



LOVELY
PROFESSIONAL
UNIVERSITY

SIX WEEKS SUMMER TRAINING REPORT

On

Object-Oriented Programming Using Python

INT 343: Training In Programming

Submitted By

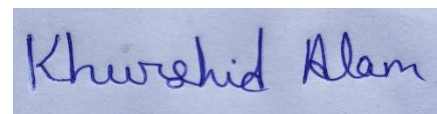
Khurshid Alam

B.Tech. (Information Technology)

Reg No- 12000571

DECLARATION

I hereby declare that I have completed my six weeks of summer training at **E-Box**(One of the world's leading online certification course providers) from 3rd May 2022 to 11th June 2022 under the guidance of **Mr. Subhasis Chaudhuri**. I also declare that I have worked with full dedication during these six weeks of training and my learning outcomes fulfill the requirements of training for the award of the degree of Bachelor of Technology in Computer Science & Engineering(IT), Lovely Professional University, Phagwara.



(Signature of the Student)

Khurshid Alam

Reg. No – 1200571

Date- 14/07/2022

ACKNOWLEDGMENT

I Khurshid Alam, a student of **Lovely Professional University, Phagwara**, Punjab would like to thank our university for giving me such an interesting opportunity to undergo the summer training program. As a part of summer training, I have opted for **Object-Oriented Programming using Python by E-Box**. The writing of this project report has been one of the significant academic challenges I have faced. Without the support, patience, and guidance of the institution Lovely Professional University, and friends this task would not have been completed. It is so they and I owe my deepest gratitude.

It gives me immense pleasure in presenting this project report on **“Object-Oriented Programming Using Python”**. I hereby take this opportunity to add a special note of thanks to Lovely Professional University and the faculty members for designing such an interesting summer training program and providing exceptional courses and training programs from different organizations. Without the help, support, and energy, this project wouldn't have kick-started nor would have reached fruitfulness.

I am extremely thankful to the course instructor **Mr. Subhasis Chaudhuri** for guiding me throughout the course duration. I also respect and thank E-Box for providing me an opportunity to do the training program with all support and guidance which made me complete the training program on the topic **“Object-Oriented Programming Using Python”**.

Khurshid Alam

Certification of Course Completion



CERTIFICATE OF COMPLETION



This is to certify that

Khurshid Alam

has successfully completed the E-Box Online Certification Course on

"LPU - Object Oriented Programming using Python - Internship"

during the period May 2021 - May 2022.

Managing Director

Amphisoft



Table of Content

1. Introduction to Python

- Introduction
- Features of Python

2. Technology Learnt

- Control Statement
- Loop in Python
- Function in Python
- OPPS in Python
- Python Classes and object
- Python Inheritance and Polymorphism
- Exception

3. Project

- Overview of the Project
- Flowchart of the Project
- Implementation
- Project Legacy

4. Bibliography

Introduction to Python

- Introduction:

Python is a programming language that combines the features of C and Java. It offer elegant style of developing programs like C. When the programmers want to go for object orientation, Python offers classes and objects like Java. In Python, the program to add two numbers will be as follows:

```
#Python program to add two numbers

a = b = 10    #take two variables and store 10 in to them
print("Sum=" (a+b))    # display their sum
```

The preceding code is easy to understand and develop. Hence, Python is gaining popularity among the programming folks. Of course, there are several other features of Python which we will discuss in future which make it the preferred choice of most programmers.

- Features of Python

- Easy to Code

Python is a very high-level programming language, yet it is effortless to learn. Anyone can learn to code in Python in just a few hours or a few days. Mastering Python and all its advanced concepts, packages and modules might take some more time. However, learning the basic Python syntax is very easy, as compared to other popular languages like C, C++, and Java.

➤ Easy to Read

Python code looks like simple English words. There is no use of semicolons or brackets, and the indentations define the code block. You can tell what the code is supposed to do simply by looking at it.

➤ Open-Source

Python is developed under an OSI-approved open source license. Hence, it is completely free to use, even for commercial purposes. It doesn't cost anything to download Python or to include it in your application. It can also be freely modified and re-distributed. Python can be downloaded from the official Python website.

➤ Interpreted

When a programming language is interpreted, it means that the source code is executed line by line, and not all at once. Programming languages such as C++ or Java are not interpreted, and hence need to be compiled first to run them. There is no need to compile Python because it is processed at runtime by the interpreter.

➤ Portable

Python is portable in the sense that the same code can be used on different machines. Suppose you write a Python code on a Mac. If you want to run it on Windows or Linux later, you don't have to make any changes to it. As such, there is no need to write a program multiple times for several platforms.

➤ Dynamically Typed

Many programming languages need to declare the type of the variable before runtime. With Python, the type of the variable can be decided during runtime.

This makes Python a dynamically typed language. For example, if you have to assign an integer value 20 to a variable “x”, you don’t need to write `int x = 20`. You just have to write `x = 15`.

Technology Learnt

■ Control Statement

➤ The if Statement:

This statement is used to execute one or more statement depending on whether a condition is True or not. The syntax or correct format of if statement is given below:

```
# Example of If statement  
Num = 1  
If num ==1  
    Print("One")
```

First, the condition is tested. If the condition is True, then the statements given after colon (:) are executed. We can write one or more statements after colon (:). If the condition is False, then the statements mentioned after colon are not executed.

➤ The if...else Statement:

This statement is used to execute one or more statement depending on whether a condition is True or not. The syntax or correct format of if statement is given below:


```
X = 10
if x%2 ==0
    Print(X, "is even number" )
Else:
    Print(X, "is a odd number")
```

First, the condition is tested. If the condition is True, then the statements given after colon (:) are executed. We can write one or more statements after colon (:). If the condition is False, then the statements mentioned after colon are not executed.

➤ The if...else Statement:

Sometimes, the programmer has to test multiple conditions and execute statements depending on those conditions. if... elif... else statement is useful in such situations Consider the following syntax of if... elif... else statement:

```
num = 5
if num == 0:
    Print(num, "is zero")
elif num >0:
    print(num, "is positive")
else:
    print(num, "is negative")
```

When condition1 is True, the statements1 will be executed. If condition1 is False, then condition2 is evaluated. When condition2 is True, the statements2 will be executed. When condition2 is False, the condition3 is tested. If condition3 is True, then statements3 will be executed.

▪ Loop in Python

➤ The while loop:

A statement is executed only once from top to bottom. For example, if is a statement that is executed by Python interpreter only once. But a loop is

useful to execute repeatedly. For example, while and for are loops in Python. They are useful to execute a group of statements repeatedly several times. The while loop is useful to execute a group of statements several times repeatedly depending on whether a condition is True or False. The syntax or format of while loop is following

```
# To display number from 1 to 10
X = 1
While x<10:
    print(x)
    x+=1
print ("End")
```

Here, 'statements' represents one statement or a suite of statements. Python interpreter first checks the condition. If the condition is True, then it will execute the statements written after colon (:). After executing the statements, it will go back and check the condition again. If the condition is again found to be True, then it will again execute the statements. Then it will go back to check the condition once again. In this way, as long as the condition is True, Python interpreter executes the statements again and again. Once the condition is found to be False, then it will come out of the while loop.

➤ The for loop:

The for loop is useful to iterate over the elements of a sequence. It means, the for loop can be used to execute a group of statements repeatedly depending upon the number of elements in the sequence. The for loop can work with sequence like string, list, tuple, range etc. The syntax of the for loop is given below:

```
# To display each character from a string
str = "Hello"
for ch in str:
    print (ch)
```

The first element of the sequence is assigned to the variable written after 'for' and then the statements are executed. Next, the second element of the sequence is assigned to the variable and then the statements are executed second time. In this way, for each element of the sequence, the statements are executed once. So, the for loop is executed as many times as there are number of elements in the sequence.

➤ Nested loop

It is possible to write one loop inside another loop. For example, we can write a for loop inside a while loop or a for loop inside another for loop. Such loops are called 'nested loops'.

```
for i in range(3):  
    for j in range(4):  
        print('i=', i, '\t', 'j=', j)
```

The outer for loop repeats for 3 times by changing i values from 0 to 2. The inner for loop repeats for 4 times by changing j values from 0 to 3. Observe the indentation (4 spaces) before the inner for loop. The indentation represents that the inner for loop is inside the outer for loop. Similarly, print() function is inside the inner for loop.

When outer for loop is executed once, the inner for loop is executed for 4 times. It means, when i value is 0, j values will change from 0 to 3.

▪ Function in Python

A function is similar to a program that consists of a group of statements that are intended to perform a specific task. The main purpose of a function is to perform a specific task or work. Thus when there are several tasks to be performed, the programmer will write several functions. There are several built-in' functions in Python to perform various tasks. For

example, to display output, Python has `print()` function. Similarly, to calculate square root value, there is `sqrt()` function and to calculate power value, there is `power()` function. Similar to these functions, a programmer can also create his own functions which are called 'user-defined' functions.

We can define functions to provide the required functionality. Here are simple rules to define a function in Python.

Function blocks begin with the keyword **def** followed by the function name and parentheses (()).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.

The code block within every function starts with a colon (:) and is indented.

The statement `return [expression]` exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as `return None`.

➤ Syntax

```
def functionname( parameters ):
    "function_docstring"
    function_suite
    return [expression]
```

By default, parameters have a positional behavior and you need to inform them in the same order that they were defined.

➤ Example

The following function takes a string as input parameter and prints it on standard screen.

```
def printme( str ):
    "This prints a passed string into this function"
    print str
    return
```

➤ Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

Following is the example to call printme() function –

```
# Now call printme function
printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

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

I'm first call to user defined function!

Again second call to the same function

➤ Pass by reference vs value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example

–

```
# Function definition
def changeme( mylist ):
    "This changes a passed list into this function"
    mylist.append([1,2,3,4]);
    print "Values inside the function: ", mylist
    return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result –

Values inside the function: [10, 20, 30, [1, 2, 3, 4]]

Values outside the function: [10, 20, 30, [1, 2, 3, 4]]

➤ Function Arguments

You can call a function by using the following types of formal arguments –

Required arguments

Keyword arguments

Default arguments

Variable-length arguments

➤ Local and Global Variable

When we declare a variable inside a function, it becomes a local variable. A local variable is a variable whose scope is limited only to that function where it is created. That means the local variable value is available only in

that function and not outside of that function. In the following example, the variable 'a' is declared inside myfunction() and hence it is available inside that function. Once we come out of the function, the variable 'a' is removed from memory and it is not available. Consider the following code:

```
# local variable in a function

def myfunction():

    a=1 # this is local var

    a+=1 # increment it

    print (a) # displays 2

myfunction()

print(a)  # error, not available
```

```
# global variable in a function

a = 1

def myfunction():

    b=1 # this is local var

    print ( 'a = ' , a)

    print ( 'b = ' , b)

myfunction()

print(a)  # available

print(b)  # error, Not Available
```

■ OPPS in Python

Object-oriented programming (OOP) is a programming pattern based on the concept of objects. Objects consists of data and methods. The object's data are it's properties, that defines what it is. And the object's methods, are it's functions, that defines what the object can do. Object-

oriented style of programming is very popular because of its ability to map the virtual world entities i.e. our code, to the real world objects. OOPS concepts are widely used by many popular programming languages due to the several advantages it provides.

In OOPS, all programs involve creation of classes and objects. This makes programs lengthy. For example, we have to write all the statements of the program inside a class and then create objects to the class. Then use the features of the class through objects. This type of programming requires much code to perform a simple task like adding to numbers. Also, the program execution takes more time. So, for simple tasks, it is better to go for procedure oriented approach which offers less code and more speed. For example, a C program executes faster than its equivalent program written in Python or Java!

Even though, Python is an object oriented programming language like Java, it does not force the programmers to write programs in complete object oriented way. Unlike Java Python has a blend of both the object oriented and procedure oriented features. Hence Python programmers can write programs using procedure oriented approach (like C) or object oriented approach (like Java) depending on their requirements. This is definitely an advantage for Python programmers!

■ **Classes and object**

The entire OOPS methodology has been derived from a single root concept called 'object'. An object is anything that really exists in the world and can be distinguished from others. This definition specifies that everything in this world is an object. For example, a table, a ball, a car, a dog, a person, etc. will come under objects. Then what is not an object? If something does not really exist, then it is not an object. For example, our thoughts, imagination, plans, ideas etc. are not objects, because they do not physically exist.

➤ **Creating Classes and Objects in Python**

Let's create a class with the name Person for which Raju and Sita are objects. A class is created by using the keyword, class. A class describes

the attributes and actions performed by its objects. So, we write the attributes (variables) and actions (functions) in the class as:

```
# This is a class  
  
class Person:  
  
    # attributes means variables  
  
    name = 'Raju'  
  
    age = 20  
  
    # actions means functions  
  
    def talk (cls):
```

➤ The self Variable

Let's create a class with the name Person for which Raju and Sita are objects. A class is created by using the keyword, class. A class describes the attributes and actions performed by its objects. So, we write the attributes (variables) and actions (functions) in the class as:

```
S1 = Student()
```

➤ Constructor

A constructor is a special method that is used to initialize the instance variables of a class. In the constructor, we create the instance variables and initialize them with some starting values. The first parameter of the constructor will be 'self' variable that contains the memory address of the instance. For example,

```
Def __init__(self):  
    self.name = "Khurshid"  
    self.marks = 900
```

Here, the constructor has only one parameter, i.e. 'self'. Using 'self.name' and 'self.marks', we can access the instance variables of the class. A

constructor is called at the time of creating an instance. So, the above constructor will be called when we create an instance as:

```
S1 = Student()
```

➤ Inner Classes

Writing a class within another class is called creating an inner class or nested class. For example, if we write class B inside class A, then B is called inner class or nested class. Inner classes are useful when we want to sub group the data of a class. For example, let's take a person's data like name, age, date of birth etc. Here, name contains a single value like Charles', age contains a single value like '30' but the date of birth does not contain a single value. Rather, it contains three values like date, month and year. So, we need to take these three values as a sub group. Hence it is better to write date of birth as a separate class Dob inside the Person class. This Dob will contain instance variables dd, mm and yy which represent the date of birth details of the person.

Generally, the inner class object is created within the outer class. Let's take Person class as outer class and Dob as inner class. Dob class object is created in the constructor of the Person class as:

```
class Person:
    def __init__(self):
        self.name = 'Charles'
        self.db = self.Dob() # this is Dob object
```

■ Python Inheritance

Inheritance is the capability of one class to derive or inherit the properties from another class.

➤ 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.

```
#Python Inheritance Syntax  
Class BaseClass:  
    {Body}  
Class DerivedClass(BaseClass):
```

■ Polymorphism

Polymorphism is a word that came from two Greek words, poly means many and morphos means forms. If something exhibits various forms, it is called polymorphism. Let's take a simple example in our daily life. Assume that we have wheat flour. Using this wheat flour, we can make burgers, rotis, or loaves of bread. It means same wheat flour is taking different edible forms and hence we can say wheat flour is exhibiting polymorphism.

In programming, a variable, object or a method will also exhibit the same nature as that of the wheat flour. A variable may store different types of data, an object may exhibit different behaviors in different contexts or a method may perform various tasks in Python. This type of behavior is called polymorphism. So, how can polymorphism? If a variable, object or method exhibits different behavior in different we define contexts, it is called polymorphism. Python has built-in polymorphism.

Project

➤ Overview of the Project

- **Project Topic:** Hotel Room book Management system(CLI) using Python

- **Introduction:** A CLI-based Python application featuring Hotel Room Book system where user can book Hotel and Calculate the total amount according to the features.

- **Overview:** Created a Hotel room book management system, where user can book Hotel and Calculate the total amount according to the features.

- **Functionality:** The application calculate the total amount according to the room type, number of person, and features.

- **Platform:** The whole project is a Code-based.

- **Development Responsibility:** I **Khurshid Alam**, would be developing this project and its related stuff.

- **Goals and Scopes:** To create a hotel room reservation system and learn about the various features of Python such as objects, classes, inheritance, etc.

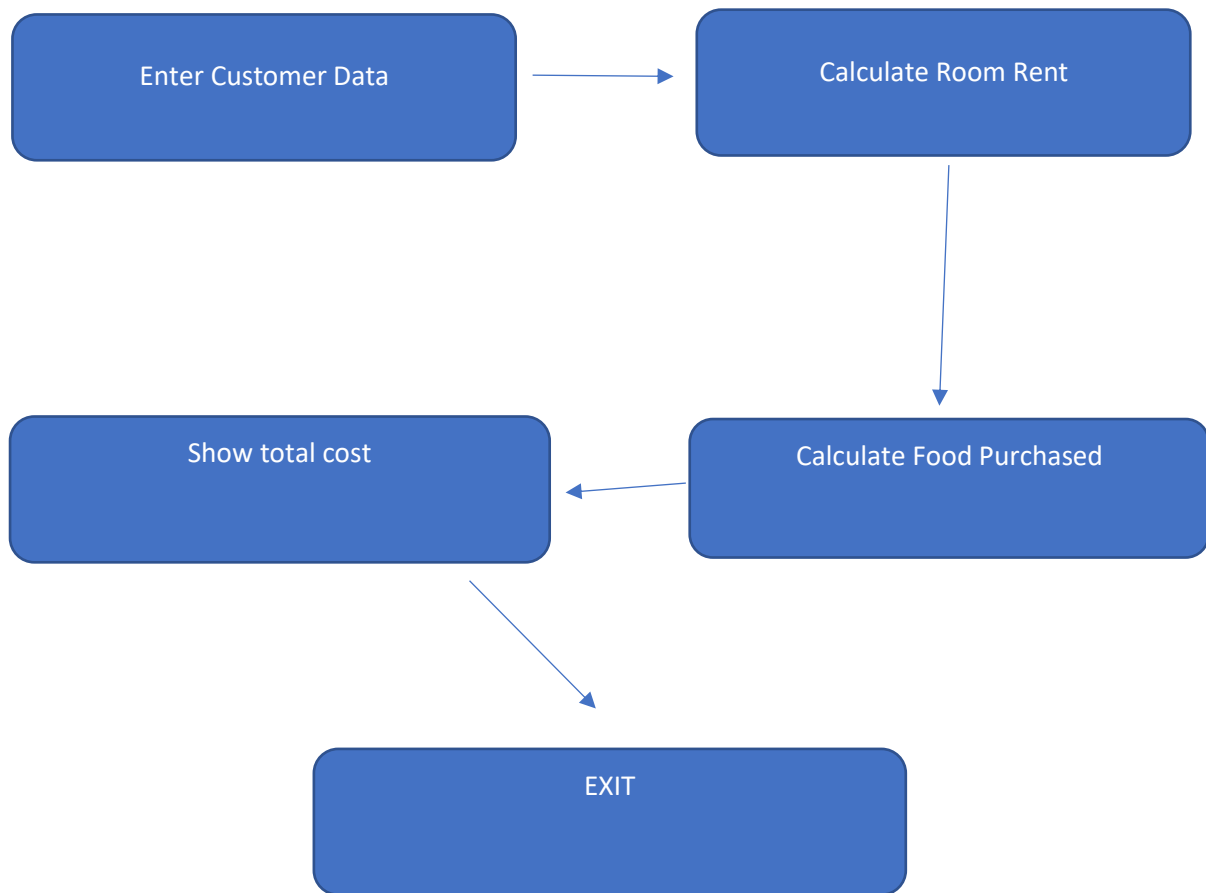
- **Risk Associated:** Maybe due to system configurations, the program may not be feasible to run properly in older IDE's.

- **Technical Process:** To make this project I have used Pycharm.

➤ Flowchart of the Project

A flowchart is a diagrammatic representation of an actual logic of a project which shows the essential overall demonstration of the actual architecture of the project.

The flowchart of the mini project on the topic “Railway Reservation System” is follows in a step by step:



➤ Implementation:

```
def __init__(self, rt='', s=0, p=0, r=0, t=0, a=1000, name='',  
address='', cindate='', coutdate='', rno=1):  
  
    print("\n\n*****WELCOME TO KHURSHID`S HOTEL*****\n")  
  
    self.rt = rt  
  
    self.r = r  
  
    self.t = t  
  
    self.p = p  
  
    self.s = s  
    self.a = a  
    self.name = name  
    self.address = address  
    self.cindate = cindate  
    self.coutdate = coutdate  
    self.rno = rno
```

```

def inputdata(self):
    self.name = input("\nEnter your Fullname:")
    self.address = input("\nEnter your address:")
    self.cindate = input("\nEnter your check in date:")
    self.coutdate = input("\nEnter your checkout date:")
    print("Your room no.:", self.rno, "\n")

def roomrent(self): # sell1353

    print("We have the following rooms for you:-")

    print("1. Class A---->4000")
    print("2. Class B---->3000")
    print("3. Class C---->2000")
    print("4. Class D---->1000")

    x = int(input("Enter the number of your choice Please->"))
    n = int(input("For How Many Nights Did You Stay:"))

    if (x == 1):

        print("you have choose room Class A")

        self.s = 4000 * n

    elif (x == 2):

        print("you have choose room Class B")

        self.s = 3000 * n

    elif (x == 3):

        print("you have choose room Class C")

        self.s = 2000 * n

    elif (x == 4):

        print("you have choose room Class D")

        self.s = 1000 * n

    else:

        print("please choose a room")

    print("your choosen room rent is =", self.s, "\n")

def foodpurchased(self):

    print("*****RESTAURANT MENU*****")

    print("1.Dessert----->100", "2.Drinks----->50", "3.Breakfast--->90", "4.Lunch---->110", "5.Dinner--->150", "6.Exit")

```

```

while (1):

    c = int(input("Enter the number of your choice:"))

    if (c == 1):
        d = int(input("Enter the quantity:"))
        self.r = self.r + 100 * d

    elif (c == 2):
        d = int(input("Enter the quantity:"))
        self.r = self.r + 50 * d

    elif (c == 3):
        d = int(input("Enter the quantity:"))
        self.r = self.r + 90 * d

    elif (c == 4):
        d = int(input("Enter the quantity:"))
        self.r = self.r + 110 * d

    elif (c == 5):
        d = int(input("Enter the quantity:"))
        self.r = self.r + 150 * d

    elif (c == 6):
        break;
    else:
        print("You've Enter an Invalid Key")

    print("Total food Cost=Rs", self.r, "\n")

def display(self):
    print("*****HOTEL BILL*****")
    print("Customer details:")
    print("Customer name:", self.name)
    print("Customer address:", self.address)
    print("Check in date:", self.cindate)
    print("Check out date", self.coutdate)
    print("Room no.", self.rno)
    print("Your Room rent is:", self.s)
    print("Your Food bill is:", self.r)

    self.rt = self.s + self.t + self.p + self.r

    print("Your sub total Purchased is:", self.rt)
    print("Additional Service Charges is", self.a)
    print("Your grandtotal Purchased is:", self.rt + self.a, "\n")
    self.rno = 12

def main():
    a = hotelmanage()

    while (1):
        print("1.Enter Customer Data")

        print("2.Calculate Room Rent")

        print("3.Calculate Food Purchased")

        print("4.Show total cost")

        print("5.EXIT")

```

```
b = int(input("\nEnter the number of your choice:"))
    if (b == 1):
        a.inputdata()

    if (b == 2):
        a.roomrent()

    if (b == 3):
        a.foodpurchased()

    if (b == 4):
        a.display()

    if (b == 5):
        quit()

main()
```

➤ **Git:**

https://github.com/Khurshid-1/Six_week_summer_training_project.git

➤ **Project Legacy**

Technical and Managerial Lessons Learnt

Throughout the training I have learnt many technological and managerial lessons which helps me to decide about in the cloud security operations and another circumstances during the project and training sessions.

➤ **Technical Lessons Learnt:**

- How to use python in real life project.
- How to implements OOPS.
- How to create a CLI.

➤ **Managerial Lessons Learnt:**

- Learnt about how to manage project.
- Learnt how to handle exceptions.

Bibliography

I have completed this report with the help of the following references from various website, books and with the training materials offered by Ebox.

<https://app.e-box.co.in/>

Books I have followed

- Core python programming (Nageswara rao)