Programare I
Curs 9
Introducere în OOP. Principii OOP. Clase și obiecte

Botescu Mihai
mihai.botescu00@e-uvt.ro

May 2021

```
In [1]: class Point(object):
   ...:     def __init__(self,x,y): #P(x,y)
   ...:         self.x = x
   ...:         self.y = y
   ...:

In [2]: p = Point(2,-2)

In [3]: p

In [4]: class Point(object):
   ...:     def __init__(self,x,y): #P(x,y)
   ...:         self.x = x
   ...:         self.y = y
   ...:     def __str__(self):
   ...:         return f'<{self.x},{self.y}>'
   ...:

In [5]: p = Point(2,-2)

In [6]: p

In [7]: print(p)
<2,-2>

In [8]: class Point(object):
   ...:     def __init__(self,x,y): #P(x,y)
   ...:         self.x = x
   ...:         self.y = y
   ...:     def __repr__(self):
   ...:         return f'<{self.x},{self.y}>'
   ...:

In [9]: p = Point(2,4)

In [10]: print(p)
<2,4>

In [11]: import math

In [12]: class Point(object):
    ...:     def __init__(self,x,y): #P(x,y)
```

```
    ...:         self.x = x
    ...:         self.y = y
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>'
    ...:     def dist(self,P):
    ...:         return sqrt((self.x-P.x)^2 + (self.y-P.y)^2)
    ...:

In [13]: P1 = Point(0,0)

In [14]: P2 = Point(0,2)

In [15]: print(P1)

In [16]: print(P2)

In [17]: P1.dist(P2)


In [18]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>'
    ...:     def dist(self,P):
    ...:         return sqrt((self.x-P.x)^2 + (self.y-P.y)^2)
    ...:

In [19]: P1.dist(P2)

In [20]: math.sqrt(2)

In [21]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>'
    ...:     def dist(self,P):
    ...:         return math.sqrt((self.x-P.x)^2 + (self.y-P.y)^2)
    ...:

In [22]: P1.dist(P2)


In [23]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>'
    ...:     def dist(self,P):
    ...:         return math.sqrt((self.x-P.x)^2 + (self.y-P.y)^2)
```

```
    ...:
```

```
In [24]: P1 = Point(0,0)
```

```
In [25]: P2 = Point(0,2)
```

```
In [26]: P1.dist(P2)
```

```
In [27]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>'
    ...:     def dist(self,P):
    ...:         return math.sqrt((self.x-P.x)**2 + (self.y-P.y)**2)
    ...:
    ...:
```

```
In [28]: P1 = Point(0,0) #instantiez obiectul P1
```

```
In [29]: P2 = Point(0,2) #instantiez obiectul P2
```

```
In [30]: P1.dist(P2)
```

```
In [31]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y,culoare): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:         self._culoare = culoare
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>; culoare={self.culoare}'
    ...:     def dist(self,P):
    ...:         return math.sqrt((self.x-P.x)**2 + (self.y-P.y)**2)
    ...:
    ...:
```

```
In [32]: P1 = Point(1,2,'rosu')
```

```
In [33]: print(P1)
```

```
In [34]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y,culoare): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:         self._culoare = culoare
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>; culoare={self._culoare}'
    ...:     def dist(self,P):
    ...:         return math.sqrt((self.x-P.x)**2 + (self.y-P.y)**2)
    ...:
    ...:
```

```
In [35]: P1 = Point(1,2,'rosu')

In [36]: print(P1)

In [37]: P1._culoare = 'albastru'

In [38]: print(P1)

In [39]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y,culoare): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:         self._culoare = culoare #atributul este semiprivat (accesibil in afara
    ...:  clasei, dar sa ne simtim vinovati daca-l accesam)
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>; culoare={self._culoare}'
    ...:     def dist(self,P):
    ...:         return math.sqrt((self.x-P.x)**2 + (self.y-P.y)**2)
    ...:
    ...:

In [40]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y,culoare): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:         self.__culoare = culoare #atributul este privat (NU e accesibil in afa
    ...: ra clasei)
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>; culoare={self._culoare}'
    ...:     def dist(self,P):
    ...:         return math.sqrt((self.x-P.x)**2 + (self.y-P.y)**2)
    ...:
    ...:

In [41]: P1 = Point(1,2,'galben')

In [42]: print(P1)


In [43]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y,culoare): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:         self.__culoare = culoare #atributul este privat (NU e accesibil in afa
    ...: ra clasei)
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>; culoare={self.__culoare}'
    ...:     def dist(self,P):
    ...:         return math.sqrt((self.x-P.x)**2 + (self.y-P.y)**2)
    ...:
    ...:

In [44]: P1 = Point(1,2,'galben')
```

```
In [45]: print(P1)

In [46]: P1.__culoare = 'mov'

In [47]: print(P1)

In [48]: import math
    ...: class Point(object):
    ...:     def __init__(self,x,y,culoare): #P(x,y)
    ...:         self.x = x
    ...:         self.y = y
    ...:         self.__culoare = culoare #atributul este privat (NU e accesibil in afa
    ...: ra clasei)
    ...:     def __repr__(self):
    ...:         return f'<{self.x},{self.y}>; culoare={self.__culoare}'
    ...:     def dist(self,P):
    ...:         return math.sqrt((self.x-P.x)**2 + (self.y-P.y)**2)
    ...:     def modifica_culoare(self, culoare):
    ...:         self.__culoare = culoare
    ...:
    ...:

In [49]: P1 = Point(1,2,'galben')

In [50]: print(P1)

In [51]: P1.__culoare = 'mov'

In [52]: print(P1)

In [53]: P1.modifica_culoare('verde')

In [54]: print(P1)
```