Department of Computer Engineering

Academic Year 2023-24

Skill Based Lab-III Python

EXPERIMENT NO. 1

Experiment No. 1

1. **Aim:** Write a python program to swap two numbers and check if the first number is positive or negative or zero.

2. **Objectives:** From this experiment, the student will be able to learn
   - Basics of Python programming and Decision Making in Python.

3. **Outcomes:**
   - Students will be to understand and apply basic concepts in python.

4. **Software Required :** Python 3

5. **Theory:**

   Python is a general-purpose interpreted, interactive, object-oriented, and high- level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). There are 3 ways of executing a Python program

   1 Using Python's command line window 2 Using Python's IDLE graphics window 3 Directly from System prompt

   The first two are called interactive modes where we can type the program one line at a time and the complier executes it immediately. The last one is called non interactive mode where we ask complier to execute our program after typing the entire program. Programmers can store Python script source code in a file with the '.py'extension, and use the interpreter

to execute the contents of the file. To execute the script by the interpreter, you have to tell the interpreter the name of the file.

Decision making is required when we want to execute a code only if a certain condition is satisfied. The if…elif…else statement is used in Python for decision making. Syntax for if statement as follows:

1. if expression:
   //execute your code
2. if expression:
   //execut
   e
   y
   o
   u
   r
   c
   o
   d
   e
   e
   l
   s
   e
   :
   //execute your code
3. if expression:
   //execute
   your code
   elif
   expression:
   //execut

e your code

else:

//execute your code

6. **Algorithm:**
   1. Start
   2. Enter two numbers n1 and n2
   3. Declare a variable temp p
   4. for swapping do, temp=n1
   5. n2=temp
   6. Print

   value n1
   and n2 7. If
   (n1>0)
   8. Print the number
   is positive 9. Else
   (n1= =0)
   10. Print number is zero
   11. Else print the number is negative
   12. stop

7. **Conclusion:**
   We have studied how to run python program and swapping of two number, control statement if else for find the given number is positive, negative or zero.

8. **Viva Questions:**

   - How to run Python Program
   - Explain the if...elif...else loop

9. **References:**
   - Dr. R. Nageswara Rao, "Core Python Programming", Dreamtech Press
   - Learn Python the Hard Way(3rd Edition)(Zed Shaw's Hard Way Series)

Experiment No 2

1. **Aim:** Write a menu driven python program to check if number and string palindrome and find the factorial of the input number

2. **Objectives:** From this experiment, the student will be able to
   - Learn basics of Python programming and Decision Making and Functions in Python
3. **Outcomes:**
   - Student will be able to understand and apply basic concepts in python.
4. **Software Required :** Python 3
5. **Theory:**

Functions in python

The four steps to defining a function in Python are the following:

1. Use the keyword def to declare the function and follow this up with the function name.
2. Add arguments to the function: they should be within the parentheses of the function. End your line with a colon.
3. Add statements that the functions should execute.
4. End your function with a return statement if the function should output something. Without the return statement, your function will return an object None.
5. docstring is short for documentation string. It is used to explain in brief, what a function does.

Syntax for functions in pyhon:

```
def
    function_name(p
    arameters):
    """docstring"""
    statement(s)
```

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 factorial of a

number is the product of all the integers from 1 to that number.For example, the factorial of 6 (denoted as 6!) is 1*2*3*4*5*6 = 720. Factorial is not defined for negative numbers and the factorial of zero is one, 0! = 1.

6. Algorithm:

To find the string is palindrome.

1. Take a string from the user and store it in a variable.
2. Reverse the string using string slicing and compare it back to the original string.
3. Print the final result.
4. Exit.

To find factorial of number

1. Take the number from the user and store it in a variable
2. Pass the number as an argument to a recursive factorial function
3. Define the base condition as the number to be lesser than or equal to 1 and return 1 if it is.
4. Otherwise call the function recursively with the number minus 1 multiplied by the number itself.
5. Then return the result and print the factorial of the number.
6. Exit

7. Conclusion:

We have studied how to write functions in python program to find palindrome and factorial of number.

8. Viva Questions:

- What is palindrome?
- Explain Recursive function

9. References:

- Dr. R. Nageswara Rao, “Core Python Programming”, Dreamtech Press
Learn Python the Hard Way(3rd Edition)(Zed Shaw’s Hard Way Series

Experiment No. 3

1. **Aim:** Write a menu driven program to demonstrate use of list in python.
   a. Put even and odd elements into two different lists.
   b. Merge and sort the two list.
   c. Update first element with X value and delete the middle element of list.
   d. Find max and min element from the list.
   e. Add N names into the existing number list and check if word python is present in list.
2. **Objectives:** From this experiment, the student will be able to
   - Learn List and basic list operations
3. **Outcomes:**
   - Student will be able to understand and apply basic concepts in python.
4. **Software Required :** Python 3
5. **Theory:**

Lists are positional ordered collections of arbitrarily typed objects, and they have no fixed size and they are mutable. Lists are contained in square brackets []. Lists can contain numbers, strings, nested sublists, or nothing

Examples:

Examples: L1 = [0,1,2,3],
L2 = ['zero', 'one'],
L3=[0,1,[2,3],'three',['
four,one']], L4 = [].

List indexing works just like string indexing.

Lists are mutable: individual elements can be reassigned in place. Moreover, they can grow and shrink in place.

Example:

```
>>> L1 = [0,1,2,3]
>>> L1[0] = 4
```

>
>
>

L
1
[
0
]

4

Basic List Operations

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

Some of basic operations of list are as follows.

Python includes the following list functions –

Sr.No.	Function	Description
1	**cmp(list1, list2)**	Compares elements of both lists.
2	**len(list)**	Gives the total length of the list.
3	**max(list)**	Returns item from the list with max value.
4	**min(list)**	Returns item from the list with min value.
5	**list(seq)**	Converts a tuple into list.

Python includes following list methods:

Sr.No.	Metods	Description
1	**list.append(obj)**	Appends object obj to list

2	**list.count(obj)**	Returns count of how many times obj occurs in list
3	**list.extend(seq)**	Appends the contents of seq to list
4	**list.index(obj)**	Returns the lowest index in list that obj appears
5	**list.insert(index, obj)**	Inserts object obj into list at offset index
6	**list.pop(obj=list[-1])**	Removes and returns last object or obj from list
7	**list.remove(obj)**	Removes object obj from list
8	**list.reverse()**	Reverses objects of list in place
9	**list.sort([func])**	Sorts objects of list, use compare func if given

6. Algorithm

1. Start
2. Take the number of elements and store it in variable.
3. Take the element of each list one by one.
4. Use for loop to transverse through the elements of the list and if statement to check if the element is odd or even.
5. If the element is even ,append into a separate list and if it is odd append it to a different one
6. Use elif loop merge the two list and also sort the element into in it.
7. For updating the list take index value and update and for deleting use delete (deleted according index)or remove(removing according to element) function. Check for the string whether the Python string is present or not.
8. Using for loop to find minimum and maximum number and also find the occurrence of element in the list.
9. Exit/Stop

7. Conclusion:

Hence from above experiment student can understand that basic concept of list and list related various operation that create the list, update the list, merge the list etc.

8. Viva Questions:

- What is List?
- What are various methods of list?

9. References:

- Core Python Programming, Dr. R. Nageswara Rao, Dreamtech Press 2
- Beginning Python: Using Python 2.6 and Python 3.1. James Payne, Wrox publication
- Learn Python the Hard Way: (3rd Edition) (Zed Shaw's Hard Way Series)
- Python Projects , Laura Cassell,Alan Gauld,wrox publication

Experiment No. 4

1. **Aim:** Write a menu driven program to demonstrate use of tuples in python.
   a. Add and show N student roll number, name and 3 subject marks in a list of tuples.
   b. Display student roll number and marks whose name is Python.
   c. Demonstrate nested tuple and sort nested tuple by name.

2. **Objectives:** From this experiment, the student will be able to

   - Learn tuples and basic operations on tuples.

3. **Outcomes:**

   - Student will be able to understand and apply basic concepts in python.

4. **Software Required :** Python 3

5. **Theory:**

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";
```

Advantages of Tuple over List

However, there are certain advantages of implementing a tuple over a list. Below listed are some of the main advantages:

- We generally use tuple for heterogeneous (different) data types and list for homogeneous (similar) data types.
- Since tuple are immutable, iterating through tuple is faster than with list. So there is a slight performance boost.
- Tuples that contain immutable elements can be used as key for a dictionary. With list, this is not possible.

- If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

1. Creating a Tuple

A tuple is created by placing all the items (elements) inside a parentheses (), separated by comma. The parentheses are optional but is a good practice to write it.

A tuple can have any number of items and they may be of different types (integer, float, list, string etc.).

2. Accessing Elements in a Tuple

There are various ways in which we can access the elements of a tuple.

a. Indexing

We can use the index operator [] to access an item in a tuple where the index starts from 0. So, a tuple having 6 elements will have index from 0 to 5. Trying to access an element other that (6, 7...) will raise an Index Error.

The index must be an integer, so we cannot use float or other types. This will result into Type Error.

b. Negative Indexing

Python allows negative indexing for its sequences.

The index of -1 refers to the last item, -2 to the second last item and so on.

c. Changing a Tuple

Unlike lists, tuples are immutable.

This means that elements of a tuple cannot be changed once it has been assigned. But, if the element is itself a mutable datatype like list, its nested items can be changed.

d. Python Tuple Methods

Methods that add items or remove items are not available with tuple. Only the following two methods are available.

Method	Description
count(x)	Return the number of items that is equal to x

index(x)	Return index of first item that is equal to x

6. Algorithm:

1. Start
2. Create the empty tuple
3. Accept the elements in the form of tuple like student roll no., Student Name and three subject marks
4. Use for loop to transverse through the elements of the tuple find average and sum.
5. Use of elif loop for check whether the string python is present in it or not
6. For sort the tuple use sorted function. Using sorting function data is sorted by names
7. Exit/Stop.

7. Conclusion:

Hence from this experiment we learned the various functions of tuples like delete, modify or insert elements of tuples since tuple are immutable, so for that how to create a new tuple and store the updated elements in it.

8. Viva Questions:

- Explain what is tuple?
- How it is differ from List?
- How list can be sorted?

9. References:

- Core Python Programming, Dr. R. Nageswara Rao, Dreamtech Press 2
- Beginning Python: Using Python 2.6 and Python 3.1. James Payne, Wrox publication
- Learn Python the Hard Way: (3rd Edition) (Zed Shaw's Hard Way Series)
- Python Projects , Laura Cassell,Alan Gauld,wrox publication

Experiment No. 5

1. **Aim:** Write a program to demonstrate, Classes objects and constructor
2. **Objectives:** From this experiment, the student will be able to learn
   - To demonstrate Classes objects and constructors
3. **Outcomes:**
   - Students will be able to demonstrate, Classes objects and constructors.
4. **Software Required:** Python 3
5. **Theory:**

A class is a user-defined blueprint or prototype from which objects are created. Classes provide a means of bundling data and functionality together. Creating a new class creates a new type of object, allowing new instances of that type to be made. Each class instance can have attributes attached to it for maintaining its state. Class instances can also have methods (defined by their class) for modifying their state.

Class creates a user-defined data structure, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

Class Objects: An Object is an instance of a Class. A class is like a blueprint while an instance is a copy of the class with actual values. An object consists of:

State: It is represented by the attributes of an object. It also reflects the properties of an object.

Behavior: It is represented by the methods of an object. It also reflects the response of an object to other objects.

Identity: It gives a unique name to an object and enables one object to interact with other objects.

Types of constructors:

Default constructor: The default constructor is a simple constructor which doesn't accept any arguments. Its definition has only one argument which is a reference to the instance being constructed.

Parameterized constructor: constructor with parameters is known as parameterized constructor. The parameterized constructor takes its first argument as a reference to the instance being constructed known as self and the rest of the arguments are provided by the programmer.

__init__ method: The __init__ method is similar to constructors in C++ and Java. Constructors are used to initialize the object's state. Like methods, a constructor also contains a collection of statements (i.e. instructions) that are executed at the time of Object creation. It runs as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object**.**

6. **Algorithm**:
    1. Create a class.
    2. Initialize the method OR constructor.
    3. Initialize the variables accessing through the method.
    4. Create an object of the class.
    5. Print details of both class and object created.

7. **Conclusion:** We have learned to demonstrate, Classes objects and constructors.

8. **Viva Questions:**

    - Explain __init__ method
    - Explain the class objects.

9. **References:**
    - Dr. R. Nageswara Rao, "Core Python Programming", Dreamtech Press
    - Learn Python the Hard Way(3rd Edition)(Zed Shaw's Hard Way Series)

Experiment No. 6

1. **Aim:** Write a python program to demonstrate inheritance in Python (with method overloading and overriding)
2. **Objectives:** From this experiment, the student will be able to learn Object oriented programming using Python
3. **Outcomes:**
    - Students will be able to Interpret Object oriented programming in Python.
4. **Software Required:** Python 3
5. **Theory:**

- Inheritance is the procedure in which one class inherits the attributes and methods of another class.
- The class whose properties and methods are inherited is known as the Parent class.
- The class that inherits the properties from the parent class is the Child class.
- The interesting thing is, along with the inherited properties and methods, a child class can have its own properties and methods.

➢ The benefits of inheritance are:

- It represents real-world relationships well.
- It provides the reusability of a code. We don't have to write the same code again and again. Also, it allows us to add more features to a class without modifying it.
- It is transitive in nature, which means that if class B inherits from another class A, then all the subclasses of B would automatically inherit from class A.

How to inherit a parent class? Use the following syntax:

```
class parent_class:
body of parent class
class child_class( parent_class):
body of child class
```

Types of Inheritance in Python:

1. Single Inheritance: Single inheritance enables a derived class to inherit properties from a single parent class, thus enabling code reusability and the addition of new features to existing code.

2. Multiple Inheritance: When a class can be derived from more than one base class this type of inheritance is called multiple inheritances. In multiple inheritances, all the features of the base classes are inherited into the derived class.

3. Multilevel Inheritance: In multilevel inheritance, features of the base class and the derived class are further inherited into the new derived class. This is similar to a relationship representing a child and a grandfather

- **Method Overloading:**

Methods in Python can be called with zero, one, or more parameters. This process of calling the same method in different ways is called method overloading. It is one of the important concepts in OOP. Two methods cannot have the same name in Python; hence method overloading is a feature that allows the same operator to have different meaning.

- **Method overriding:**

Method overriding in Python is when you have two methods with the same name that each perform different tasks.

6. **Algorithm:**
   1. Create a class person.
   2. Initialize the ID.
   3. Initialize the variables such as name.
   4. Use super class function.
   5. Access the parent class properly.
   6. Display the ID and name of the student.
   7. Enter the other marks if in sports.
   8. Display the result.
   9. If not include the sport marks then display the result of student without including sport marks.

7. **Conclusion:**
   Hence from this experiment we learned the concept of inheritance in python with overloading and overriding methods.

8. **Viva Questions:**
   - What is inheritance?
   - Explain different types of inheritances?

9. **References:**
   - Dr. R. Nageswara Rao, "Core Python Programming", Dreamtech Press
   - Learn Python the Hard Way(3rd Edition)(Zed Shaw's Hard Way Series)

Experiment No. 7

1. **Aim:** Write a program to demonstrate exception handling in python
2. **Objectives:** From this experiment, the student will be able to
   - Demonstrate exception handling in python
3. **Outcomes:**
   - Student will be able to demonstrate exception handling in python
4. **Software Required:** Python 3
5. **Theory**: Errors in Python can be of two types i.e. Syntax errors and Exceptions. Errors are the problems in a program due to which the program will stop the execution. On the other hand, exceptions are raised when some internal events occur that change the normal flow of the program.

   Exceptions: Exceptions are raised when the program is syntactically correct, but the code resulted in an error. This error does not stop the execution of the program; however, it changes the normal flow of the program.

   Try and Except Statement – Catching Exceptions
   Try and except statements are used to catch and handle exceptions in Python. Statements that can raise exceptions are kept inside the try clause and the statements that handle the exception are written inside except clause.

   Catching Specific Exception
   A try statement can have more than one except clause, to specify handlers for different exceptions.

   Finally Keyword in Python

   Python provides a keyword finally, which is always executed after the try and except blocks. The final block always executes after normal termination of try block or after try block terminates due to some exception.

   Raising Exception
   The raise statement allows the programmer to force a specific exception to occur. The sole argument in raise indicates the exception to be raised. This must be either an exception instance or an exception class (a class that derives from Exception)

6. **Algorithm:**
   1. Step 1: Start the program.
   2. Step 2: Declare the variables
   3. Step 3: Read the values
   4. Step 4: Inside the try block check the condition.
      (Perform operations .b. otherwise throw the exception.)
   5. Step 5: Catch the exception and display the appropriate message.
   6. Step 6: Stop the program.

7. **Conclusion:**
   We have studied the exception handling in python

8. **Viva Questions:**
   - What to you understand by try and except?
   - Explain Zero Division Error.

9. **References:**
   Dr. R. Nageswara Rao, “Core Python Programming”, Dreamtech Press
   Learn Python the Hard Way (3rd Edition)(Zed Shaw’s Hard Way Series

Experiment No. 8

1. **Aim:** Implement Python programs to explore files and directories.

   a. Python program to read the content of file and write it in another file.

   b. Python program to append data to existing file and then display the entire file

2. **Objectives:** From this experiment, the student will be able to

   - Learn File handling in Python

3. **Outcomes:**

   - Student will be able to understand and summarize different File handling operations.

4. **Software Required:** Python 3

5. **Theory:**

   - Python supports file handling and allows users to handle files i.e., to read and write files, along with many other file handling options, to operate on files.
   - Python treats files differently as text or binary and this is important.
   - Each line of code includes a sequence of characters and they form a text file.
   - Each line of a file is terminated with a special character, called the EOL or End of Line characters like comma {,} or newline character. It ends the current line and tells the interpreter a new one has begun.
   - Before performing any operation on the file like read or write, first we have to open that file. For this, we should use Python's inbuilt function open()
   - But at the time of opening, we have to specify the mode, which represents the purpose of the opening file.
   - Syntax: f = open(filename, mode)
   - Where the following mode is supported:
   - r: open an existing file for a read operation.
   - w: open an existing file for a write operation. If the file already contains some data then it will be overridden.
   - a: open an existing file for append operation. It won't override existing data.

- r+: To read and write data into the file. The previous data in the file will not be deleted.
- w+: To write and read data. It will override existing data.
- a+: To append and read data from the file. It won't override existing data.

6. Algorithm:

1. Start
2. Create a file to store the data
3. Open file for writing the data
4. Enter the character from keyboard
5. Write the string into file
6. For reading the character from the file
7. Open the file for reading data.
8. Read all character from file and display in screen.
9. For appending and then reading the data
10. Open the file for reading the data and accept data.
11. Write the string into file and put file pointer of beginning of file
12. Read the strings from file
13. For counting number of lines, words and characters in a file
14. Open the file for reading the data .use if else loop for checking the file existence.

7. Conclusion:

Hence from this experiment we learned the various file handling operations such as reading the content of file and writing it in another file. Also learned to append the data to existing file and then display the entire file.

8.Viva Questions:

- Explain different file handling operations
- What are the different applications of file handling?

9.References:

- Core Python Programming, Dr. R. Nageswara Rao, Dreamtech Press 2
- Beginning Python: Using Python 2.6 and Python 3.1. James Payne, Wrox publication
- Learn Python the Hard Way: (3rd Edition) (Zed Shaw's Hard Way Series)
- Python Projects , Laura Cassell,Alan Gauld,wrox publication

Experiment No. 9

1. **Aim:** Implement Python programs to explore files and directories.
   a. Python program to count number of lines, words, and characters in a file.
   b. Python program to display files available in the current directory

2. **Objectives:** From this experiment, the student will be able to learn
   - To demonstrate File handling in Python

3. **Outcomes:**
   - Students will be able to demonstrate to Understand and summarize different File handling operations

4. **Software Required:** Python 3

5. **Theory:**

Types of files:

- **ASCII Text Files**

  A text file is a stream of characters that can be sequentially processed by a computer in forward direction. For this reason a text file is usually opened for only one kind of operation (reading, writing, or appending) at any given time. Because text files can process characters, they can only read or write data one character at a time. In Python, a text stream is treated as a special kind of file.

  In a text file, each line contains zero or more characters and ends with one or more characters that specify the end of line. Each line in a text file can have maximum of 255 characters. When data is written to a text file, each newline character is converted to a carriage return/line feed character. Similarly, when data is read from a text file, each carriage return/line feed character is converted to newline character.

- **Binary Files:** A binary file is a file which may contain any type of data, encoded in binary form for computer storage and processing purposes. It includes files such as word processing documents, PDFs, images, spreadsheets, videos, zip files, and other executable programs. Like a text file, a binary file is a collection of bytes.

- A binary file is also referred to as a character stream with following two essential differences.

  A binary file does not require any special processing of the data and each byte of data is transferred to or from the disk unprocessed.

- **DIRECTORY METHODS:**
  As we all know a directory is a collection of files where each file may be of a different format. Python has various methods in the os module that help programmers to work with directories. These methods allow users to create, remove, and change directories.
  - **The mkdir () Method**: The mkdir () method of the os module is used to create directories in the current directory. The method takes the name of the directory (the one to be created) as an argument. The syntax of mkdir() is os.mkdir("new_dir_name")

  - **getcwd() Method**: The get cwd () method is used to display the current working directory (cwd). We have already read that, all files and folders whose path does not exist in the root folder are assumed to be present in the current working directory. So to know your cwd is quite important at times and for this getcwd() method is used. The syntax of getcwd() is, os.getcwd()

  - **chdir() Method**: The chdir() method is used to change the current directory. The method takes the name of the directory which you want to make the current directory as an argument. Its syntax is os.chdir("dir_name")

6. Algorithm:

1. Start
2. Create a file to store the data
3. Open file for writing the data
4. Enter the character from keyboard
5. Write the string into file
6. For reading the character from the file
7. Open the file for reading data.
8. Read all character from file and display in screen.
9. For appending and then reading the data

10. Open the file for reading the data and accept data.
11. Write the string into file and put file pointer of beginning of file
12. Read the strings from file
13. For counting number of lines, words and characters in a file
14. Open the file for reading the data .use if else loop for checking the file existence.

7. Conclusion:

8. Viva Questions:

- What is file handle?
- Differentiate between read() and read lines().
- Differentiate write() and write lines()

9. References:

- Dr. R. Nageswara Rao, "Core Python Programming", Dreamtech Press
- Learn Python the Hard Way(3rd Edition)(Zed Shaw's Hard Way Series)

Experiment No. 10

1. **Aim:** For the given application, create a database

2. **Objectives:** From this experiment, the student will be able to learn
   - To demonstrate GUI programming and database operations in Python

3. **Outcomes:**

   - Students will be able to understand different database operations.

4. **Software Required:** Python 3

5. **Theory:** Python programming language has powerful features for database programming. Python supports various databases like SQLite, MySQL, Oracle, Sybase, PostgreSQL, etc. Python also supports Data Definition Language (DDL), Data Manipulation Language (DML), and Data Query Statements. The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard**.**

   In this experiment, we will see the use of SQLite database in the python programming language. It is done by using python's inbuilt, sqlite3 module. You should first create a connection object that represents the database and then creates some cursor objects to execute SQL statements.

   In this experiment we are going to perform following operations:
   1. Database Creation on SQLite
   2. Create a table
   3. Insert Values
   4. Display values
   5. Update Values

6. **Algorithm: (** Student have to write algorithm according to program**)**

7. **Conclusion: In this experiment by using SQLite we have created a database of using SQLite**

8. Viva Questions:

- Name the database supported by Python
- What are the different database operations?

9. References:

- Dr. R. Nageswara Rao, "Core Python Programming", Dreamtech Press
- Learn Python the Hard Way(3rd Edition)(Zed Shaw's Hard Way Series)

Experiment No. 11

1. **Aim:** Design a GUI using Tkinter for a given application

2. **Objectives:** From this experiment, the student will be able to learn
   - To demonstrate GUI programming and database operations in Python

3. **Outcomes:**
   - Students will be able to understand different database operations.

4. **Software Required:** Python 3

5. **Theory:**

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, Tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with Tkinter is the fastest and easiest way to create GUI applications. Creating a GUI using Tkinter is an easy task.

- To create a Tkinter app:

✓ Importing the module – tkinter
✓ Create the main window (container)
✓ Add any number of widgets to the main window
✓ Apply the event Trigger on the widgets.

- Tkinter exposes the following geometry manager classes: pack, grid, and place.
- The *pack()* Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The *grid()* Method − This geometry manager organizes widgets in a table-like structure in the parent widget.
- The *place()* Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

6. **Algorithm:(** Student have to write algorithm according to program**)**

7. **Conclusion:**

8. **Viva Questions:**

   - How to disable an Entry widget in Tkinter?
   - How to resize an Entry Box by height in Tkinter?

9. **References:**

   - Dr. R. Nageswara Rao, "Core Python Programming", Dreamtech Press
   - Learn Python the Hard Way(3rd Edition)(Zed Shaw's Hard Way Series)

Experiment No. 12

1. **Aim:** Write a python program to demonstrate Data Series and Data Frames using Pandas
2. **Objectives:** From this experiment, the student will be able to learn
    - to demonstrate Data Series and Data Frames using Pandas
3. **Outcomes:**
    - Students will be able to write a python program to demonstrate Data Series and Data Frames using Pandas.
4. **Software Required:** Python 3
5. **Theory:**
    - Pandas is an open source library in Python. It provides ready to use high-performance data structures and data analysis tools.
    - Pandas module runs on top of NumPy and it is popularly used for data science and data analytics.
    - NumPy is a low-level data structure that supports multi-dimensional arrays and a wide range of mathematical array operations. Pandas has a higher-level interface. It also provides streamlined alignment of tabular data and powerful time series functionality.
    - DataFrame is the key data structure in Pandas. It allows us to store and manipulate tabular data as a 2-D data structure.
    - Pandas provides a rich feature-set on the DataFrame. For example, data alignment, data statistics, slicing, grouping, merging, concatenating data, etc.
    - DataFrame is the most important and widely used data structure and is a standard way to store data.
    - DataFrame has data aligned in rows and columns like the SQL table or a spreadsheet database.
    - We can either hard code data into a DataFrame or import a CSV file, tsv file, Excel file, SQL table, etc

- Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.
- A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns.

6. **Algorithm:** (Student have to write algorithm according to program)

7. **Conclusion:**

8. **Viva Questions:**

   - What is pandas and why it is used in python?
   - What is the difference between data series and data frames?

9. **References:**

   - Dr. R. Nageswara Rao, "Core Python Programming", Dreamtech Press
   - Learn Python the Hard Way(3rd Edition)(Zed Shaw's Hard Way Series)