



### Experiment No 2:

**Aim:** To implement functions and object oriented concepts in python.

### Theory:

Python allows us to divide a large program into the basic building blocks known as a function. The function contains the set of programming statements enclosed in a block. A function can be called multiple times to provide reusability and modularity to the Python program.

### Creating a Function

Python provides the **def** keyword to define the function. The syntax of the define function is given below.

### Syntax:

1. **def** my\_function(parameters):
2. function\_block
3. **return** expression

### Function Calling

In Python, after the function is created, we can call it from another function. A function must be defined before the function call; otherwise, the Python interpreter gives an error. To call the function, use the function name followed by the parentheses.

### Recursive function

A function that calls itself is a recursive function. This method is used when a certain problem is defined in terms of itself. Although this involves iteration, using an iterative approach to solve such a problem can be tedious.

### Classes and Objects

Classes are used to create user-defined data structures. Classes define functions called **methods**, which identify the behaviors and actions that an object created from the class can perform with its data.

While the class is the blueprint, an **instance** is an object that is built from a class and contains real data

### Class Definition Syntax:



```
class ClassName:
    # Statement-1
    .
    .
    .
    # Statement-N
```

### Instantiate an Object in Python

```
Instance_name=ClassName()
```

### Instance attributes and class attributes

Attributes created in `__init__()` are called **instance attributes**. An instance attribute's value is specific to a particular instance of the class.

On the other hand, **class attributes** are attributes that have the same value for all class instances. You can define a class attribute by assigning a value to a variable name outside of `__init__()`.

### The self

Class methods must have an extra first parameter in the method definition. We do not give a value for this parameter when we call the method, Python provides it. If we have a method that takes no arguments, then we still have to have one argument.

### \_\_init\_\_ method

The `__init__` method is similar to constructors in C++ and Java. Constructors are used to initializing the object's state.

### Inheritance in Python:

Inheritance is the capability of a class to inherit all properties and methods of base class from which it is derived and can add new features to the class without modifying it. The syntax to define a derived class when one or more base classes are to be inherited is as follows:

```
class derivedClassName(baseClassName,...):
```

```
<statement  
1>.
```

```
<statement  
n>
```



# Vidyavardhini's College of Engineering and Technology

## Department of Artificial Intelligence & Data Science

---

### Different forms of Inheritance:

#### 1. Single inheritance:

When a child class inherits from only one parent class, it is called single inheritance. We saw an example above.

#### 2. Multiple inheritance:

When a child class inherits from multiple parent classes, it is called multiple inheritance. Unlike java, python shows multiple inheritance.

#### 3. Multilevel inheritance:

When we have a child and grandchild relationship.

#### 4. Hierarchical inheritance

More than one derived classes are created from a single base.

#### 5. Hybrid inheritance:

This form combines more than one form of inheritance. Basically, it is a blend of more than one type of inheritance.

### PROGRAM:

class Student:

# Constructor

def \_\_init\_\_(self, name, rollno, m1, m2):

self.name = name

self.rollno = rollno

self.m1 = m1

self.m2 = m2

# Function to create and append new student

def accept(self, Name, Rollno, marks1, marks2):

# use 'int(input())' method to take input from user

ob = Student(Name, Rollno, marks1, marks2)

ls.append(ob)

# Function to display student details

def display(self, ob):

print("Name : ", ob.name)

print("RollNo : ", ob.rollno)

```
print("Marks1 : ", ob.m1)
print("Marks2 : ", ob.m2)
print("\n")
```

```
# Search Function
def search(self, rn):
    for i in range(len(ls)):
        if ls[i].rollno == rn:
            return i
    return -1
```

```
# Delete Function
def delete(self, rn):
    i = self.search(rn)
    if i != -1:
        del ls[i]
        print("Deleted successfully!")
    else:
        print("Student not found!")
```

```
# Update Function
def update(self, rn, No):
    i = self.search(rn)
    if i != -1:
        roll = No
        ls[i].rollno = roll
        print("Updated successfully!")
    else:
        print("Student not found!")
```

```
# Create a list to add Students
ls = []
```

```
# an object of Student class
obj = Student('', 0, 0, 0)
```

```
print("\nOperations used, ")
print("\n1.Accept Student details\n2.Display Student Details\n" "3.Search Details of a
Student\n4.Delete Details of Student" "\n5.Update Student Details\n6.Exit")
```

```
while True:
    ch = int(input("Enter choice: "))
    if ch == 1:
        obj.accept("A", 1, 100, 100)
        obj.accept("B", 2, 90, 90)
        obj.accept("C", 3, 80, 80)
    elif ch == 2:
        print("\nList of Students\n")
        for i in range(len(ls)):
            obj.display(ls[i])
    elif ch == 3:
        print("\n Student Found\n")
        s = obj.search(2)
        if s!= -1:
            obj.display(ls[s])
        else:
```

```
            print("Student not found!")
    elif ch == 4:
        obj.delete(2)
        print("List after deletion")
        for i in range(len(ls)):
            obj.display(ls[i])
    elif ch == 5:
        obj.update(3, 2)
        print("List after updation")
        for i in range(len(ls)):
            obj.display(ls[i])
    elif ch == 6:
        print("Thank You!")
        break
    else:
        print("Invalid Choice")
```

**OUTPUT**

```
PS F:\AIDS_BARI_ANKIT\PP\PRACTICALS> python -u "f:\AIDS_BARI_ANKIT\PP\PRACTICALS\pp_prac_code.p
Operations used,

1.Accept Student details
2.Display Student Details
3.Search Details of a Student
4.Delete Details of Student
5.Update Student Details
6.Exit
Enter choice: 
```

**Conclusion:**

The implementation of functions and object-oriented concepts in Python proved to be essential in enhancing code modularity, reusability, and maintainability. By effectively utilizing these programming paradigms, we were able to achieve greater flexibility and scalability in our software design, laying a solid foundation for future development endeavors.