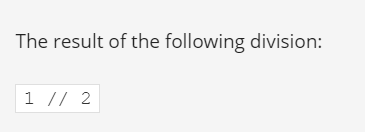
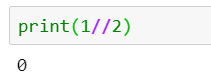
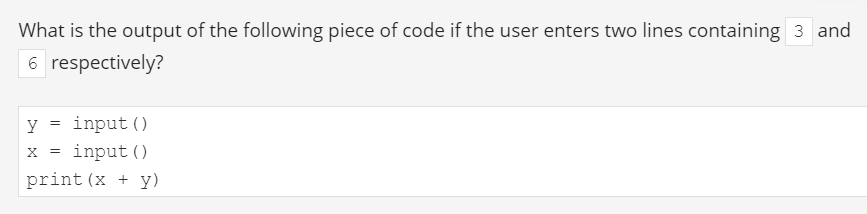
🡪



Ans:0

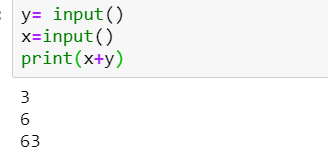


🡪

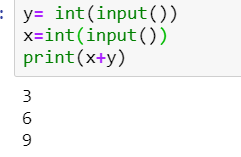


Ans: 63

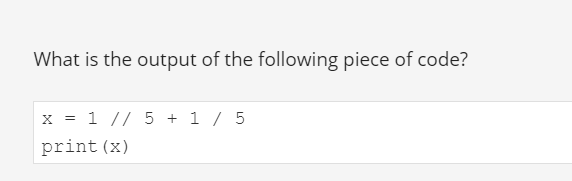
It wil treat 6 n 3 as string so + will do conact



We convert input into int then + will treated as addition



🡪



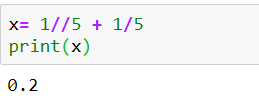
Ans:0.2

Explanation: // and / are high priority

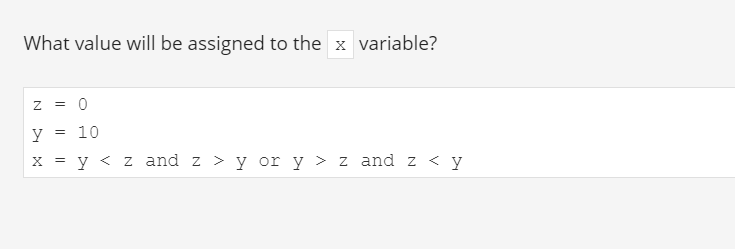
1//5 means zero

1/5 means 0.2

1//5 + 1/5 =0.2



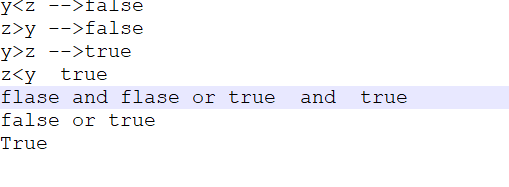
🡪



Ans : True

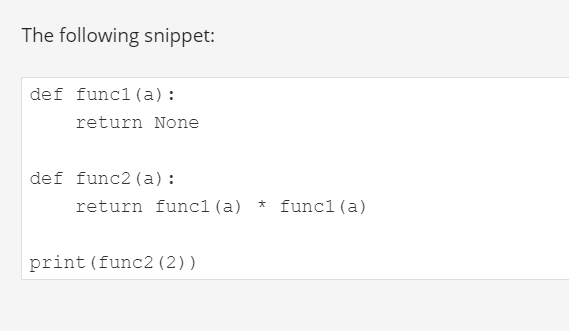
Comparing operators has high priority over logical operators so first we will do comparing then logical

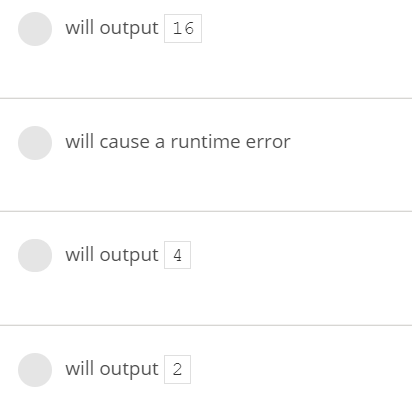
In logical operators ‘ and ‘has high priority compare to ‘ or ‘





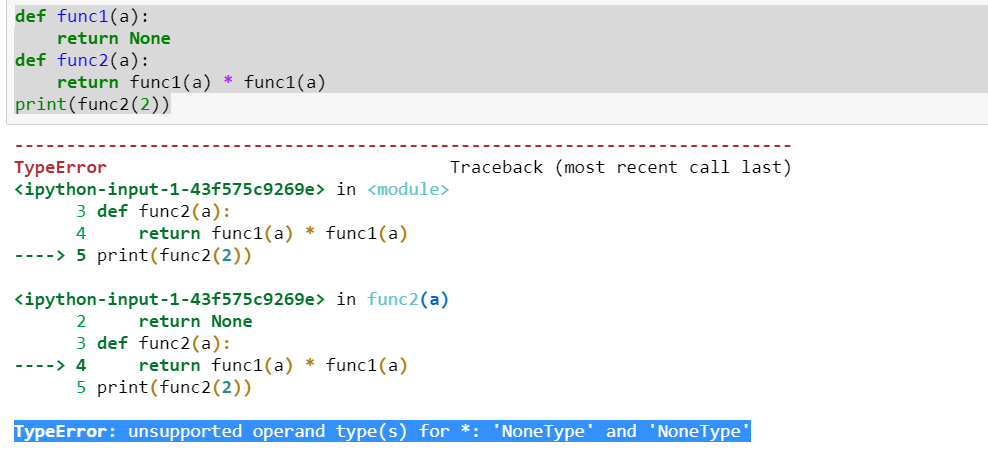
🡪



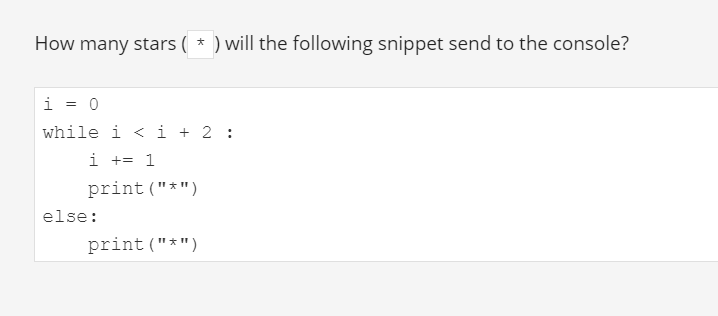


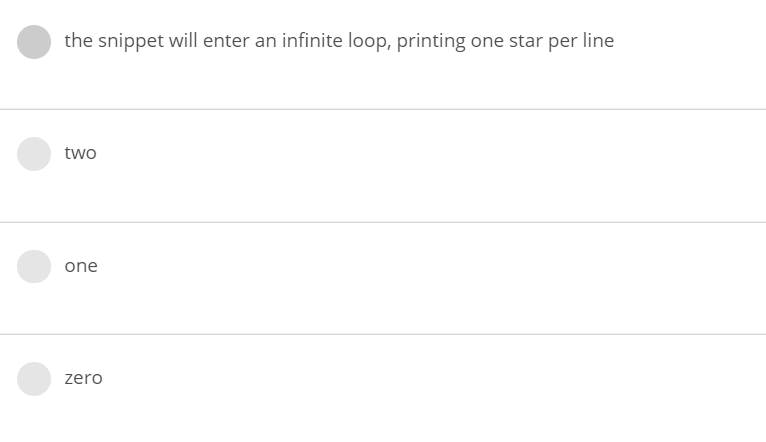
Ans: will cause a runtime error.

We will get error as **TypeError**: unsupported operand type(s) for \*: 'NoneType' and 'NoneType'



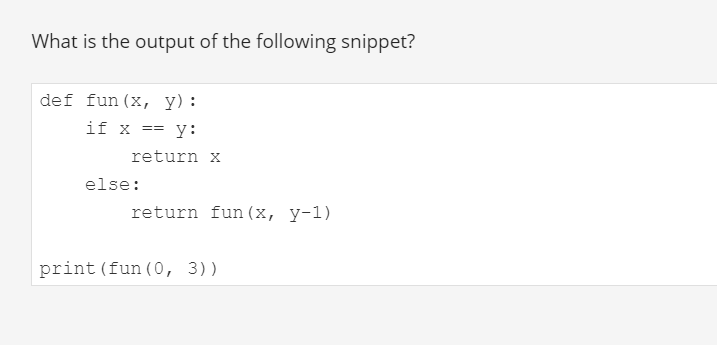
🡪

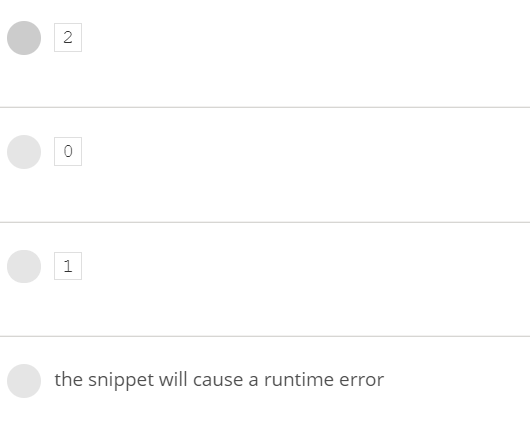




Ans: the snippet will enter an infinite loop, printing one star per line.

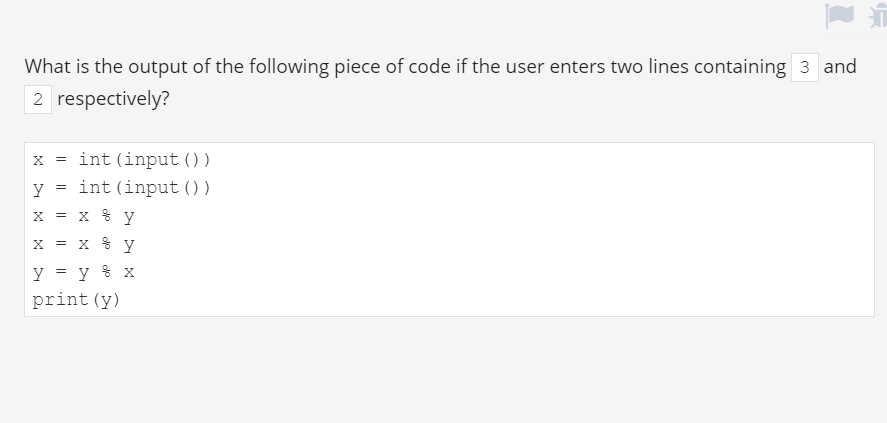
🡪





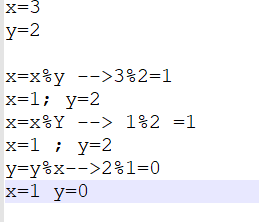
Ans: 0

🡪

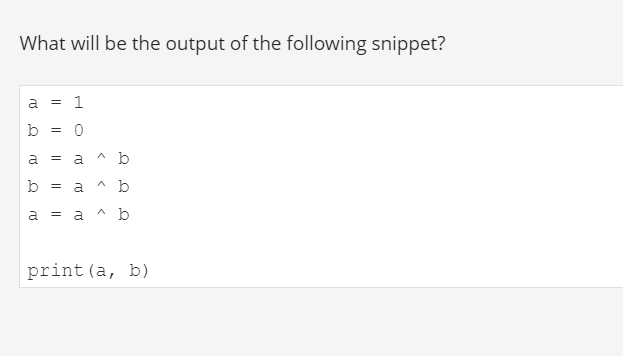


Ans: 0

Explanation:



🡪



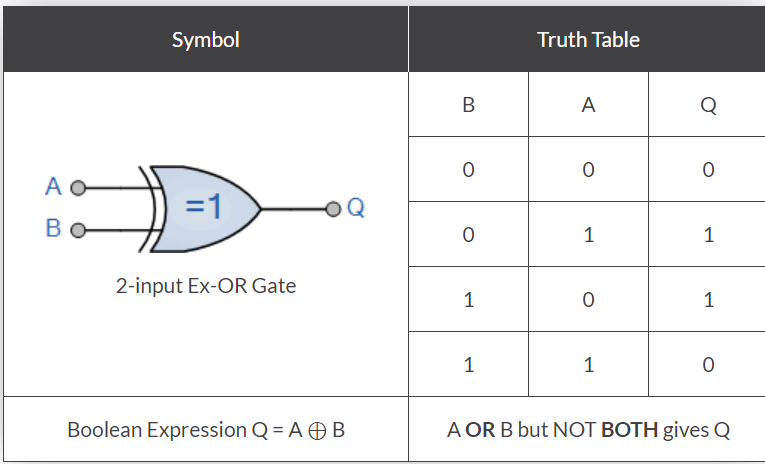
Ans: 01

Explanation:

^ is for bit wise xor

Means for odd input of true o/ p is true

For 2 bytes either one is 1 theno/ps one



a= 1means 01

b= 00 means 00

a=a^b 🡪 a will be 01

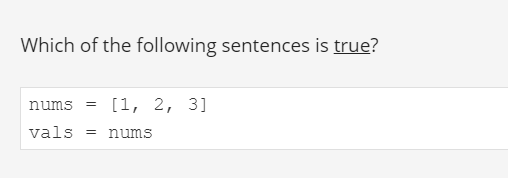
b=a^b 🡪b will be 01

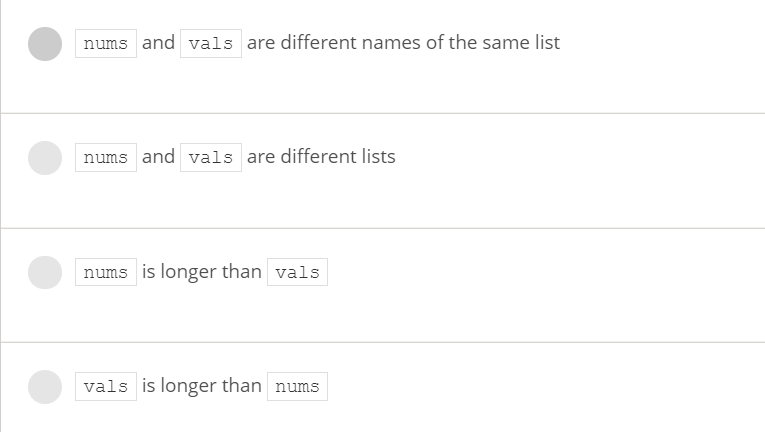
now a=01 b=01

now a= a^b 🡪00 means a=0

b=01 means 1

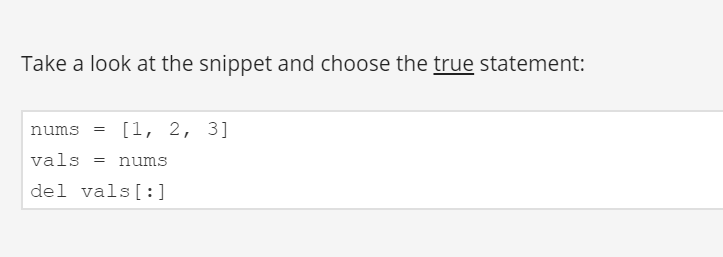
🡪

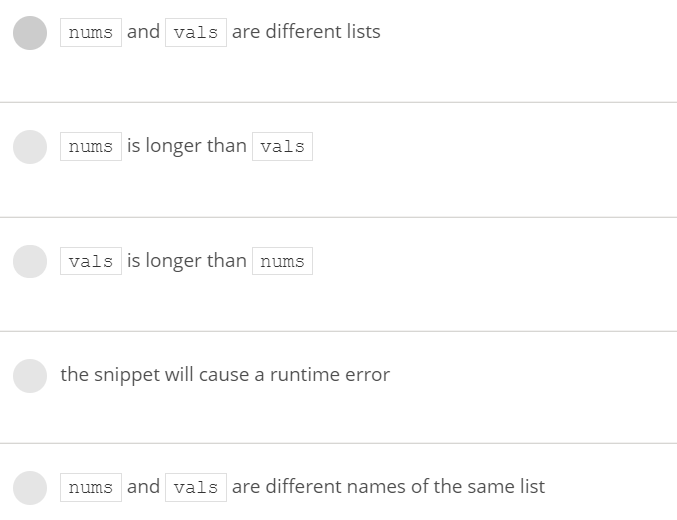




Ans: nums and vals are different names of the same list

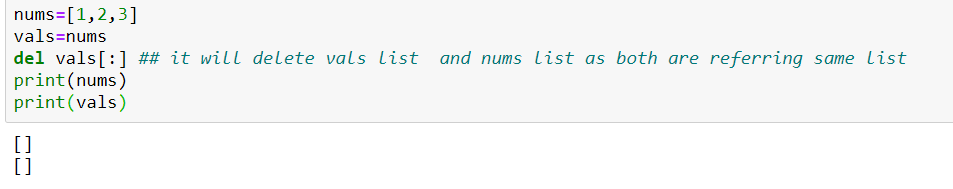
🡪





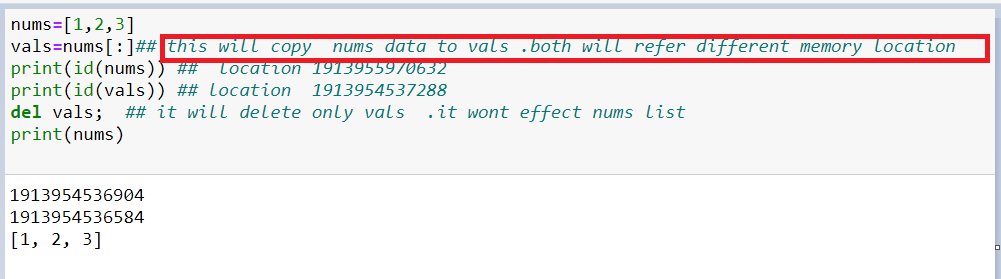
Ans: nums and vals are different names of the same list

Exaplantion:

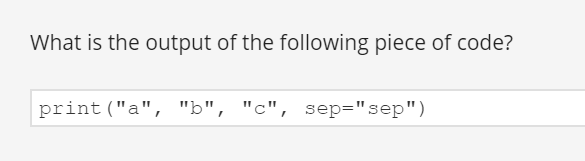


del vals[:] 🡪 makes list as null; but vals variable will be there

If we use command like del vals then it will delete entire list and variable also



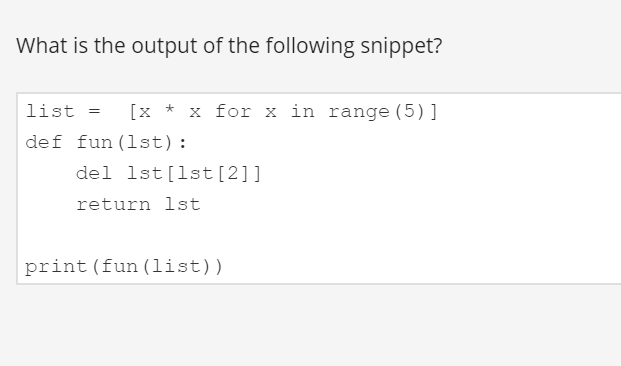
🡪

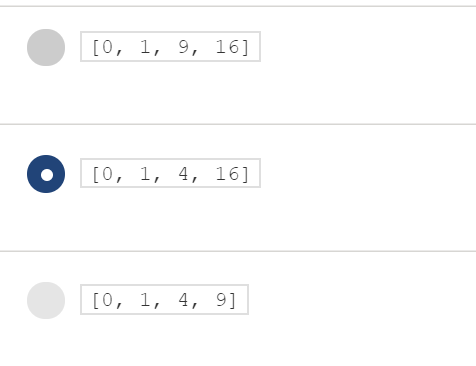


Ans:



-->





🡪

**Ans: [0,1,4,9]**

Explanation:

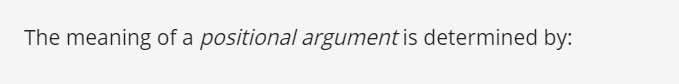
0,1,4,9,16

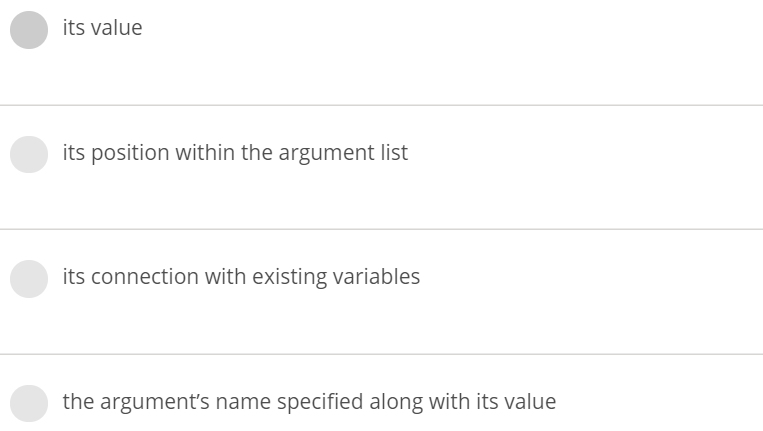
lst[2] means 4

lst[lst[2]]🡪lst[4]

lst[4]🡪16

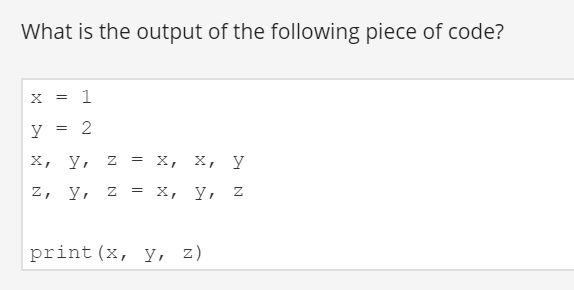
so it will delete 16



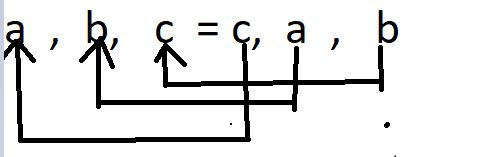


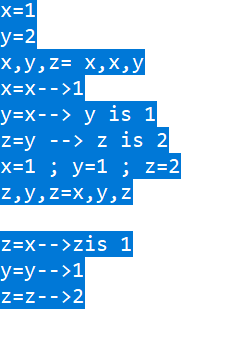
Ans: The position within the argument list

🡪

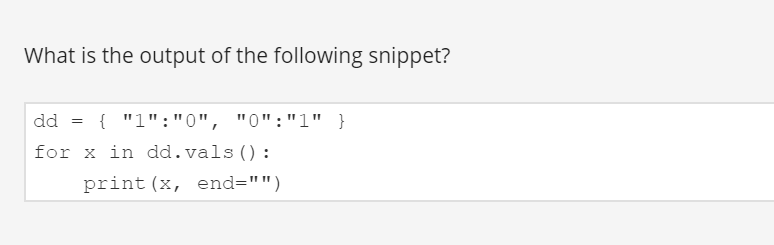


Ans:1 1 2



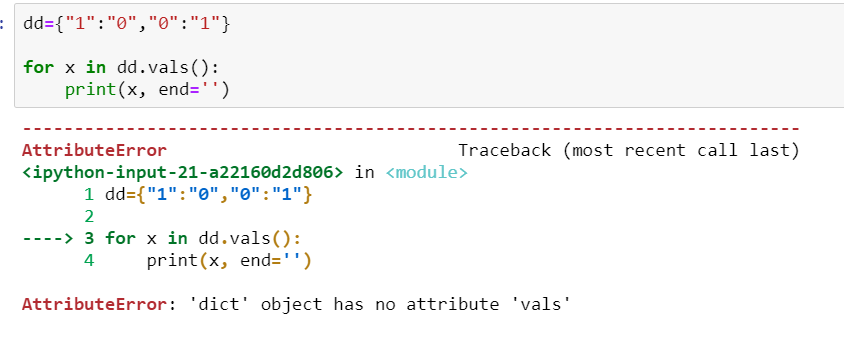


🡪

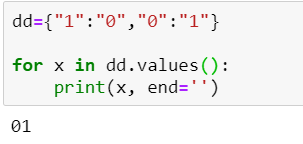


Ans: 

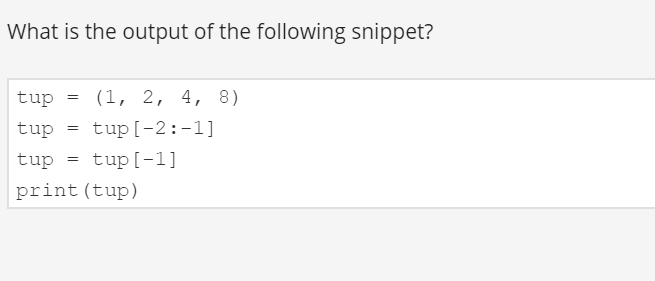
Explanation:



We have values() method for dictionary .there is no method with name vals()

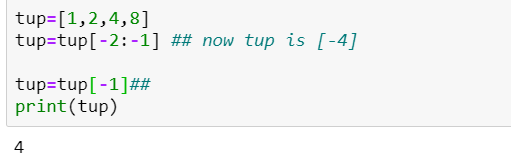


🡪

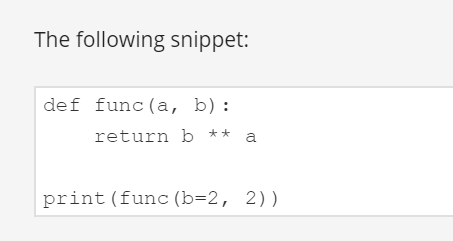


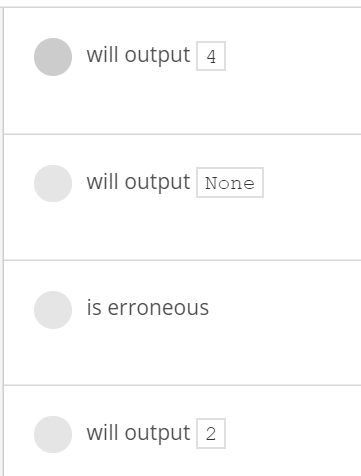
Ans:4

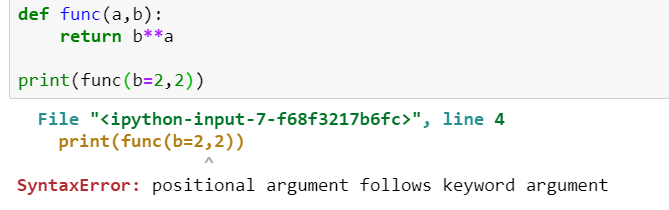
Explanation:



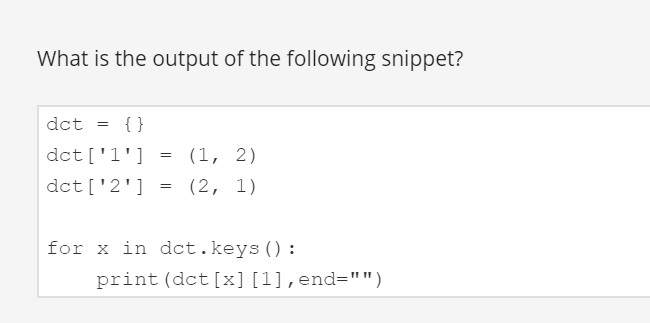
🡪





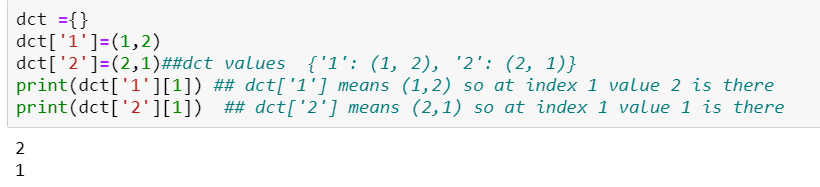
Ans: is erroneous

🡪

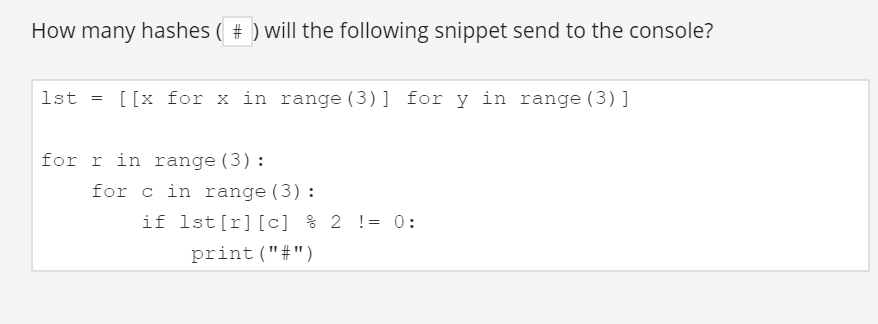


Ans: 21

Explanation:



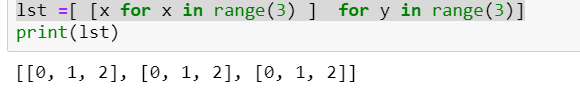
In print end =”” is there so o/p will be printed without space so o/p is 21

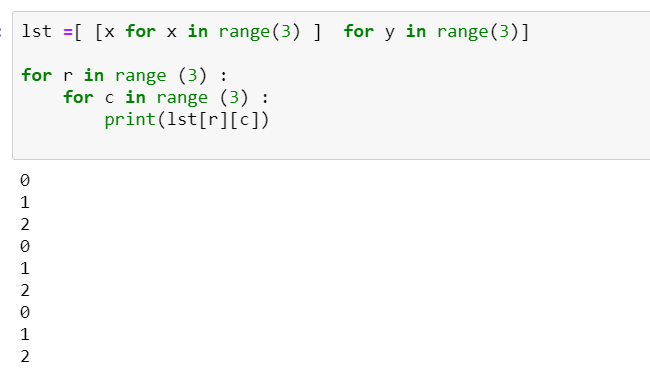


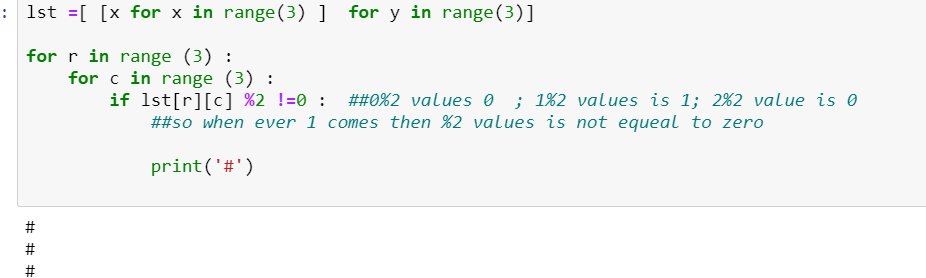


Ans: three

Explanation:







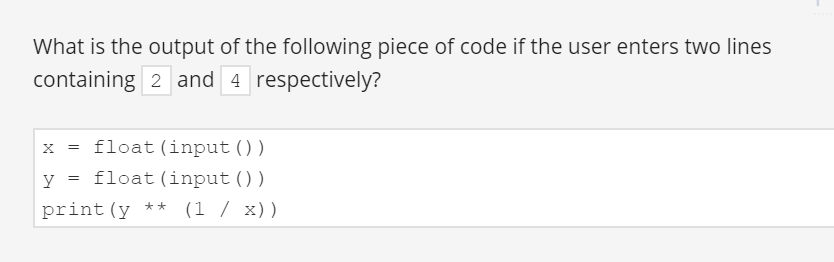
🡪





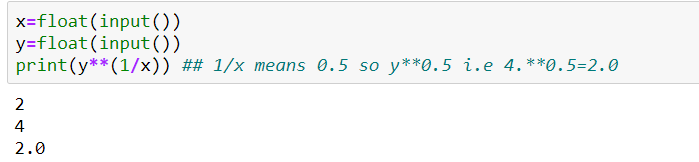
Ans: in is python operator so we cant use that a variable

🡪

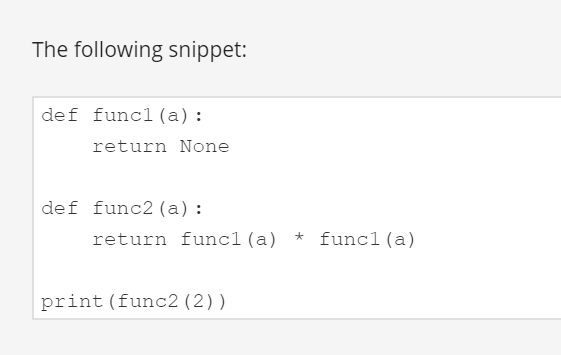


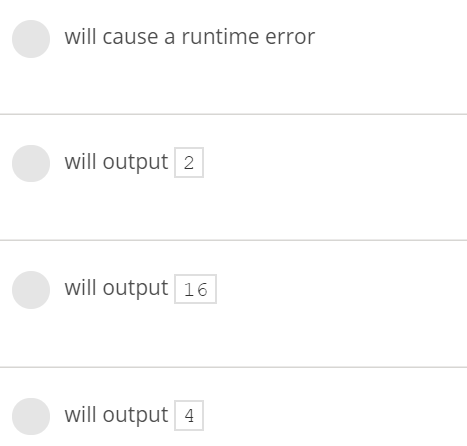


Ans:2.0



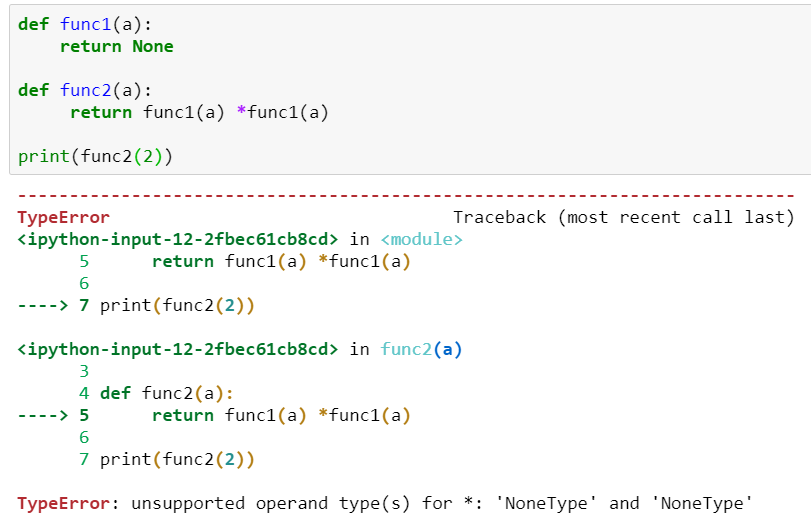
🡪



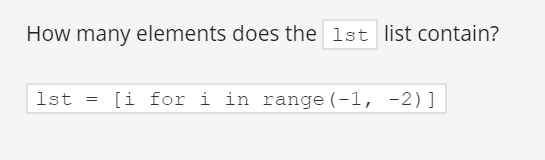


Ans: will cause a runtime error

Bcz None\* None not supported

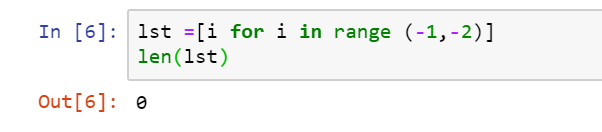


🡪





Ans: zero



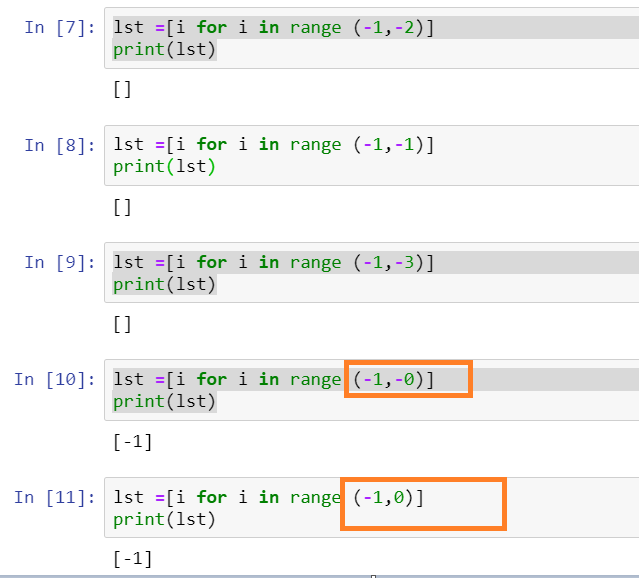
Important point below are

If we give range(-1,-2) means empty list

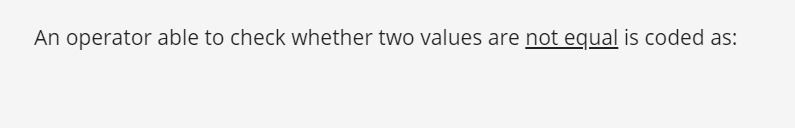
Range (-1,-1) also empty list

Ranger(-1,-0) gives [-1]

Range(-1,0) also gives [-1]



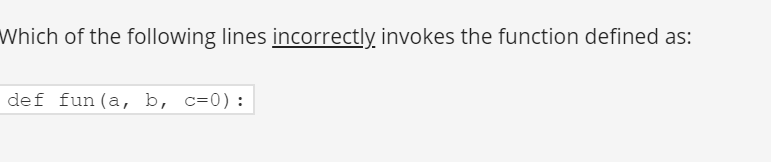
🡪





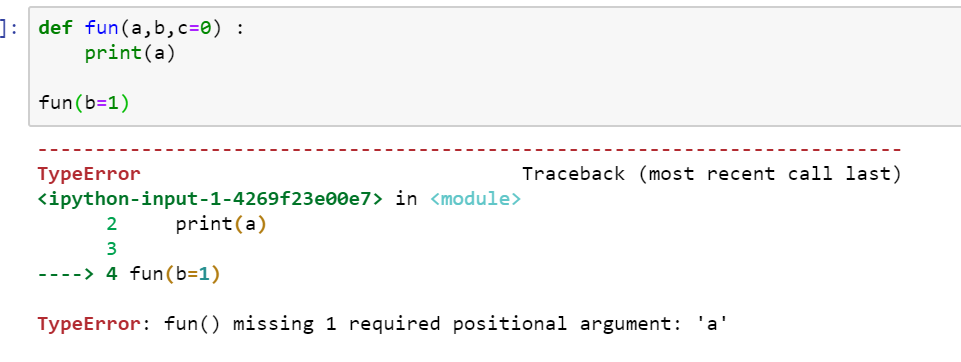
Ans: !=

🡪

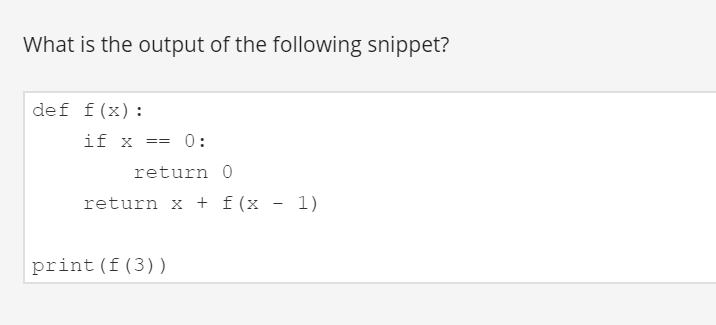




Ans: fun(b=1)

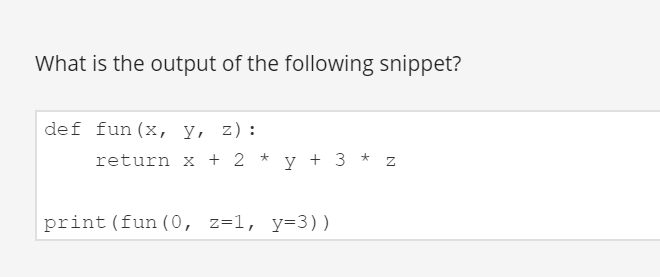


🡪

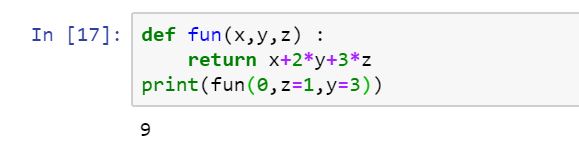


o/p:6

🡪



Ans:9



o/p: 9

x= 0

y =3

z= 1

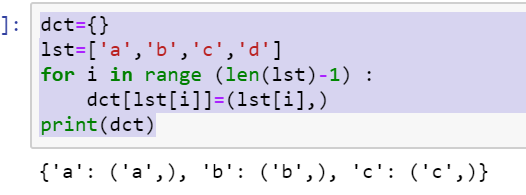
0+2\*3 + 3\*1

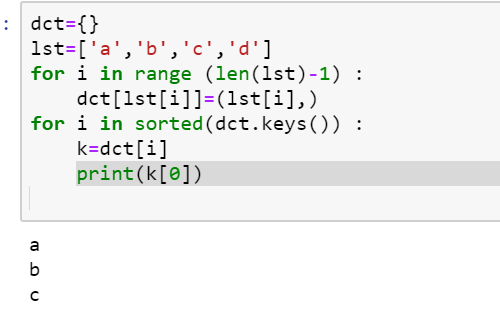
0+6+3 =9

🡪



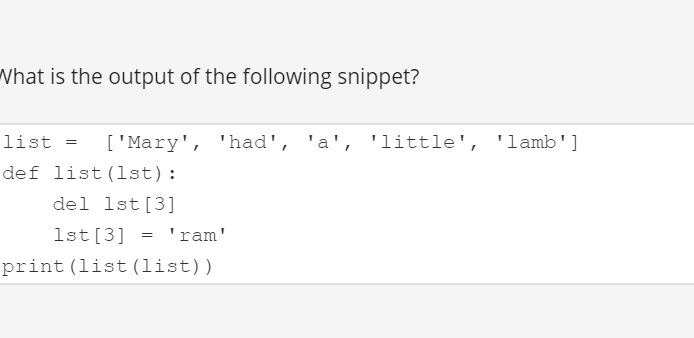
Ans: print(k[0])





K=dct[i] is of tuple type so we have to use k[0] to print a,b,c values

🡪



Ans: snippet is erroneous

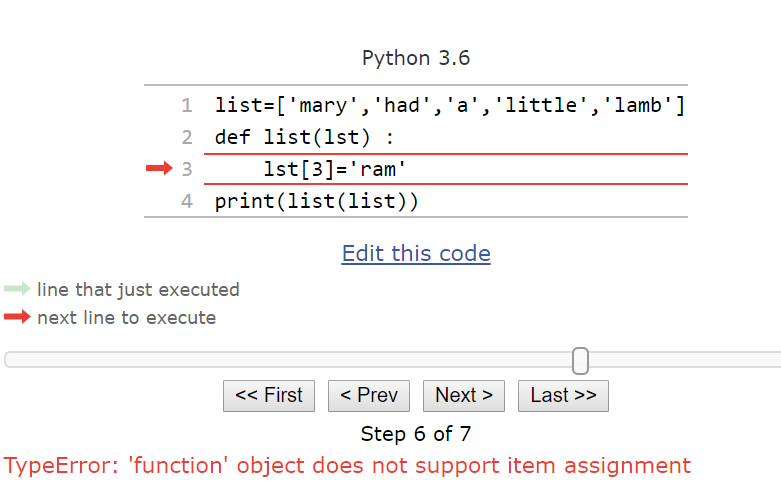
Because function name and list name both are same

Solution: change the function name

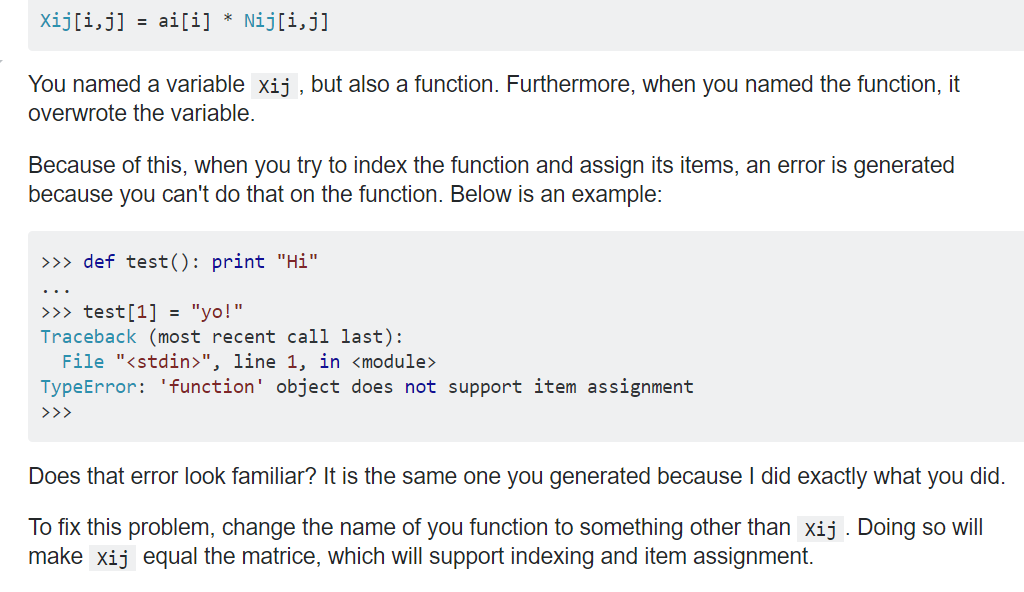
Explanation :

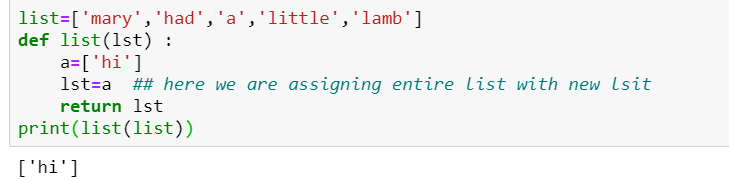
Item deletion and item assignment not possible inside a function for i/p parameter if the function name and i/p list name both are same

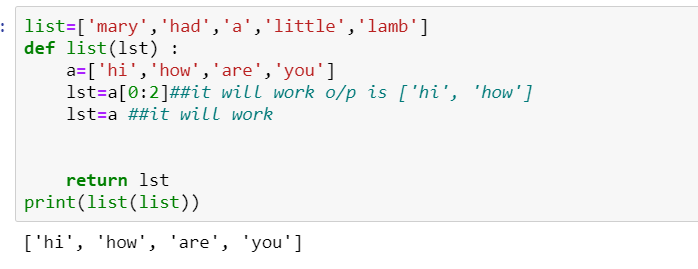




Refer: <https://stackoverflow.com/questions/18232313/typeerror-function-object-does-not-support-item-assignment>

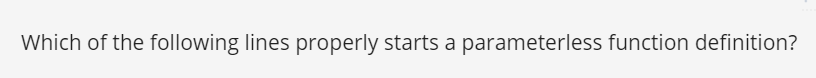




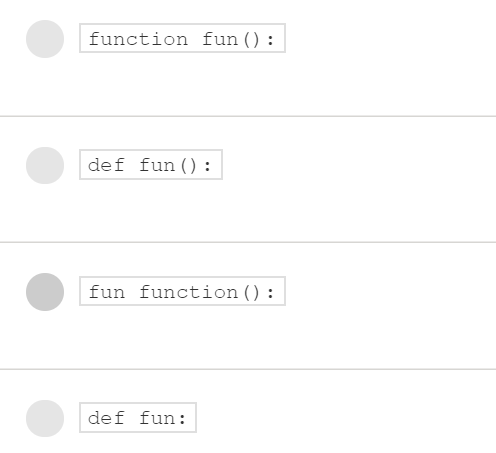




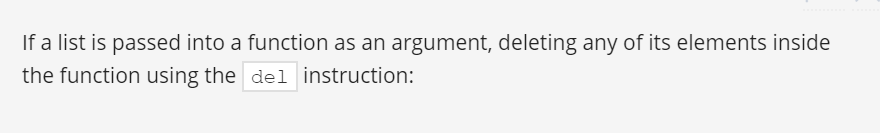
🡪

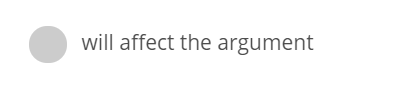


Options:

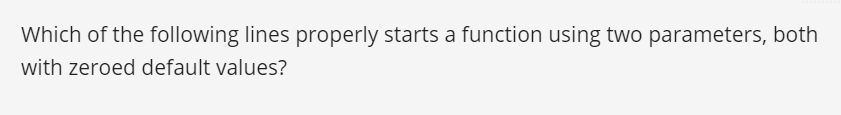


Ans: def fun():

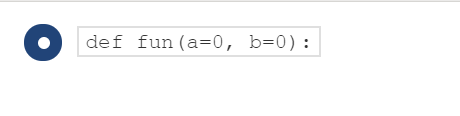


Ans: 

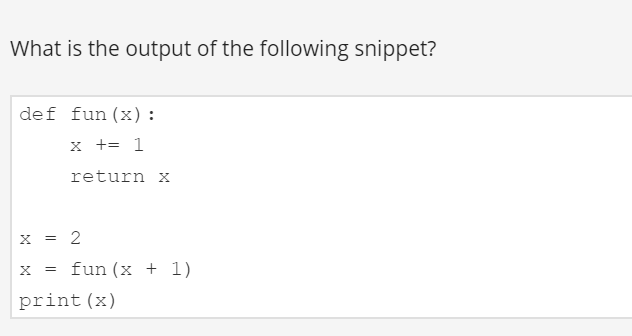
--🡪



Ans:

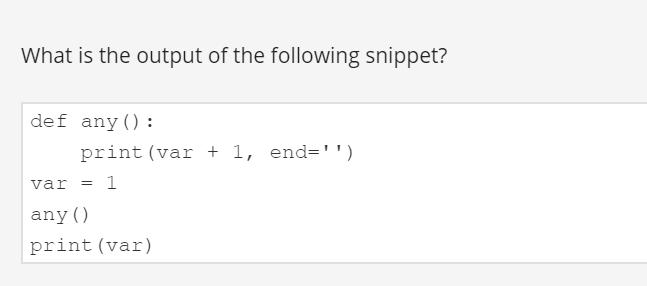


--🡪

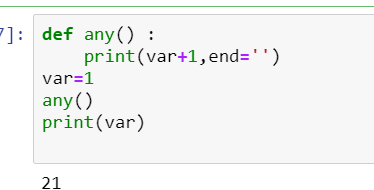


o/p is: 4

----🡪



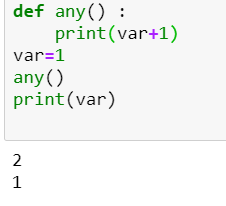
o/p is: 21



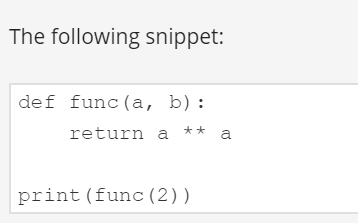
Inside function it will print 2 bcz var+1 i.e 1+1

After calling any function print(var) will print 1 as var is

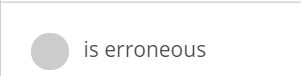
Bcz of end=’ ‘ so o/o will be printed in same line

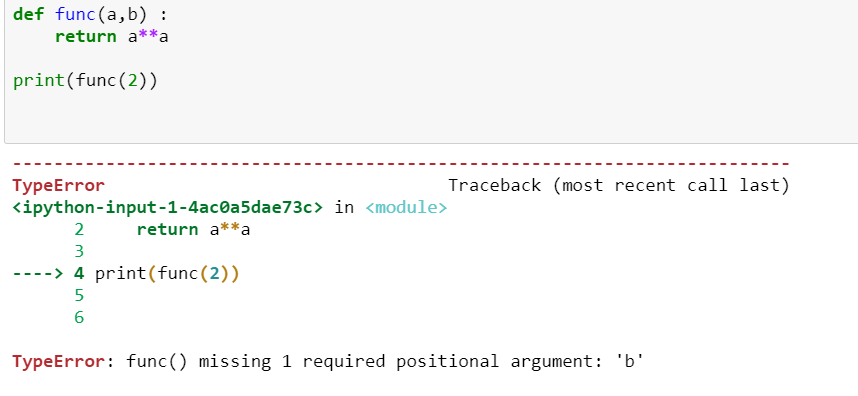


--🡪

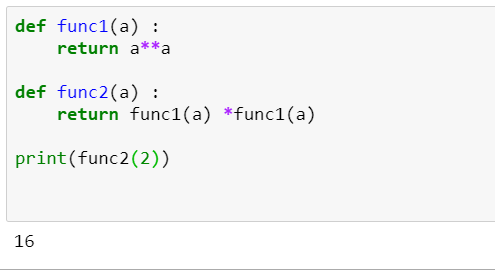


Ans:



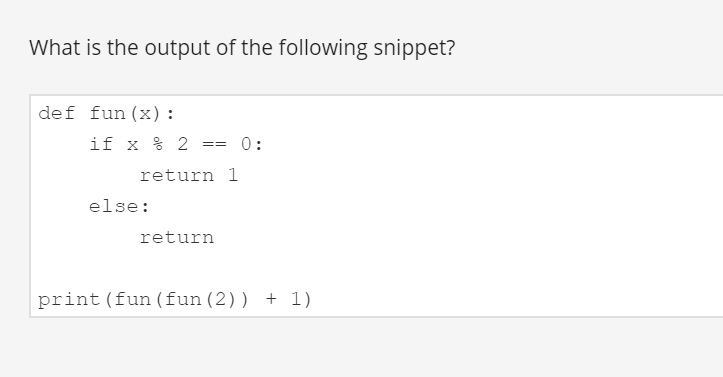


----🡪



Ans:16

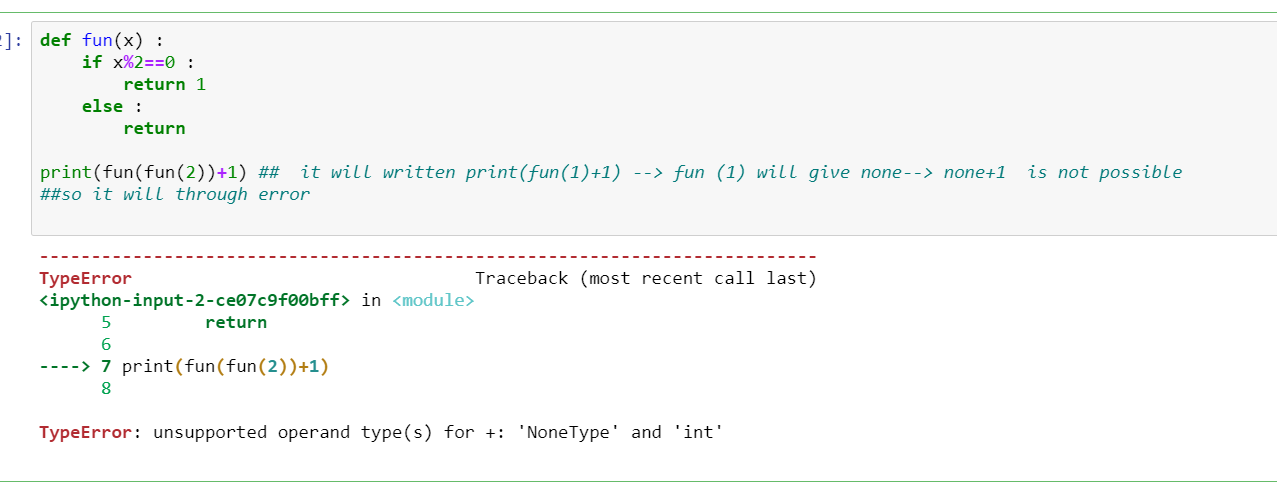
-🡪



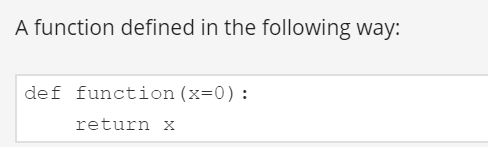
Ans:



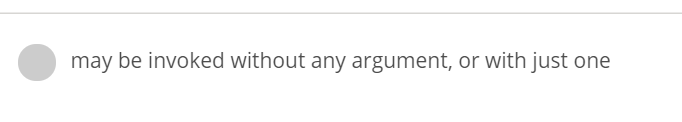
Explanation is below



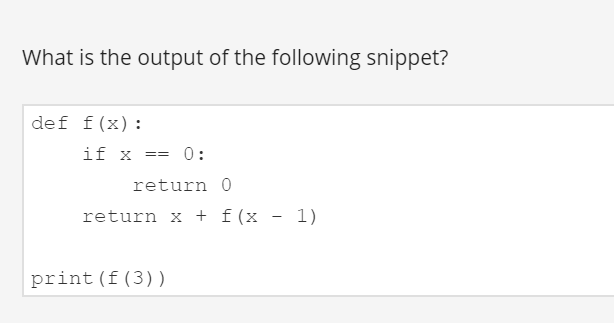
-🡪



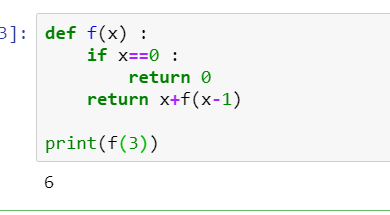
Ans :



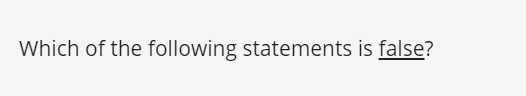
-🡪



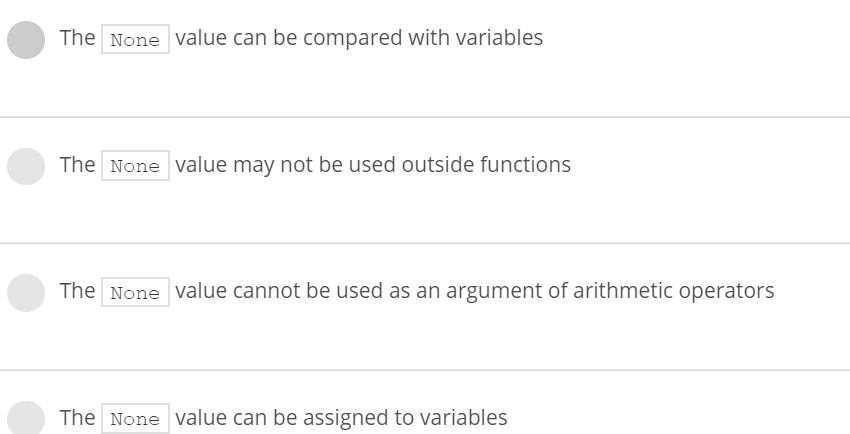
Ans :6



-🡪



Options:



None value can be compared with values 🡪True

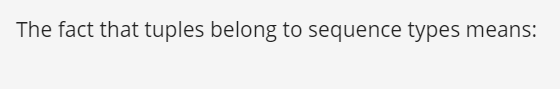
The None value may not be used outside functions 🡪 False

The None Value cannot be used as an argument of arithmetic operators🡪true

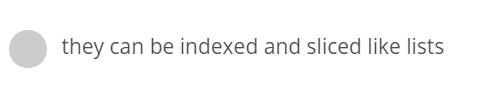
The None value can be assigned to variables🡪True

QUESTION ON TUPLE:

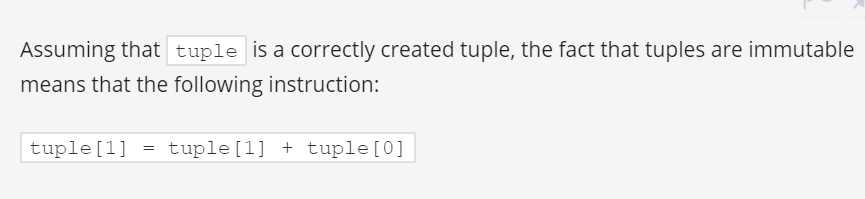
🡪



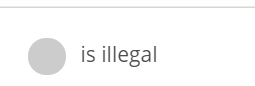
Ans:

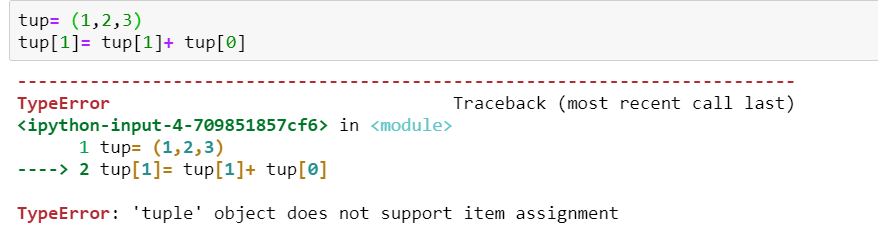


🡪

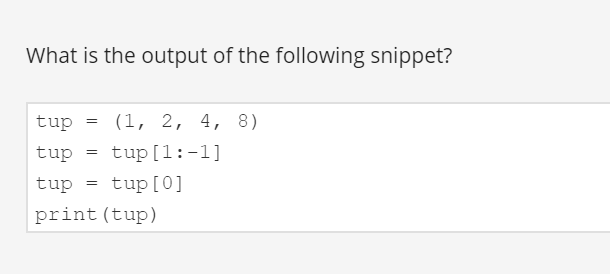


Ans: it will throw error





🡪



Ans: 2

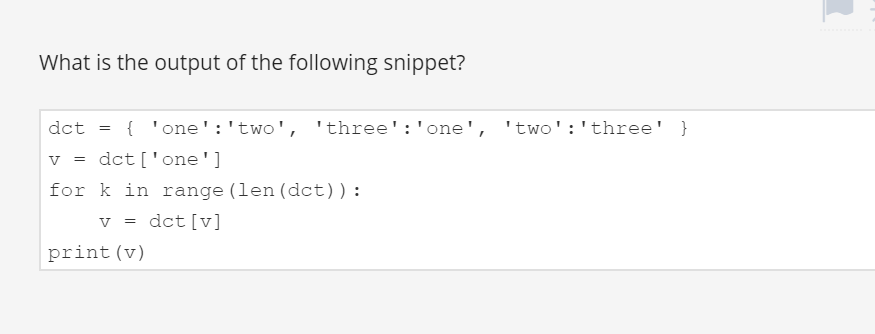
Explanation:

Tup[1:-1 ] gives as (2,4) which is tuples

Now tup=tup[0] means we are assigning the 2 value to tup which is integer

So print(tup) gives o/p as 2

🡪



Ans: two

Explanation:

V=dct[‘one’] means v is ‘two’

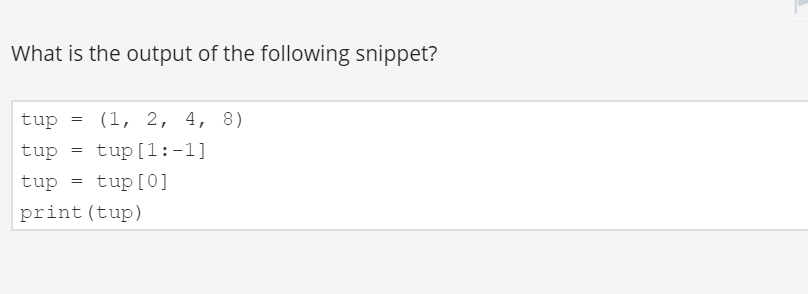
The loop will have length as 3 means it will loop through 3 times

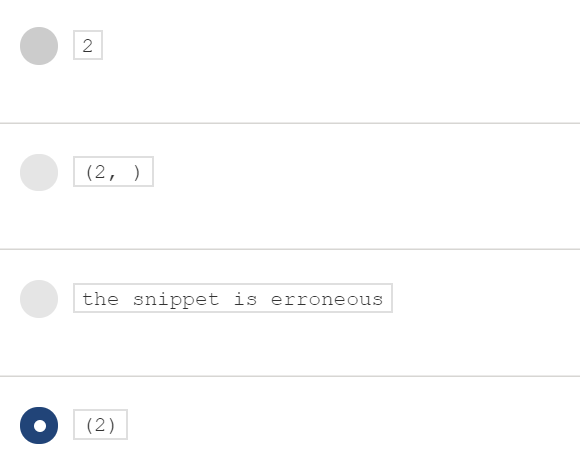
V=dct[‘two’] gives ‘three’ – first time loop

V=dct[‘three] gives ‘one’—2nd time loop

V=dct[‘one’] gives two --3rd time loop

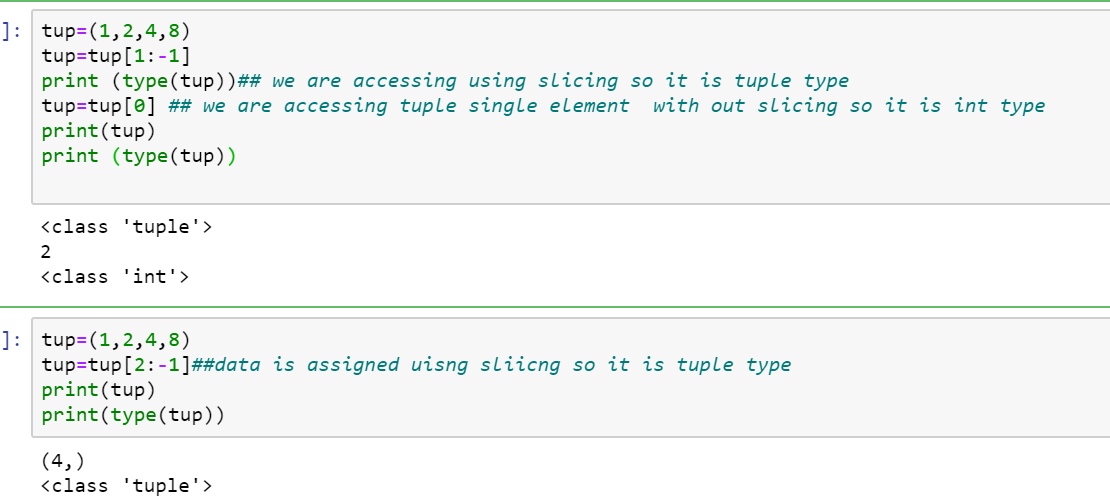
🡪



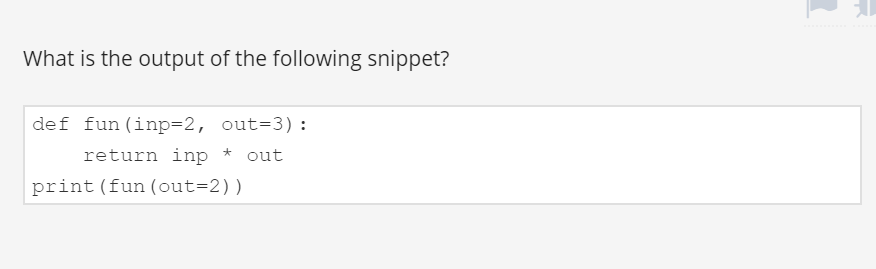


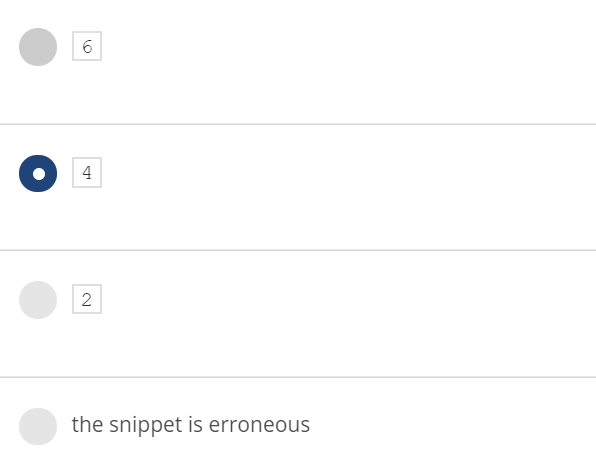
ANS: 2

Explanation:



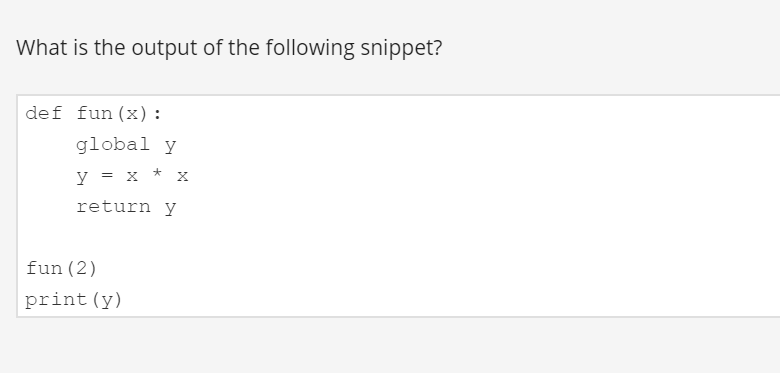
--🡪

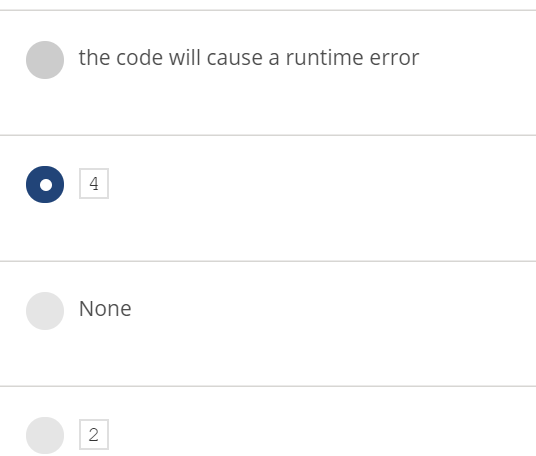




o/p is : 4

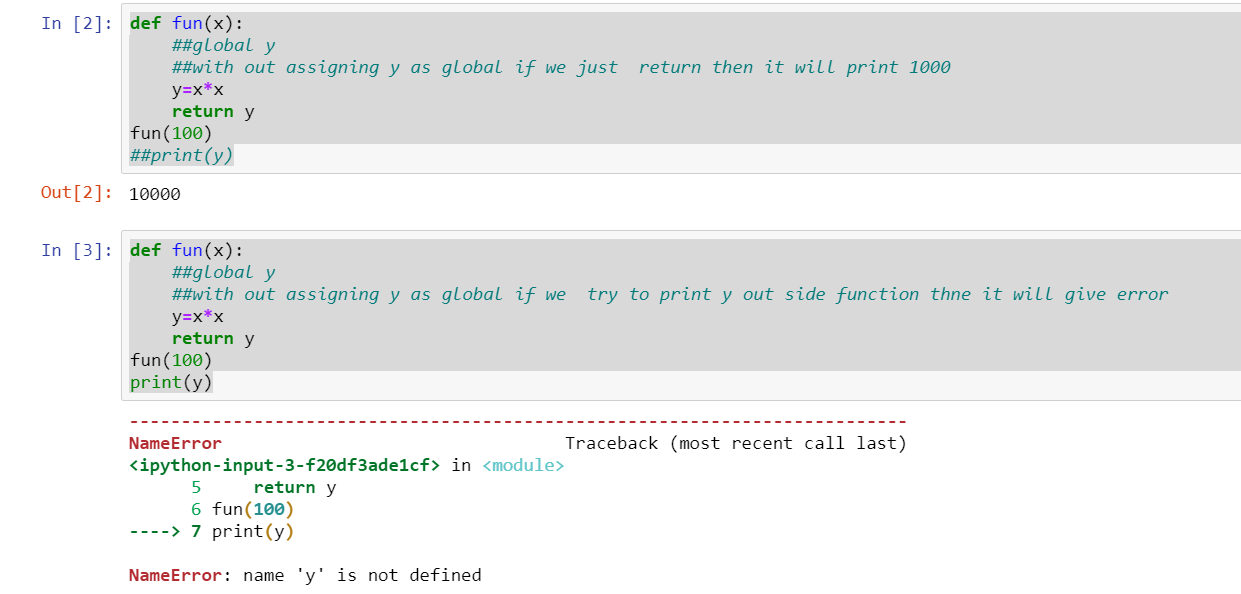
🡪

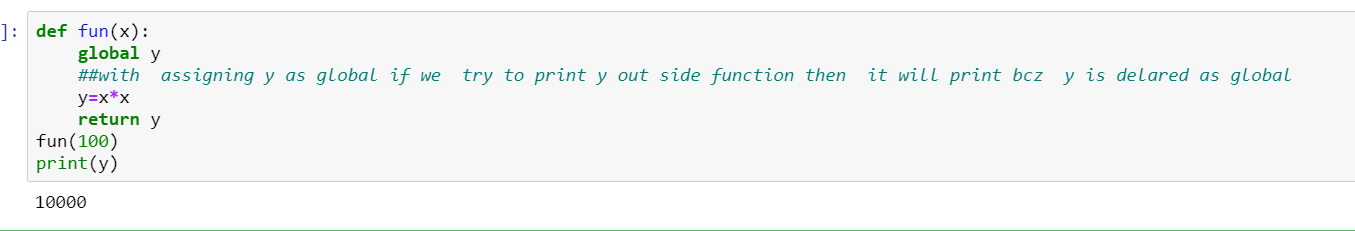




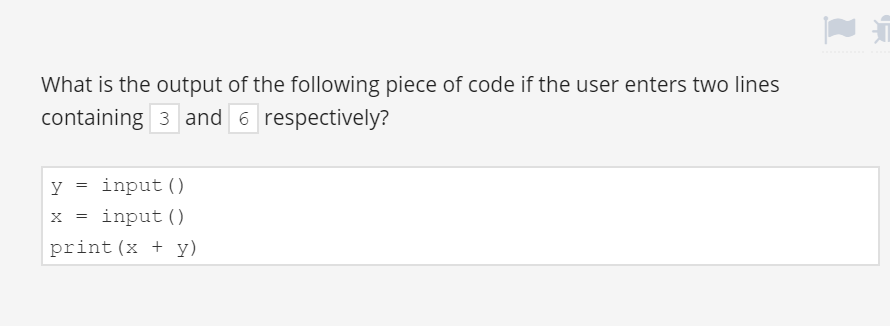
Ans : 4

Explanation:





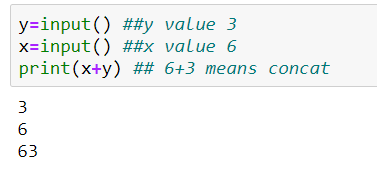
🡪



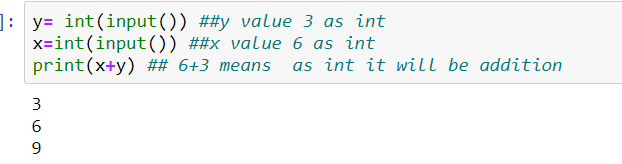
Ans: 63

Explanation:

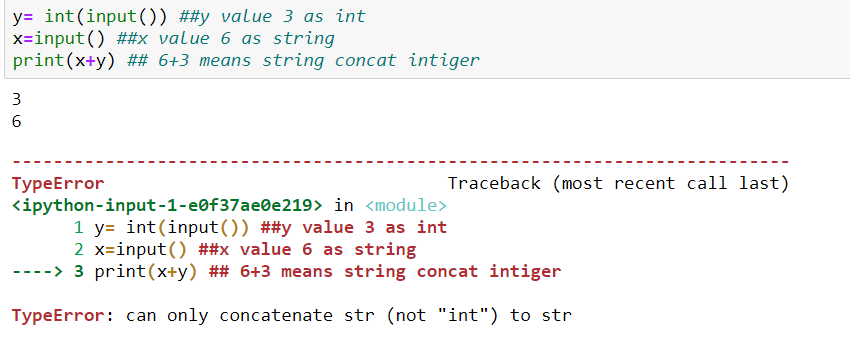
In below case it will consider as string so concat will be done



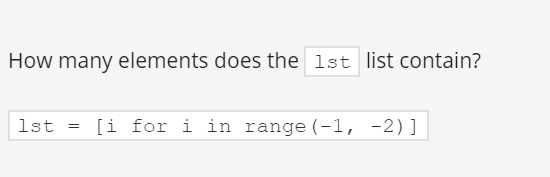
In below case input converted into int data type so + means addition will be done



In below case y is int and x is int if we do + means itwill try for concat n will give error

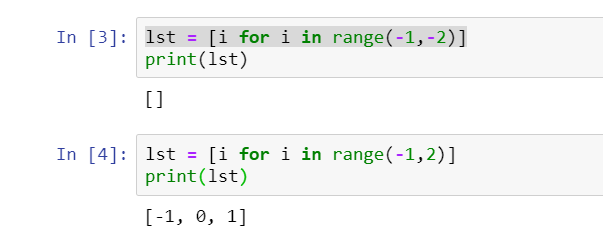


🡪

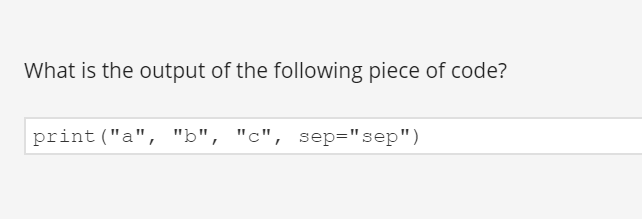


Ans: zero

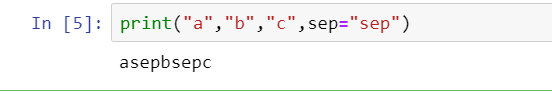
Explanation:



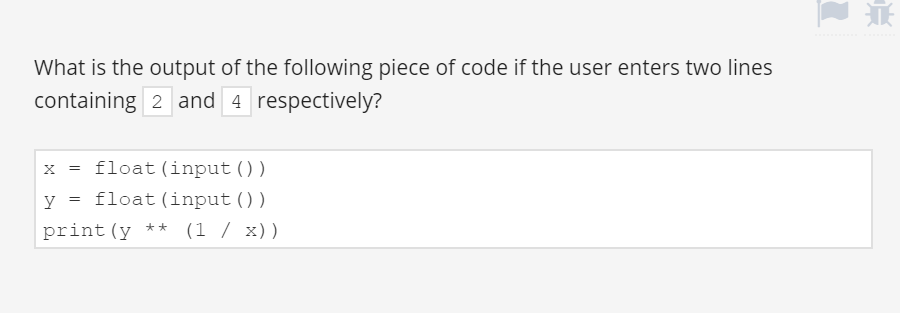
🡪



O/p: asepbsepc

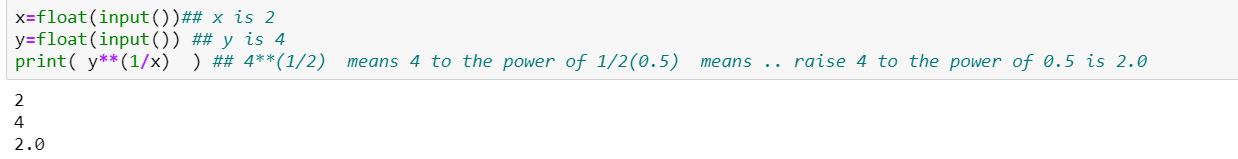


--🡪

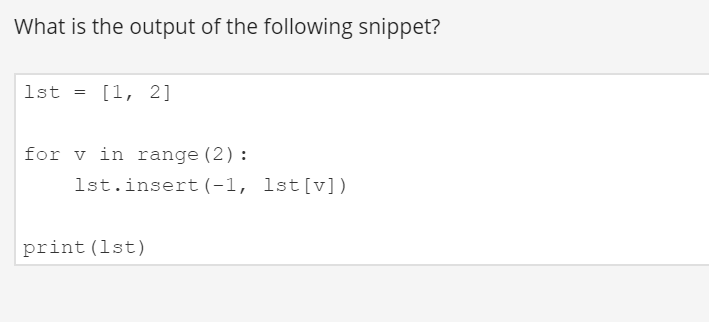


Ans: 2.0

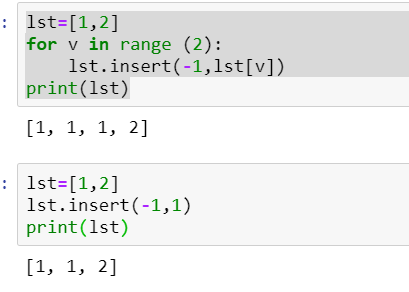
Explanation in below



🡪



o/p is: [1,1,1,2]

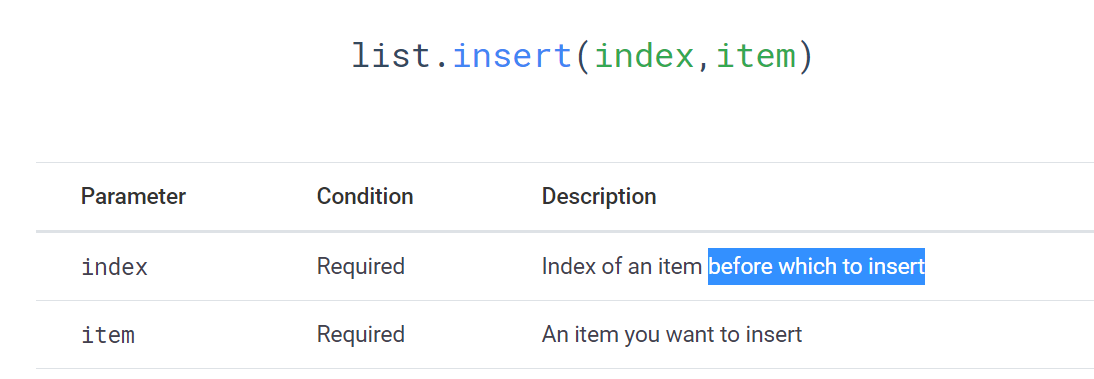


Examples:

Refer: inserT append extend in LISTS

## IMP ABOUT LISTS

<https://www.learnbyexample.org/python-list-insert-method/>



Examples

Example: Insert ‘yellow’ at 2nd position

L = ['red', 'green', 'blue']

L.insert(1,'yellow')

print(L) # ['red', 'yellow', 'green', 'blue']

You can also use [negative indexing](https://www.learnbyexample.org/python-list/#negative-list-indexing) with insert() method.

Example: Insert ‘yellow’ at 2nd position with Negative indexing

L = ['red', 'green', 'blue']

L.insert(-2,'yellow')

print(L) # ['red', 'yellow', 'green', 'blue']

in above -2 means green . so it will insert yellow before green element

Index greater than list length

When you specify an *index* greater than list length, you do not get any exception. Instead, the item is inserted at the end of the list.

Example: Index greater than list length inserts item at the end

L = ['red', 'green', 'blue']

L.insert(10,'yellow')

print(L) # ['red', 'green', 'blue', 'yellow']

## insert() vs append()

Inserting item at the end of the list with insert() method is equivalent to append() method.

Example: Inserting item at the end with insert()

L = ['red', 'green', 'blue']

L.insert(len(L),'yellow')

print(L) # ['red', 'green', 'blue', 'yellow']

is equivalent to

L.append('yellow')

## insert() vs extend()

insert() method treats its argument as a single object.

Example:

L = ['red', 'green']

L.insert(2,'blue')

print(L) # ['red', 'green', 'blue']

Use [extend()](https://www.learnbyexample.org/python-list-extend-method/) method, if you want to add every item of an iterable to a list.

Example:

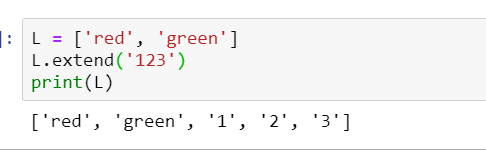
L = ['red', 'green']

L.extend('blue')

print(L) # ['red', 'green', 'b', 'l', 'u', 'e']

ex:1

below 123 is treated as string and iterable



In below examples 123 is int data type and it is not iterable

