

# CODE :

```
1 class Person:
    tabnine: test | explain | document | ask
2     def __init__(self, fname, lname):
3         self.firstname = fname
4         self.lastname = lname
5
6     tabnine: test | explain | document | ask
7     def printname(self):
8         print(self.firstname, self.lastname)
9
10 class Student(Person):
11     tabnine: test | explain | document | ask
12     def __init__(self, fname, lname, year): # Ensure it accepts three arguments
13         super().__init__(fname, lname)
14         self.graduationyear = year
15
16     tabnine: test | explain | document | ask
17     def welcome(self):
18         print("Welcome", self.firstname, self.lastname, "to the class of", self.graduationyear)
19
20 x = Student("Aditya", "Sharma", 2005)
21 # Changed names
22 x.printname()
23 x.welcome()
24
25 class Polygon:
26     tabnine: test | explain | document | ask
27     def __init__(self, no_of_sides):
28         self.n = no_of_sides
29         self.sides = [0 for i in range(no_of_sides)]
30
31     tabnine: test | explain | document | ask
32     def inputSides(self):
33         self.sides = [float(input("Enter side "+str(i+1)+" : ")) for i in range(self.n)]
34
35     tabnine: test | explain | document | ask
36     def dispSides(self):
37         for i in range(self.n):
38             print("Side",i+1,"is",self.sides[i])
39
40 class Triangle(Polygon):
41     tabnine: test | explain | document | ask
42     def __init__(self):
43         super().__init__(3)
44
45     tabnine: test | explain | document | ask
46     def findArea(self):
47         a, b, c = self.sides
48         s = (a + b + c) / 2
49         area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
50         print('The area of the triangle is %0.2f' %area)
51
52 t = Triangle()
53 t.inputSides()
54 t.dispSides()
55 t.findArea()
```

```

49 class SuperClass:
50     tabnine: test | explain | document | ask
51     def super_method(self):
52         print("Super Class method called")
53
54 class DerivedClass1(SuperClass):
55     tabnine: test | explain | document | ask
56     def derived1_method(self):
57         print("Derived class 1 method called")
58
59 class DerivedClass2(DerivedClass1):
60     tabnine: test | explain | document | ask
61     def derived2_method(self):
62         print("Derived class 2 method called")
63
64 d2 = DerivedClass2()
65 d2.super_method()
66 d2.derived1_method()
67 d2.derived2_method()
68
69 class shapes:
70     tabnine: test | explain | document | ask
71     def __init__(self, no_sides):
72         self.n = no_sides
73         self.sides = [0 for i in range(no_sides)]
74
75     tabnine: test | explain | document | ask
76     def takeSides(self):
77         self.sides = [float(input("Enter side "+str(i+1)+" : ")) for i in range(self.n)]
78
79     tabnine: test | explain | document | ask
80     def disSides(self):
81         for i in range(self.n):
82             print("Side",i+1,"is",self.sides[i])
83
84 class rec(shapes):
85     tabnine: test | explain | document | ask
86     def __init__(self):
87         shapes.__init__(self,2) # Changed number of sides
88
89     tabnine: test | explain | document | ask
90     def findArea(self):
91         a, b = self.sides
92         area = a * b # Changed area calculation
93         print('The area of the rectangle is', area)
94
95 t = rec()
96 t.takeSides()
97
98 mytuple = ("apple", "orange", "grapes") # Changed fruits
99 myit = iter(mytuple)
100 print(next(myit))
101 print(next(myit))
102 print(next(myit))
103
104 mystr = "orange" # Changed fruit
105 myit = iter(mystr)
106 print(next(myit))
107 print(next(myit))
108 print(next(myit))
109 print(next(myit))
110 print(next(myit))

```

```

105 mytuple = ("apple", "orange", "grapes") # Changed fruits
106 for x in mytuple:
107     print(x)
108
109 mystr = "orange" # Changed fruit
110 for x in mystr:
111     print(x)
112
113 class MyNumbers:
114     tabnine: test | explain | document | ask
115     def __iter__(self):
116         self.a = 2 # Changed starting number
117         return self
118
119     tabnine: test | explain | document | ask
120     def __next__(self):
121         if self.a <= 22: # Changed upper limit
122             x = self.a
123             self.a += 2 # Changed increment value
124             return x
125         else:
126             raise StopIteration
127
128 myclass = MyNumbers()
129 myiter = iter(myclass)
130 for x in myiter:
131     print(x)
132
133 def myfunc():
134     x = 500 # Changed value
135     print(x)
136
137 myfunc()
138
139 def myfunc():
140     x = 500 # Changed value
141     def myinnerfunc():
142         print(x)
143     myinnerfunc()
144
145 myfunc()
146
147 x = 500 # Changed value
148 def myfunc():
149     print(x)
150
151 myfunc()
152 print(x)
153
154 x = 500 # Changed value
155 def myfunc():
156     x = 400 # Changed local value
157     print(x)
158
159 myfunc()
160 print(x)
161
162 tabnine: test | explain | document | ask
163 def myfunc():
164     global x
165     x = 500 # Changed value
166
167 myfunc()
168 print(x)

```

---

# OUTPUT :

---

```
Aditya Sharma
Welcome Aditya Sharma to the class of 2005
Enter side 1 : [user input]
Enter side 2 : [user input]
Enter side 3 : [user input]
Side 1 is [side 1 value]
Side 2 is [side 2 value]
Side 3 is [side 3 value]
The area of the triangle is [calculated area value]
Super Class method called
Derived class 1 method called
Derived class 2 method called
Enter side 1 : [user input]
Enter side 2 : [user input]
Side 1 is [side 1 value]
Side 2 is [side 2 value]
```