

```
1 class CSStudent:
2     # Class Variable
3     stream = 'cse'
4
5     # The init method or constructor
6     def __init__(self, roll):
7         # Instance Variable
8         self.roll = roll
9
10    # Objects of CSStudent class
11
12
13 a = CSStudent(101)
14 b = CSStudent(102)
15
16 print(a.stream) # prints "cse"
17 print(b.stream) # prints "cse"
18 print(a.roll)  # prints 101
19
20 # Class variables can be accessed using class
21 # name also
22 print(CSStudent.stream) # prints "cse"
23
24 ##End of Programclass CSStudent:
25     # Class Variable
26     stream = 'cse'
27
28     # The init method or constructor
29     def __init__(self, roll):
30         # Instance Variable
31         self.roll = roll
32
33         # Adds an instance variable
34
35     def setAddress(self, address):
36         self.address = address
37
38         # Retrieves instance variable
39
40     def getAddress(self):
41         return self.address
42
43     # Driver Code
44
45
46 a = CSStudent(101)
47 a.setAddress("Patna, Bihar")
48 print(a.getAddress())
49
50 ##End of Program# Write Python code here
51 class sampleclass:
52     count = 0     # class attribute
53
```

```

54     def increase(self):
55         sampleclass.count += 1
56
57 # Calling increase() on an object
58 s1 = sampleclass()
59 s1.increase()
60 print(s1.count)
61
62 # Calling increase on one more
63 # object
64 s2 = sampleclass()
65 s2.increase()
66 print(s2.count)
67
68 print(sampleclass.count)
69
70 ##End of Program# A Python program to demonstrate inheritance
71
72 # Base or Super class. Note object in bracket.
73 # (Generally, object is made ancestor of all classes)
74 # In Python 3.x "class Person" is
75 # equivalent to "class Person(object)"
76 class Person(object):
77
78     # Constructor
79     def __init__(self, name):
80         self.name = name
81
82     # To get name
83
84     def getName(self):
85         return self.name
86
87     # To check if this person is employee
88
89     def isEmployee(self):
90         print("I am in Person Class")
91         return False
92
93
94 # Inherited or Sub class (Note Person in bracket)
95 class Employee(Person):
96
97     # Here we return true
98     def isEmployee(self):
99         print("I am in Employee Class")
100        return True
101
102
103 # Driver code
104 emp = Person("Geek1") # An Object of Person
105 print(emp.getName(), emp.isEmployee())
106

```

```

107 emp = Employee("Geek2") # An Object of Employee
108 print(emp.getName(), emp.isEmployee())
109
110 ##End of Program#Super Python code to demonstrate how parent constructors
111 # are called.
112
113 # parent class
114 class Person(object):
115
116     # __init__ is known as the constructor
117     def __init__(self, name, idnumber):
118         self.name = name
119         self.idnumber = idnumber
120
121     def display(self):
122         print(self.name)
123         print(self.idnumber)
124
125     # child class
126
127
128 class Employee(Person):
129     def __init__(self, name, idnumber, salary, post):
130         self.salary = salary
131         self.post = post
132
133         # invoking the __init__ of the parent class
134         Person.__init__(self, name, idnumber) #important to declare to get
base class vars
135
136     def display(self):
137         Person.display(self)
138         print(self.salary)
139         print(self.post)
140
141
142
143     # creation of an object variable or an instance
144
145
146 p1 = Person('Rahul', 886012)
147 # calling a function of the class Person using its instance
148 p1.display()
149
150
151 p2 = Employee('Ashish', 105, 5000, "VP")
152 # calling a function of the class Person using its instance
153 p2.display()
154
155
156 ##End of Program# If you forget to invoke the __init__() of the parent class
then its instance variables would not be available to the child class.
157 # The following code produces an error for the same reason.

```

```

158 # Python program to demonstrate error if we
159 # forget to invoke __init__() of parent.
160
161 class A:
162     def __init__(self, n='Rahul'):
163         self.name = n
164
165
166 class B(A):
167     def __init__(self, roll):
168         self.roll = roll
169
170
171 object = B(23)
172 print(object.name) #AttributeError: 'B' object has no attribute 'name'
173
174
175 ##End of Program# Python example to show working of multiple
176 # inheritance
177 class Base1(object):
178     def __init__(self):
179         self.str1 = "Geek1"
180         print("Base1")
181
182
183 class Base2(object):
184     def __init__(self):
185         self.str2 = "Geek2"
186         print("Base2")
187
188
189 class Derived(Base1, Base2):
190     def __init__(self):
191         # Calling constructors of Base1
192         # and Base2 classes
193         Base1.__init__(self)
194         Base2.__init__(self)
195         print("Derived")
196
197     def printStrs(self):
198         print((self.str1, self.str2))
199
200
201 ob = Derived()
202 ob.printStrs()
203
204 ##End of Program#3. Multilevel inheritance: When we have child and grand
    child relationship.
205 # A Python program to demonstrate inheritance
206
207 # Base or Super class. Note object in bracket.
208 # (Generally, object is made ancestor of all classes)
209 # In Python 3.x "class Person" is

```

```

210 # equivalent to "class Person(object)"
211 class Base(object):
212
213     # Constructor
214     def __init__(self, name):
215         self.name = name
216
217     # To get name
218     def getName(self):
219         return self.name
220
221     # Inherited or Sub class (Note Person in bracket)
222
223
224 class Child(Base):
225
226     # Constructor
227     def __init__(self, name, age):
228         Base.__init__(self, name)
229         self.age = age
230
231     # To get name
232     def getAge(self):
233         return self.age
234
235     # Inherited or Sub class (Note Person in bracket)
236
237
238 class GrandChild(Child):
239
240     # Constructor
241     def __init__(self, name, age, address):
242         Child.__init__(self, name, age)
243         self.address = address
244
245     # To get address
246     def getAddress(self):
247         return self.address
248
249     # Driver code
250
251
252 g = GrandChild("Geek1", 23, "Noida")
253 print((g.getName(), g.getAge(), g.getAddress()))
254
255
256 ##End of Program# Python program to demonstrate private members
257 # of the parent class
258 class C(object):
259     def __init__(self):
260         self.c = 21
261
262         # d is private instance variable

```

```
263         self.__d = 42
264 class D(C):
265     def __init__(self):
266         self.e = 84
267         C.__init__(self)
268 object1 = D()
269
270 # produces an error as d is private instance variable
271 print(D.d)
272
273 ##End of Program
274 class Cricket_Personalities:
275     sport_played = "Cricket"
276
277     def __init__(self, fname, lname):
278         print("I am called always on declaration of object")
279         self.firstname = fname
280         self.lastname = lname
281
282
283     def printname(self):
284         print(self.firstname, self.lastname)
285
286
287 # Use the Person class to create an object, and then execute the printname
    method:
288
289 x = Cricket_Personalities("Sachin", "Tendulkar")
290 x.printname()
291 print(x.sport_played)
292
293 y = Cricket_Personalities("Rahul", "Dravid")
294 y.printname()
295
296 ##End of Program
```