





| Experiment No.4 |
| --- |
| Creating functions, classes and objects using Python |
| Date of Performance: 14/02/2024 |
| Date of Submission: 21/02/2024 |

















**Code:**

class Student:

def \_\_init\_\_(self,name="",marks=0):

self.name = name

self.marks = marks

def GradeCalculator(self):

mark = self.marks

if mark>60:

return "Grade A"

elif mark>50:

return "Grade B"

elif mark>40:

return "Grade C"

else:

return "Fail"

s1=Student(input("Enter your name:"),int(input("Enter marks:")))

print("Student name: ",s1.name)

print("Student marks: ",s1.marks)

print("Student Grade:",s1.GradeCalculator())

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

s2=Student(input("Enter your name:"),int(input("Enter marks:")))

print("Student name: ",s2.name)

print("Student marks: ",s2.marks)

print("Student Grade:",s2.GradeCalculator())

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

s3=Student(input("Enter your name:"),int(input("Enter marks:")))

print("Student name: ",s3.name)

print("Student marks: ",s3.marks)

print("Student Grade:",s3.GradeCalculator())

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

**Output:**

Enter your name:Sarwadeep

Enter marks:91

Student name: Sarwadeep

Student marks: 91

Student Grade: Grade A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter your name:Ab

Enter marks:80

Student name: Ab

Student marks: 80

Student Grade: Grade A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter your name:Vcet123

Enter marks:100

Student name: Vcet123

Student marks: 100

Student Grade: Grade A

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Conclusion:**

In Python, classes serve as blueprints for creating objects, which are instances of those classes. Classes define the behavior and properties of objects through attributes (variables) and methods (functions). Objects encapsulate data and behavior, promoting a more organized and modular code structure. Functions, on the other hand, are blocks of reusable code that perform specific tasks and can be defined both within classes as methods or independently.The combination of classes, objects, and functions enables the implementation of object-oriented programming (OOP) principles such as encapsulation, inheritance, and polymorphism. Encapsulation allows for the bundling of data and methods within objects, providing abstraction and data hiding. Inheritance facilitates code reuse and the creation of hierarchies of classes, while polymorphism allows for flexible and dynamic behavior through method overriding and method overloading.

Together, classes, objects, and functions form the backbone of Python programming, providing a powerful framework for building scalable and flexible applications. They promote modular design, code reusability, and maintainability, making it easier to develop and maintain complex systems.