## Write OOP classes to handle the following scenarios:

- A user can create and view 2D coordinates
- A user can find out the distance between 2 coordinates
- A user can find find the distance of a coordinate from origin
- A user can check if a point lies on a given line
- A user can find the distance between a given 2D point and a given line

```
class Point:

  def __init__(self,x,y):
    self.x_cod = x
    self.y_cod = y

  def __str__(self):
    return '<{},{}>'.format(self.x_cod,self.y_cod)

  def euclidean_distance(self,other):
    return ((self.x_cod - other.x_cod)**2 + (self.y_cod - other.y_cod)**2)**0.5

  def distance_from_origin(self):
    return (self.x_cod**2 + self.y_cod**2)**0.5
    # return self.euclidean_distance(Point(0,0))


class Line:

  def __init__(self,A,B,C):
    self.A = A
    self.B = B
    self.C = C

  def __str__(self):
    return '{}x + {}y + {} = 0'.format(self.A,self.B,self.C)

  def point_on_line(line,point):
    if line.A*point.x_cod + line.B*point.y_cod + line.C == 0:
      return "lies on the line"
    else:
      return "does not lie on the line"

  def shortest_distance(line,point):
    return abs(line.A*point.x_cod + line.B*point.y_cod + line.C)/(line.A**2 + line.B**2)**0.5



l1 = Line(1,1,-2)
p1 = Point(1,10)
print(l1)
print(p1)

l1.shortest_distance(p1)
```

```
1x + 1y + -2 = 0
<1,10>
6.363961030678928
```

## How objects access attributes

```
class Person:

  def __init__(self,name_input,country_input):
    self.name = name_input
    self.country = country_input

  def greet(self):
    if self.country == 'india':
      print('Namaste',self.name)
    else:
      print('Hello',self.name)



# how to access attributes
p = Person('nitish','india')


p.name
```

```
'nitish'
```

```
# how to access methods
p.greet()
```

```
Namaste nitish
```

```
# what if i try to access non-existent attributes
p.gender
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-49-39388d77d830> in <module>
      1 # what if i try to access non-existent attributes
----> 2 p.gender

AttributeError: 'Person' object has no attribute 'gender'
```

## Attribute creation from outside of the class

```
p.gender = 'male'


p.gender
```

```
'male'
```

## Reference Variables

- Reference variables hold the objects
- We can create objects without reference variable as well
- An object can have multiple reference variables
- Assigning a new reference variable to an existing object does not create a new object

```
# object without a reference
class Person:

  def __init__(self):
    self.name = 'nitish'
    self.gender = 'male'

p = Person()
q = p


# Multiple ref
print(id(p))
print(id(q))
```

```
140655538334992
140655538334992
```

```
# change attribute value with the help of 2nd object
```

```
print(p.name)
print(q.name)
q.name = 'ankit'
print(q.name)
print(p.name)
```

```
nitish
nitish
ankit
ankit
```

## Pass by reference

```
class Person:

  def __init__(self,name,gender):
    self.name = name
    self.gender = gender

# outside the class -> function
def greet(person):
  print('Hi my name is',person.name,'and I am a',person.gender)
  p1 = Person('ankit','male')
  return p1

p = Person('nitish','male')
x = greet(p)
print(x.name)
print(x.gender)
```

```
Hi my name is nitish and I am a male
ankit
male
```

```
class Person:

  def __init__(self,name,gender):
    self.name = name
    self.gender = gender

# outside the class -> function
def greet(person):
  print(id(person))
  person.name = 'ankit'
  print(person.name)

p = Person('nitish','male')
print(id(p))
greet(p)
print(p.name)
```

```
140655538334288
140655538334288
ankit
ankit
```

## Object ki mutability

```
class Person:

  def __init__(self,name,gender):
    self.name = name
    self.gender = gender

# outside the class -> function
def greet(person):
  person.name = 'ankit'
  return person

p = Person('nitish','male')
print(id(p))
p1 = greet(p)
print(id(p1))
```

```
140655555218960
140655555218960
```

## Encapsulation

```
# instance var -> python tutor
class Person:

  def __init__(self,name_input,country_input):
    self.name = name_input
    self.country = country_input

p1 = Person('nitish','india')
p2 = Person('steve','australia')
```

```
p2.name
```

```
'steve'
```

```
class Atm:

  # constructor(special function)->superpower ->
  def __init__(self):
    print(id(self))
    self.pin = ''
    self.__balance = 0
    #self.menu()

  def get_balance(self):
    return self.__balance

  def set_balance(self,new_value):
    if type(new_value) == int:
      self.__balance = new_value
    else:
      print('beta bahot maarenge')

  def __menu(self):
    user_input = input("""
    Hi how can I help you?
    1. Press 1 to create pin
    2. Press 2 to change pin
    3. Press 3 to check balance
    4. Press 4 to withdraw
    5. Anything else to exit
    """)

    if user_input == '1':
      self.create_pin()
    elif user_input == '2':
      self.change_pin()
    elif user_input == '3':
      self.check_balance()
    elif user_input == '4':
      self.withdraw()
    else:
      exit()

  def create_pin(self):
    user_pin = input('enter your pin')
    self.pin = user_pin

    user_balance = int(input('enter balance'))
    self.__balance = user_balance

    print('pin created successfully')

  def change_pin(self):
    old_pin = input('enter old pin')

    if old_pin == self.pin:
      # let him change the pin
      new_pin = input('enter new pin')
      self.pin = new_pin
      print('pin change successful')
    else:
      print('nai karne de sakta re baba')

  def check_balance(self):
    user_pin = input('enter your pin')
    if user_pin == self.pin:
      print('your balance is ',self.__balance)
    else:
      print('chal nikal yahan se')

  def withdraw(self):
    user_pin = input('enter the pin')
    if user_pin == self.pin:
      # allow to withdraw
      amount = int(input('enter the amount'))
      if amount <= self.__balance:
        self.__balance = self.__balance - amount
        print('withdrawl successful.balance is',self.__balance)
      else:
        print('abe garib')
```

```
    else:
      print('sale chor')


obj = Atm()
```

```
 140655538526416
```

```
obj.get_balance()
```

```
 1000
```

```
obj.set_balance(1000)
```

```
obj.withdraw()
```

```
 enter the pin
 enter the amount5000
 ---------------------------------------------------------------------------
 TypeError                                 Traceback (most recent call last)
 <ipython-input-93-826ea677aa70> in <module>
 ----> 1 obj.withdraw()

 <ipython-input-86-f5bffac7e2a0> in withdraw(self)
      67       # allow to withdraw
      68       amount = int(input('enter the amount'))
 ---> 69       if amount <= self.__balance:
      70         self.__balance = self.__balance - amount
      71         print('withdrawl successful.balance is',self.__balance)

 TypeError: '<=' not supported between instances of 'int' and 'str'
```

## Collection of objects

```
Start coding or generate with AI.
```

```
# list of objects
class Person:

  def __init__(self,name,gender):
    self.name = name
    self.gender = gender

p1 = Person('nitish','male')
p2 = Person('ankit','male')
p3 = Person('ankita','female')

L = [p1,p2,p3]

for i in L:
  print(i.name,i.gender)
```

```
 nitish male
 ankit male
 ankita female
```

```
# dict of objects
# list of objects
class Person:

  def __init__(self,name,gender):
    self.name = name
    self.gender = gender

p1 = Person('nitish','male')
p2 = Person('ankit','male')
p3 = Person('ankita','female')

d = {'p1':p1,'p2':p2,'p3':p3}

for i in d:
  print(d[i].gender)
```

```
male
male
female
```

## Static Variables(Vs Instance variables)

```
# need for static vars
```

```
class Atm:

  __counter = 1

  # constructor(special function)->superpower ->
  def __init__(self):
    print(id(self))
    self.pin = ''
    self.__balance = 0
    self.cid = Atm.__counter
    Atm.__counter = Atm.__counter + 1
    #self.menu()

  # utility functions
  @staticmethod
  def get_counter():
    return Atm.__counter


  def get_balance(self):
    return self.__balance

  def set_balance(self,new_value):
    if type(new_value) == int:
      self.__balance = new_value
    else:
      print('beta bahot maarenge')

  def __menu(self):
    user_input = input("""
    Hi how can I help you?
    1. Press 1 to create pin
    2. Press 2 to change pin
    3. Press 3 to check balance
    4. Press 4 to withdraw
    5. Anything else to exit
    """)

    if user_input == '1':
      
c1 = Atm()
```