

```
L = [1,2,3]
```

```
L.upper()
```

```
-----  
-----  
AttributeError                                Traceback (most recent call  
last)
```

```
<ipython-input-1-af1f83522ab7> in <module>
```

```
1 L = [1,2,3]
```

```
2
```

```
----> 3 L.upper()
```

```
AttributeError: 'list' object has no attribute 'upper'
```

```
s = 'hello'
```

```
s.append('x')
```

```
-----  
-----  
AttributeError                                Traceback (most recent call  
last)
```

```
<ipython-input-2-2cb7c5babec0> in <module>
```

```
1 s = 'hello'
```

```
----> 2 s.append('x')
```

```
AttributeError: 'str' object has no attribute 'append'
```

```
L = [1,2,3]
```

```
print(type(L))
```

```
<class 'list'>
```

```
s = [1,2,3]
```

```
# syntax to create an object
```

```
#objectname = classname()
```

```
# object literal
```

```
L = [1,2,3]
```

```
L = list()
```

```
L
```

```
[]
```

```
s = str()
```

```
s
```

```
{"type": "string"}
```

```
# Pascal Case
```

```
HelloWorld
```

```
class Atm:
```

```
    # constructor(special function)->superpower ->
```

```
    def __init__(self):
```

```
        print(id(self))
```

```
        self.pin = ''
```

```
        self.balance = 0
```

```
        #self.menu()
```

```
    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')
```

```
        self.menu()
```

```
    def change_pin():
```

```
        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')
        self.menu()
    else:
        print('nai karne de sakta re baba')
        self.menu()

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')
    self.menu()

```

```
obj1 = Atm()
```

```
140289660099024
```

```
id(obj1)
```

```
140289660099024
```

```
obj2 = Atm()
```

```
140289660586384
```

```
id(obj2)
```

```
140289660586384
```

```
L = [1,2,3]
```

```
len(L) # function ->bcos it is outside the list class
```

```
L.append()# method -> bcos it is inside the list class
```

```
class Temp:
```

```
def __init__(self):  
    print('hello')
```

```
obj = Temp()
```

```
hello
```

```
3/4*1/2
```

```
0.375
```

```
class Fraction:
```

```
    # parameterized constructor
```

```
    def __init__(self,x,y):  
        self.num = x  
        self.den = y
```

```
    def __str__(self):  
        return '{}/{ {}'.format(self.num,self.den)
```

```
    def __add__(self,other):  
        new_num = self.num*other.den + other.num*self.den  
        new_den = self.den*other.den  
  
        return '{}/{ {}'.format(new_num,new_den)
```

```
    def __sub__(self,other):  
        new_num = self.num*other.den - other.num*self.den  
        new_den = self.den*other.den  
  
        return '{}/{ {}'.format(new_num,new_den)
```

```
    def __mul__(self,other):  
        new_num = self.num*other.num  
        new_den = self.den*other.den  
  
        return '{}/{ {}'.format(new_num,new_den)
```

```
    def __truediv__(self,other):  
        new_num = self.num*other.den  
        new_den = self.den*other.num  
  
        return '{}/{ {}'.format(new_num,new_den)
```

```
    def convert_to_decimal(self):  
        return self.num/self.den
```

```
fr1 = Fraction(3,4)
fr2 = Fraction(1,2)

fr1.convert_to_decimal()
# 3/4
```

0.75

```
print(fr1 + fr2)
print(fr1 - fr2)
print(fr1 * fr2)
print(fr1 / fr2)
```

10/8
2/8
3/8
6/4

```
s1={1,2,3}
s2={3,4,5}
```

s1 + s2

```
-----
-----
TypeError                                Traceback (most recent call
last)
<ipython-input-32-3a417afc75fb> in <module>
      2 s2={3,4,5}
      3
----> 4 s1 + s2
```

TypeError: unsupported operand type(s) for +: 'set' and 'set'

```
print(fr1 - fr2)
```

```
-----
-----
TypeError                                Traceback (most recent call
last)
<ipython-input-39-929bcd8b32dc> in <module>
----> 1 print(fr1 - fr2)
```

TypeError: unsupported operand type(s) for -: 'Fraction' and
'Fraction'