D Y PATIL
RAMRAO ADIK
INSTITUTE OF
TECHNOLOGY
NAVI MUMBAI

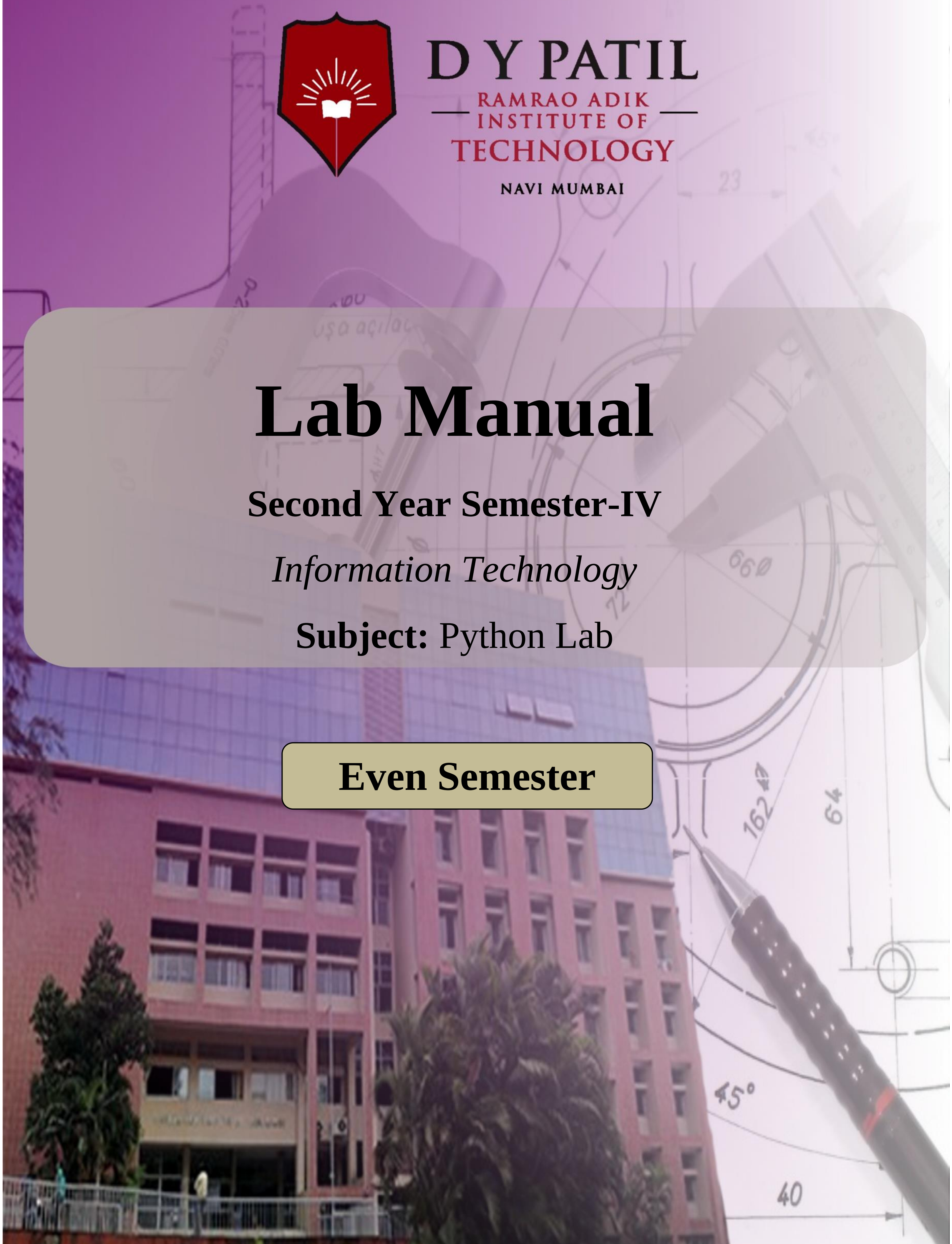
Lab Manual

Second Year Semester-IV

Information Technology

Subject: Python Lab

Even Semester



Institute Vision & Mission

Vision

To foster and permeate higher and quality education with value added engineering, technology programs, providing all facilities in terms of technology and platforms for all round development with societal awareness and nurture the youth with international competencies and exemplary level of employability even under highly competitive environment so that they are innovative adaptable and capable of handling problems faced by our country and world at large.

Mission

The Institution is committed to mobilize the resources and equip itself with men and materials of excellence thereby ensuring that the Institution becomes pivotal center of service to Industry, academia, and society with the latest technology. RAIT engages different platforms such as technology enhancing Student Technical Societies, Cultural platforms, Sports excellence centers, Entrepreneurial Development Center and Societal Interaction Cell. To develop the college to become an autonomous Institution & deemed university at the earliest with facilities for advanced research and development programs on par with international standards. To invite international and reputed national Institutions and Universities to collaborate with our institution on the issues of common interest of teaching and learning sophistication.

Department Vision & Mission

Vision

To pervade higher and quality education with value added engineering, technology programs to deliver the IT graduates with knowledge, skills, tools and competencies necessary to understand and apply the technical knowledge and to become competent to practice engineering professionally and ethically in tomorrow's global environment. To contribute to the overall development by imparting moral, social and ethical values.

Mission

The mission of the IT department is to prepare students for overall development including employability, entrepreneurship and the ability to apply the technology to real life problems by educating them in the fundamental concepts, technical skills/programming skills, depth of knowledge and development of understanding in the field of Information Technology. To develop entrepreneurs, leaders and researchers with exemplary level of employability even under highly competitive environments with high ethical, social and moral values.

Departmental Program Educational Objectives

PEO1: To produce IT graduates who have strong foundation in mathematics, sciences and basic engineering and prepare them with strong engineering knowledge.

PEO2: To provide technical competence to use techniques, skills and modern engineering tools that allows them to work effectively in areas such as Algorithms, Computer Organization, Information Systems, Networks and Multimedia.

PEO3: To train students with good practical knowledge by use of modern equipped laboratory and conduct the experiments, record, analyze and interpret data and also in scientific and engineering breadth that cover multi-disciplinary subjects enabling them to comprehend, analyze the Information Technology problems and develop solutions.

PEO4: To inculcate the students in professional and ethical attitude, effective communication skills, teamwork skills, multidisciplinary approach and an ability to relate engineering.

PEO5: To produce graduates who have the ability to pursue advanced studies and research in disciplines and develop consciousness on the issues of social concerns and enabling them to upgrade their personality and experiences through community services to have a meaningful linkage with the society.

PEO6: To provide graduates with the ability to upgrade their personality and develop an attitude for self-learning and life-long learning.

Index

Sr. No.	Contents	Page No.
1.	List of Experiments	iv
2.	Experiment Plan and Lab Outcomes	v
3.	Study and Evaluation Scheme	vi
4.	Experiment No. 1	1
5.	Experiment No. 2	6
6.	Experiment No. 3	13
7.	Experiment No. 4	22
8.	Experiment No. 5	25
9.	Experiment No. 6	29
10.	Experiment No. 7	33
11.	Experiment No. 8	37
12.	Experiment No. 9	40
13.	Experiment No. 10	45

List of Experiments

Sr. No.	Experiments Name
1	Write a python program to implement Comments, Datatypes, Expressions, Input and Output Functions.
2	Write a python program to implement Byte Array, Range, Set and STRING Functions
3	Write python programs to implement the Control Structures
4	Write python programs to implement the For Loop
5	Write python programs to implement Classes, object, Static method and inner class
6	Write python programs to implement Inheritance and Polymorphism with Method overloading and Method Overriding
7	Write python program to implement different types of Exceptions
8	Write a python programs to implement different file handling operations using pickle
9	Write a python program to implement GUI Canvas Application using Tkinter
10	Write a python program to implement Client-Server programming using TCP Socket
11	Mini Project

Experiment Plan & Course Outcome

Course Outcomes:

CO1	Describe the Numbers, Math functions, Strings, List, Tuples and Dictionaries in Python and express different Decision Making statements and Functions
CO2	Interpret Object oriented programming in Python and understand different File handling operations
CO3	Explain how to design GUI Applications in Python and evaluate different database operations
CO4	Design and develop Client Server network applications using Python

Module No.	Week No.	Experiments Name	Course Outcome
1	W1	Write a python program to implement Comments, Datatypes, Expressions, Input and Output Functions.	CO1
2	W2	Write a python program to implement Byte Array, Range, Set and STRING Functions	CO1
3	W3	Write python programs to implement the Control Structures	CO1
4	W4	Write python programs to implement the For Loop	CO1
5	W5	Write python programs to implement Classes, object, Static method and inner class	CO2
6	W6	Write python programs to implement Inheritance and Polymorphism with Method overloading and Method Overriding	CO2
7	W7	Write python program to implement different types of Exceptions	CO2
8	W8	Write a python programs to implement different file handling operations using pickle	CO2
9	W9	Write a python program to implement GUI Canvas Application using Tkinter	CO3
10	W10	Write a python program to implement Client-Server programming using TCP Socket	CO4
11	W11	Mini Project	All COs

Study and Evaluation Scheme

Course Code	Course Name	Teaching Scheme			Credits Assigned			
		Theory	Practical	Tutorial	Theory	Practical	Tutorial	Total
ITL404	Python Lab	--	02+02*	--	-	02	--	02

Course Code	Course Name	Examination Scheme		
		Term Work	Practical & Oral	Total
ITL404	Python Lab	50	50	100

Term Work:

1. Term work assessment must be based on the overall performance of the student with every experiment graded from time to time. The grades should be converted into marks as per the Credit and Grading System manual and should be added and averaged.
2. The final certification and acceptance of term work ensures satisfactory performance of laboratory work and minimum passing marks in term work.
3. Mini Project has a weightage of 10 marks in term work

Practical & Oral:

1. Practical & Oral exam will be based on the entire syllabus of Python Lab respectively.

Python Lab

Experiment no. :1

Write a python program to implement Comments, Datatypes, Expressions, Input and Output Functions.

Experiment No. 1

Aim: Write a python program to implement Comments, Datatypes, Expressions, Input and Output Functions.

What will you learn by performing this experiment?
The basic components of Python Programming.

Software Required: Python 3x version, IDLE(IDE)
Theory: In python single and multiline comments are used to give code information to the user or fellow programmer. For a single line comment # keyword is used. In case a user want to enter some information which may extend for multi lines then in that case.

Python supports two types of comments:

1) Single lined comment:

In case user wants to specify a single line comment, then comment must start with #

Eg:
# This is single line comment.

2) Multi lined Comment:
Multi lined comment can be given inside triple quotes.
eg:
“””This
 Is
 Multipline comment”””

Python has five standard data types −

Numbers
String
List
Tuple
Dictionary.

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity) −	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -1

Basic Python Operators

A simple hello world example is given below. Write below code in a file and save with .py extension. Python source file has .py extension.

hello.py

```
print("hello world by python!")
```

Execute this example by using following command.

```
Python3 hello.py
```

After executing, it produces the following output to the screen.

Output

```
hello world by python!
```

In order to take input from a user we use input() function.
e.g.
a=input("enter the value") will ask user to enter the value and assign that to variable/tag 'a'.

Conclusion and Discussion: We have learnt the basic components which a python learner must know before the actual coding logic begins.

QUIZ / Viva Questions:

1. What is the use and meaning of single line and multiline comment statements?
2. What are the different datatypes and their usage?
3. How to take input from user and print?

References:

1. Dr. R. Nageswara Rao,"Core Python Programming" , Dreamtech Press, Wiley Publication.
2. James Payne, "Beginning Python: Using Python 2.6 and Python 3.1",Wrox Publication.
3. Wesley J Chun," Core Python Applications Programming",Third Edition, Pearson Publication.

Python Lab

Experiment no. : 2

Write a python program to implement Byte Array, Range, Set and String Functions.

Experiment No.2

Aim: Write a python program to implement Byte Array, Range, Set and String Functions

What will you learn by performing this experiment?
The use of Byte Array , Range, Set and String Functions which are useful in large data storage and to know how to perform different operations on it.

Software Required: Python3x, IDLE

Theory:

Byte represents a group of byte numbers just like an array does(0 to 255). It can't be modified.

e.g. list1=[10,20,0,40,50]
x=bytes(list1)
print(x[0])

Try using a for loop:

```
for i in x:
        print(i)
```

Bytearray is same as bytes datatypes. But this can be modified
list2=[10,20,0,40]
x=bytearray(list2)
print(x[0])

e.g. we can assign x[0]=49 then try for loop

```
for i in x:
        print(i)
```

Range describes a sequence of numbers. For e.g.

```
r=range(30,40,2)
For i in r:
        print(i)
```

will print 30,32,34,36,38. Here 30 is the starting value, 40 is the end value and 2 is the step function value.

Sets are the unordered collection of elements. No duplicate value is printed in sets.

s={10,20,30,20,40}
print(s) may display 10,20,30,40

ch=set("Hello")
print(ch) # may display {'H', 'e','l','o'}

Strings are amongst the most popular types in Python. We can create them simply by enclosing characters in quotes. Python treats single quotes the same as double quotes.

var1 = 'Hello World!'
var2 = "Python Programming"

String Functions

Sr.No.	Functions with Description
1	**capitalize()** Capitalizes first letter of string
2	**center(width, fillchar)** Returns a space-padded string with the original string centered to a total of width columns.
3	**count(str, beg= 0,end=len(string))** Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.
4	**decode(encoding='UTF-8',errors='strict')** Decodes the string using the codec registered for encoding. encoding defaults to the default string encoding.
5	**encode(encoding='UTF-8',errors='strict')** Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.
6	**endswith(suffix, beg=0, end=len(string))** Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise.

7	**expandtabs(tabsize=8)** Expands tabs in string to multiple spaces; defaults to 8 spaces per tab if tabsize not provided.
8	**find(str, beg=0 end=len(string))** Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.
9	**index(str, beg=0, end=len(string))** Same as find(), but raises an exception if str not found.
10	**isalnum()** Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.
11	**isalpha()** Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.
12	**isdigit()** Returns true if string contains only digits and false otherwise.
13	**islower()** Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.
14	**isnumeric()** Returns true if a unicode string contains only numeric characters and false otherwise.
15	**isspace()** Returns true if string contains only whitespace characters and false otherwise.
16	**istitle()** Returns true if string is properly "titlecased" and false otherwise.
17	**isupper()** Returns true if string has at least one cased character and all cased characters are in

	uppercase and false otherwise.
18	**join(seq)** Merges (concatenates) the string representations of elements in sequence seq into a string, with separator string.
19	**len(string)** Returns the length of the string
20	**ljust(width[, fillchar])** Returns a space-padded string with the original string left-justified to a total of width columns.
21	**lower()** Converts all uppercase letters in string to lowercase.
22	**lstrip()** Removes all leading whitespace in string.
23	**maketrans()** Returns a translation table to be used in translate function.
24	**max(str)** Returns the max alphabetical character from the string str.
25	**min(str)** Returns the min alphabetical character from the string str.
26	**replace(old, new [, max])** Replaces all occurrences of old in string with new or at most max occurrences if max given.
27	**rfind(str, beg=0,end=len(string))** Same as find(), but search backwards in string.
28	**rindex(str, beg=0, end=len(string))** Same as index(), but search backwards in string.

29	**rjust(width,[, fillchar])** Returns a space-padded string with the original string right-justified to a total of width columns.
30	**rstrip()** Removes all trailing whitespace of string.
31	**split(str="", num=string.count(str))** Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.
32	**splitlines(num=string.count('\n'))** Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed.
33	**startswith(str, beg=0,end=len(string))** Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.
34	**strip([chars])** Performs both lstrip() and rstrip() on string.
35	**swapcase()** Inverts case for all letters in string.
36	**title()** Returns "titlecased" version of string, that is, all words begin with uppercase and the rest are lowercase.
37	**translate(table, deletechars="")** Translates string according to translation table str(256 chars), removing those in the del string.
38	**upper()** Converts lowercase letters in string to uppercase.
39	**zfill (width)** Returns original string leftpadded with zeros to a total of width characters; intended

	for numbers, zfill() retains any sign given (less one zero).
40	**isdecimal()** Returns true if a unicode string contains only decimal characters else false

Conclusion and Discussion: We have learnt data types like bytes, bytearray, sets and strings. Many operations are possible using built string functions.

QUIZ / Viva Questions:

1. What is the difference between bytes and bytearray datatype.
2. What is the set data type?
3. What is a string data type and list some string functions?

References:

1. Dr. R. Nageswara Rao,"Core Python Programming" , Dreamtech Press, Wiley Publication.
2. James Payne, "Beginning Python: Using Python 2.6 and Python 3.1",Wrox Publication.
3. Wesley J Chun," Core Python Applications Programming",Third Edition, Pearson Publication.

Python Lab

Experiment No. : 3

Write python programs to understand the Control Structures.

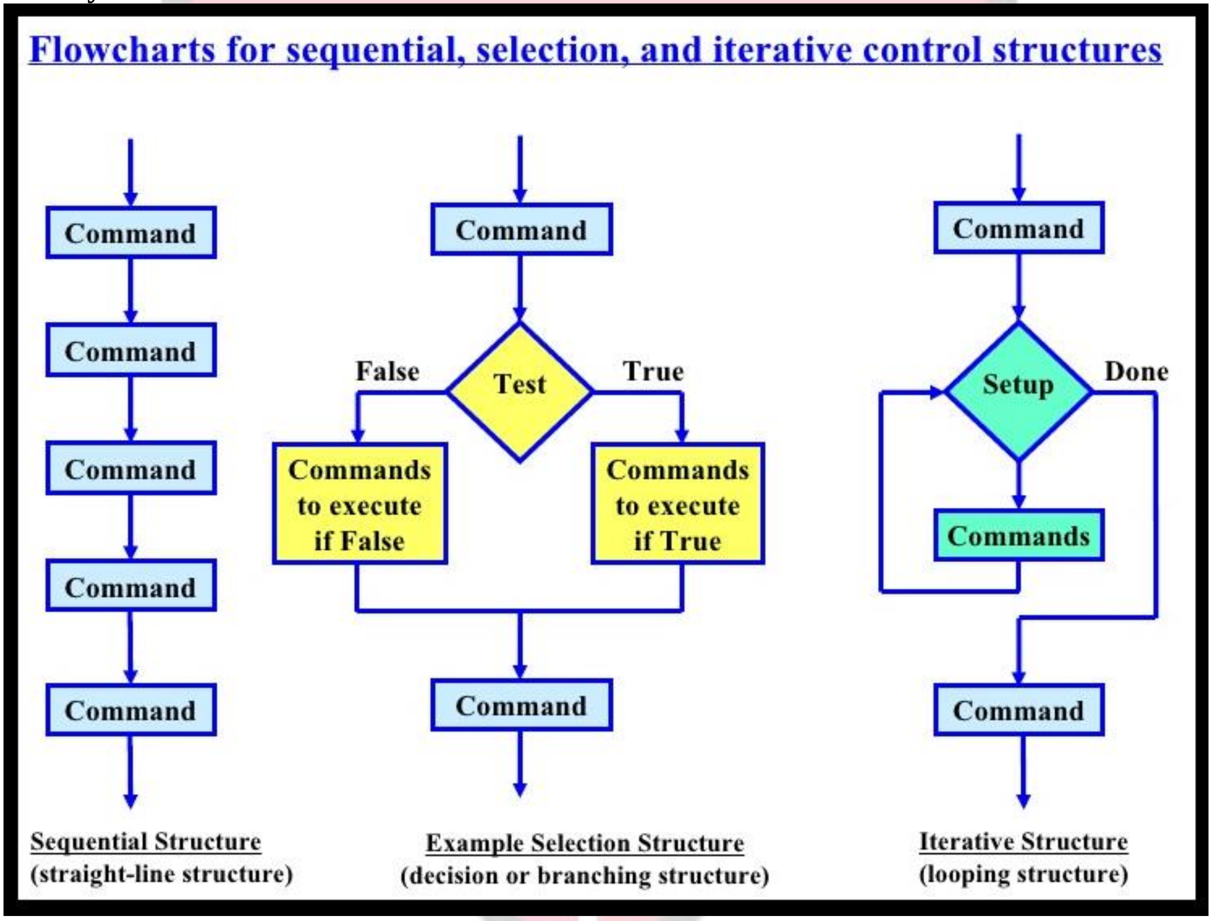
Experiment No. 3

Aim: Write python programs to implement the Control Structures

What will you learn by performing this experiment?
The use of different control structures, the use of the control structures. Using the control structures you will be able to increase the efficiency of programming and implement various operations.

Software Required: Python3x, IDLE

Theory:


Flowcharts for sequential, selection, and iterative control structures
Command
Command
Command
Command
Command
Command
Test
False
True
Commands to execute if False
Commands to execute if True
Command
Command
Setup
Done
Commands
Command
Sequential Structure
(straight-line structure)
Example Selection Structure
(decision or branching structure)
Iterative Structure
(looping structure)

Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise. Following is the general form of a typical decision making structure found in most of the programming languages –

Python programming language assumes any non-zero and non-null values as TRUE, and if it is either zero or null, then it is assumed as FALSE value.
Python programming language provides following types of decision making statements.
An **if statement** consists of a boolean expression followed by one or more statements.

An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is FALSE.
Python provide various control structures that allow for more complicated execution paths. A loop statement allows us to execute a statement or group of statements multiple times.
While loop Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
For loop Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
Nested loop you can use one or more loop inside any another while, for or do..while loop.

<u>FOR Loop</u>

Figure - for loop Flowchart:

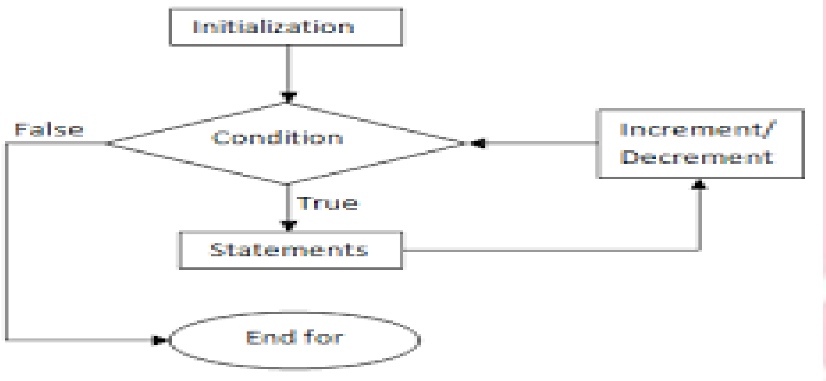


fig: Flowchart for for loop

<u>Syntax:</u>

```
for iterating_var in sequence:

   //execute your code
```

<u>Example 01:</u>

```
#!/usr/bin/python



for x in range (0,3) :

    print 'Loop execution %d' % (x)
```

<u>Output:</u>

```
Loop execution 0
Loop execution 1

Loop execution 2
```

Example 02:

```
#!/usr/bin/python
```

```
for letter in 'TutorialsCloud':

   print 'Current letter is:', letter
```

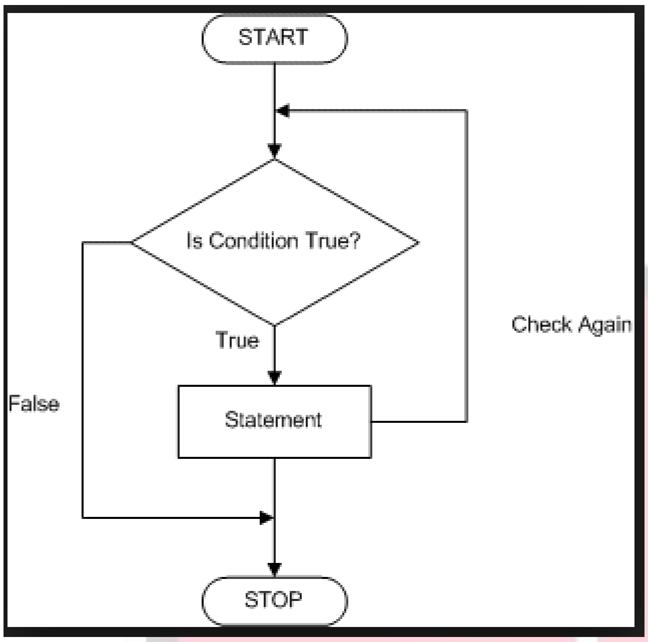
Output:

```
Current letter is : T

Current letter is : u

Current letter is : t

Current letter is : o

Current letter is : r

Current letter is : i

Current letter is : a

Current letter is : l

Current letter is : s

Current letter is : C

Current letter is : l

Current letter is : o

Current letter is : u

Current letter is : d
```

while Loop:

The graphical representation of the logic behind while looping is shown below:

Figure - while loop Flowchart:



<u>Syntax:</u>

```
while expression:
  //execute your code
```

<u>Example:</u>

```
#!/usr/bin/python


#initialize count variable to 1

count =1


while count < 6 :

   print count

   count+=1

#the above line means count = count + 1
```

Output:

```
1

2

3

4

5
```

The *elif* Statement:

The **elif** statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

Similar to the **else**, the **elif** statement is optional. However, unlike **else**, for which there can be at most one statement, there can be an arbitrary number of **elif** statements following an **if**.

syntax

```
if expression1:
   statement(s)

elif expression2:
   statement(s)

elif expression3:
   statement(s)

else:
   statement(s)
```

Core Python does not provide switch or case statements as in other languages, but we can use if..elif...statements to simulate switch case as follows −

Example

```
#!/usr/bin/python


var = 100
```

```
if var == 200:
   print "1 - Got a true expression value"
   print var
elif var == 150:
   print "2 - Got a true expression value"
   print var
elif var == 100:
   print "3 - Got a true expression value"
   print var
else:
   print "4 - Got a false expression value"
   print var

print "Good bye!"
```

When the above code is executed, it produces the following result −

```
3 - Got a true expression value
100
Good bye!
```

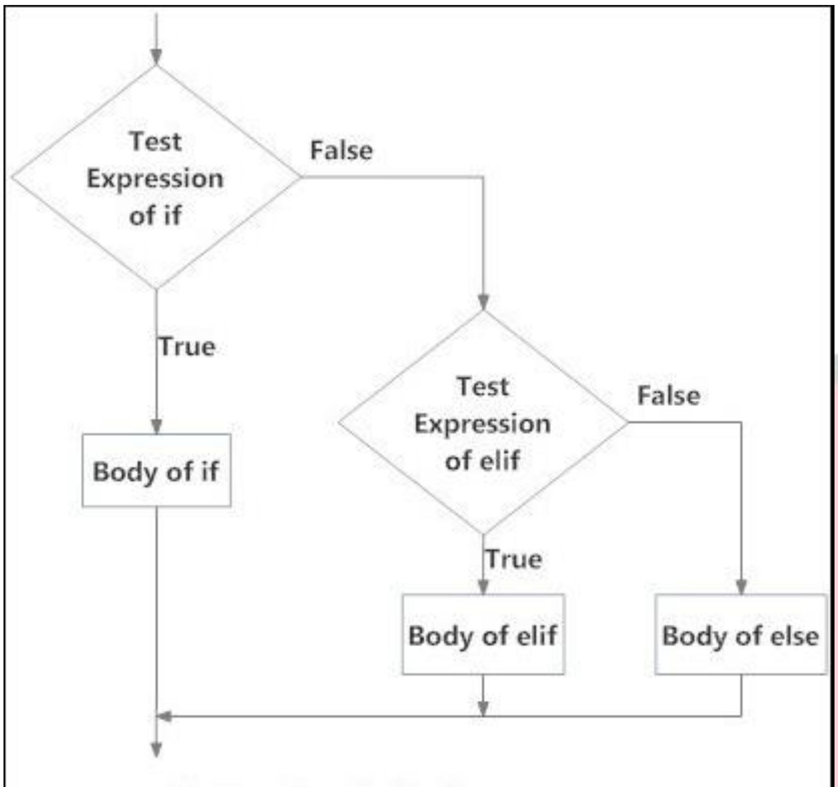


Fig: Operation of if...elif...else statement

Loop Control Statements

```
#Control Structures remember the indentations
print "--Finding MAX BETWEEN 3 Nos using IF-Else-Elif STATEMENT--"
Inter Value of three integers, a, b,
if (a>b and a>c)
a is Greater
elif (b>c)
b is Greater
else
c is Greater
```

WHILE LOOP for calculating the factorial of the given number.

```
Initialize fact=1, num, i=1
while
{(i <= num)
fact=fact*i
i=i+1
}
Print fact
```

Generate Fibonacci series using If statement & WHILE LOOP

```
Initialize variables n, f1=0, f2=1, c=2
if (n==1)
```

```
print f1
elif (n==2)
print f1,'\n',f2
else
print f1,'\n',f2
while (c<n)
{
f=f1+f2
print f
f1,f2=f2,f
c+=1
}
```

Output:

Output generates the greatest value among the three give variable, factorial value of the given value, and Fibonacci series of the given number.

Conclusion and Discussion: We have learnt data types like bytes, bytearray, sets and strings. Many operations are possible using built string functions.

QUIZ / Viva Questions:

1. What is the difference between if-else structure and for loop?
2. When is while loop and do while loop initialized?
3. What is the use of control structures in programming?

References:

1. Dr. R. Nageswara Rao,"Core Python Programming" , Dreamtech Press, Wiley Publication.
2. James Payne, "Beginning Python: Using Python 2.6 and Python 3.1",Wrox Publication.
3. Wesley J Chun," Core Python Applications Programming",Third Edition, Pearson Publication.

Python Lab

Experiment No. : 4

Write a python program to implement the for loop

Experiment No.4

Aim: Write a python program to implement Pascal triangle using For loop.

What will you learn by performing this experiment?
How to implement for loop to increase programming efficiency and print different patterns using the loop.

Software Required: Python 3, IDLE

Theory:

For loop Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
Nested loop you can use one or more loop inside any another while, for or do..while loop.

FOR Loop

Figure - for loop Flowchart:

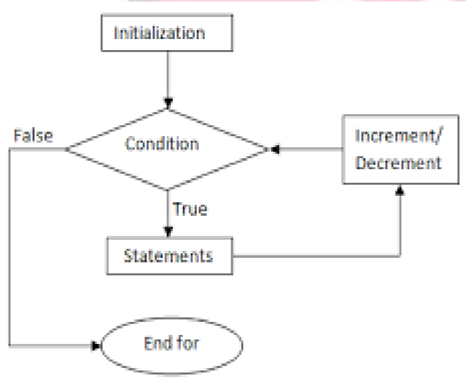


fig: Flowchart for for loop

Syntax:

```
for iterating_var in sequence:

   //execute your code
```

Pascal's triangle: It is a triangular array of the binomial coefficients. In much of the Western world, it is named after French mathematician Blaise Pascal, although other mathematicians studied it centuries before him in India,[1] Persia (Iran), China, Germany, and Italy.[2]

The rows of Pascal's triangle are conventionally enumerated starting with row $n = 0$ at the top (the 0th row). The entries in each row are numbered from the left beginning with $k = 0$ and are usually staggered relative to the numbers in the adjacent rows. The triangle may be constructed in the following manner: In row 0 (the topmost row), there is a unique nonzero entry 1. Each entry of each subsequent row is constructed by adding the number above and to the left with the number above and to the right, treating blank entries as 0. For example, the initial number in the first (or any other) row is 1 (the sum of 0 and 1), whereas the numbers 1 and 3 in the third row are added to produce the number 4 in the fourth row.

```
                    1
                 1      2
              1      4      4
           1      6      12      8
        1      8      24      32      16
     1      10      40      80      80      32
  1      12      60      160      240      192      64
1      14      84      280      560      672      448      128
```

Conclusion: Thus we have learnt how to print different patterns using for loop and successfully implemented Pascal triangle using for loop.

QUIZ / Viva Questions:

1. What is for loop?
2. What is Pascal triangle?
3. What is the use of control structures in programming?

References:

1. Dr. R. Nageswara Rao,”Core Python Programming” , Dreamtech Press, Wiley Publication.
2. James Payne, ”Beginning Python: Using Python 2.6 and Python 3.1”,Wrox Publication.
3. Wesley J Chun,” Core Python Applications Programming”,Third Edition, Pearson Publication.

Python Lab

Experiment No. : 5

Write a python program to implement Classes, Object, Static method and Inner class.

Experiment No.5

Aim: Write a python program to implement Classes, Object, Static method and Inner class

What will you learn by performing this experiment?
How to create class, its object, creating inner class and implementation of static method.

Software Required: Python 3, IDLE

Theory:

Class is a model to create Objects. Variables represent Attributes and Methods represent actions performed. Methods and variables are also available in Objects aka Instance Variables. A function written inside a class is known as method.

A method can be called in 2 ways:

```
classname.methodname()

instancename.methodname()
```

A sample class definition:

```
class Classname(object):

        attributes

        def __init__(self):

        def method1():

        def method2():
```

__init__(self) is a special method used to initialize the variables.

We can create new object instances of the classes. The procedure to create an object is similar to a function call. We can access attributes of objects by using the object name prefix.

Syntax of creating instance:

```
instancename= Classname()
```

Static Methods:

They are involved when some processing is related to class level but we need not involve the class or instances to perform any work. E.g. setting environment variables, counting no. of class instances or changing an attribute in other class.

e.g:

```
class Myclass:

n=0

def __init__(self):

Myclass.n=Myclass.n+1

@staticmethod

def noObjects():

        print('No. Of instances created:', Myclass.n)

obj1=Myclass()

obj2=Myclass()

obj3=Myclass()

Myclass.noObjects()
```

Inner Class:

Creating a class inside another class is known as Inner class. It is used when we want to replace inheritance, i.e. when a class(to be extended) is needed for only 1 class.

```
class Person:
        def __init__(self):
                self.name='Charles'
        def display(self):
                print('Name=',self.name)
        class Dob:
                def __init__(self):
                        self.dd=10
```

```
                    self.mm=5
                    self.yy=1988
            def display(self):
                    print('Dob ={}/{}/{}'.format(self.dd,self.mm,self.yy))
p=Person()
p.display
x=Person().Dob()
x.display()
print(x.yy)
```

Conclusion: Thus we have learnt how to create class, object , static method and inner class.

QUIZ / Viva Questions:

1.What is meant by class and object?

2.When do we need a static method and inner class?

References:

1. Dr. R. Nageswara Rao,"Core Python Programming" , Dreamtech Press, Wiley Publication.
2. James Payne, "Beginning Python: Using Python 2.6 and Python 3.1",Wrox Publication.
3. Wesley J Chun," Core Python Applications Programming",Third Edition, Pearson Publication.

Experiment No. : 6

Write python programs to understand Inheritance and Polymorphism with Method overloading and Method Overriding

Experiment No.6

Aim: Write python programs to implement Inheritance and Polymorphism with Method overloading and Method Overriding

What will you learn by performing this experiment?
You will understand the concepts of Inheritance, Polymorphism, Method overloading and Overriding. You will be able to distinguish between method overloading and Method overriding.

Software Required: Python 3, IDLE

Theory:

Inheritance:-Inheritance allows programmer to create a general class first then later extend it to more specialized class. It also allows programmer to write better code. Using inheritance you can inherit all access data fields and methods, plus you can add your own methods and fields, thus inheritance provides a way to organize code, rather than rewriting it from scratch. In object-oriented terminology when class X extend class Y , then Y is called **super class** or **base class** and X is called **subclass or derived class**. One more point to note that only data fields and method which are not private are accessible by child classes, private data fields and methods are accessible only inside the class.

Polymorphism:- Sometimes an object comes in many types or forms. If we have a button, there are many different draw outputs (round button, check button, square button, button with image) but they do share the same logic: onClick(). We access them using the same method . This idea is called *Polymorphism*. Polymorphism is based on the greek words Poly (many) and morphism (forms). We will create a structure that can take or use many forms of objects.

Method Overloading:- In Python you can define a method in such a way that there are multiple ways to call it. Given a single method or function, we can specify the number of parameters ourself. Depending on the function definition, it can be called with zero, one, two or more parameters. This is known as *method overloading.*

```
#!/usr/bin/env python

class Human:

    def sayHello(self, name=None):

        if name is not None:
            print 'Hello ' + name
        else:
            print 'Hello '
```

```
# Create instance
obj = Human()

# Call the method
obj.sayHello()

# Call the method with a parameter
obj.sayHello('Guido')
```

Method Overiding:- Overriding is the ability of a class to change the implementation of a method provided by one of its ancestors. Overriding is a very important part of OOP since it is the feature that makes inheritance exploit its full power. Through method overriding a class may "copy" another class, avoiding duplicated code, and at the same time enhance or customize part of it. Method overriding is thus a strict part of the inheritance mechanism.

```
class Thought(object):
    def __init__(self):
        pass
    def message(self):
        print "I feel like I am diagonally parked in a parallel universe."

class Advice(Thought):
    def __init__(self):
        super(Advice, self).__init__()
    def message(self):
        print "Warning: Dates in calendar are closer than they appear"
        super(Advice, self).message()
```

Output:- The output of the program shows the concept of inheritance and polymorphism with method overloading and method overriding.

Conclusion: Thus we have learnt how to implement inheritance and polymorphism using Method overloading and method overriding.

QUIZ / Viva Questions:

1. 1.What is meant polymorphism?

2. 2.What is Inheritance?

3. What is the difference between Method overloading and Method Overriding?

References:

1. Dr. R. Nageswara Rao,"Core Python Programming" , Dreamtech Press, Wiley Publication.
2. James Payne, "Beginning Python: Using Python 2.6 and Python 3.1",Wrox Publication.
3. Wesley J Chun," Core Python Applications Programming",Third Edition, Pearson Publication.

Python Lab

Experiment No. : 7

Write python program to implement different types of Exceptions

Experiment No.7

Aim: Write python program to implement different types of Exceptions.

What will you learn by performing this experiment?
How to use exception classes in the try, except, else and finally construct.

Software Required: Python 3, IDLE

Theory:

Exception can be said to be any abnormal condition in a program resulting to the disruption in the flow of the program.
Whenever an exception occurs the program halts the execution and thus further code is not executed. Thus exception is that error which python script is unable to tackle with.
Exception in a code can also be handled. In case it is not handled, then the code is not executed further and hence execution stops when exception occurs.
Hierarchy Of Exception:

1. ZeroDivisionError: Occurs when a number is divided by zero.
2. NameError: It occurs when a name is not found. It may be local or global.
3. IndentationError: If incorrect indentation is given.
4. IOError: It occurs when Input Output operation fails.
5. EOFError: It occurs when end of file is reached and yet operations are being performed , etc..

Exception Handling mechanism:
The suspicious code can be handled by using the try block. Enclose the code which raises an exception inside the try block. The try block is followed except statement. It is then further followed by statements which are executed during exception and in case if exception does not occur.
Syntax:

```
try:
   malicious code
except Exception1:
   execute code
except Exception2:
    execute code
....
....
except ExceptionN:
    execute code
else:
    In case of no exception, execute the else block code.
```

eg:

```
try:
    a=10/0
    print a
except ArithmeticError:
        print "This statement is raising an exception"
else:
    print "Welcome"
```

Output:

```
>>>
This statement is raising an exception
>>>
```

Explanation:

The malicious code (code having exception) is enclosed in the try block.

Try block is followed by except statement. There can be multiple except statement with a single try block.

Except statement specifies the exception which occurred. In case that exception is occurred, the corresponding statement will be executed.

At the last you can provide else statement. It is executed when no exception is occurred.

Except with no Exception:

Except statement can also be used without specifying Exception.

Syntax:

```
try:
        code
    except:
        code to be executed in case exception occurs.
    else:
        code to be executed in case exception does not occur.
```

eg:

```
try:
    a=10/0;
except:
    print "Arithmetic Exception"
else:
    print "Successfully Done"
```

Output:

```
>>>
Arithmetic Exception
>>>
```

Conclusion and Discussion: Thus we have learnt exception handling mechanism.

QUIZ / Viva Questions:

1. What is Exception Handling?

2. What is difference between error and exception?

3. Explain the use of try,except,else,finally?

References:

1. Dr. R. Nageswara Rao,"Core Python Programming" , Dreamtech Press, Wiley Publication.
2. James Payne, "Beginning Python: Using Python 2.6 and Python 3.1",Wrox Publication.
3. Wesley J Chun," Core Python Applications Programming",Third Edition, Pearson Publication.

Python Lab

Experiment No. : 8

Write python programs to understand different file handling operations with pickle

Experiment No. 8

Aim: Write a python programs to implement different file handling operations using pickle

What will you learn by performing this experiment?
How to use files and learn file handling. You'll understand the use of pickle and learn how and where to implement it.

Software Required: Python 3, IDLE

Theory:- When you're working with Python, you don't need to import a library in order to read and write files. It's handled natively in the language, albeit in a unique manner. The first thing you'll need to do is use Python's built-in ***open*** function to get a ***file object***. The ***open*** function opens a file. It's simple. When you use the ***open*** function, it returns something called a ***file object***. ***File bjects*** contain methods and attributes that can be used to collect information about the file you opened. They can also be used to manipulate said file.

Pickle is used for serializing and de-serializing a Python object structure. Any object in python can be pickled so that it can be saved on disk. What pickle does is that it "serialises" the object first before writing it to file. Pickling is a way to convert a python object (list, dict, etc.) into a character stream. The idea is that this character stream contains all the information necessary to reconstruct the object in another python script. Following functions we can perform on the file such as

Open():-he open() function is used to open files in our system, the filename is the name of the file to be opened. The mode indicates, how the file is going to be opened "r" for reading, "w" for writing and "a" for a appending.
Read ():-The read functions contains different methods, read(),readline() and readlines(). The read return a biog string, readline return a big line, and readlines return a list of lines.
Write ():-This method writes a sequence of strings to the file.
Append ():-The append function is used to append to the file instead of overwriting it.
To append to an existing file, simply open the file in append mode ("a"):
Close():-When you're done with a file, use close() to close it and free up any system resources taken up by the open file.

Algorithm:-
Filename-: studentpickle.py

```
class student:
def __init__(self,roll,name,age):
self.roll=roll
self.name=name
self.age=age
def display(self):
```

The file data.pkl can be read in again by Python in the same or another session or by a different program:

```
>>> import pickle
>>> f = open("data.pkl", "rb")
>>> villes = pickle.load(f)
>>> print(villes)
['Paris', 'Dijon', 'Lyon', 'Strasbourg']
>>>
```

Output:- Output of the experiment demonstrated the various file handling function functions using pickle in python.

QUIZ / Viva Questions:

1. What is Pickle?

2. How are File operations handled using Pickle?

3. What are the different ways of file handling other than Pickle?

References:

1. Dr. R. Nageswara Rao,"Core Python Programming" , Dreamtech Press, Wiley Publication.
2. James Payne, "Beginning Python: Using Python 2.6 and Python 3.1",Wrox Publication.
3. Wesley J Chun," Core Python Applications Programming",Third Edition, Pearson Publication.

Python Lab

Experiment No. : 9

Write a python program to implement GUI Canvas Application using Tkinter.

Experiment No. 9

Aim: Write a python program to implement GUI Canvas Application using Tkinter.

What will you learn by performing this experiment?
How to make GUI Application using Tkinter.

Software Required: Python, Tkinter

Theory:

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.

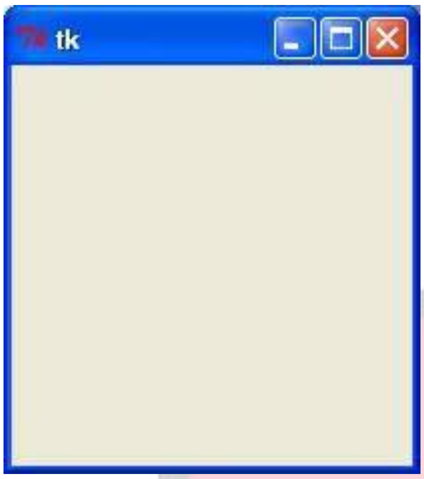
Create the GUI application main window.

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.

Example
```
#!/usr/bin/python

import tkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

This would create a following window −



TK Window

Tkinter Widgets

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table −

Sr.No. Operator & Description

1 Button

The Button widget is used to display buttons in your application.

2 Canvas

The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.

3 Checkbutton

The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.

4 Entry

The Entry widget is used to display a single-line text field for accepting values from a user.

5 Frame

The Frame widget is used as a container widget to organize other widgets.

6 Label

The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

7 Listbox
The Listbox widget is used to provide a list of options to a user.

8 Menubutton
The Menubutton widget is used to display menus in your application.

9 Menu
The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.

10 Message
The Message widget is used to display multiline text fields for accepting values from a user.

11 Radiobutton
The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.

12 Scale
The Scale widget is used to provide a slider widget.

13 Scrollbar
The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.

14 Text
The Text widget is used to display text in multiple lines.

15 Toplevel
The Toplevel widget is used to provide a separate window container.

16 Spinbox
The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.

17 PanedWindow
A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.

18 LabelFrame
A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.

19 tkMessageBox

This module is used to display message boxes in your applications.

Conclusion and Discussion: Thus we have learnt how to make a GUI based application using Tkinter.

QUIZ / Viva Questions:

1. What is Tkinter?
2. What are the attributes used in Tkinter based GUI development?

References:

1. Dr. R. Nageswara Rao,"Core Python Programming" , Dreamtech Press, Wiley Publication.
2. James Payne, "Beginning Python: Using Python 2.6 and Python 3.1",Wrox Publication.
3. Wesley J Chun," Core Python Applications Programming",Third Edition, Pearson Publication.

Python Lab

Experiment No. : 10

Write python programs to understand Client-Server programming using TCP Socket.

Experiment No. 10

Write python programs to understand Client-Server programming using TCP Socket.

Aim: Write a python program to implement Client-Server programming using TCP Socket

What will you learn by performing this experiment?
You will learn about TCP Socket. You will learn abut Client-server programming and how client server programming is implemented using TCP socket.

Software Required: Python 3, IDLE.

Theory

TCP/IP Client and Server:- Sockets can be configured to act as a *server* and listen for incoming messages, or connect to other applications as a *client*. After both ends of a TCP/IP socket are connected, communication is bi-directional.

Echo Server:- This sample program, based on the one in the standard library documentation, receives incoming messages and echos them back to the sender. It starts by creating a TCP/IP socket. Then **bind()** is used to associate the socket with the server address. In this case, the address is localhost, referring to the current server, and the port number is 10000. Calling **listen()** puts the socket into server mode, and **accept()** waits for an incoming connection. **accept()** returns an open connection between the server and client, along with the address of the client. The connection is actually a different socket on another port (assigned by the kernel). Data is read from the connection with **recv()** and transmitted with **sendall()**.When communication with a client is finished, the connection needs to be cleaned up using **close()**. This example uses a try:finally block to ensure that **close()** is always called, even in the event of an error.

Echo Clint:- The client program sets up its socket differently from the way a server does. Instead of binding to a port and listening, it uses connect() to attach the socket directly to the remote address. After the connection is established, data can be sent through the socket with **sendall()** and received with **recv()**, just as in the server. When the entire message is sent and a copy received, the socket is closed to free up the port.

Algorithm:-

```
import socket
host='127.0.0.1'
port=8000
#Create server side socket
s=socket.socket()
s.bind((host,port))
print "Server is Waiting........"
#Allow max clients connections=1
```

```
s.listen(1)
# wait till client connects
c,addr=s.accept()
print "A Client is connected"
#Server runs continiously
while True:
#receive 1024 byte data from client
data=c.recv(1024)
# if client send empty string then come out
if not data:
break
print "From Client: "+str(data.decode())
# Enter response from server
data1=raw_input("From Server : ")
# Send data to the client
c.send(data1.encode())
#Close Connection
c.close()
```

Syntax:

Echo Server

This sample program, based on the one in the standard library documentation, receives incoming messages and echos them back to the sender. It starts by creating a TCP/IP socket.

```
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Then **bind()** is used to associate the socket with the server address. In this case, the address is localhost, referring to the current server, and the port number is 10000.

```
# Bind the socket to the port
server_address = ('localhost', 10000)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address)
```

Calling **listen()** puts the socket into server mode, and **accept()** waits for an incoming connection.

```
# Listen for incoming connections
sock.listen(1)

while True:
   # Wait for a connection
```

```
    print >>sys.stderr, 'waiting for a connection'
    connection, client_address = sock.accept()
```

accept() returns an open connection between the server and client, along with the address of the client. The connection is actually a different socket on another port (assigned by the kernel). Data is read from the connection with **recv()** and transmitted with **sendall()**.

Echo Client

The client program sets up its **socket** differently from the way a server does. Instead of binding to a port and listening, it uses **connect()** to attach the socket directly to the remote address.

```
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect the socket to the port where the server is listening
server_address = ('localhost', 10000)
print >>sys.stderr, 'connecting to %s port %s' % server_address
sock.connect(server_address)
```

After the connection is established, data can be sent through the **socket** with **sendall()** and received with **recv()**, just as in the server.

Conclusion and Discussion: Thus we have learnt how client server programming is implemented using TCP socket.

QUIZ / Viva Questions:

1. What is TCP Socket?
2. What do you understand by the term client-server programming?
3. How is client server programming implemented using TCP socket?

References:

1. Dr. R. Nageswara Rao,”Core Python Programming” , Dreamtech Press, Wiley Publication.
2. James Payne, ”Beginning Python: Using Python 2.6 and Python 3.1”,Wrox Publication.
3. Wesley J Chun,” Core Python Applications Programming”,Third Edition, Pearson Publication.