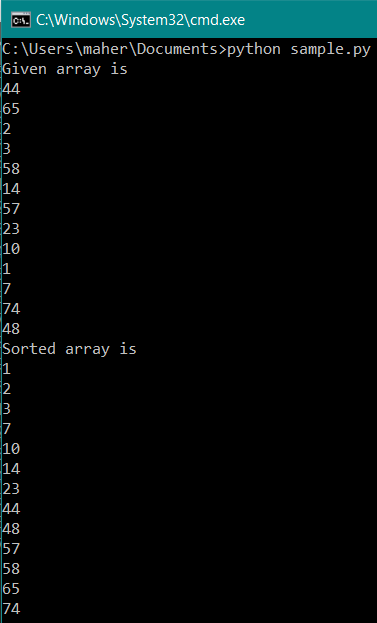
sorted\_index = sorted\_index + 1

list1 = [44, 65, 2, 3, 58, 14, 57, 23, 10, 1, 7, 74, 48]

merge\_sort(list1, 0, len(list1) -1)

print(list1)

**Output:**

****

## Post Practical Questions:

1. Which of the following is not a stable sorting algorithm in its typical implementation.
2. Insertion Sort
3. **Merge Sort**
4. Quick Sort
5. Bubble Sort
6. Consider a situation where swap operation is very costly. Which of the following sorting algorithms should be preferred so that the number of swap operations are minimized in general?
7. Insertion Sort
8. **Merge Sort**
9. Quick Sort
10. Bubble Sort
11. You have to sort 1 GB of data with only 100 MB of available main memory. Which sorting technique will be most appropriate?
12. Insertion Sort
13. **Merge Sort**
14. Quick Sort
15. Bubble Sort
16. In a modified merge sort, the input array is splitted at a position one-third of the length(N) of the array. What is the worst case time complexity of this merge sort?

a) N(logN base 3) b)N(logN base 2/3) c)**N(logN base 1/3)** d)N(logN base 3/2)

1. Which sorting algorithm will take least time when all elements of input array are identical? Consider typical implementations of sorting algorithms.
2. Insertion Sort
3. **Merge Sort**
4. Quick Sort
5. Bubble Sort

1. Which of the following sorting algorithms has the lowest worst-case complexity?
2. Insertion Sort
3. **Merge Sort**
4. Quick Sort
5. Bubble Sort

## Conclusion:

We learned to implement merge sort using python.