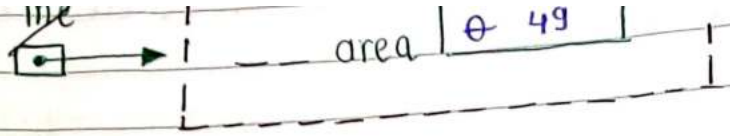


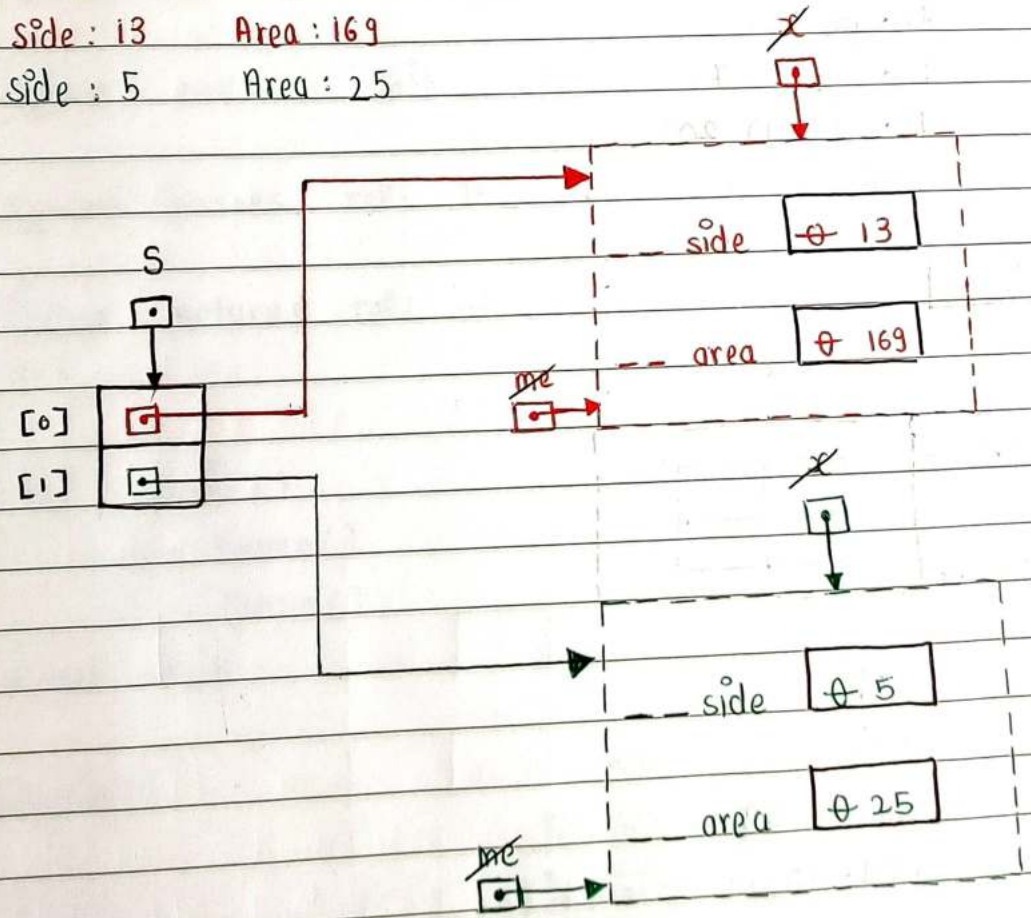
P308.py

```
class Square :
    __side = 0
    __area = 0
    def input (me):
        print ('enter side :', end = ' ')
        me.__side = int (input ())
        me.__area = me.__side * * 2
    # end : input
    def output (me):
        print (f' side : {me.__side} \t'
               f' Area : {me.__area} ')
    # end : output
#end : square
def main ():
    s = [None] * 2
```

```
        for i in range(len(s)):
            s[i] = Square()
        # end: for
        for x in s:
            x.input()
        # end: for
        for x in s:
            x.output()
        # end: for
# end: main
```

main()

enter side : 13 ↵
enter side : 5 ↵
Side : 13 Area : 169
side : 5 Area : 25



DATE

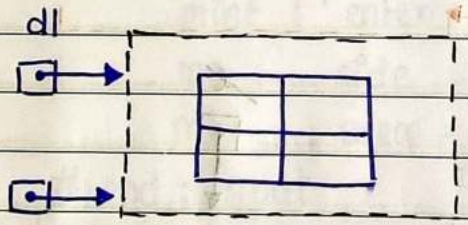
P309.py

```
class Demo:
    def greet(me):
        print('Hello')
    # end: greet
    def square(me, x):
        print(f'square: {x**2}')
    # end: square
    def add(me, a, b):
        print(f'add: {a+b}')
    # end: add
# end: Demo
def main():
    d1 = Demo()
    d1.greet()
    d1.square(7)
    d1.add(10, 20)
# end: main
```

x / 7

a / 10 b / 20

main()



o/p

Hello
Square: 49
add: 30

CLASSMATE

PAGE

DATE

ps10.py

```
class Demo:
    def __init__(me):
        print('in init class Demo')
    #end: init
# end: Demo
'''
object = ClassName()
```

e.g. d1 = Demo()

- object is created i.e. memory is allocated
- reference (ref.) of object is created
- Python calls __init__ of class Demo.
- Python passes ref. to __init__ fn
- Python returns ref. at place of object creation

'''

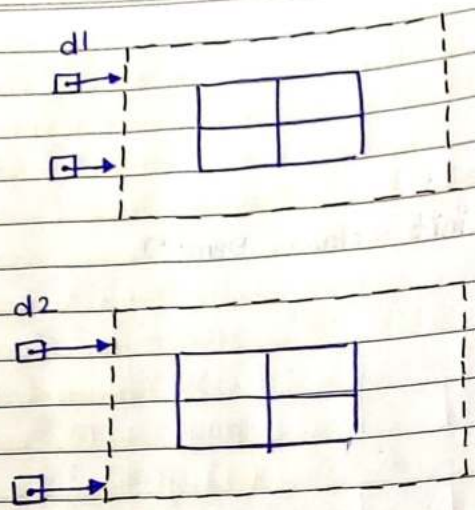
```
def main():
    d1 = Demo()
    d2 = Demo()
#end: main
```

main()

o/p

In init class Demo
In init class Demo

DATE



magic fn / methods

↓

__functionName()__

__int__

(initialise)

↓

Constructor

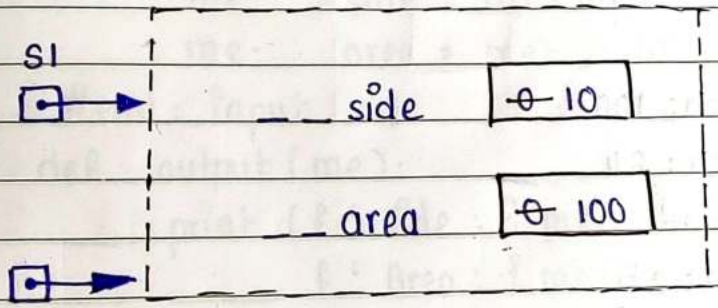
P311.py

```
class Square:
    def __init__(me):
        me.__side = 0
        me.__area = 0
    #end: int
    def input(me):
        print('enter side :')
        me.__side = int(input())
        me.__area = me.__side**2
    #end: input
```

```
    def output(me):
        print(f'side : {me.__side}\t'
              f'Area : {me.__area}')
    # end : output
# end: square
def main():
    S1 = square()
    S1.input()
    S1.output()
# end : main

main()
```

enter side :
10 ↵
side : 10 Area : 100



P312.py

side
10

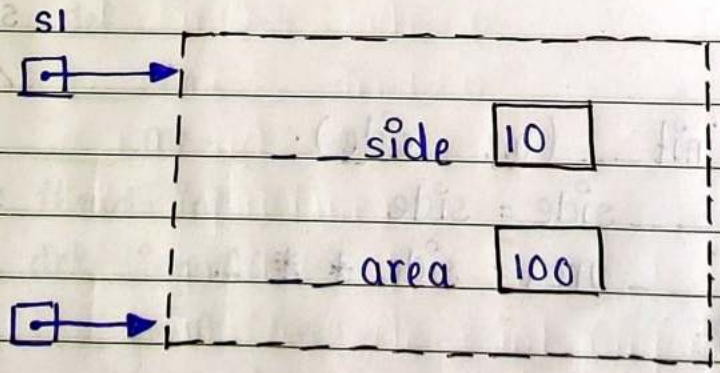
side
8

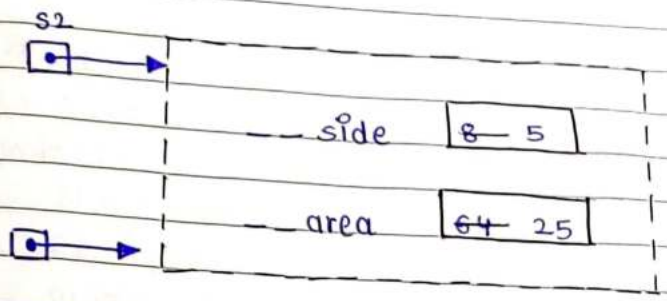
```
class Square:
    def __init__(me, side):
        me.__side = side
        me.__area = side ** 2
    # end: init
    def input(me):
        print('enter side :')
        me.__side = int(input())
```

```
        me.__area = me.__side ** 2
    #end: input
    def output(me):
        print(f'side: {me.__side}\t'
              f'Area: {me.__area}')
    #end: output
#end: square
def main():
    s1 = square(10)
    s1.output()
    s2 = Square(8)
    s2.output()
    s2.input()
    s2.output()
#end: main
```

main()

```
side : 10    Area: 100
side : 8     Area: 64
enter side : 5 ↵
side : 5     Area: 25
```





P313.py

```
class Square :
    def __init__(me, side = 0):
        me.__side = side
        me.__area = side ** 2
    #end: int
    def input(me):
        print('enter side :')
        me.__side = int(input())
        me.__area = me.__side ** 2
    #end: input
    def output(me):
        print(f'side : {me.__side} \t'
              f'Area : {me.__area}')
    #end: output
#end: Square
def main():
    S1 = Square()
    S1.output()
    S2 = square(8)
    S2.output()
    S1.input()
    S1.output()
# end: main
```

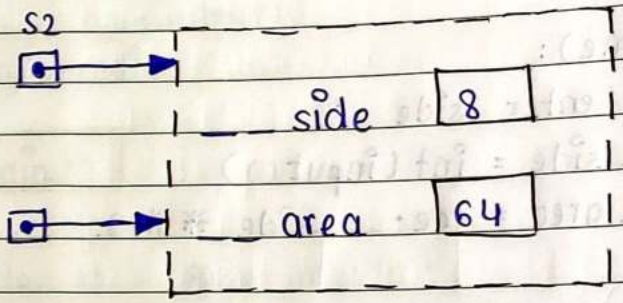
side 0

side 8

main()

DATE

```
%p
side : 0      Area : 0
side : 8      Area : 64
  :
  :
```



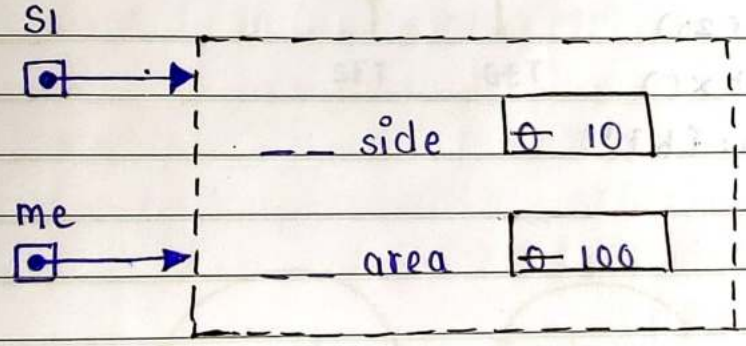
p314.py

```
class Square :
    def __init__(me):
        me.__side = 0
        me.__area = 0
    #end: int
    def input(me):
        print('enter side :')
        me.__side = int(input())
    #end: input
    def __str__(me):
        return(f'side : {me.__side} \t'
```

DATE

```
        f'Area : {me.__area}')
    #end : str
# end : Square
def main():                                    msg
    s1 = Square()                              'side : 10 \t Area : 100'
    s1.input()
    msg = str(s1)
    # msg = s1.__str__()
    print(msg)
    print(s1)
        ↓
    # print(s1.__str__())
    print(f'S1 obj {s1}')
        ↓
    # print(f'S1 obj {s1.__str__()}')
# end: main

main()
```




```
enter side :
10 ↵
side : 10     Area : 100
side : 10     Area : 100
S1 obj  side : 10  Area : 100
```


PAGE

P315.py

```
class Data:
    def __init__(me):
        me.__x = 0
    #end: init
    def setx(me, x):
        me.__x = x
    #end: set x
    def getx(me):
        return me.__x
    #end: getx
#end: Data
def main():
    d1 = Data()
    d1.setx(10)
    a = d1.getx()
    print(f'a: {a}')
    d2 = Data()
    d2.setx(20)
    b = d2.getx()
    print(f'b: {b}')
#end: main

main()
```

x / 10

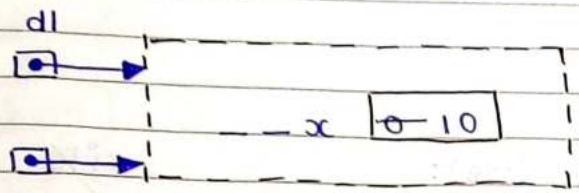
x / 20

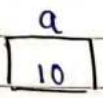
o/p

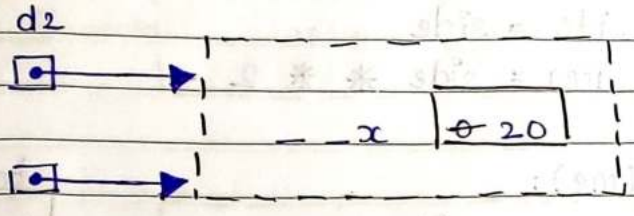
a : 10

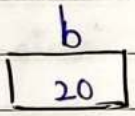
b : 20

DATE

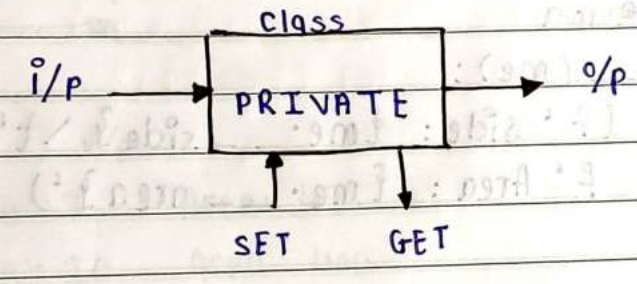


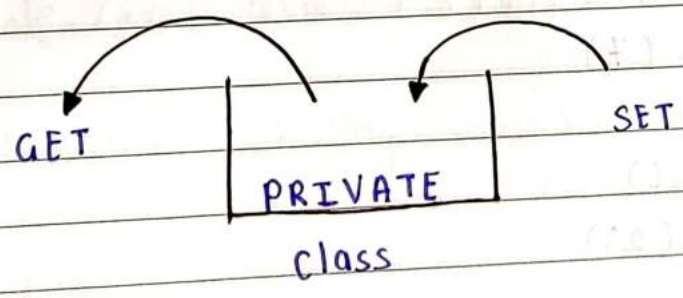






*





P316.py

```
class Square:
    def __init__(me):
        me.__side = 0
        me.__area = 0
    #end: init
    def setside(me, side):
        me.__side = side
        me.__area = side ** 2
    # end: setside
    def getside(me):
        return me.__side
    # end: getside
    def getarea(me):
        return me.__area
    # end: getarea
    def __str__(me):
        return (f'side: {me.__side}\t'
                f'Area: {me.__area}')
    #end: str
# end: square
def main():
    s1 = Square()
    s1.setside(7)
    print(s1)
    s2 = square()
    s2.setside(20)
    print(s2)
    ta = s1.getArea() + s2.getArea()
    print(f'Total Area: {ta}')
    print(f's1 obj. side: {s1.getside()}')
    print(f's2 obj. area: {s2.getArea()}')
```

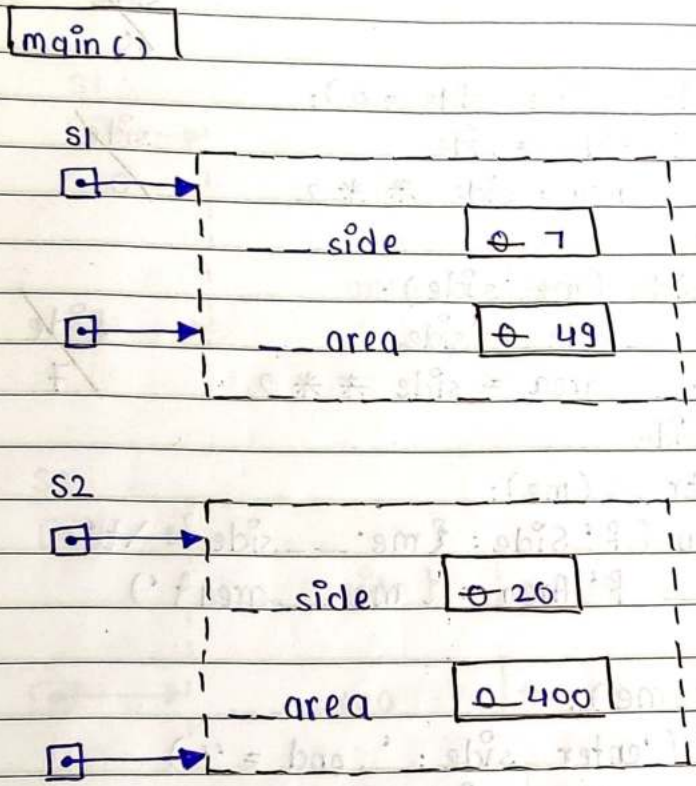
side 7

side 20

ta 449

DATE

end: main



o/p

Side : 7 Area : 49
Side : 20 Area : 400
Total Area : 449
S1 obj. side : 7
S2 obj. area : 400

* init : inihalistion (starting value)

set : change

DATE

P317.PY

side
10

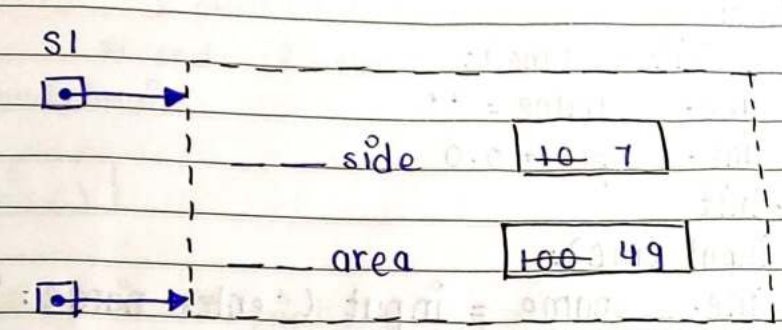
```
class Square:
    def __init__(me, side = 0):
        me.__side = side
        me.__area = side ** 2
    # end: init
    def setside(me, side):
        me.__side = side
        me.__area = side ** 2
    #end: setside
    def __str__(me):
        return (f'Side: {me.__side} \t'
                f'Area: {me.__area}')
    # end: str
    def input(me):
        print('enter side: ', end = '')
        me.__side = int(input())
        me.__area = me.__side ** 2
    # end: input
#end: Square
def main():
    s1 = Square(10)
    print(s1) # print(s1.__str__())
    s1.setside(7)
    print(s1)
    s2 = Square()
    print(s2)
    s2.input()
    print(s2)
    s2.setside(5)
    print(s2)
#end: main
```

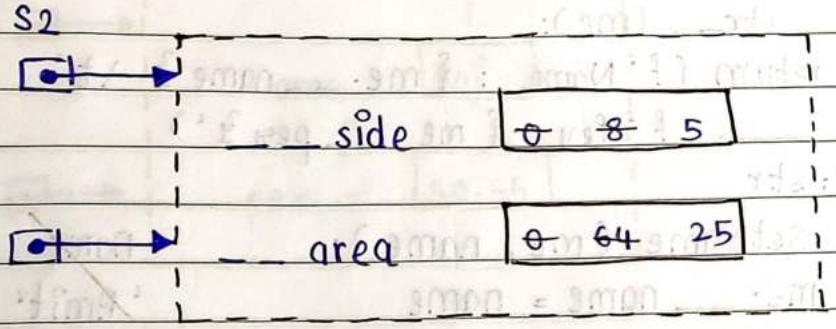
side
0

side
7

DATE

main ()





o/p

side : 10 Area : 100
side : 7 Area : 49
side : 0 Area : 0
enter side : 8 ↵
side : 8 Area : 64
side : 5 Area : 25

P318.py

```
class student:
    def __init__(me):
        me.__name = ''
        me.__per = 0.0
    #end: init
    def input(me):
        me.__name = input('enter name:')
        me.__per = float(input('enter per:'))
    # end: input
    def __str__(me):
        return (f'Name : {me.__name} \t'
                f'Per : {me.__per}')
    #end: str
    def setName(me, name):
        me.__name = name
    # end: setName
    def setPer(me, name):
        me.__per = per
    #end: setPer
    def getName(me): return me.__name
    def getPer(me): return me.__per
#end: student
def main():
    s1 = student()
    s1.input()
    print(s1) # print(s1.__str__())
    s2 = student()
    s2.setName('Amit')
    s2.setPer(70.77)
    print(s2)
    ff s1.getPer() > s2.getPer():
```

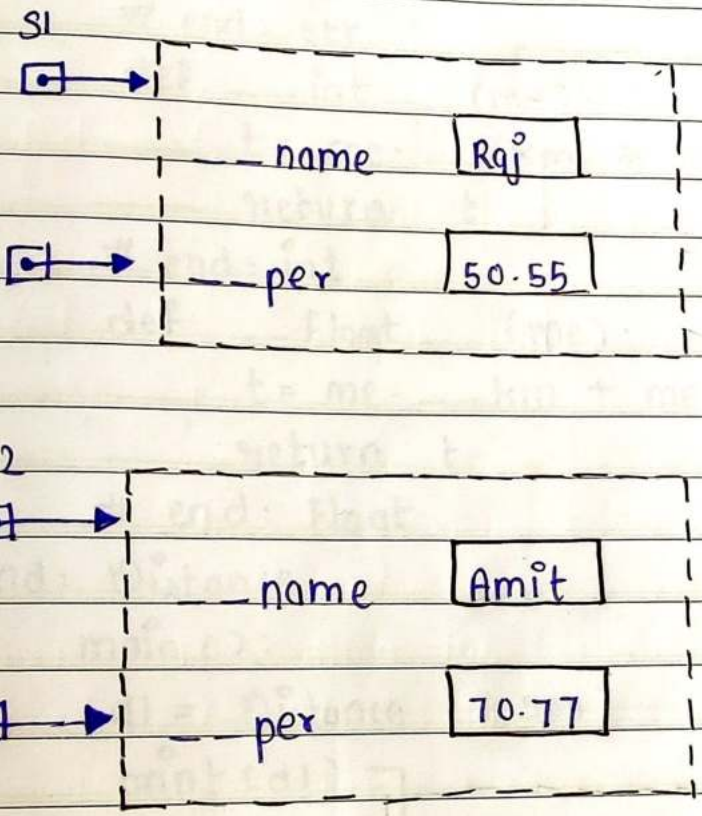
name
~~'Amit'~~

Per
~~70.77~~

```
        print(f'{s1.getName()} has higher %')
    else:
        print(f'{s2.getName()} has higher %')
    # end: if-else
# end: main

main()
```

solve in 1 print



enter name : Raj ↵
enter per : 50.55 ↵
Name : Raj Per : 50.55
Name : Amit Per : 70.77
Amit has higher %

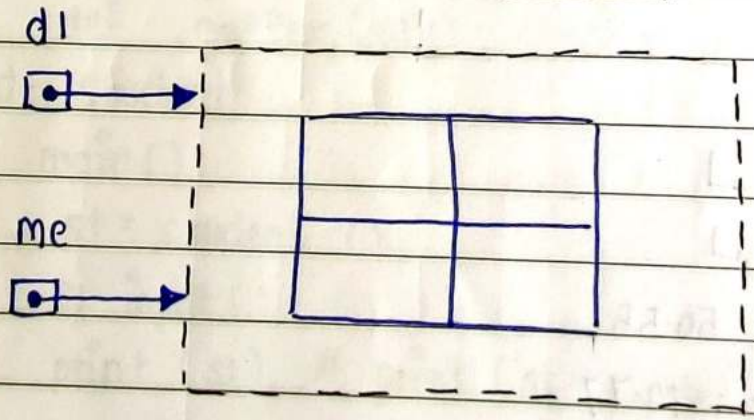
P319.py

```
Class  Demo:
    def  greet (me):
        print ('Hello ', end = '')
        me.name()
    # end: greet
    def  name(me):
        print ('Raj')
    # end: name
# end: Demo
def  main():
    dl = Demo()
    dl.greet()
# end : main

main()
```

%p

Hello Raj



DATE

P320.py

```
class Distance :
    def __init__(me, km, m):
        me.__km = km
        me.__m = m
    # end: init
    def __str__(me):
        return (f'KM: {me.__km} \n'
                f'M: {me.__m}')
    # end: str
    def __int__(me):
        t = me.__km * 1000 + me.__m
        return t
    # end: int
    def __float__(me):
        t = me.__km + me.__m / 1000
        return t
    # end: float
# end: Distance
def main():
    d1 = Distance(7,500)
    print(d1)

    # print(d1.__str__())
    x = int(d1)

    # x = d1.__int__()
    print(f'Meters: {x}')
    y = float(d1)

    # y = d1.__float__()
    print(f'KM: {y}')
```

km / 7

m / 500

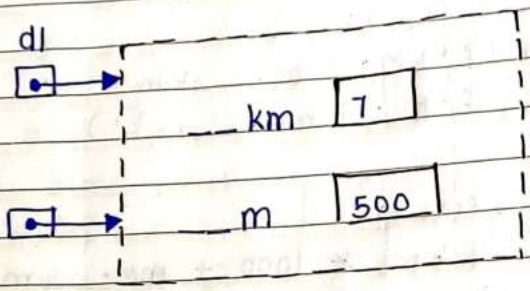
x
7500

y
7.5

7.5

print (f' Kilometers : {y} ')
end : main

main ()



%p

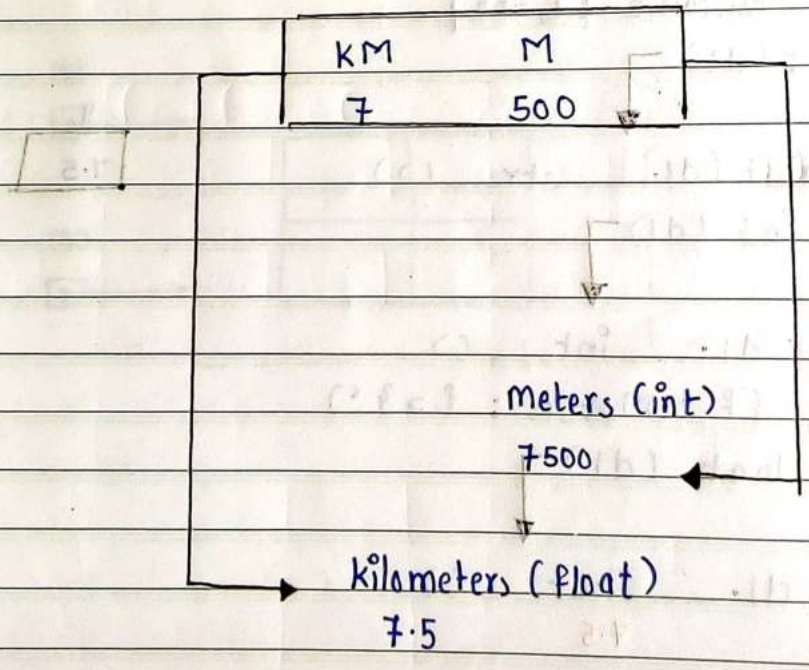
KM : 7

M : 500

Meters : 7500

Kilometers : 7.5

class Distance



DATE

* Conversion of class to basic datatype

class → str

class → int

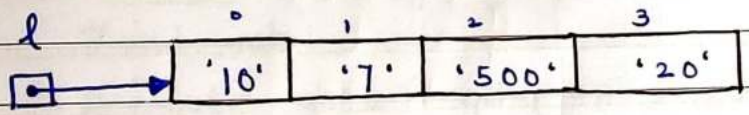
class → float

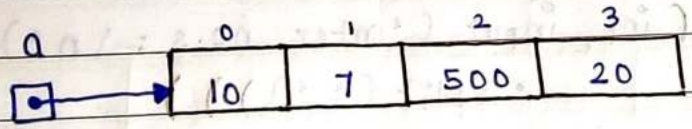
P321.py

```
print ('enter no.s :')
s = input()
l = s.split(' ')
a = list(map(int, l))
s = sum(a)
print(f'sum: {s}')
```

s

'10 7 500 20'





enter no.s :
10 7 500 20 ↵
sum: 537

PAGE

DATE

P322.py

```
s = input ('enter no.s : \n')
l = s.split (' ')
a = list (map (int, l))
s = sum (a)
print (f 'sum: {s}')
```

P323.py

```
l = input ('enter no.s : \n').split (' ')
a = list (map (int, l))
s = sum (a)
print (f 'sum: {s}')
```

P324.py

```
a = list (map (int, input ('enter no.s : \n').split(' ')))
sb = sum (a)
print (f 'sum: {s}')
```

P325.py

```
s = sum (list (map (int, input ('enter no.s : \n')
                   .split (' '))))

print (f 'sum: {s}')
```

DATE

P326.py

```
print (f 'sum : {sum (list (map ( int , input ("enter
                    no.s:\n") . split (" ")))) }')
```

P327.py

```
class Point :
    def __init__ (me):
        me.__x = 0
        me.__y = 0
    # end: init
    def __str__ (me):
        return (f'x : {me.__x} \t'
                f'y : {me.__y }')
    #end: str
    def set (me, x, y):
        me.__x = x
        me.__y = y
    #end: set
    # input getx gety
    def getx(me):
        return me.__x
    #end: getx
    def gety (me):
        return me.__y
    #end: gety
    def __str__ (me):
        return (f 'x : {me.__x} \t'
                f 'y : {me.__y }')
    #end: str
```

o/p

x : 10 y : 20
x : 5 y : 6

x/10 y/20

①

PAGE

DATE

```
    def __add__(me, t):
        r = Point()
        r.__x = me.__x + t.__x
        r.__y = me.__y + t.__y
        return r
    # end: add
    def __sub__(me, t):
        r = Point()
        r.__x = me.__x - t.__x
        r.__y = me.__y - t.__y
        return r
    # end: sub
    """
    + __add__
    * __mul__
    % __mod__
    // __floordiv__
    - __sub__
    ** __pow__
    / __truediv__
    """
    def __neg__(me):
        r = Point()
        r.__x = -me.__x
        r.__y = -me.__y
        return r
    # end: neg
    def __pos__(me):
        r = Point()
        r.__x = +me.__x
        r.__y = +me.__y
        return r
    # end: pos
```

p1 + p2 → binary op → two

\+ → __add__

o/p
x: 10 y: 20
x: 5 y: 6
x: 15 y: 26

a b c
10 20
c = a + b
a b c
10 20 30

\- → __sub__

o/p
x: 100 y: 200
x: 30 y: 40
x: 70 y: 160

def main

① ② ③ ④ ⑤

DATE

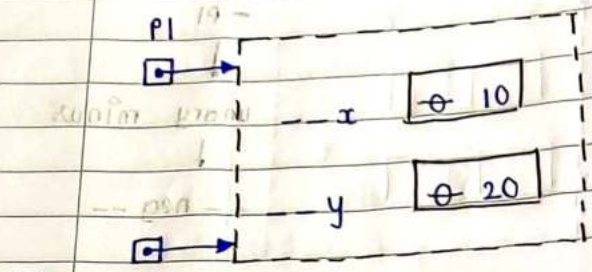
```
def main():
    p1 = Point()
    p1.set(10,20)
    print(p1)
    # print(p1.__str__())
    p2 = Point()
    p2.set(5,6)
①   print(p2)
    p3 = p1 + p2
    # p3 = p1.__add__(p2)
②   print(p1, p2, p3, sep='\n')
    p1.set(100,200)
    p2.set(30,40)
    p3 = p1 - p2
    # p3 = p1.__sub__(p2)
③   print(p1, p2, p3, sep='\n')
    p1.set(10,-20)
    p2 = -p1
    # p2 = p1.__neg__()
④   print(p1, p2, sep='\n')
    p1.set(10,-20)
    p2 = +p1
    # p2 = p1.__pos__()
⑤   print(p1, p2, sep='\n')
#end: main

main()
```

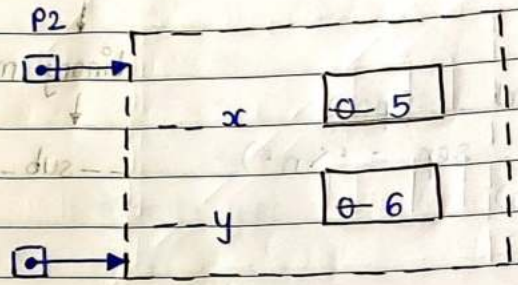
-p1 → unary minus → __neg__

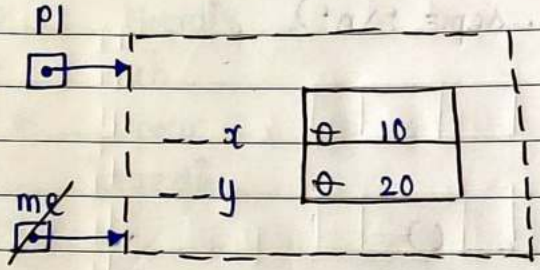
p1 - p2 → binary minus → __sub__

PAGE

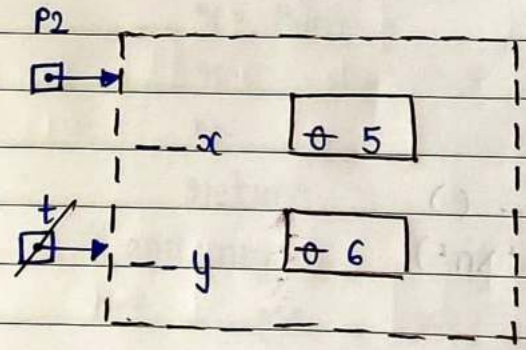


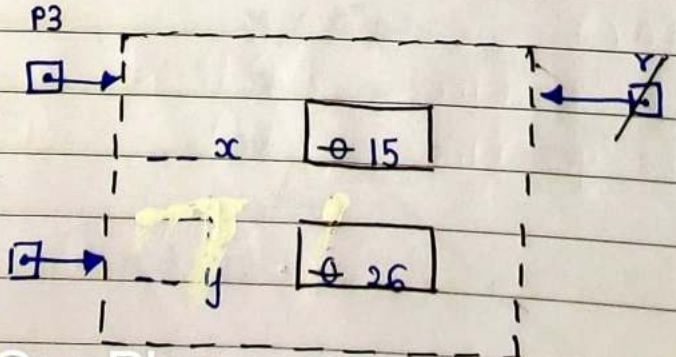
①





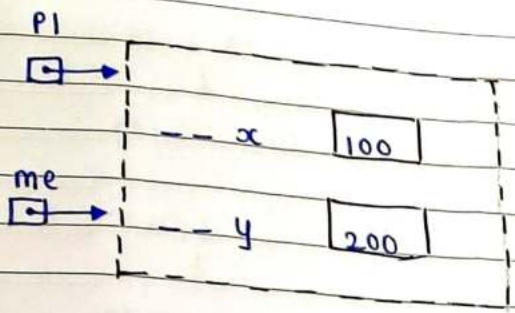
②

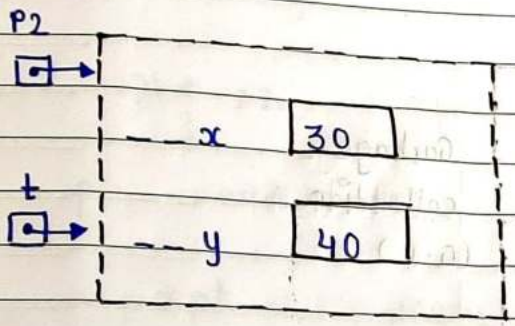




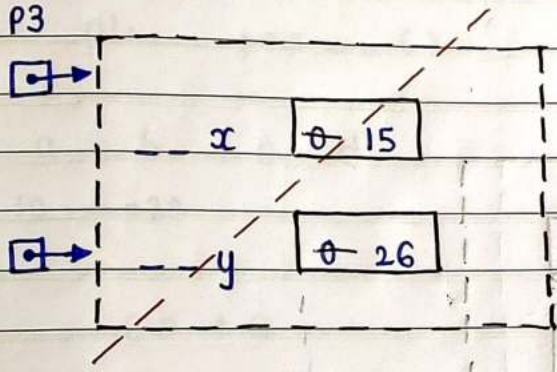
③

DATE

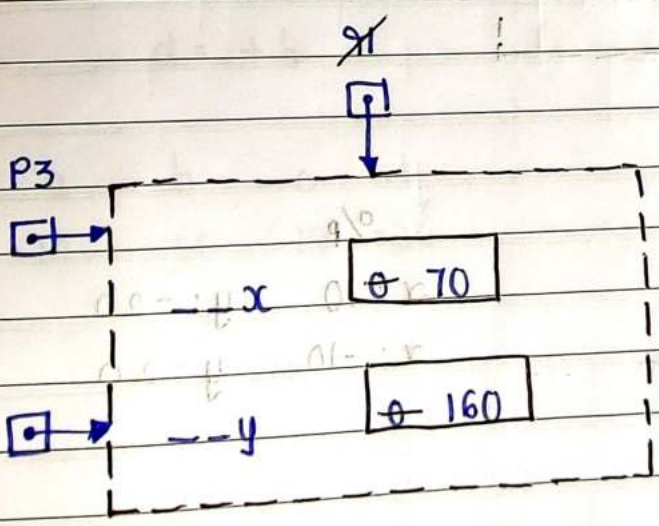




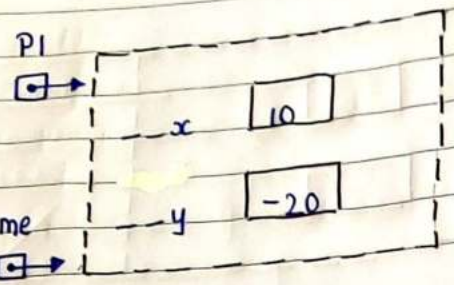
③

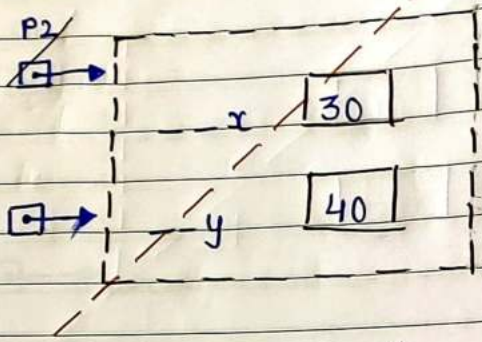


Python
G.C
garbage
collection



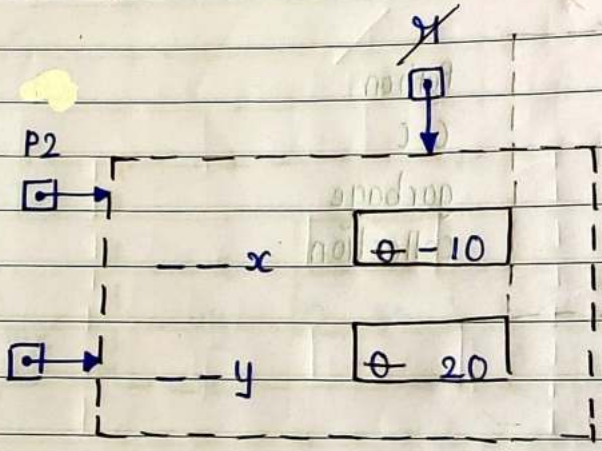
DATE





4

5



a	b	c	d
10	-20		

c = -a

d = -b

a	b	c	d
10	-20	-10	20

%p

x : 10 y : -20

x : -10 y : 20

DATE

p1 - p2 binary minus

↓

p1.__sub__(p2)

-p1 unary minus

↓

p1.__neg__()

p1 + p2 binary plus

↓

p1.__add__(p2)

(5)

+p1 unary plus

↓

p1.__pos__()

a	b	c	d
10	-20		

c = +a

d = +b

a	b	c	d
10	-20	10	-20

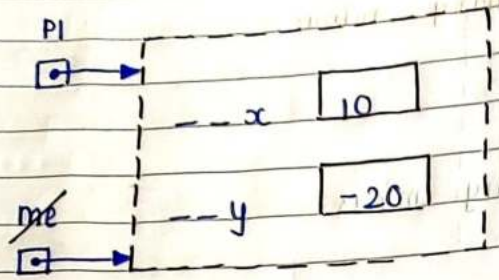
%p

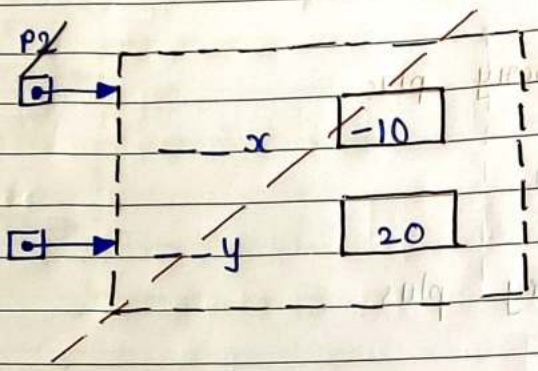
x : 10 y : -20

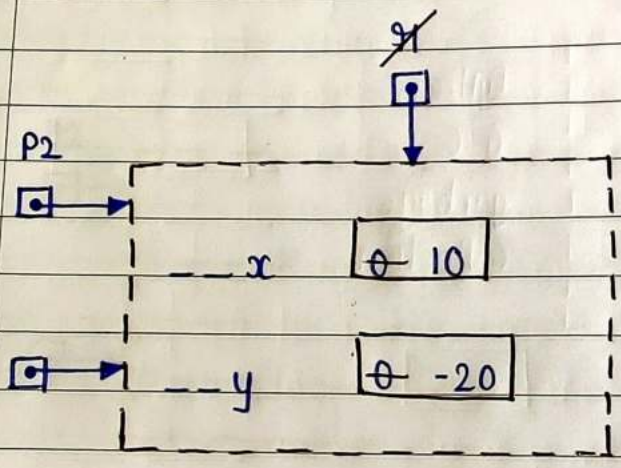
x : 10 y : -20

PAGE

DATE







1

DATE

P328.py

```
class Square:
    def __init__(me):
        me.__side = 0
        me.__area = 0
    # end: init
    def setside(me, side):
        me.__side = side
        me.__area = side ** 2
    # end: setside
    def __str__(me):
        return (f'side: {me.__side}\t'
                f'Area: {me.__area}')
    #end: str
    # input getside getArea
    def getside(me):
        return me.__side
    #end: getside
    def getArea(me):
        return me.__area
    # end: getArea
    def input(me):
        print('enter side:')
        me.__side = int(input())
        me.__area = me.__side**2
    #end: input
```

side
0

(1)

o/p

side : 10 Area: 100
side : 10 Area: 100
side : 20 Area: 400

[P.T.O]

DATE

```
(2)  def __eq__(me, t):
         return me.__area == t.__area
     # end: eq
"""
==   __eq__
!=   __ne__
>    __gt__
<    __lt__
>=   __ge__
<=   __le__
"""
```

o/p

```
side: 10   Area: 100
side: 10   Area: 100
side: 20   Area: 400
s1 eq. s2
s1 not eq s3
```

```
(3)  def __ge__(me, t):
         return me.__area >= t.__area
     # end: ge
     def __iadd__(me, v):
(4)      me.__side = me.__side + v
         me.__area = me.__side ** 2
         return me
     # end: iadd
     def __isub__(me, v):
(5)      me.__side = me.__side - v
         me.__area = me.__side ** 2
         return me
     # end: isub
"""
+=   __iadd__
-=   __isub__
*=   __imul__
%=   __imod__
**=  __ipow__
/=   __itruediv__
//=  __ifloordiv__
"""
```

o/p

```
side : 10   Area : 100
side : 10   Area : 100
side : 20   Area : 400
s1 eq. s2
s1 not eq s3
s1 gt. eq s2
s1 lt s3
```

def mai

(1)

(2)

(3)

DATE

```
def main():
    s1 = square()
    s1.setside(10)
    print(s1) # print(s1.__str__())
    s2, s3 = square(), square()
    s2.setside(10)
    s3.setside(20)
①   print(s2, s3, sep='\n')
    if s1 == s2:
        # s1.__eq__(s2)
        print('s1 eq. s2')
    else:
        print('s1 not eq. s2')
②   # end: if-else
    if s1 == s3:
        # s1.__eq__(s3)
        print('s1 eq. s3)
    else:
        print('s1 not eq s3')
    # end: if-else
    s1.setside(15)
    if s1 >= s2:
        # s1.__ge__(s2)
        print('s1 gt. eq. s2')
    else:
        print('s1 lt. s2')
③   # end: if-else
    if s1 >= s3:
        # s1.__ge__(s3)
```


PAGE

```
        print ('s1 gt eq s3')
    else:
        print ('s1 lt s3')
# end: ff-else
s1.setside(5)
s1 += 3
```

④ `# s1 = s1.__iadd__(3)`

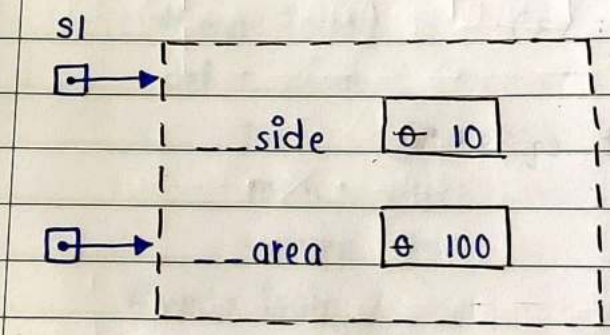
```
print(s1)
s2.setside(10)
print(s2)
s2 -= 4
```

`# s2 = s2.__isub__(4)`

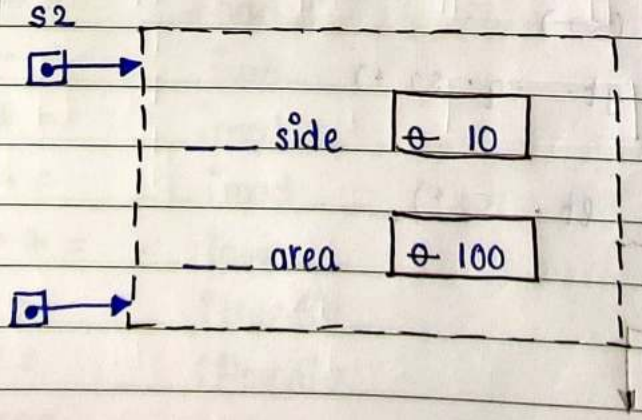
⑤ `print(s2)`

`# end: main`

`main()`

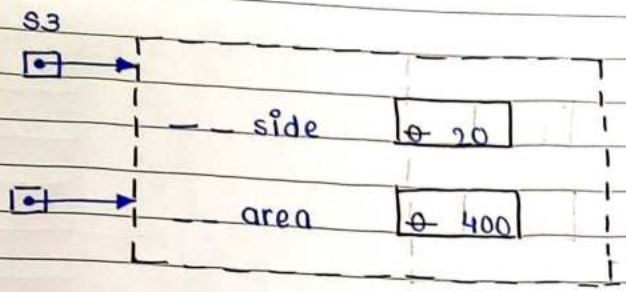


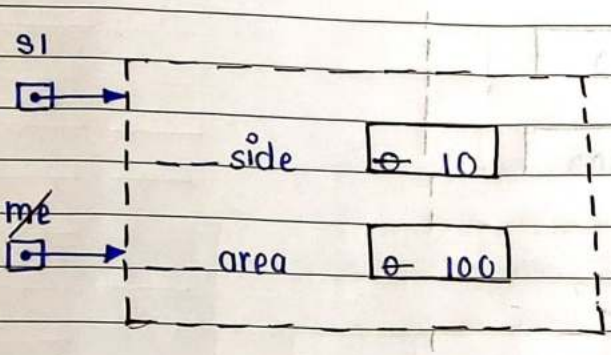
①



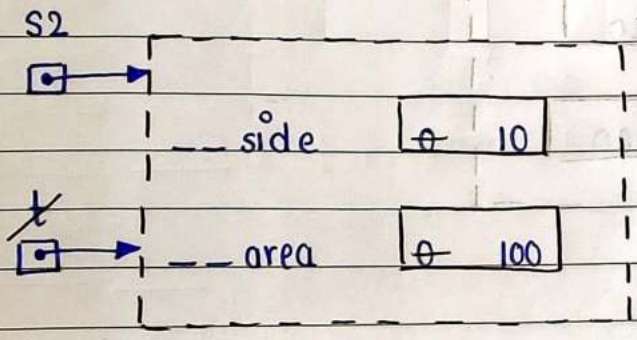
②

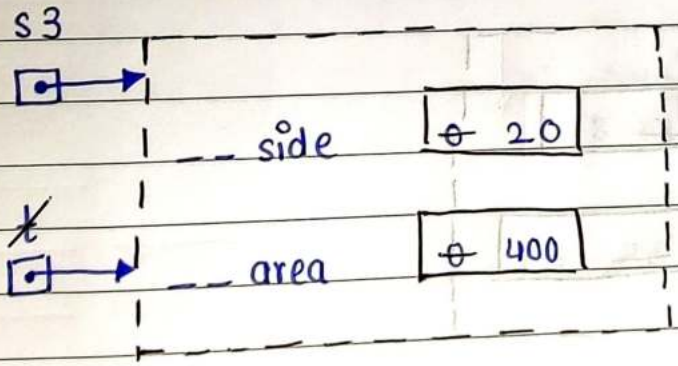
DATE





(2)

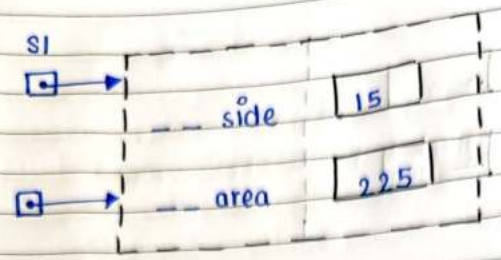


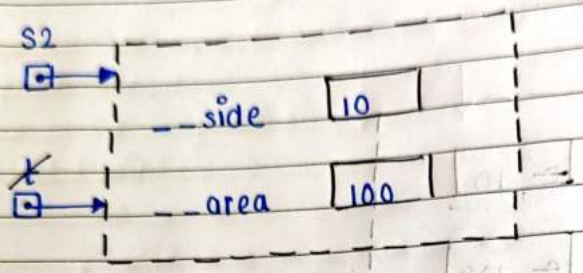


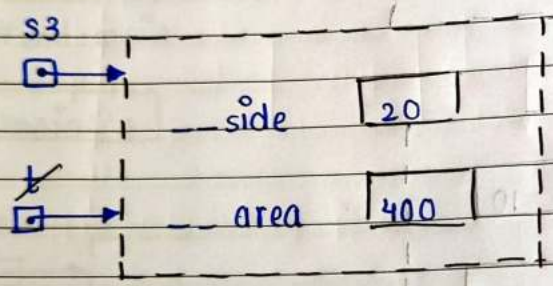
PAGE

DATE

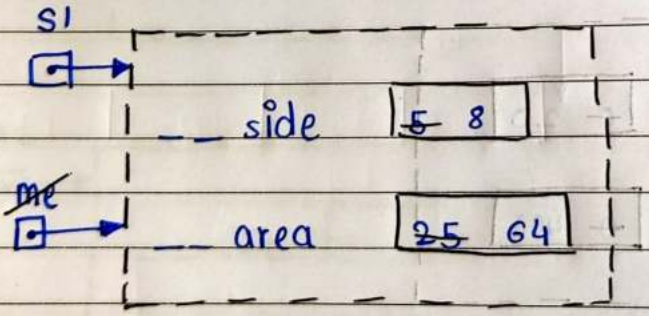
③







④



o/p

Side: 8 Area: 64

PAGE

DATE

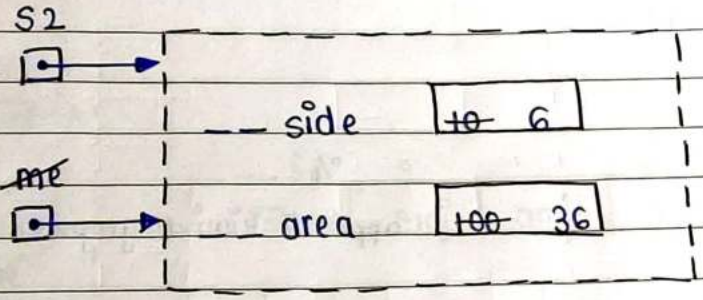
a	b
5	3

a += b

a	b
8	3

---

(5)



o/p

side : 10 Area : 100

side : 6 Area : 36

P329.py

```
print ('raj' in ['amit', 'raj', 'ravi'])

print ('joe' in ['amit', 'raj', 'ravi'])
```

o/p

True

False

DATE

P330.py

```
a = ['apple', 'kiwi', 'grapes']
print (a)
```

%p

['apple', 'kiwi', 'grapes']

```
s1 = '\n'.join(a)
print (s1)
s2 = '#'.join(a)
print (s2)
```

%p

apple
kiwi
grapes

%p

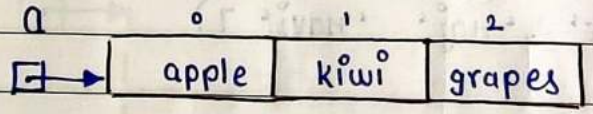
apple # kiwi # grapes

s1

'apple \n kiwi \ngrapes'

s2

'apple # kiwi # grapes'

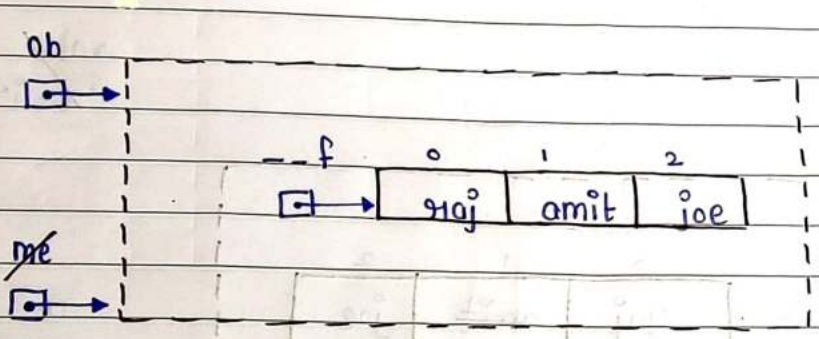


DATE

P331.py

```
class Friends:
    def __init__(me):
        me.__f = ['raj', 'amit', 'joe']
    # end: init
    def __str__(me):
        return '\n'.join(me.__f)
    # end: str
```

o/p
raj
amit
joe




```
    def __contains__(me, t):
        return t in me.__f
    # end: contains
    # 'raj' in me.__f (True)
    # 'ravi' in me.__f (False)
```

t t
'raj' 'ravi'

```
'''
in
|
|
__contains__
'''
```

o/p
raj is my friend
ravi is not my friend

PAGE

classmate

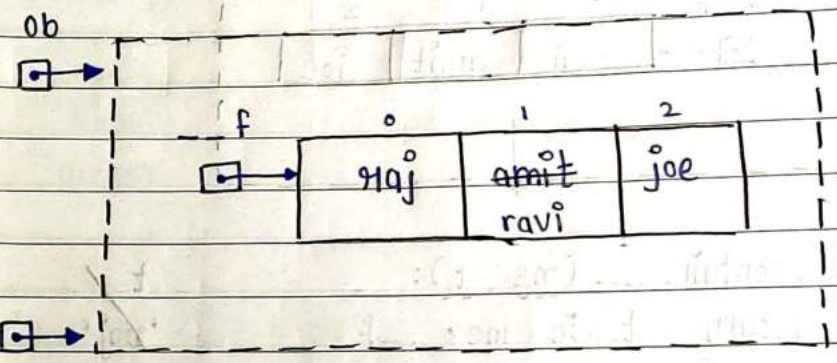
DATE

```
# ____ = []
# ___getitem___
def ___getitem___(me, index):
    return me.__f[index]
# end: getitem
# [] = ____
# ___setitem___
def ___setitem___(me, index, value):
    me.__f[index] = value
# end: setitem
```

index
1

index
1

value
'ravi'



x
'amit'

```
%p
x: amit
old list:
raj
amit
joe
new list:
raj
ravi
joe
```


16/08/22
4:07 p.m.

classmate

PAGE