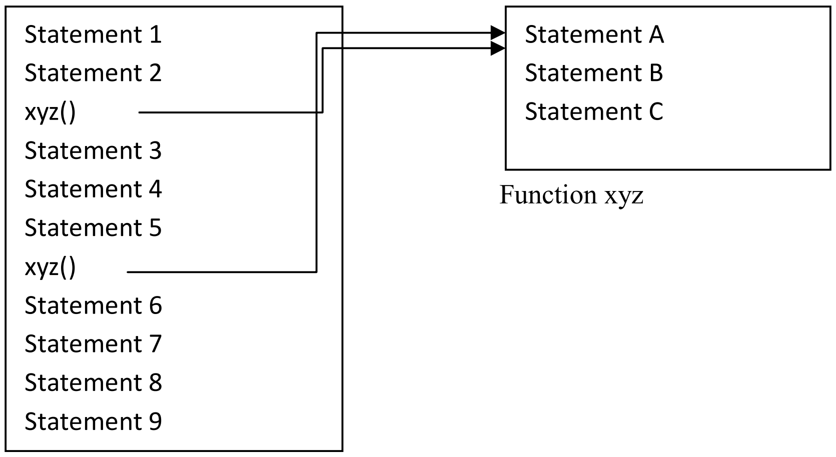
Unit-IV

Structures: Numbers, Strings, Lists, Tuples, Dictionary, Date & Time, Modules, Defining Functions, Exit function, default arguments

Introduction to Advanced Python: Objects and Classes, Inheritance, Regular Expressions, Event Driven Programming, GUI Programming

Topic 1 : What is a function ?

Function is a self contained block of programming code which perform a specific task. Each function is identified by a specific name which is known as function name. When you want the function to perform its task you need to call the function. On calling the function the control is transferred to the function body, after the function execution the control is transferred to the point in the program that called the function.



Program ABC.py

Topic 2 : Advantages of Functions

Use of function in a program provide following advantages :

1. Simpler structure of a program : Functions divides the program into smaller pieces. Thus, programs containing functions tends to be simper, easier to read/write and understand.

2. Provide code reusability : Functions can be reused any number of times in a program or any number of programs.

3. Easier to isolate and fix errors : Functions divide the program into smaller independent sub programs, thus the debugging and error finding in programs becomes easier.

4. Quicker development of large programs : By using the function we can divide the software development activity into small parts and can assign the task to different programmers. Thus, the overall development time of the software gets reduced.

5. Reduces the overall development cost : The functions can be reused in any number of programs. This substantially reduces the cost of writing, testing and debugging the program. Thus, the overall cost of software development reduces.

Topic 3

Types of functions

Functions can be classified into two types :

1. Built in functions or Library functions 2. User defined functions

Built in functions are predefined function which are available to user for programmers , these functions are also known as library functions. There are many library functions in Python. Some of the most commonly used functions are :

1. input () : This function give a message to the user, read input from the user in the form of string. Message given to the user is optional.

Example :

```
>>>name=input("Enter name : ")

Enter name : Computer science

>>>print(name)

Computer science
```

2. int() : This function convert the number or string into an integer object.

Example :

```
Num1=int("4")
```

3. sum() : This function add the numbers given to it as arguments and return their sum.

Example :

```
print(sum(2,5,6,8))

21
```

User Defined Functions

The functions which are defined specifically by the user to meet his requirements are called user defined functions.

Every function in python consists of the following two components,

a) Function Definition

b) Function Call

Function definition : The function definition contains the code of the function, it consists of function header and function body.

```
Syntax :   def  function_name(parameter 1, parameter 2,…………….parameter n) :

                 Statement 1
                 Statement 2
                 …………………..
                 return value
```

The first line of the function definition is called function header, the header start with the def followed by the function name and the set or parameters or arguments that the function will receive. The arguments given in the function definition are called formal arguments or formal parameters. The body of the function contain the statements that define the logic of the function.

Some function also return a value, these function are called value returning functions. A function return value by using return keyword.

Example :

```
        1. message() :
                print("Welcome to Python")
                print("B.Sc. 1st Year Students")

        2. def SI(P,R,T) :
                SI=P*R*T/100
                return SI
```

Function Call

When we want a specific function to execute we need to call it, this is accomplished using a function call. Function call is made by writing the name of the function and passing the desired arguments/ parameters. The parameters given to the function during function call are called actual arguments.

Syntax : function_name(parameter 1, parameter 2,…….parameter n)

Example : result= **SI(100,4,2)**

```
          Print("Simple Interest = ", result)
          message()
```

Output : Simple Interest=8.0

```
         Welcome to Python
         B.Sc. 1st Year
```

Function can also return more than value : this can be understool with the help of following example.

```
def area_peri_rectangle(l,b):
a=l*b
p=2*(l+b)
return a,p

length=int(input("Enter Length = "))
breadth=int(input("Enter Breadth = "))
area,perimeter = area_peri_rectangle(length,breadth)
```

Types of Arguments

There are 4 types of arguments in Python

1. Positional argument

2. Keyword argument

3. Default argument

4. Variable length arguments

Positional Argument is an argument that identifies the parameter based on its position in the argument list passed into the function. Positional arguments in a function called are matched with the parameter in the function from left to right.

Example :

```
def student info (name, age, gender)
        print("student name ",name)
        print("student age", age)
        print("student gender", gender)

student_info("Suresh",32,"Male")
```

Keyword argument is an argument where the argument is passed to a function by referring arguments names rather than their position

Example :
student_info (name=”Suresh”, age=32, gender=”Male”)
student_info (name=”Fatima”, gender=”Female”, age=28)

Advantages of Keyword arguments
1. Arguments can be passed in any order
2. You can easily omit the optional parameters from arbitrary position.
   Student_info(name=”Suresh”, ,gender=”Male”)
3. It makes the function call clearer and easy to understand

Default arguments provide default values. These values are used as formal arguments when corresponding values are not provided in the actual arguments.

This can be understood with the help of following example

```
Def displayArea(length=2, breadth=3):
        Area=length*breadth
        print(“ Area of rectangle = ”,Area)

displayArea()
output:  Area of rectangle=6

displayArea(10,20)
output: Area of rectangle=200

displayArea(length=8,breadth=5)
output:  Area of rectangle=40

displayArea(length=10)
output:  Area of rectangle=30
```

Some important point regarding default arguments
1.Every default parameter in function definition must specify a default value.
2. A function can have both required and default parameter, but all required parameters must be placed before any default parameters. For example
 def function_display (a,b=8,c,d) is a syntax error.
3.You can omit the parameters starting for the end parameter only. For example, function_display(23,45, ,7) is a syntax error

Variable length Arguments
This type of argument is used in situations where a function can receive one or more arguments. This is done by adding * before the variable which will contain variable no of values. The variable which store the variable length of values is stored as tuple. This can be easily understood with the help of following example.

```
def show(*num) :
for i in num :
        print (i)
```

List: It is an ordered collection of elements. By ordered we mean that the elements are stored one after the other**. Lists are mutable this means that the content of the list can be modified by adding, removing or changing objects during the execution of the program.**

Creating a list

Method 1 : Using square brackets

Name_of_list=[item1, item2, item3,…….. item1]
Examples :
1. Booknames=["programming using python", "mathematics101","Thermodynamics"]

2. Booksprices=[230,345,567]

3. Height_of_stu=[75, 85.6, 45.9, 69.4]

Method 2 : using list constructor

A list can be created by using a list construct function. This function take the list its as arguments and create a list.
Syntax :
list(list items as arguments)

examples :
Booknames= list (["programming using python", "mathematics101","Thermodynamics"])

stu_list= list() it is an empty list

Accessing a list :
List can be accessed with the help of
1. Positive indexing : positive indexes starts from 0 and goes upto (no.of list elements-1). First element of list have 0 index, second element have 1 index and so on
Example : stu_height=[74,80,56,69,80]
>>>stu_height
this statement will show all the elements of the list
>>>stu_height[2]
This element will show the 3rd element of the list i.e. 56
2. Negative indexing : Negative indexes starts from -1 and goes upto –no.of list elements. The index of last element is -1 and the index of first element is –no. of elements in the list.
Example : stu_height=[74,80,56,69,80]
>>>stu_height[-2]
This element will show the 2nd element of the list starting from the last element i.e. 69
3. List slicing : by using this method we can access the subset of list elements. This is done by supplying the start and end index+1 of the list elements to be accessed.
Example : stu_height=[74,80,56,69,80]
>>>stu_height[1:3]
This element will show the 2nd and 3rd element of the list i.e. 80 and 56
4. with the help of for loop
The list elements can be accessed with the help of for loop
Example : stu_height=[74,80,56,69,80]
```
for i in stu_height
        print(i)
```

This element will show all the elements of the list

List Methods
Python has provided various inbuilt methods to apply various operation on lists. These functions are called list methods. Some of the most list methods are as follow :
append () : This method is used to add an element to the end of existing list.
For Example :
colors=[“Red”, “Blue”,”Green”]
colors.append(“Black”) here Black will be added at the end of color list

count(): This method count the number of times an element occur in a list.
For Example :
colors=[“Red”, “Blue”,”Green”,”Red”]
colors.count(“Red”) here the result will be 2 because “Red” occur 2 times in the list

clear(): This method clears all the elements from the list.
colors.clear() This statement will delete all the elements of list

insert(): This method insert an element in the existing list at a specific index.
For Example :
colors=[“Red”, “Blue”, ”Green”, ”Red”]
colors.insert(1,“Pink”) This statement will insert Pink at index number 1. The list elements will now become “Red”, “Pink”, “Blue”, ”Green”, ”Red”

pop() : This method remove the element from a specific index, if index will not be provided then the last element of the list will be removed.
colors=[“Red”, “Blue”, ”Green”, ”Red”]
colors.pop() This statement will remove the last element of the list i.e “Red”
colors.pop(2) This statement will remove the element stored at index value 2 i.e. Green.

remove() : This method remove the first occurrence of the specified element from the existing list.
colors.remove(“Red”) This method remove the first Red occurrence of Red.

reverse() : This method will reverse the list
colors=[“Red”, “Blue”, ”Green”, ”Red”]
colors.pop() This statement will remove the last element of the list i.e “Red”
colors.reverse() This statement will reverse the elements of the colors list

Few Examples of List
1. Create a list which store the name of fruits and show the elements of the list
Solution 1
```
fruits=[“Apple”,”Orange”,”Mango”,”Grapes”,”Banana”]
for i in fruits :
        print(i)
```
Solution 2
```
fruits=[“Apple”,”Orange”,”Mango”,”Grapes”,”Banana”]
for i in range(len(fruits)) :
        print(fruits[i])
```

2. Create a list that store the name of N students of your class and also display their names after the creation of list.
```
n= int(input(“Enter the number of students whose names you want to store in the list”))
```

```
stu_name=[ ]
for i in range(n):
        name=input("Enter student name ")
        stu_name.insert(i,name)

print("The names entered by you are : ")
for i in stu_name :
        print(i, end=" ")
```

Tuples : Just like list tuple is also an ordered collection of elements. However unlike lists tuples are immutable. Which means that once a tuple is created we cannot add, delete, modify the elements of the tuple.

Creating tuple

Method 1: using ()
Syntax : tuple_name(item1, item2,……..item N)
Example : Stu_Gender("Male","Female","Other")
print(Stu_Gender)

Method 2: by using tuple constructor
This method creates a tuple with the help of tuple constructor function.
Syntax : tuple_name=tuple(item1, item2,……..item N)
Example : Stu_Gender=tuple(("Male","Female","Other"))

Accessing Tuples : The tuple elements can be accessed with the help of following
a) Using Positive Indexing
Example : Stu_Gender("Male","Female","Other")
Stu_Gender[1]   This will access the tuple element with index value 1 i.e, "Female"
b) Using Negative Indexing
Example : Stu_Gender("Male","Female","Other")
Stu_Gender[-1]   This will access the tuple element with index value -1 i.e, "Other"
c) Using list Indexing
Example : Stu_Gender("Male","Female","Other")
Stu_Gender[0,2]   This will access the tuple element with index value 0,1 i.e, "Male", "Female"

Tuple support all the methods supported by list except those which change the tuple.

Comparison of Tuple and List
1. Tuple uses less space than list
2 Tuple is Immutable where as List is mutable
3. Function arguments are passed as tuples and as list (as discussed in variable number of positional arguments)
4. Function return multiple values in the form of a tuple.
5. It takes less computer resources and time to create a tuple.

Procedure/Function Oriented Programming

In this approach, the problem is viewed as the sequence of things to be done. The primary focus here is on functions. The programming languages that follow this approach are

COBOL, FORTRAN, C etc. A typical structure for procedural programming is shown in the fig given below.

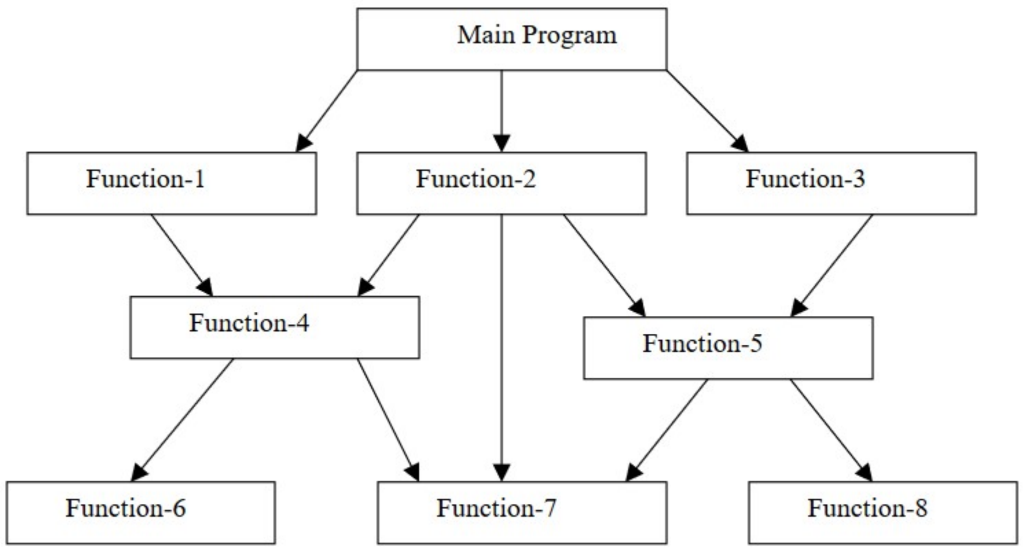
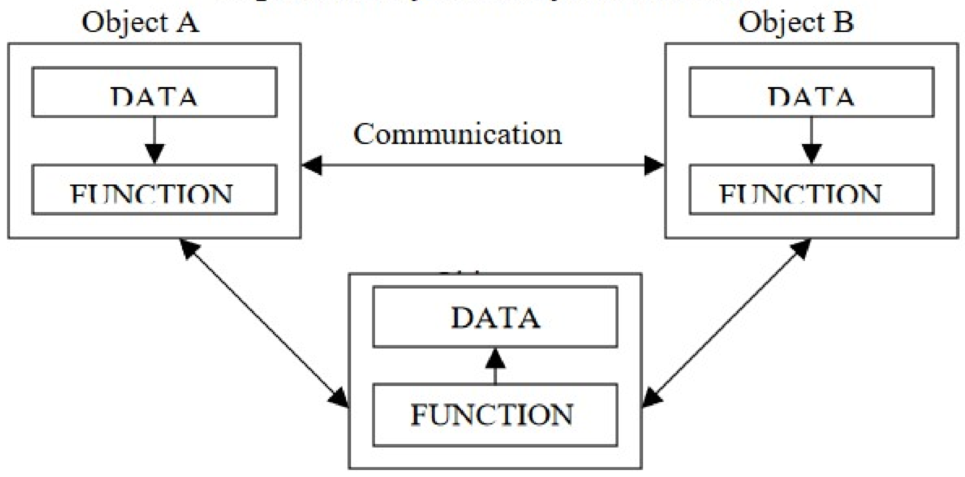


Fig. 1.2 Typical structure of procedural oriented programs

Procedure oriented programming basically consists of writing a list of instructions for the computer to follow, and organizing these instructions into groups known as functions.
In a large program it is very difficult to identify what data is used by which function. In case we need to revise an external data structure, we also need to revise all functions that access the data. This provides an opportunity for bugs to creep in. Another serious drawback with the procedural approach is that we do not model real world problems very well. This is because functions are action-oriented and do not really corresponding to the element of the problem.
Some Characteristics exhibited by procedure-oriented programming are:
• Emphasis is on doing things (algorithms).
• Large programs are divided into smaller programs known as functions.
• Most of the functions share global data.
• Data move openly around the system from function to function.
• Functions transform data from one form to another.
• Employs top-down approach in program design.

Object Oriented Paradigm The major motivating factor in the invention of object-oriented approach is to remove some of the flaws encountered in the procedural approach. OOP treats data as a critical element in the program development and does not allow it to flow freely around the system. It ties data more closely to the function that operate on it, and protects it from accidental modification from outside function. OOP allows decomposition of a problem into a number of entities called objects and then builds data and function around these objects. The organization of data and function in object-oriented programs is shown in fig. given below The data of an object can be accessed only by the function associated with that object. However, function of one object can access the function of other objects.

Organization of data and function in OOP



Some of the features of object oriented programming are:

• Emphasis is on data as well as procedures.
• Programs are divided into what are known as objects.
• Data is hidden and cannot be accessed by external function.
• New data and functions can be easily added whenever necessary.
• Follows bottom up approach in program design.

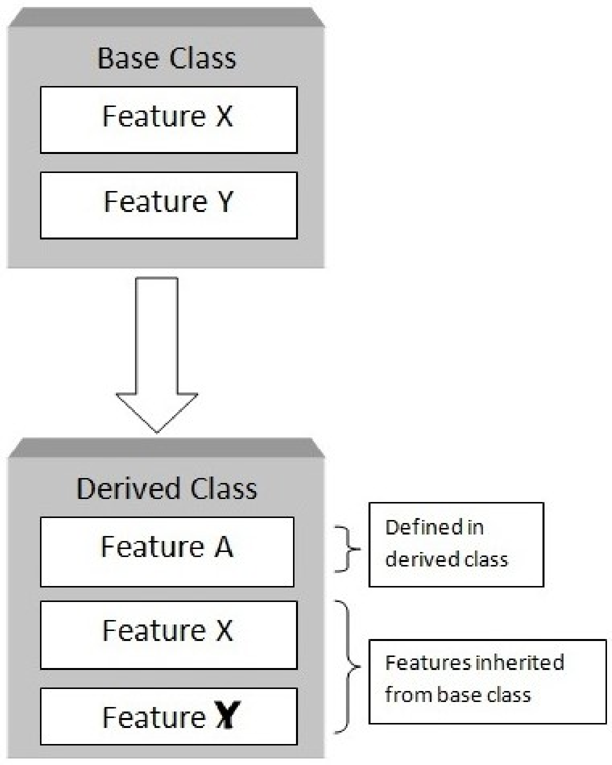
Object-oriented programming is the most recent concept among programming paradigms.

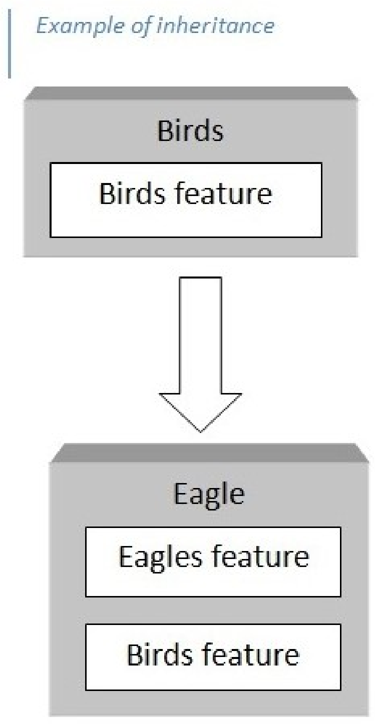
Some of the concepts used extensively in object-oriented programming are as follow :

• **Classes** : A class is a user define data type that define the structure of an object. A class mainly has two sections data and functions. Class do not occupy memory space because it is just the design for the particular type of objects. Once a class has been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with which they are created. A class is thus a collection of objects similar types. For examples, Mango, Apple and orange members of class fruit.

• **Objects** : Objects are the variables of a particular class. For example if we have a class named Student than Student with name XYZ will be an object of that class. Program objects should be chosen such that they match closely with the real-world objects. Objects take up space in the memory and have an associated address like a record in Pascal, or a structure in c. When a program is executed, the objects interact by sending messages to one another.

• **Inheritance :** Inheritance is the process by which objects of one class acquired the properties of objects of another classes. It supports the concept of hierarchical classification. For example, the bird, 'eagle' has specific features of its own but it also posses the general features of birds which all other types of birds possess. The principal behind this sort of division is that each derived class shares common characteristics with the class from which it is derived. In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it.





Other features of OOPs are :

- Data Abstraction and Data Hiding
- Polymorphism
- Message Passing

GUI Programming/What Are IDEs ? Discuss the IDE's available for Python Programming.

An IDE (or Integrated Development Environment) is a software that is dedicated to software development. IDEs integrate several tools that are used for the software development. The main tools included in a GUI are as follow:

- **Code editor:** It is used to write code of the software in programming languages such as Python, Java, C++ etc.
- **Build Tool:** This tool is used to convert the programming code into machine language. This is done with the help of compiler/interpreter.
- **Debugging Tool:** This tool help to find the errors/bugs in the program. Debugging tools provides various debugging techniques such as check point, unit testing etc.
- **Execute Tool:** This tool execute the program and show the output.

Most of the IDE's also provide additional features and tools such as :

- **Syntax highlighting**
  Being able to quickly spot keywords, variables, and symbols in your code makes reading and understanding code much easier.
- **Automatic code formatting**
  Any editor or IDE worth it's salt will recognize the colon at the end of a while or for statement, and know the next line should be indented.

Most IDEs support many different programming languages and contain many more features. They can, therefore, be large and take time to download and install.

General Editors and IDEs with Python Support

1. PyCharm

Category: IDE
Website: https://www.jetbrains.com/pycharm/

2. Atom

Category: Code Editor
Website: https://atom.io/

3. Eclipse + PyDev

Category: IDE
Website: www.eclipse.org
Python tools: PyDev, www.pydev.org

4. GNU Emacs

Category: Code Editor
Website: https://www.gnu.org/software/emacs/

5. Vi / Vim

Category: Code Editor
Website: https://www.vim.org/

6. Visual Studio

Category: IDE
Website: https://www.visualstudio.com/vs/
Python tools: Python Tools for Visual Studio, aka PTVS

Event Driven Programming

Event-driven programming is a programming paradigm in which the flow of program execution is determined by *events* - for example a user action such as a mouse click, key press, or a message from the operating system or another program. An event-driven application is designed to detect events as they occur, and then deal with them using an appropriate *event-handling procedure*. Almost all modern programming languages such as Visual Basic, Visual C++, Python and Java support event-driven programming.

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

- **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- **wxPython** − This is an open-source Python interface for wxWindows http://wxpython.org.

- **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine http://www.jython.org.

There are many other interfaces available, which you can find them on the net.

Tkinter Programming

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

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

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