Agenda

03 February 2025 17:57

1. Why we need to study OOP (The problem and solution)
2. Class & Object
3. Four pillars of oops
   3.1 Inheritance
   3.2 Polymorphism
   3.2 Encapsulation
   3.4 Abstraction

Why we need to study OOP (The problem and solution)

03 February 2025 18:15

The Problem:

==========

What is Procedural Programming?

Procedural programming follows a step-by-step approach, where a program is structured as a sequence of procedures (functions) that operate on data. It focuses on writing functions that perform specific tasks.

Example:

Bank application

1. ATM functions(deposit, withdraw, check balance)
2. Loan Functions(interest calc, foreclosure)

Disadvantages:

1. As features grows it is not easy to maintain
2. Need to have mappings for each feature (mapping data & functions)

The Solution:

==========

OOP:

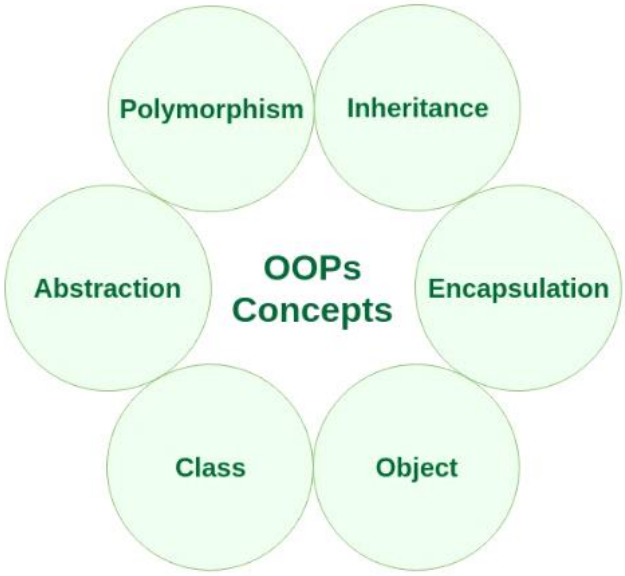
OOP organizes data(variables) and functions(behaviour) together into **objects** using **classes**.

Advantages:

Feature	**Procedural Programming**	**Object-Oriented Programming**
Structure	Uses functions and variables separately	Groups together (data and functions) into objects (classes)
Data Security	Data is global, risk of accidental modification	Data is encapsulated inside objects/class
Code Reusability	Difficult to reuse functions	Easy to reuse objects and classes
Scalability	Becomes complex handling in multiple features	Easy to extend with new features

Note: Using OOPS we can create our own data types(application specific).
In-built data types in python are nothing but classes

OOPS pillars/principles:


Polymorphism
Inheritance
Abstraction
OOPs Concepts
Encapsulation
Class
Object

Class & Object

03 February 2025 19:20

Class:

It is a blueprint and it is virtual(doesn't have memory), A class defines a set of rules/principles(attributes/data/variables and methods) that object can use them.

Object:

It is an instance of class and it has physical memory, Can access those class(attributes and methods)

Examples:

Class	Objects
Car	Toyota, Suzuki, Honda
Mobile	iPhone, Samsung
Television	Sony, BPL

Let us create a banking example with below functionalities:

- Change pin
- Check balance
- Withdraw
- Deposit

Functions (V) Methods

functions :- These are global and they can be applied on all other classes

Methods :- These are local to class. where these are defined, Any function defined inside a class are called methods.

Re-visit the example with granularity

06 February 2025 20:30

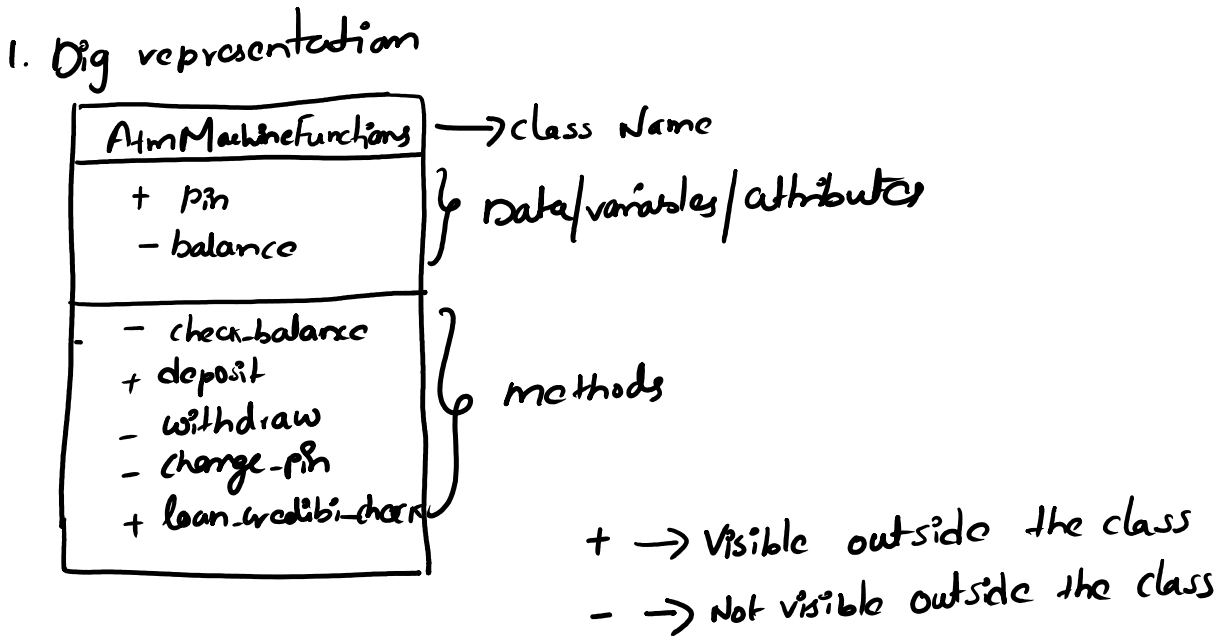
Let us create a banking example with below functionalities:

- Change pin
- Check balance
- Withdraw
- Deposit
- loan credibility check

Class Diagram

06 February 2025 18:55

1. Diagrammatic representation of class
2. Public vs Private(visibility)



self keyword

06 February 2025 19:04

Definition: self is nothing but current object.

Why we need self?

A Python class does not allow methods or data to be accessed directly within other methods. To enable this interaction (calling methods interchangeably within the class), we use self

Note: We can use any word instead of self, such as shewag, honda, ram, etc.

self keyword

```
class Sample():

  def __init__(self):
    print(id(self))

  def func1(self):
    print("I am here in func1")
```

```
[2] s= Sample()
```

```
133365005766672
```

```
[3] print(id(s))
```

```
133365005766672
```

```
s1 = Sample()
```

```
133365004513296
```

```
[5] print(id(s1))
```

```
133365004513296
```

-> We can use any word instead of self

```
[28] class Sample():

       def __init__(vishnu):
```

will behave as self

```
[28] class Sample():

       def __init__(vishnu):
         print(id(vishnu))
         vishnu.a=1000

       def func1(vishnu):
         print("I am here in func1")
         print(vishnu.a)

       def func2(vishnu):
         print("I am here in func2")
         print(vishnu.a)
         vishnu.func1()
```

Will behave as self

```
[29] s= Sample()
```

```
133365005687952
```

```
s.func2()
```

```
I am here in func2
1000
I am here in func1
1000
```

Magic Methods

06 February 2025 19:52

1. __init__ - special method**(constructor)** will get called **implicitly** while **object creation time**
   Usage - writing configurations code
   Types : non-parameterized and parameterized

2. __str__ : It is special method in Python and gets executed when an object is **printed using print()**
   Usage: This method allows you to customize the output while printing the object, making it more readable and meaningful.

3. __add__: Performs **addition between 2 objects**, (python class doesn't know how to perform addition between 2 objects)
   Usage: when you want to perform **addition** between two objects of a custom class using the **+** operator

4. __sub__: Performs **subtraction between 2 objects**, (python class doesn't know how to perform addition between 2 objects)
   Usage: when you want to perform **subtraction** between two objects of a custom class using the **-** operator

5. __mul__: Performs **multiplication between 2 objects**, (python class doesn't know how to perform addition between 2 objects)
   Usage: when you want to perform **multiplication** between two objects of a custom class using the * operator

6. __truediv__: Performs **division between 2 objects**, (python class doesn't know how to perform addition between 2 objects)
   Usage: when you want to perform **division** between two objects of a custom class using the / operator

Constructor(- - init - -) :-

```
[32] class Sample():

       def __init__(self,b,c,d):
         print(id(self))
         self.a=b
         print(self.a)
         print(b)
         print(c,d)


       def func1(self):
         print("I am here in func1")
         print(self.a)

       def func2(self):
         print("I am here in func2")
         print(self.a)
         self.func1()
```

parametrized const

```
s= Sample(1,2,3)
```

This how you should call param const

```
133364824488656
1
1
2 3
```

```
class Sample():

  def __init__(self):
    print(id(self))


  def func1(self):
    print("I am here in func1")
    print(self.a)

  def func2(self):
    print("I am here in func2")
    print(self.a)
    self.func1()
```

Non-param const

```
[38] s= Sample()
```

This is how you should call non-param const

```
        self.func1()
```

This is how

```
[38] s= Sample()
```

```
133364824706128
```

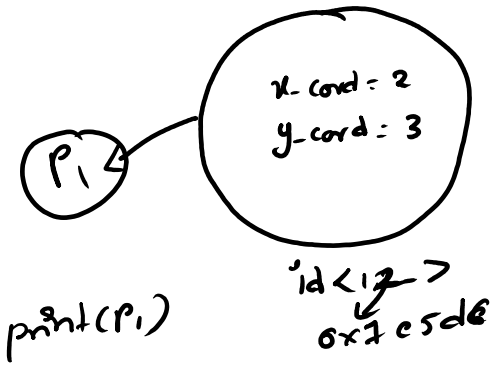
```
[19] class Point():

       def __init__(self,a,b):
         self.x_cord = a
         self.y_cord = b
```

```
p1 = Point(2,3)
```

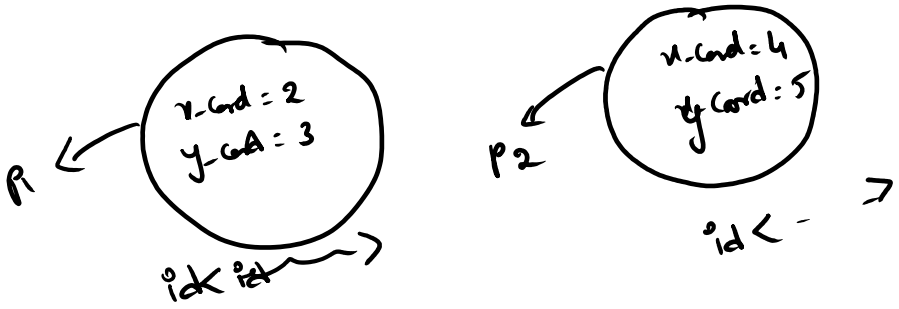
```
print(p1)
```

```
<__main__.Point object at 0x7e5d6ca67f10>
```

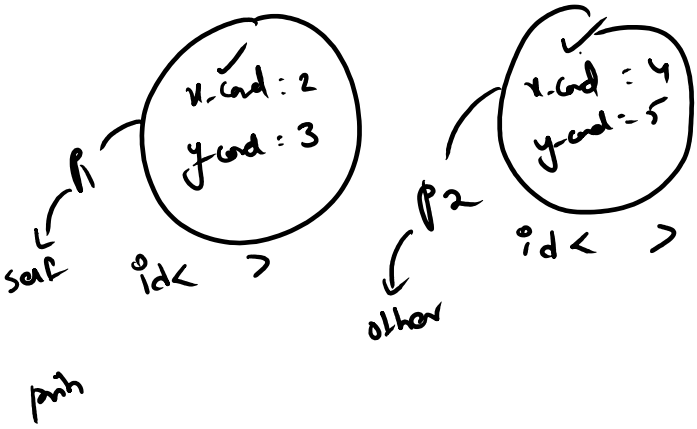
(2,3) <2,3>




```
p1 = Point(2,3)
p2=Point(4,5)
```




```
class Point():

  def __init__(self,a,b):
    self.x_cord = a
    self.y_cord = b

  def __str__(self):
    return "({},{})".format(self.x_cord,self.y_cord)


  def __add__(self,other):
    x = self.x_cord + other.x_cord
    y = self.y_cord + other.y_cord
    return "({},{})".format(x,y)
```



p1 = Point(2,3) ✓

p2 = Point(4,5) ✓

print(p1) ✓ (2,3)

print(p2) ✓ (4,5)

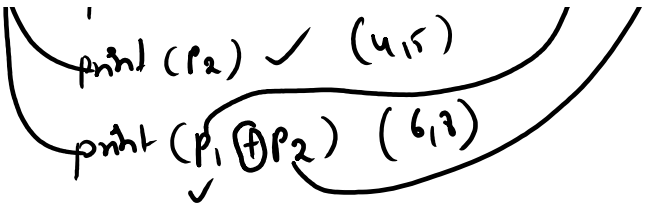
print

x = 6

y = 18

a = 3

b = a

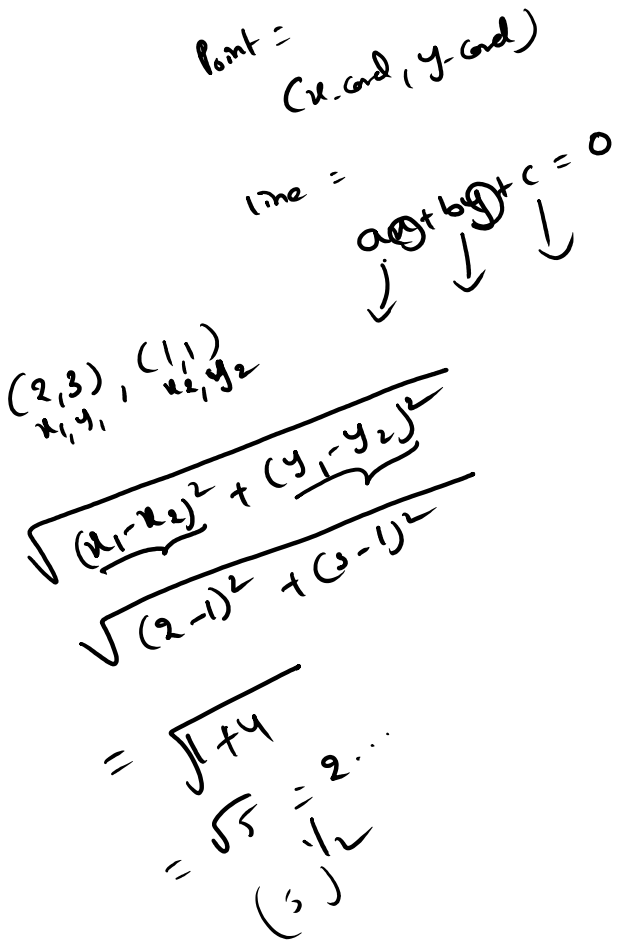

print (p2) ✓ (4,5)
print (p1 + p2) (6,7)
y=18

Sample Question

10 February 2025 19:13

Let us create own data type called Geometry which performs operations on point and line

1. Find the distance between 2 points?
2. Find the distance between point and line?
3. Check whether the point exists on line or not?
4. Find the distance from a point to an origin?
5. Find the mid point between 2 given points?



Find the distance between 2 points?

```
class Point():
  def __init__(self,a,b):
    self.x_cord = a
    self.y_cord = b
  def __str__(self):
    return "<{},{}>".format(self.x_cord,self.y_cord)

  def distance_between_points(self,other):
    return ((self.x_cord - other.x_cord)**2 + (self.y_cord - other.y_cord)**2)**0.5
```

```
p1=Point(2,3)
print(p1)
```

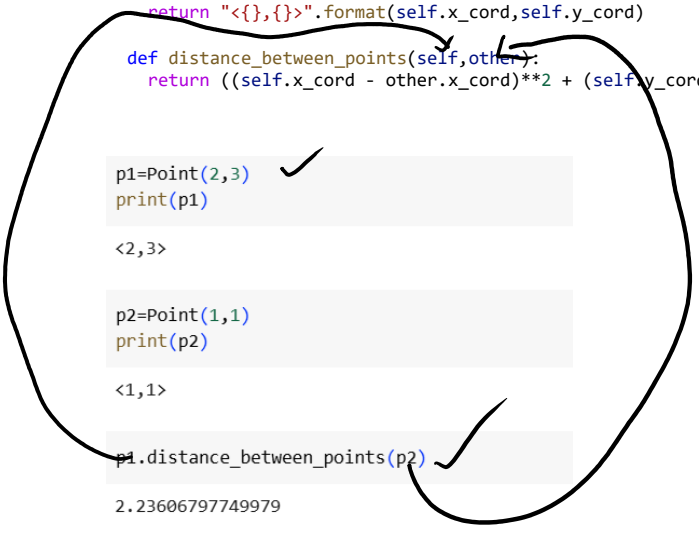
```
<2,3>
```

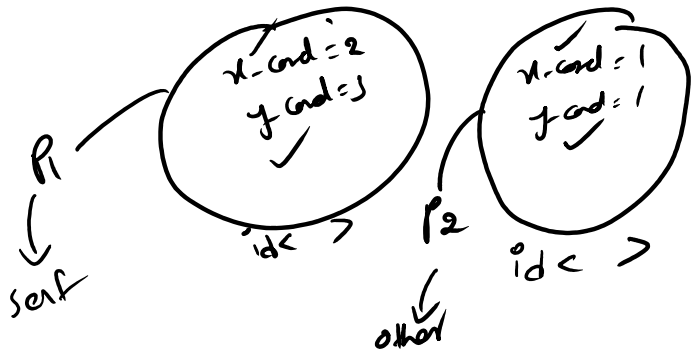
```
p2=Point(1,1)
print(p2)
```

```
<1,1>
```

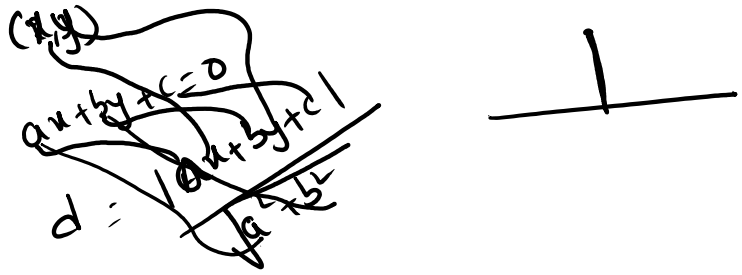
```
p1.distance_between_points(p2)
```

```
2.23606797749979
```

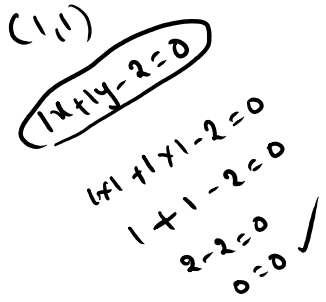




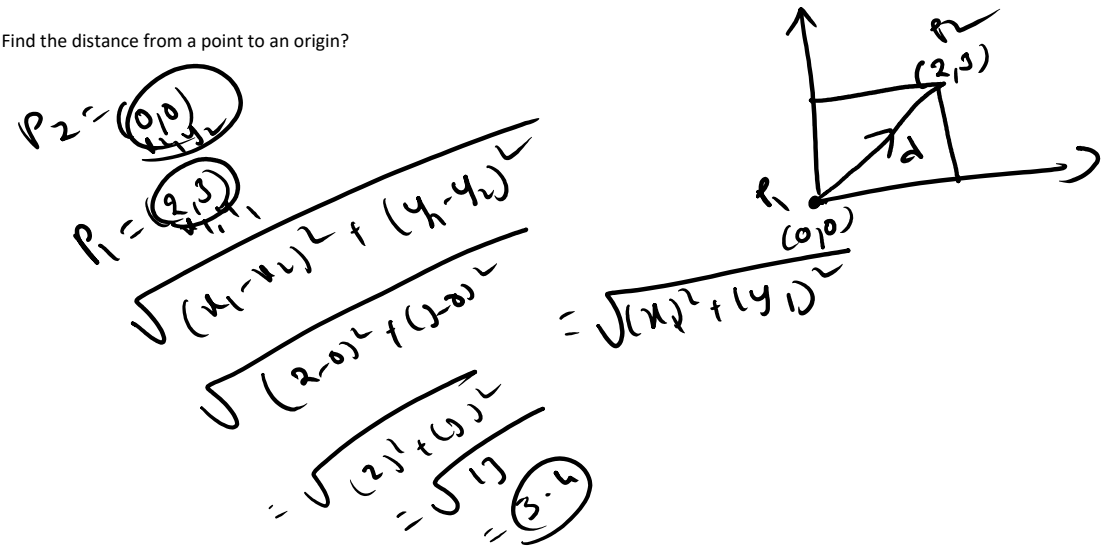
Find the distance between point and line?



Check whether the point exists on line or not?



Find the distance from a point to an origin?



Find the mid point between 2 given points?

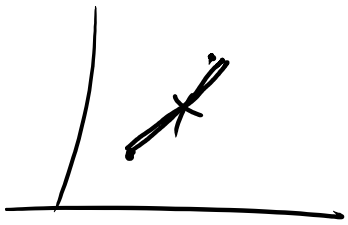
$P_1 = (x_1, y_1)$

$P_2 = (x_2, y_2)$

$\left(\frac{x_1+x_2}{2}, \frac{y_1+y_2}{2}\right)$

$= \left(\frac{3}{2}, \frac{4}{2}\right)$

mid p $= (1.5, 2)$



Encapsulation, class & static methods

10 February 2025 21:08

Encapsulation:

Encapsulation is the process of bundling data (variables) and methods (functions) together into a single unit (class) while restricting direct access outside the class

This ensures that sensitive information is **hidden** from direct modification and can only be accessed through controlled mechanisms like **getter and setter methods**.