برنامه سازی پیشرفته
(برنامه نویسی شیءگرا: مقدمه)

صادق اسکندری - دانشکده علوم ریاضی، گروه علوم کامپیوتر

eskandari@guilan.ac.ir

یادآوری ...

یک نوع داده عبارت است از مجموعه ای از مقادیر به همراه مجموعه ای از عملگرها بر روی آن مقادیر

Data type = set of values (domain) + set of operators

Integer = Z + {+,*,/, ...}

انواع داده در پایتون

- انواع داده درون ساخت (Built-in data type)

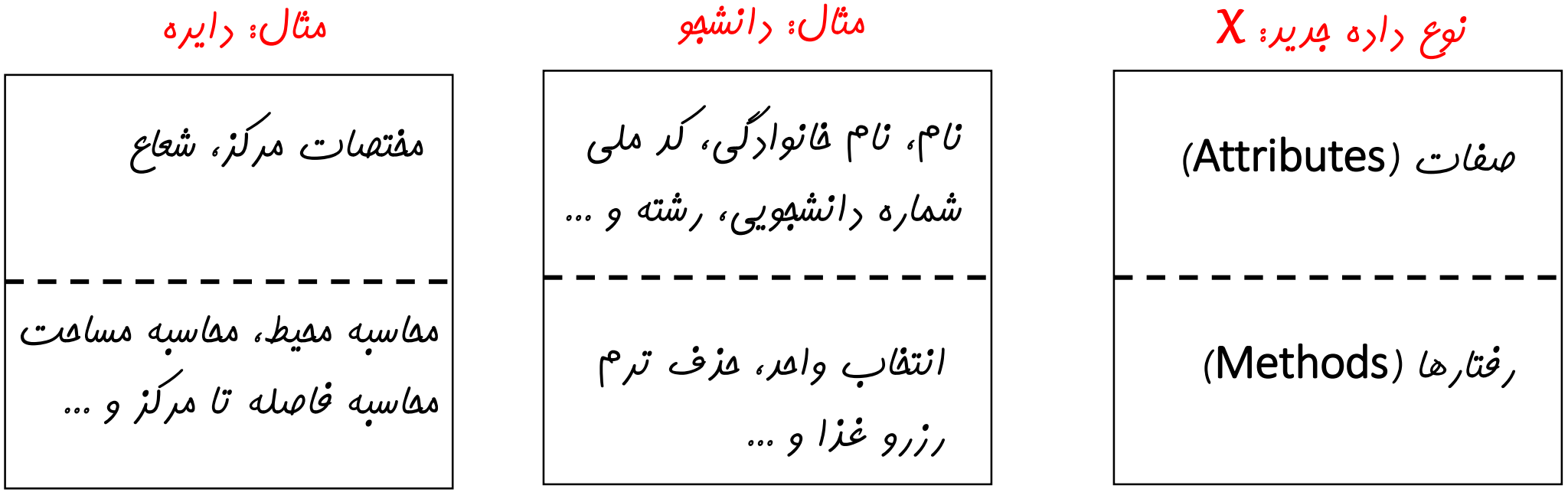
Numbers, Strings, Lists, Dictionaries, Tuples, Files, Sets,

- انواع داده کلاسی

Student, Teacher, Car, TV, ...

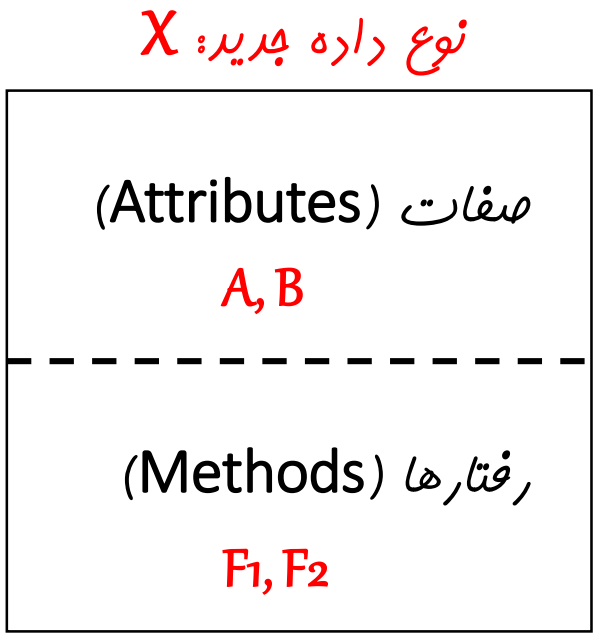
کلاس: ایجاد انواع داده جدید

برای تعریف یک نوع داده جدید، از مفهوم کلاس (class) استفاده می شود.



کلاس: ایجاد انواع داده جدید

کد نویسی یک کلاس در پایتون




```
class X:
    def __init__(self, a, b):
        self.A =a
        self.B = b

    def F1(self,params):
        #F1 Body

    def F2(self,params):
        #F2 Body
```

سازنده (Constructor): اشیاء از این نوع داده را مقدار دهی اولیه می کند.

یعنی یک متغیر (شیء) از نوع X را با صفات A=a و B=b ایجاد می کند.

کلاس: ایجاد انواع داده جدید

به متغیرهایی که از نوع داده جدید (داده کلاسی) ایجاد می کنیم، یک نمونه یا یک شیء از آن نوع داده می گوییم.

```
class X:
    def __init__(self, a, b):
        self.A =a
        self.B = b

    def F1(self,params):
        #F1 Body

    def F2(self,params):
        #F2 Body
```

```
Obj1 = X(10,"aabb")
Obj2 = X(a = 44, b = "jfdk")
```

ایجاد اشیاء مختلف از نوع داده X

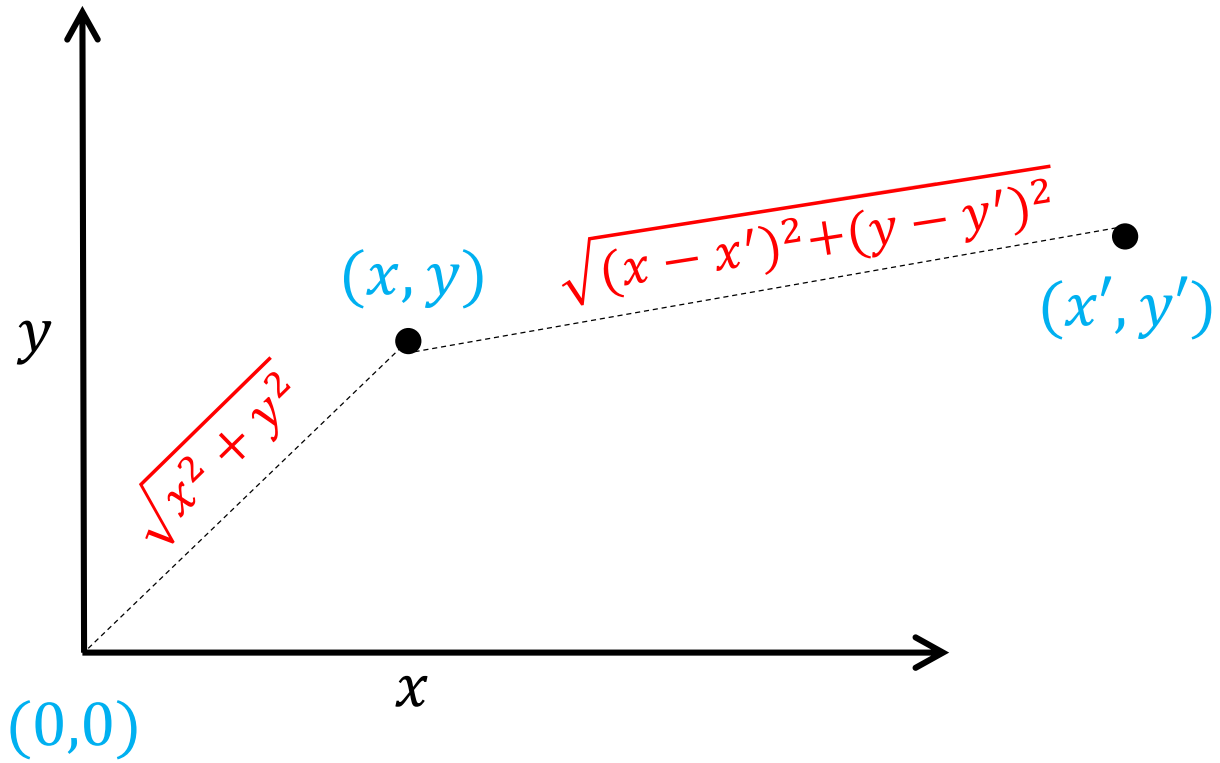
```
Obj1.A = 18
Obj2.B = 32

Obj1.F1(…)
```

با استفاده از نقطه، می توان به صفات و متدهای یک شیء دسترسی داشت

توجه مهم: یک کلاس، یک الگو برای یک نوع داده است ولی یک شیء یک نمونه واقعی از آن نوع داده است.

کلاس: ایجاد انواع داده جدید

مثال: نوع داده نقطه (Point)



نوع داده: Point

صفات (Attributes)

x, y: مختصات

رفتارها (Methods)

distance_from_origin(): فاصله این نقطه از مبدأ مختصات

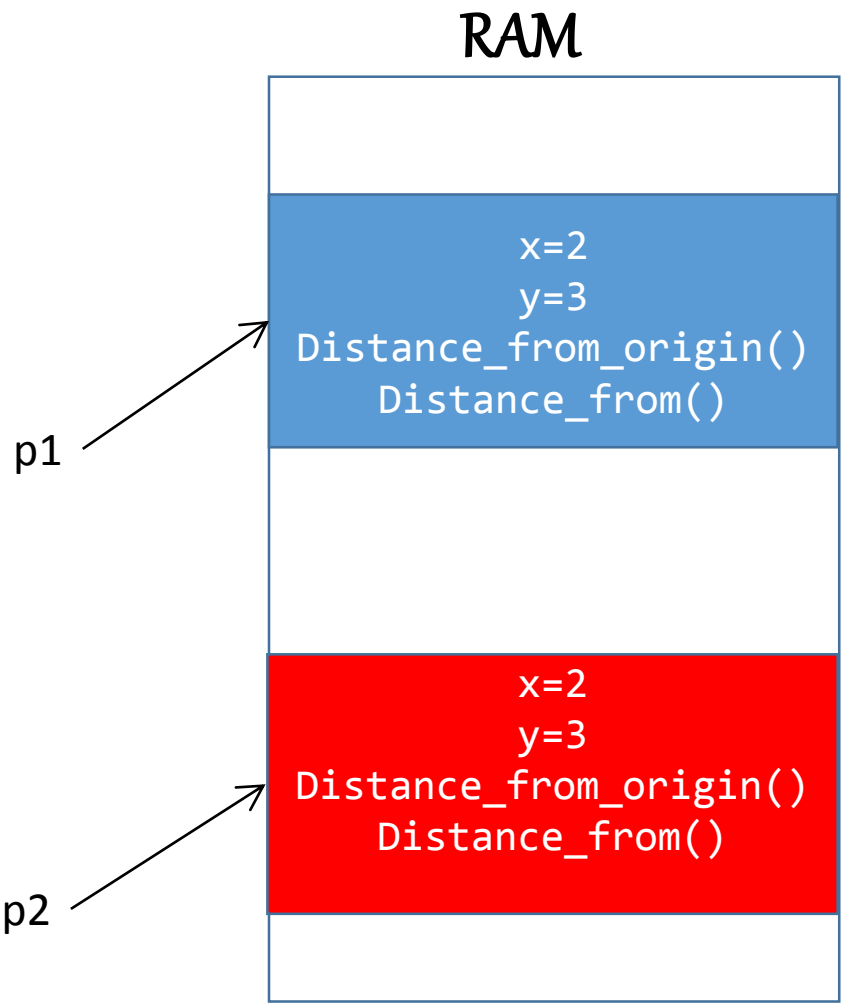
distance_from(p): فاصله این نقطه از نقطه p

کلاس: ایجاد انواع داده جدید

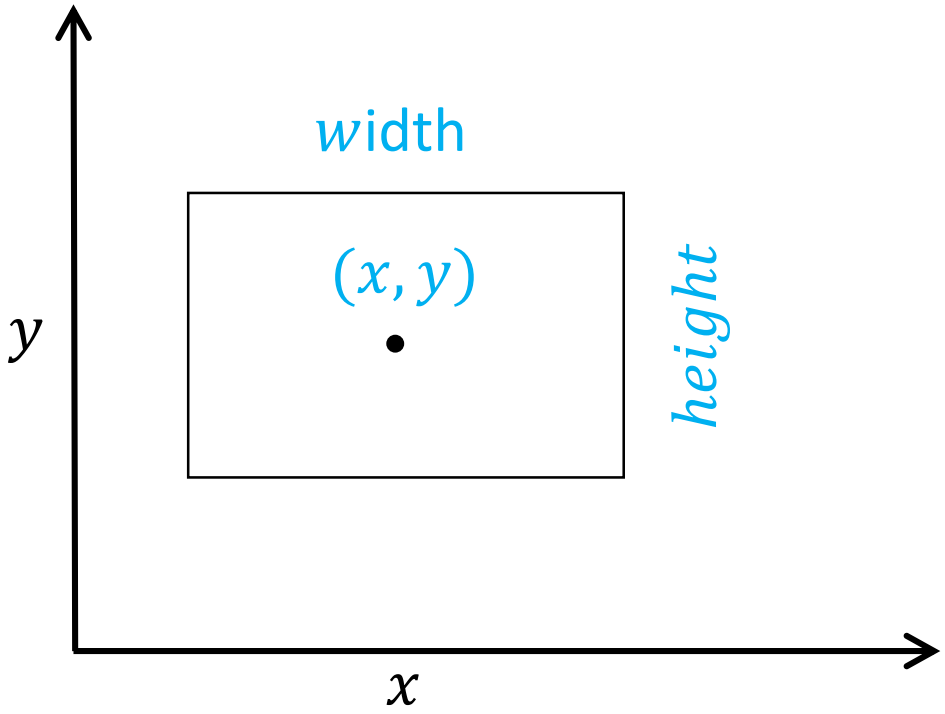
مثال: نوع داده نقطه (Point)

```python
import math

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance_from_origin(self):
        return math.sqrt( self.x**2 + self.y**2 )

    def distance_from(self,p):
        return  math.sqrt( (self.x - p.x)**2
                           + (self.y - p.y)**2 )

p1 = Point(2,3)
p2 = Point(3,4)

print(p1.distance_from_origin())
print(p2.distance_from_origin())
print(p1.distance_from(p2))
print(p2.distance_from(p1))
print(p1.distance_from(p1))
```

```
3.605551275463989
5.0
1.4142135623730951
1.4142135623730951
0.0
```



کلاس: ایجاد انواع داده جدید



مثال: نوع داده مستطیل (Rectangle)

نوع داده: **Rectangle**

صفات (Attributes)

- **center**: نقطه مرکز مستطیل
- **width**: عرض مستطیل
- **height**: طول مسطیل

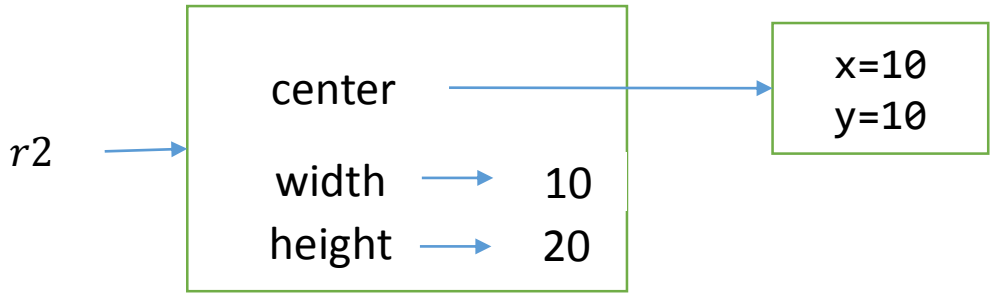
رفتارها (Methods)

- **area()**: مساحت مستطیل
- **perimeter()**: محیط مستطیل
- **distance(r)**: فاصله مرکز این مستطیل تا مرکز مستطیل **r**
- **Is_square()**: آیا این مستطیل، مربع است؟

کلاس: ایجاد انواع داده جدید

مثال: نوع داده مستطیل (Rectangle)

```
1  class Rectangle:
2      def __init__(self, c=Point(0,0), w=1, h=1):
3          self.center = c
4          self.width = w
5          self.height = h
6
7      def area(self):
8          return self.width * self.height
9
10     def perimeter(self):
11         return 2*self.width + 2*self.height
12
13     def distance(self,r):
14         return self.center.distance_from(r.center)
15
16     def is_square(self):
17         return self.width == self.height
18
19 r1 = Rectangle()
20 r2 = Rectangle(Point(10,10), w=10, h=20)
21
22 print(r1.area())                    1
23 print(r2.area())                    200
24 print(r2.distance(r1))              14.142135623730951
25 print(r1.is_square())               True
26 print(r2.is_square())               False
```



وضعیت ارجاعات شیء **r2**

کلاس: ایجاد انواع داده جدید

تمرین: کلاس Rectangle را به گونه ای بازتعریف کنید که دارای رفتارهای زیر نیز باشد:

show(): رسم مستطیل با استفاده از لاکپشت

Intersect(r): ناحیه اشتراک این مستطیل با مستطیل **r**: پاسخ باید در قالب یک مستطیل جدید برگردانده شود.

Contains(r): بررسی می کند که آیا مستطیل **r** به شکل کامل درون این مستطیل قرار گرفته است؟

Equals(r): آیا این مستطیل از نظر ابعاد (نه نقطه مرکزی)، با مستطیل **r** یکسان است؟

کلاس: مقایسه اشیاء

زمانی که می گوییم، علی و مریم ماشین های یکسانی دارند، منظورمان این است که مدل ماشین های آنها (رنگ، مارک، ...) با هم برابر است. در اینجا منظورمان این نیست که آنها یک ماشین را با هم استفاده می کنند. به چنین مقایسه ای، مقایسه سطحی (Shallow Comparison) گفته می شود.

زمانی که می گوییم علی و مریم مادر یکسانی دارند، منظورمان این است که مادر هر دوی آنها یک نفر است، نه اینکه مادرهای آنها شبیه به هم است. به چنین مقایسه ای، مقایسه عمیق (deep Comparison) گفته می شود.

کلاس: مقایسه اشیاء

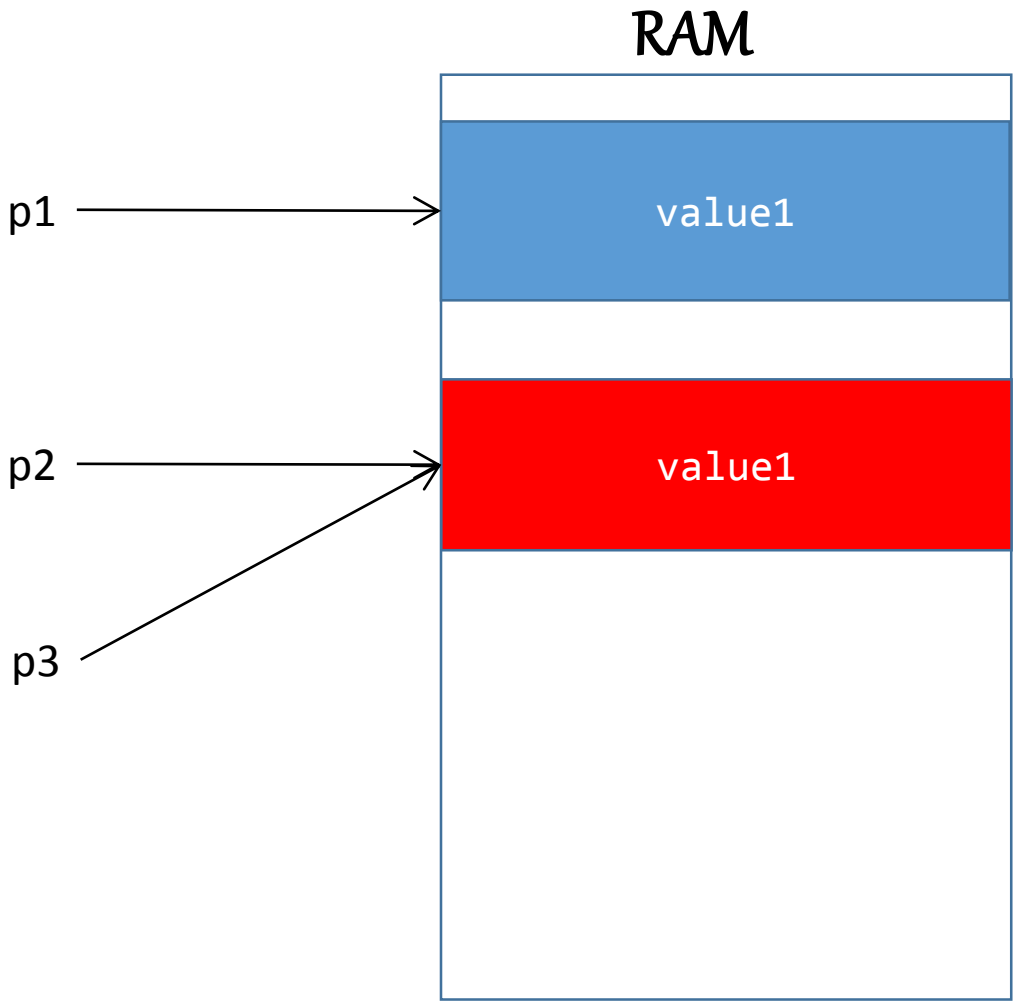
در زبانهای برنامه نویسی نیز مفهوم مقایسه سطحی و عمیق به صورت زیر قابل بیان است:

مقایسه سطحی دو شیء عبارت است از مقایسه مقادیر آنها

بنابراین مقایسه سطحی **p1** با **p2** دارای پاسخ، **True** خواهد بود. زیرا، هر دو دارای مقادیر یکسان **value1** هستند.

مقایسه عمیق دو شیء عبارت است از مقایسه آدرس های آنها

بنابراین مقایسه عمیق **p1** با **p2** دارای پاسخ **False** خواهد بود. زیرا، اشیاء آنها یکی نیستند. ولی مقایسه عمیق **p2** با **p3** دارای پاسخ **True** خواهد بود زیرا هر دو به یک چیز اشاره می کنند.



کلاس: مقایسه اشیاء

در پایتون، عملگرهای اصلی جهت مقایسه اشیاء عبارتند از عملگر رابطه ای == و کلمه کلیدی is

نوع داده	تأثیر عملگر ==	تأثیر عملگر is
اولیه	سطحی l1 = [1,2,3] l2 = [1,2,3] l1 == l2 #True	عمیق l1 = [1,2,3] l2 = [1,2,3] l1 is l2 #False
کلاسی	عمیق r1 = Rectangle(Point(10,10), 10,20) r2 = Rectangle(Point(10,10), 10,20) print(r1 == r2) #False	عمیق r1 = Rectangle(Point(10,10), 10,20) r2 = Rectangle(Point(10,10), 10,20) print(r1 is r2) #False

سوال: اگر بخواهیم دو شیء کلاسی را به صورت سطحی مقایسه کنیم، چه باید کرد؟

جواب: چنین مقایسه ای باید به شکل یک متد درون تعریف کلاس گنجانده شود.

کلاس: مقایسه اشیاء

```python
import math

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def distance_from_origin(self):
        return math.sqrt( self.x**2 + self.y**2 )

    def distance_from(self,p):
        return  math.sqrt( (self.x - p.x)**2
                           + (self.y - p.y)**2 )

    def equals(self, p):
        return self.x == p.x and self.y == p.y

p1 = Point(2,3)
p2 = Point(3,4)
p3 = Point(2,3)

print(p1.equals(p2))
print(p1.equals(p1))
print(p1.equals(p3))
print(p1 is p3)
print(p1 is p1)
print(p1 == p3)
```

```
False
True
True
False
True
False
```