Basic Python Programming

[Session 1] Lab session

ITinerary X University of Ghana

Contents

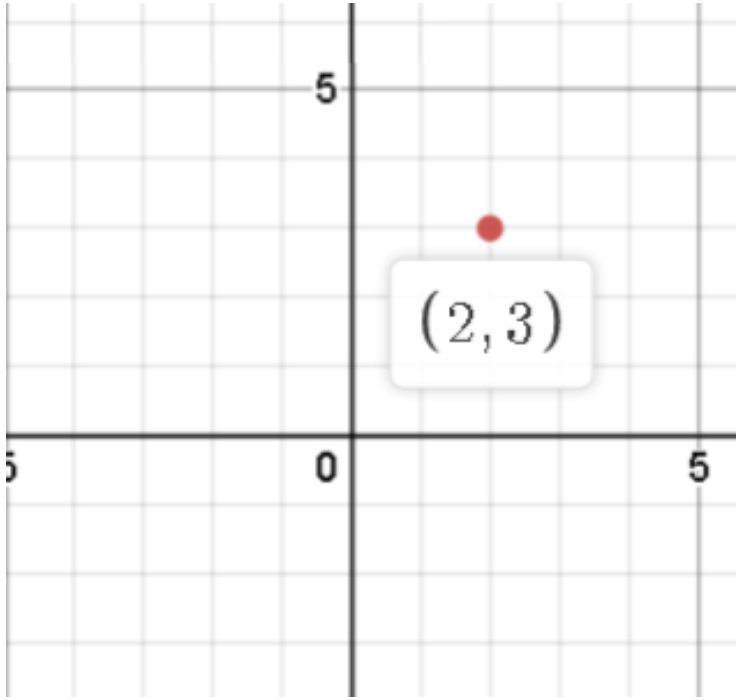
- **Class: Dot**
- **Class: Car**

Class: Dot

Dot [1]

- We will implement “Dot”

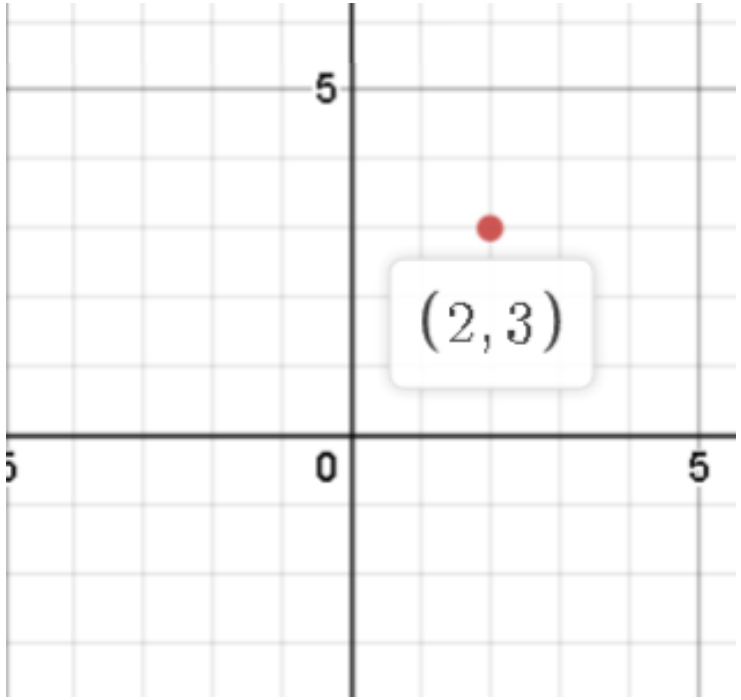
- dot?



Dot [2]

- Basically, Dot contains two data
  - x and y

- Okay, that's all for member variables
  - And then?
  - Define methods!



Review: Methods [1]

- **Note that, every method should be like…**

```
class A:
    def some_method(self, param1, param2):
        pass
```

Review: Methods [2]

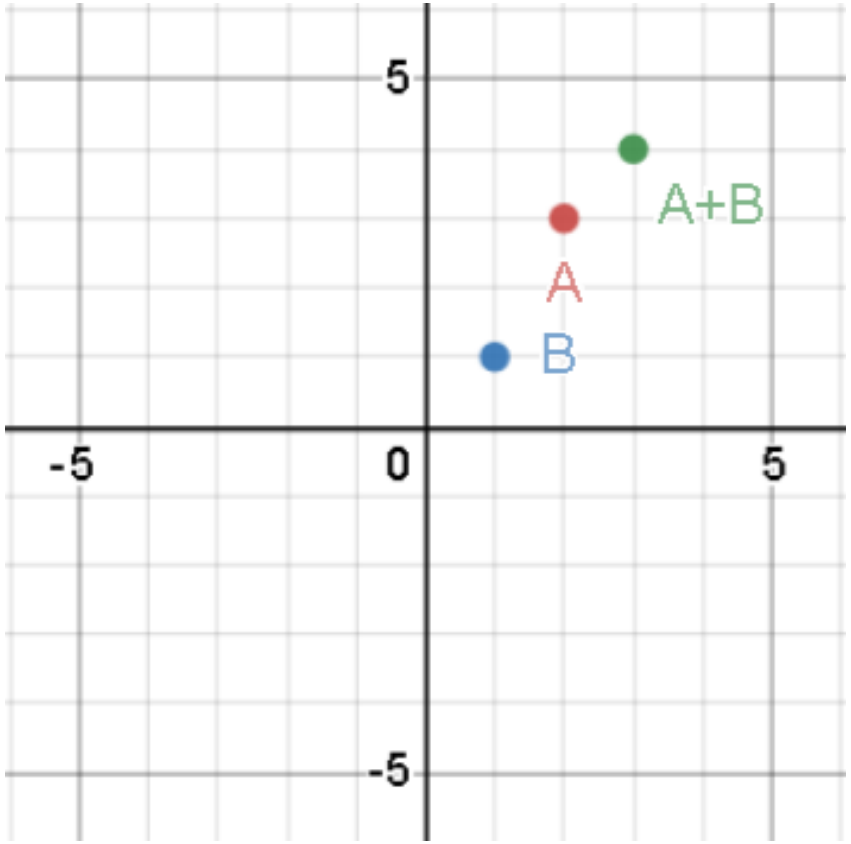
- **You can access the member variable inside method, by using self!**

```python
class Student:
    def __init__(self, my_name):
        self.name = my_name

    def get_name(self):
        return self.name

john = Student("John")
print(john.get_name())
```

```
C:\Users\user\Documents\GitHub\EunSeong-Park.github.io\conte
John
```

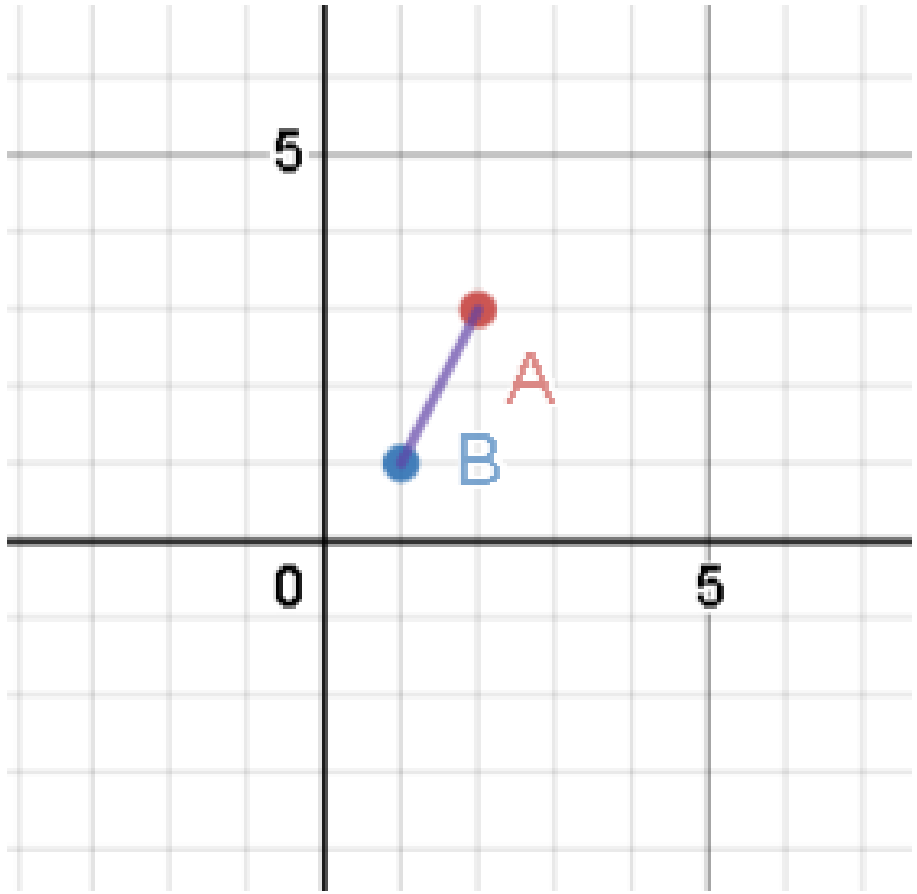
Dot: Methods [1]

- **dot_add, dot_sub method adds / subtracts each coordinate**
  - Example: (2, 3) + (1, 1) = (3, 4)



Dot: Methods [2]

- dot_dist returns the distance between two points
  - in this case, root(5).



Dot: Methods [3]

- Other methods are defined in the skeleton code.
  - You have to modify this

```python
# skeleton
import math

class Dot:
    def __init__(self, x_input, y_input):
        pass
    def dot_get_x(self):
        # return x (number)
        pass
    def dot_get_y(self):
        # return y (number)
        pass
    def dot_add(self, other):
        # return (x, y) = (x1 + x2, y1 + y2) (Dot)
        pass
    def dot_sub(self, other):
        # return (x, y) = (x1 - x2, y1 - y2) (Dot)
        pass
    def dot_dist_origin(self):
        # return the distance from the origin (0,0) (number)
        pass
```

Let’s start!

- **Use the skeleton, dot.py**

Class: Car

Car

- **A car can do**
  - drive (as long as the fuel remains)
  - show the its status:
    - fuel (L)
    - total distance (km)
    - etc.

- **Cars have their own features:**
  - car name
  - mileage (km/L)
  - max fuel (L)

Car: Member Variables

```
class Car:
  def __init__(self, name, mileage, max_fuel):
    self.name = name
    self.mileage = mileage
    self.max_fuel = max_fuel
    self.fuel = self.max_fuel
    self.dist = 0
```

Car: Methods [1]

- **Brrr(self, km): drive X km**
  - The car should consume the fuel, as you drive
  - If the fuel is not enough, it cannot go
  - After that, show the current state

- **gas_station(self): Full the fuel**
  - It’s free!
  - After that, show the current state

Car: Methods [2]

- **status(self): show the current status**
  - warn if the fuel is too low

```
Car name: BMW
 Mileage: 16km/L
    Fuel: 25.0L / 50L
Distance: 400km
```

```
Car name: BMW
 Mileage: 16km/L
    Fuel: 0.0625L / 50L
Distance: 800549km
WARNING: remaining fuel is too low
```

Let’s start!

- **Use the skeleton, car.py**

Thank you