Laboratory Manual

Python Lab
(CS-506)

For

Third Year Students
Department: Computer Science & Engineering

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Department of Computer Science and Engineering

Vision of CSE Department:

The department envisions to nurture students to become technologically proficient, research competent and socially accountable for the welfare of the society.

Mission of the CSE Department:

I. To provide high quality education through effective teaching-learning process emphasizing active participation of students.

II. To build scientifically strong engineers to cater to the needs of industry, higher studies, research and startups.

III. To awaken young minds ingrained with ethical values and professional behaviors for the betterment of the society.

Program Educational Objectives:

Graduates will be able to

I. Our engineers will demonstrate application of comprehensive technical knowledge for innovation and entrepreneurship.

II. Our graduates will employ capabilities of solving complex engineering problems to succeed in research and/or higher studies.

III. Our graduates will exhibit team-work and leadership qualities to meet stakeholder business objectives in their careers.

IV. Our graduates will evolve in ethical and professional practices and enhance socioeconomic contributions to the society.

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Program Outcomes (POs):

Engineering Graduates will be able to:

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

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

Course Outcomes

Python(CS-506)

CO1:	Identify the basic data types, operators, variables and functions.
CO2 :	Ability to analyze the importance of object oriented programming over structural programming.
CO3 :	Determine the list, tuples, dictionary and set build in container data types.
CO4 :	Able to Implement object oriented database and Graphical user interface application using packages.
CO5 :	Develop the ability to analyse and write database applications in Python programming.

Course	Course Outcomes	CO Attainment	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	Identify the basic data types,operators, variables and functions.		3				1								2		
CO2	Ability to analyze the importance of object oriented programming over structural Programming.		2				2								1	2	
CO3	Determine the list, tuples, dictionary and set build in container data types.										1					1	2
CO4	Able to Implement object oriented database and Graphical user interface application using Packages.				1		1					1			2		1
CO5	Develop the ability to analyses and write database applicationsin Python programming.											1		1	1		1

INSTITUTE OF TECHNOLOGY & MANAGEMENT
GWALIOR • MP • INDIA

List of Program

S. No.	List	Course Outcome	Page No.
	INTRODUCTION TO PYTHON		**1-11**
1	To write a python program that takes in command line arguments as input and print thenumber of arguments. To write a python program find the square root of a number (Newton's method)	CO1	12 - 13
2	To write a python program exponentiation (power of a number). To write a python program to compute the GCD of two numbers. To write a python program first n prime numbers.	CO2	14 – 16
3	To write a python program find the maximum of a list of numbers. To write a python program to perform Matrix Multiplication.	CO1	17 - 18
4	To write a python program to find the most frequent words in a text file.	CO3	19
5	To write a python program linear search. To write a python program Binary search	CO4	20 - 21
6	To write a python program selection sort. To write a python program Insertion sort.	CO4	22 - 23
7	To write a python program merge sort.	CO4	24
8	To write a python program simulate bouncing ball in Pygame.	CO1	25
Value added programs			
9	To demonstrate working of classes and objects To demonstrate class method and static method To demonstrate constructors	CO4	26 – 29
10	To demonstrate inheritance. To Concept of polymorphism in python (method overloading and overriding)	CO4	30 - 31
Develop database application using python as project base learning		CO5	

INTRODUCTION TO PYTHON

In Python programming section, the applications of Python are taken into account. Applications of Python which are taken into details according to the syllabus prescribed by RGPV Bhopal for this lab are:

a) Console Based Programming
b) OOPs Based Programming
c) GUI Based Programming
d) String
e) List
f) Tuple
g) Dictionary

LAB REQUIREMENTS

For Python Programming

- Python 3.7
- PyCharm
- Anaconda

This Interpreter has no special hardware requirements as such. Any System with a minimum 256 MBRAM and any normal processor can use for this lab.

Python

The objective of this exercise is to become familiar with the Python IDE for version 3.X while introducing basic mathematical operations, variable types, and printing options.

Background

Virtually all modern programming languages make us of an IDE, or Integrated Development Environment, which allows the creation, editing, testing, and saving of programs and modules. InPython, the IDE is called IDLE (like many items in the language, this is a reference to the Britishcomedy group Monty Python, and in this case, one of its members, Eric Idle).
Before opening IDLE, it is worth recalling that there are three basic types of simple variables in Python: integers (whole numbers, commonly used as an index), floats (that is, numbers

with a decimal point, AKA real numbers), and strings (collections of alphanumeric characters such as names, sentences, or numbers that are not manipulated mathematically such as a part number or zip code). A legal variable name must start with a letter. It is then optionally followed by some collection of letters, numerals and the underscore. It cannot contain any other characters or spaces,and cannot be a reserved word (i.e., a word with a special meaning in the language such as a command or operator). In Python, variables may be created by simply declaring them and assigning a value to them. Examples include:

a=2.3

name=”Joe”

It is best to think of the equal sign as “gets”. That is, think of the first example as “the variable a gets the floating point value 2.3” and the second as “the variable name gets the string Joe”. An assignment command such as these literally reserves space in the computer’s memory for the variable and tags it with the variable name. Then, it stores the appropriate value at that location for future use.

OBTAINING USER DATA

Objective

Interactive programs require data from the user (i.e., the person running the program, who is not necessarily the programmer). In this exercise, the function input() will be examined in order to create a simple Ohm's Law calculator

Introduction

The most general means of obtaining information from the user is the input() function. When Python executes this command it will wait for the user to enter a string of characters until the userhits the Enter key. These characters are then assigned as a string to a variable. Usually, some formof user prompt will be required (i.e., the question posed to the user). This can be accomplished viaa separate print statement or as an argument to the function. Below is a simple example which asks the user for their name and then prints it back out.

print("What is your name? ")n = input()

print("Hello", n)

Alternately, this can be shortened with the following:

n = input("What is your name? ")print("Hello", n)

It is important to remember that this function always returns a string variable. If the entered data is numeric, it must be turned into either a float or integer. This can be accomplished via the float()and int() functions. For example:

p = float(input("What is your weight in pounds? "))kg = p / 2.2

print("You weigh approximately", kg, "kilograms")

Conditionals: if

The objective of this exercise is to become familiar with the concept of branching. We introducethe if conditional statement for simple decision making. We shall also introduce the concept of menu-driven programs. In the process, we shall create a program which estimates battery life.

Introduction

Our prior programs could be classified as simple linear or straight-line programs. The program flow was fairly straight forward: Give the user directions, ask the user for data, perform a few calculations based on those data and then print out appropriate results. The next level up in sophistication is the concept of branching. That is, the execution path through the code can vary depending on certain conditions. You might think of this as the program making decisions to do one thing or another. The fundamental conditional operation is the if statement. It looks somethinglike this:

if conditional expression:Resulting action

The conditional expression is some manner of test, for example to see if one variable is larger thananother. The tests include = = (same as), != (not same as), >, <, >= and <=. The logical directivesand and or are also available. The resulting action is any legal block of Python code. It may be a single line or a multitude of lines. So, if the conditional expression is true, the resulting action is performed. If the expression is not true, the action is skipped. In either case, program execution picks up at the next line after the resulting action block. It is extremely important to note that theresulting action block must be indented. All lines of the block must be indented by the same amount. This is how Python recognizes that it is a single block of code.

As an example, suppose we'd like to test to see if variable A is larger than variable B. If it is, we'dlike to print out the message: "It's bigger". After this, we want to print out the message "Done", whether or not A was larger.

```
if A > B:
        print( "It's bigger" )print( "Done" )
```

Because the second print statement is not indented, it is not part of the block, therefore it is alwaysexecuted. If the second print statement had been indented instead, then "Done" would

only be printed if A was larger than B. A common beginner's syntax error is to forget the colon at the endof the if statement.

For another example, consider that you have a floating point variable named T that represents a computed time in hours. Instead of printing this out as hours with a fraction, you prefer to presentit as hours and minutes. A floor divide can be used to obtain the whole hours:

```
h = T // 1.0
```

Similarly, a modulo can be used to obtain the fractional portion, which when multiplied by 60.0 will yield the minutes:

```
m = (T % 1.0) * 60.0
```

So, you could print out the result as follows:

```
print( "The time is", h, "hours and", m, "minutes" )
```

Of course, what it the minutes portion works out to zero? Reading something like "The time is 5 hours and 0 minutes" looks a little strange. We'd prefer to leave off the "and 0 minutes" portion.This can be achieved with a simple set of if tests:

```
if m != 0.0:
        print( "The time is", h, "hours and", m, "minutes" )if m == 0.0:
        print( "The time is", h, "hours" )
```

This sort of "one-or-the-other" construct is fairly common. To make life a little simpler, we can use the else clause:

```
if m != 0.0:
        print( "The time is", h, "hours and", m, "minutes" )
else:
        print( "The time is", h, "hours" )
```

If m is non-zero, the full print statement is used, otherwise (else) the simplified version is used. Enter the completed program below and try it with several different values, some whole numbers,others not, and inspect the results:

```
T = float(input("Please enter a time value: "))h = T // 1.0
m = (T % 1.0) * 60.0
if m != 0.0:
        print( "The time is", h, "hours and", m, "minutes" )
else:
```

```
print( “The time is”, h, “hours” )
print( “Done!” )
```

If you look carefully, you might note that under certain circumstances the printout may still beless than satisfactory (hint: what about seconds?). How might this issue be fixed?

ITERATION

Objective

The objective of this exercise is to become familiar with the concept of iteration, also known as looping. We shall also investigate the creation of simple text-based graphs. In the process, a program that will illustrate the Maximum Power Transfer Theorem will be created.

Introduction

The ability to repeat a series of instructions with controlled variance is an extremely powerful computing tool. There are a few different ways to do this in Python, each with their own strengths.The first loop control structure is the while loop. At first glance this looks something like an if statement:

```
while control expression:statement block
```

The statement block, which might be many lines long, will be repeated as long as the control expression is true. Like the if statement, this block must be indented. The control expression is basically the same as those used in if statements: They tend to be simple variable tests. Further, it is important that one or more of the variables used in the control expression change duringthe looping process otherwise the loop will try to run forever. For example:

```
x=1
while x<10:
    print( x )
```

A second technique to create a loop is through the for statement. The template looks similar to the while structure:

```
for variable in value list:
    statement block
```

value list can be a simple listing of values such as:

```
for x in 1,3,25,17:
```

For example:

```
for x in range(5):
    print( x )
```

The code above will print the numbers 0, 1, 2, 3 and 4. Separate starting and ending values may alsobe used:

```
for x in range(3,7):
    print( x )
```

This will print out the values 3, 4, 5, and 6. Next, an increment value may be included:

```
for x in range(3,11,2):
    print( x )
```

TUPLES

Objective

The objective of this exercise is to become familiar with tuples and have a little fun besides.

Introduction

In broad terms, Python has two kinds of variables: Those that are simple and contain single items, examples being floats and integers; and compound types that contain many instances of items. Theseare called sequences. A string is a type of sequence because it is made up of a collection of individualcharacters. Although strings are often treated as a single item, it is possible to extract individual characters or groups of characters from them (a process known as slicing). Another type of sequenceis the tuple (think of this as a contraction of multiple). A basic tuple can contain a group of floats or ints. It could also contain a group of sequences such as strings or even other tuples. For example, consider a tuple that contains a series of voltage settings. It might be initialized like this:

V = (3.5, 2.0, 4.5, 6.0, 50.0, 10.0)

The sequence is defined with enclosing parentheses and the individual items are separated by commas. Square brackets are used to access any given item or slice with the initial item at location 0, as shown in the examples below:

print(V[1])

print(V[4])

These yield 2.0 and 50.0, respectively. A slice refers to a range of locations. Two values are specifiedseparated by a colon: The first is the starting point while the second is the ending point (which is itself not included). It is also worth noting that a slice is itself a sequence and therefore will be printedwith surrounding parentheses. For example:
print(V[1:4])

This prints (2.0, 4.5, 6.0) If the start point is left off, it is assumed to be 0.
Thus,print(V[:4])

yields (3.5, 2.0, 4.5, 6.0).

Finally, a third argument may be added which indicates an increment. For example:
print(V[0:4:2])
yields (3.5, 4.5).

You can determine the number of items in a sequence by using the len()
function:print(len(V))

FUNCTIONS, LIST AND FILES

Objective

There are three major objectives to this exercise: First, to become familiar with programmer-createdfunctions, second, to utilize lists, and third, to explore methods to read and manipulate data containedin external files.

Introduction - Functions

Programmer defined functions are very useful as a means of compartmentalizing and reusing code. Once a bit of code has been created that performs a certain task, it can be reused over and over. Thisalso makes the subsequent code more readable. If these functions prove to be widely applicable, theymay be placed into custom modules so that they can be used conveniently in other programs. In Python, these functions must be defined (or imported if they're in a module) before they are called within the main program. They may or may not have arguments and they may or may not return a value (possibly more than one).

By comparison, think of a typical function such as round(). It takes two arguments: The variable youwish to round and the number of places to round to. It returns a single value, namely the rounded version of the original argument. For example:

```
x = round( y, 2 )
```

y and 2 are the arguments to the function and it returns a result which we then assign to the variablex (that is, x gets yrounded to 2 places). Suppose you wish to create a function that produced (returned)the parallel equivalent of two resistors. The function would look something like this, using product-sum rule:

```
def parallel( r1, r2 ):
        rp = r1*r2/(r1+r2)return rp
```

Introduction - lists

The second item of interest is the list. Lists are sequences like tuples, but unlike tuples, lists are mutable, that is, the elements within a list maybe changed. Tuples are best thought of as a collectionof constants in comparison. Lists are defined using square brackets [] instead of using parentheses() like a tuple. Like tuples and strings, lists are accessed by using square brackets []. Lists can be sliced just like tuples.

```
T = (12,43,17)          # This is a tuple definition
L = [12,43,17]          # This is a list definition
print( T[0] )           # print first element of tuple
print( L[0] )           # print first element of list
```

Introduction - Files

Often, it is not practical to hold data within a program or expect the user to re-enter the data each time the program is used. Imagine how impractical a word processor would be if documents could not be saved external to the program itself. This is where the concept of files comes in. Ultimately,there are only a handful of things programmer's normally need to do with files: Open them (i.e., gain access to them, often exclusively, through the use of an appropriate filename and/or path), read data from them, write data to them, move around in them (for example, to skip unneeded sections), and release or close them (so that other programs can gain access). While any program that deals with files will have to open and close them, whether or not they are read, written to andmoved within will depend on what the program needs from the file. In the exercise that follows, only opening, reading, and closing will be used. Also, files can be generally of two types, binary and text. Binary files are potentially more powerful but text files are usually easier to use. In the exercise that follows we shall only look at text files. These files contain strings that may be read with any text editor while binary files are generally not readable with ordinary text editors.

In order to gain access to a file, it must first be opened:

fil = open(filename, mode)

filename is the literal disk file name such as H:\myfolder\myfile.dat. This is a string that can be hardcoded as a constant (rarely) or more commonly obtained from the user via a input() statement.mode is a string that describes how the file is to be accessed. "r" is used for reading and "w" may be used for writing. There are other modes as well. fil is a file object that is returned to you. It willbe needed for subsequent read and write calls. Note that Python allows several files to be open at once, hence the need for file objects. So, a read mode access might look like this:

fn = input("Please enter the file name: ")fil = open(fn, "r")

Once the file object is obtained, data may be read from the file. Data can be read character by character or line by line. Line oriented files are easily read as follows:

str = fil.readline()

When you are done with the file, you need to close it. close() is another file object

method:fil.close()

Program 1(a)

Object: To write a python program that takes in command line arguments as input and print the number of arguments.

Procedure: Python provides a getopt module that helps you parse command-line options and arguments.

$ python test.py arg1 arg2 arg3

The Python sys module provides access to any command-line arguments via the sys.argv. This serves two purposes −

- sys.argv is the list of command-line arguments.
- len(sys.argv) is the number of command-line arguments.

Here sys. argv[0] is the program ie. script name.

Command to be executed on command line:

$ python test.py arg1 arg2 arg3

Result:

Number of arguments: 4

Argument List: ['test.py', 'arg1', 'arg2', 'arg3']

Program-1(b)

Object: Write a Python Program to find the square root of a number by Newton's Method

Newton's Method

1. Define a function named newtonSqrt().
2. Initialize approx. as 0.5*n and better as 0.5*(approx. + n/approx.)
3. Use a while loop with a condition better!=approx. to perform the following,
    i. Set approx.=better
    ii. Better=0.5*(approx. + n/approx.)
4. Print the value of approx.

Input: 9

Output: 3

Program 2(a)

Object: Write a Python program to find the exponentiation of a number.

Algorithm:

1. Define a function named power()
2. Read the values of base and exp
3. Use 'if' to check if exp is equal to 1 or not
    i. if exp is equal to 1, then return base
    ii. if exp is not equal to 1,
4. then return (base*power(base,exp-1))
5. Print the result.

Input:

Enter base: 7

Enter exponential value: 2

Output:

49

Program-2(b)

Object: To write a python program to compute the GCD of two numbers.

Algorithm:

1. Define a function named compute GCD()
2. Find the smallest among the two inputs x and y
3. Perform the following step till smaller+1

Check if ((x % i == 0) and (y % i == 0)), then assign GCD=i

4. Print the value of gcd

Input:

M=12, N=17

Output:

1

Program-2(C)

Object: To write a python program first n prime numbers.

Algorithm:

1. Read the value of n
2. for num in range(0,n + 1), perform the following
3. if num%i is 0 then breakelse print the value of num
4. Repeat step 3 for i in range(2,num)

Input:

Enter the upper limit: 20

Output:

Prime numbers are2 3 5 7 11 13 17 19

Program-3(a)

Object: To write a python program find the maximum of a list of numbers.

Algorithm:

1. Create an empty list named l
2. Read the value of n
3. Read the elements of the list until n
4. Assign l[0] as maxno
5. If l[i]>maxno then set maxno=l[i]
6. Increment i by 1
7. Repeat steps 5-6 until i<n
8. Print the value of maximum number

Input:

list1 = [10, 20, 4, 45, 99]

Output-

Max=99

Program 3(b)

Object: To write a python program to perform Matrix Multiplication.

Algorithm:

1. Define two matrices X and Y
2. Create a resultant matrix named 'result'
3. for i in range(len(X)):
   for j in range(len(Y[0])):
   for k in range(len(Y))
   result[i][j] += X[i][k] * Y[k][j]
4. for r in result, print the value of r

Input:

```
# 3x3 matrix
X = [[12,7,3], [4 ,5,6], [7 ,8,9]]
# 3x4 matrix
Y = [[5,8,1,2], [6,7,3,0], [4,5,9,1]]
```

Output:

```
[114, 160, 60, 27]

[74, 97, 73, 14]

[119, 157, 112, 23]
```

Program-4

Object: To write a python program to find the most frequent words in a text file.

Algorithm:

1. Open the file in read mode and assign to fname variable.
2. Initialize count, maxcount to 0.
3. Spilt the file into no. of lines.
4. Spilt the lines into words.
5. Loop through each words and count the occurrences.

Input:

Let us consider you have a text file with contents like thisHi, friends this program is found in text. This program works perfectly

Output:

Most frequent words–"program"

Program 5(a)

Object: Write a Python Program to perform Linear Search

Algorithm:

1. Read n elements into the list
2. Read the element to be searched
3. If alist[pos]==item, then print the position of the item
4. else increment the position and repeat step 3 until pos reaches the length of the list

Input:

list of items is: [5, 7, 10, 12, 15])Enter item to search: 7

Output:

item found at position: 1

Program-5(b)

Object: To write a python program Binary search.

Algorithm:

1. Read n elements into the list
2. Read the element x to be searched
3. Compare x with the middle element.
4. If x matches with middle element, we return the mid index.
5. Else If x is greater than the mid element, then x can only lie in right half subarray
6. a. After the mid element. So we recur for right half.
7. Else (x is smaller) recur for the left half.

Input:

arr =[2, 3, 4, 10, 40]

x =10

Output:

Element is present at index 3

Program-6(a)

Object: To write a python program selection sort.

Algorithm:

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning. The algorithm maintains two subarrays in a given array.

1) The subarray which is already sorted.
2) Remaining subarray which is unsorted.

In every iteration of selection sort, the minimum element (considering ascending order) from the unsorted subarray is picked and moved to the sorted subarray.

Steps:
Step 1 − Set MIN to location 0
Step 2 − Search the minimum element in the listStep 3 − Swap with value at location MIN
Step 4 − Increment MIN to point to next elementStep 5 − Repeat until list is sorted

Input:

Array=[22, 12, 11, 64, 25]

Output:

Sorted Array: [11, 12, 22, 25, 64]

Program-6(b)

Object: To write a python program Insertion sort.

Algorithm:

Insertion sort is a simple sorting algorithm that works the way we sort playing cards in our hands. Insertion sort is a simple sorting algorithm that builds the final sorted array (or list) one item at a time.It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort. However, insertion sort provides several advantages: Efficient for (quite) small data sets,much like other quadratic sorting algorithms. More efficient in practice than most other simple quadratic (i.e., O(n2)) algorithms such as selection sort or bubble sort

Steps:

Step 1 − If it is the first element, it is already sorted. return 1;
Step 2 − Pick next element
Step 3 − Compare with all elements in the sorted sub-list
Step 4 − Shift all the elements in the sorted sub-list that is greater than the value to be sorted
Step 5 − Insert the value
Step 6 − Repeat until list is sorted

Input:

Array: 11, 6, 13, 12, 5

Output:

Sorted array is: 5, 6, 11, 12, 13

Program-7

Object: To write a python program Merge sort.

Algorithm:

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

MergeSort(arr[], l, r) If r > l

1. Find the middle point to divide the array into two halves:middle m = (l+r)/2
2. Call mergeSort for first half:
   Call mergeSort(arr, l, m)
3. Call mergeSort for second half:
   Call mergeSort(arr, m+1, r)
4. Merge the two halves sorted in step 2 and 3:
   Call merge(arr, l, m, r)

Input:

Given array is1211 13 5 6 7

Output:

Sorted array is5 6 7 11 12 13

Program 8

Object: To write a python program to simulate bouncing ball in Pygame.

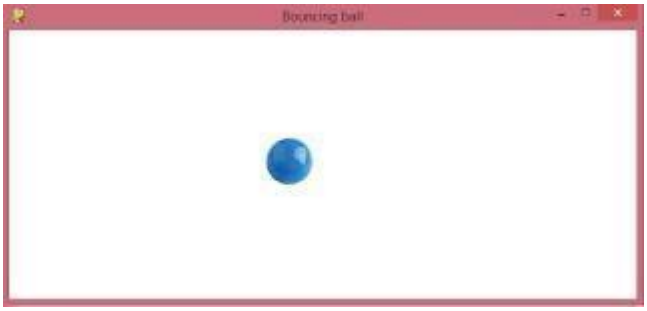
Algorithm:

STEP 1: Define the class Ball and initialize the screen background, image and the circle for theball.

STEP 2: Define the functions for update and for checking the boundary for the ball to

hitSTEP 3: Define the main function for the actual bouncing ball simulation

Output:



Program-9
Value Addition

Objective: To demonstrate working of classes and objects

Procedure:

Creating Classes

The class statement creates a new class definition. The name of the class immediately followsthe keyword class followed by a colon as follows:

class ClassName:
'Optional class documentation string'class_suite

- The class has a documentation string, which can be accessed via ClassName.__doc___
- The class_suite consists of all the component statements defining class members, dataattributes and functions.

Creating Instance Objects

To create instances of a class, you call the class using class name and pass in whatever arguments its __init_() method accepts.

"This would create first object of Employee class" emp1 = Employee("Zara", 2000)

"This would create second object of Employee class" emp2 = Employee("Manni", 5000)

Accessing Attributes

You access the object's attributes using the dot operator with object. Class variable would beaccessed using class name as follows:

emp1.displayEm

ployee()

emp2.displayEm

ployee()

print "Total Employee %d" % Employee.empCount

Instead of using the normal statements to access attributes, you can use the following functions

- The getattr(obj, name[, default]) − to access the attribute of object.
- The hasattr(obj,name) − to check if an attribute exists or not.

- The setattr(obj,name,value) − to set an attribute. If attribute does not exist, then it would becreated.
- The delattr(obj, name) − to delete an attribute.

```
hasattr(emp1, 'age')    # Returns true if 'age' attribute
existsgetattr(emp1, 'age')    # Returns value of 'age'
attribute setattr(emp1, 'age', 8)    # Set attribute
'age' at 8 delattr(empl, 'age')   # Delete attribute 'age'
```

Built-In Class Attributes

Every Python class keeps following built-in attributes and they can be accessed using dot operatorlike any other attribute −

- dict − Dictionary containing the class's namespace.
- doc − Class documentation string or none, if undefined.
- name − Class name.
- module − Module name in which the class is defined. This attribute is " main " ininteractive mode.
- bases − A possibly empty tuple containing the base classes, in the order of their occurrence in the base class list.

Destroying Objects (Garbage Collection)

Python deletes unneeded objects (built-in types or class instances) automatically to free the memory space. The process by which Python periodically reclaims blocks of memory that no longer are in use is termed Garbage Collection.

Python's garbage collector runs during program execution and is triggered when an object's reference count reaches zero. An object's reference count changes as the number of aliases that point to it changes.

An object's reference count increases when it is assigned a new name or placed in a container (list,tuple, or dictionary). The object's reference count decreases when it's deleted with del, its referenceis reassigned, or its reference goes out of scope. When an object's reference count reaches zero, Python collects it automatically.

```
a = 40   # Create object <40>
b = a     # Increase ref. count of <40> c = [b]          # Increase ref. count of
<40> del a                                              # Decrease ref. count of
<40> b = 100                                            # Decrease ref. count of
<40>c[0] = -1                                           # Decrease ref. count of
<40>
```

You normally will not notice when the garbage collector destroys an orphaned instance and reclaims its space. But a class can implement the special method del (), called a destructor, thatis invoked when the instance is about to be destroyed. This method might be used to clean up anynon-memory resources used by an instance.

Objective: To demonstrate class method & static method

Procedure:

Class Method

The @classmethod decorator, is a builtin function decorator that is an expression that gets evaluated after your function is defined. The result of that evaluation shadows your function definition.
A class method receives the class as implicit first argument, just like an instance method receivesthe instance
Syntax:

```
class C(object): @classmethod

    def fun(cls, arg1, arg2, ...):
            ....
            …
```

fun: function that needs to be converted into a class

methodreturns: a class method for function.

- A class method is a method which is bound to the class and not the object of the class.
- They have the access to the state of the class as it takes a class parameter that points to theclass and not the object instance.
- It can modify a class state that would apply across all the instances of the class. For exampleit can modify a class variable that will be applicable to all the instances.

Static Method

A static method does not receive an implicit first argument.
Syntax:

```
class C(object): @staticmethod

    def fun(arg1, arg2, ...):
            ...
            ...
```

returns: a static method for function fun.

- A static method is also a method which is bound to the class and not the object of theclass.
- A static method can’t access or modify class state.
- It is present in a class because it makes sense for the method to be present in class.

Objective: To demonstrate constructors

Procedure:

Constructors are generally used for instantiating an object. The task of constructors is to initialize (assign values) to the data members of the class when an object of class is created. In Python the __init_() method is called the constructor and is always called when an object is created.

Syntax:

```
def___init___(self):
    # Body of the constructor
```

Types of constructors:

- Default constructor: The default constructor is simple constructor which doesn't accept anyarguments. Its definition has only one argument which is a reference to the instance beingconstructed.
- Parameterized constructor: constructor with parameters is known as parameterized constructor. The parameterized constructor take 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.

Program-10

Value Addition

Objective: To demonstrate inheritance

Class Inheritance:

Instead of starting from scratch, you can create a class by deriving it from a pre- existing class bylisting the parent class in parentheses after the new class name.

The child class inherits the attributes of its parent class, and you can use those attributes as if theywere defined in the child class. A child class can also override data members and methods from the parent.
Derived classes are declared much like their parent class; however, a list of base classes to inheritfrom is given after the class name –

Syntax:

class SubClassName (ParentClass1[, ParentClass2, ...]):

'Optional class documentation

string'class_suite

Objective: Concept of polymorphism in python (method overloading and overriding)

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 astructure that can take or use many forms of objects.

Overriding Methods

You can always override your parent class methods. One reason for overriding parent's methods is because you may want special or different functionality in your subclass.

Example:

```
class Bear(object):def sound (self):
        print "ITM"
class Dog(object):def sound(self):
        print "Gwalior"
def makeSound(animalType):animalType.sound()

bearObj = Bear()dogObj = Dog()

makeSound(bearObj)
makeSound(dogObj)
```

Output:

```
ITM
Gwalior
```