

 sandeepsuryaprasad / python\_tutorials Private[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Security](#) [Ins](#)

master ▾

python\_tutorials / 10\_oops /  
1\_simple\_classes.py / <> Jump to ▾

Go to file

...



Sandeep Suryaprasad typos

Latest commit 3335379 on 29 Mar [History](#) 0 contributors

188 lines (161 sloc) | 6.21 KB

Raw

Blame



```
1  """
2  1. A class is collection/set of functions that carry out various operations o
3  "Instances"
4
5  2. Instances are the actual objects/data that your function manipulate on.
6  """
7  # Different ways of storing data using built-in data structures.
8  a = [1, 2]
9  t = (1, 2)
10 d = {'a': 1, 'b': 2}    # advantage of storing data in dict is that you can a
11
12 # Performing different operations on the data stored in the list.
13 # -----
14 # 1. sort
15 # 2. reverse
16 # 3. __len__()
17 # 4. __getitem__()
18 # 5. __contains__()
19 # -----
20 # Want to perform other operations apart from built-in
21 # -----
22 # 1. Adding a[0] = a[0] + 0.5, a[1] = a[1] + 0.5
23 # 2. Get the total of two co-ordinates total = a[0] + a[1]
24 # 3. Swap two co-ordinates temp = a[0], a[0] = a[1], a[1] = temp
25 # 4. Sorting two co-ordinates a.sort()
26 # 5. Resetting the co-ordinates a[0] = 0, a[1] = 0
27 # -----
```

```
28 # User defined class or datatype
29 class Point:
30     # Data is being saved inside a dictionary
31     def __init__(self, a, b):
32         self.a = a
33         self.b = b
34
35 p1 = Point(1, 2)
36 p2 = Point(10, 20)
37
38 # The values are internally stored in a dictionary. It is also called instance
39 print(p1.__dict__) # {"a": 1, "b": 2}
40 print(p2.__dict__) # {"a": 10, "b": 20}
41 # =====
42 # "Point" class with a some methods
43 class Point:
44     def __init__(self, a, b):
45         self.a = a
46         self.b = b
47
48     # Takes data from instance dictionary
49     def move(self, dx, dy):
50         self.a += dx
51         self.b += dy
52
53     # resets the value of self.a and self.b to zero
54     def reset(self):
55         self.a = 0
56         self.b = 0
57
58     # Method that sorts points
59     def sort(self):
60         if self.a < self.b:
61             return (self.a, self.b)
62         return (self.b, self.a)
63
64     # Method that swaps values of 'a' and 'b'
65     def swap_points(self):
66         temp = self.a
67         self.a = self.b
68         self.b = temp
69         return (self.a, self.b)
70
71     def total(self):
72         return self.a + self.b
```

```
73
74 p1 = Point(1, 2)
75 p2 = Point(1.4, 1.2)
76
77 # The information about data is present in instance dictionary (obj.__dict__)
78 # The information about methods is present in class dictionary (class_name.__dict__)
79 # =====
80 class Calculator:
81     def __init__(self, x, y):
82         self.a = x
83         self.b = y
84
85     def add(self):
86         return self.a + self.b
87
88     def sub(self):
89         return self.a - self.b
90
91     def mul(self):
92         return self.a * self.b
93
94 c1 = Calculator(1, 2)
95 c2 = Calculator(4, 5)
96 c3 = Calculator(10, 20)
97 # =====
98 # Employee Class
99 class Employee:
100     def __init__(self, fname, lname, pay):
101         self.fname = fname
102         self.lname = lname
103         self.pay = pay
104
105     def email(self):
106         return f'{self.fname}.{self.lname}@company.com'
107
108 e1 = Employee("Steve", "Jobs", 1000)
109 e2 = Employee("Bill", "Gates", 2000)
110 # =====
111 class Player:
112     def __init__(self, x, y):
113         self.x = x
114         self.y = y
115         self.health = 100
116
117     def move(self, dx, dy):
```

```
118         self.x += dx
119         self.y += dy
120
121     def attack(self, pts):
122         self.health -= pts
123
124 p1 = Player(1, 2)
125 p2 = Player(3, 4)
126 p3 = Player(5, 6)
127
128 print(p1.__dict__)
129 print(p1.__class__.__dict__)
130 print(Player.__dict__)
131 # Please note that __dict__ attribute is available only for custom classes and
132 # =====
133 class Point:
134     # a and b with default values
135     def __init__(self, a=0, b=0):
136         self.a = a
137         self.b = b
138
139 p1 = Point()
140 p2 = Point()
141 # =====
142 class Employee:
143     def __init__(self, fname, lname, pay, *args):
144         self.fname = fname
145         self.lname = lname
146         self.pay = pay
147         self.args = args
148
149 e1 = Employee('steve', 'jobs', 1000, 'python', 26, '2200 valley view lane')
150 # =====
151 # Overloading constructor using optional arguments
152 class Point:
153     def __init__(self, a=0, b=0, c=0):
154         self.a = a
155         self.b = b
156         self.c = c
157
158 p1 = Point()
159 p2 = Point(1)
160 p3 = Point(1, 2)
161 p4 = Point(1, 2, 3)
162 # =====
```

```
163 # We can have multiple __init__ method's, but the latest implementation of __
164 class Point:
165     def __init__(self, a, b):
166         self.a = a
167         self.b = b
168
169     # Re-defining __init__ method (new implementation)
170     def __init__(self, a, b, c):
171         self.a = a
172         self.b = b
173         self.c = c
174
175 # Python maintains the information about methods in class dictionary.
176 # we can access the dictionary using Point.__dict__
177 # Method name will be the key of the dictionary and the reference of the meth
178 # e.g.
179 # >>> Point.__dict__
180 # >>> mappingproxy({'__module__': '__main__', '__init__': <function Point.__i
181 # '__weakref__': <attribute '__weakref__' of 'Point' objects>, '__doc__': Non
182
183 # when you have multiple method's with the same name, the key of the class di
184 # So the new reference of the function object (with new implementation) would
185 # Point.__dict__
186 # >>> mappingproxy({'__module__': '__main__', '__init__': <function Point.__i
187 # '__weakref__': <attribute '__weakref__' of 'Point' objects>, '__doc__': Non
188 # =====
```