



|  |
| --- |
| 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:Vaidehi

Enter marks:90

Student name: Vaidehi

Student marks: 90

Student Grade: Grade A

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

Enter your name:Abc

Enter marks:80

Student name: Abc

Student marks: 80

Student Grade: Grade A

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

Enter your name:Vcet

Enter marks:100

Student name: Vcet

Student marks: 100

Student Grade: Grade A

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

**Conclusion:**

In Python, classes are 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, allowing for a more organized and modular code structure. Functions are blocks of reusable code that perform specific tasks, and they can be defined both within classes as methods or independently. Classes enable the implementation of object-oriented programming (OOP) principles such as encapsulation, inheritance, and polymorphism, facilitating the creation of complex systems with modular and maintainable code. Together, classes, objects, and functions form the backbone of Python programming, providing a powerful framework for building scalable and flexible applications.