

Swipe >>>

# Guess the Output



# Guess The Output



Language: Python

```
count = 1
def doThis():
    global count
    for i in (1, 2, 3):
        count += 1
doThis()
print (count)
```

cs mock

# Guess The Output



**Output:** 4

**Explanation:** The variable count declared outside the function is global variable and also the count variable being referenced in the function is the same global variable defined outside of the function. As a result, the changes made to variables in the function are reflected to the original variable. So, the output of the program is 4.

# Guess The Output



Language: Python

```
a = True
b = False
c = False
if not a or b:
    print ("CS")
elif not a or not b and c:
    print ("Mock")
elif not a or b or not b and a:
    print ("Interview")
else:
    print ("Mentorship")
```

cs mock

# Guess The Output



## Output: Interview

**Explanation:** In Python, the precedence order is first NOT then AND and in last OR. So the if condition and second elif condition evaluates to False while the third elif condition is evaluated to be True resulting in “Interview” as output.

# Guess The Output



Language: Python

```
a = True
b = False
c = False
if a or b and c:
    print ("Mock Interview")
else:
    print ("Mentorship")
```

cs mock

# Guess The Output



## Output: Mock Interview

**Explanation:** In Python, AND operator has higher precedence than OR operator. So, it is evaluated first. i.e, (b and c) evaluates to false. Now the OR operator is evaluated. Here, (True or False) evaluates to True. So the if condition becomes True and “Mock Interview” is printed as output.

# Guess The Output



Language: Python

```
a = 4.5
b = 2
print (a//b)
```

# Guess The Output



**Output:** 2.0

**Explanation:** This type of division is called truncating division where the remainder is truncated or dropped. The “//” operator is used to return the closest integer value which is less than or equal to a specified expression or value.

# Guess The Output



Language: Python

```
class Mock:  
    def __init__(self, id):  
        self.id = id  
        id = 555  
    mock = Mock(111)  
    print mock.id
```

cs mock

# Guess The Output



**Output:** 111

**Explanation:** Instantiation of the class “Mock” automatically calls the method `__init__` and passes the object as the `self` parameter. 111 is assigned to the data attribute of the object called `id`. The value “555” is not retained in the object as it is not assigned to a data attribute of the class/object. So, the output of the program is “111”

**cs** mock

# Guess The Output



Language: Python

```
for i in range(1):  
    print (i)
```

# Guess The Output



**Output:** 0

**Explanation:** If only a single argument is passed to the range method, Python considers this argument as the end of the range and the default start value of range is 0. So, it will print all the numbers starting from 0 and before the supplied argument.

# Guess The Output



Language: Python

```
counter = {}
def addToCounter(product):
    if product in counter:
        counter[product] += 1
    else:
        counter[product] = 1
addToCounter('CS')
addToCounter('Mock')
addToCounter('cs')
print (len(counter))
```

cs mock

# Guess The Output



**Output:** 3

**Explanation:** The task of the “len” function is to return the number of keys in a dictionary. The keys to a dictionary are case sensitive. Here 3 keys are added to the dictionary.

# Guess The Output



Language: Python

```
dictionary = {}
dictionary[1] = 2
dictionary[1] = 2
sum = 0
for k in dictionary:
    sum += dictionary[k]
print (sum)
```

# Guess The Output



**Output:** 4

**Explanation:** In the above dictionary, key 1 is enclosed between single quotes and only 1 represents two different keys as one of them is an integer and the other is a string. So, the output of the program is 4.

# Guess The Output



Language: Python

```
a = "1"  
b = 2  
print (a + b)
```



Error



3

# Guess The Output



**Output:** Error

**Explanation:** Python is a strongly typed language. Variable 'b' is of type integer and the variable 'a' is of type string so trying to concatenate an integer variable to a string variable, gives "TypeError" exception.

# Guess The Output



Language: Python

```
def addToList(listcontainer):  
    listcontainer += [10]  
mylistContainer = [10, 20, 30, 40]  
addToList(mylistContainer)  
print len(mylistContainer)
```

cs mock

# Guess The Output



**Output:** 5

**Explanation:** In Python, everything is a reference, and references are passed by value. Here the task of the function “addToList” is to add an element 10 in the list, So this will increase the length of the list by 1. So the output of the program is 5.

# Guess The Output



Language: Python

```
mockId = [1, 2]
mockId.append([3,4])
print (len(mockId))
```

**cs** mock

# Guess The Output



**Output: 3**

**Explanation:** The task of the `append()` method is to append a passed obj into an existing list. Append method will not merge the two lists, the entire list which is passed is added as an element of the list. Final `mockId` will look like `[1, 2, [3, 4]]`. So the output is 3.

# Guess The Output



Language: Python

```
list = ['CS', 'Mock', 'Interview', 'Mentorship']
pos = list.index("crack")
print (pos)
```

cs mock

# Guess The Output



## Output: Error

**Explanation:** The task of the index is to find the position of a supplied value in a given list. In the above program the supplied value is “crack” and the list is nameList. As crack is not present in the list, an exception is thrown.

# Guess The Output



Language: Python

```
list = ['CS', 'Mock', 'Interview', 'Mentorship']
print (list[1][-1])
```

cs mock

# Guess The Output



**Output:** k

**Explanation:** The index position -1 represents either the last element in a list or the last character in a String. In the above given list, the index 1 represents the second element i.e, the second string “Mock” and the index -1 represents the last character in the string “Mock”. So, the output is “k”.

# Guess The Output



Language: Python

```
def mock(x,l=[]):
    for i in range(x):
        l.append(i*i)
    print(l)

mock(1)
mock(2,[3])
mock(2)
```

**cs** mock

# Guess The Output



**Output:** [0] [3, 0, 1] [0, 0, 1]

**Explanation:** The first mock function call appends 0 to the empty list, l. l is a name for a variable that points to a list stored in memory. The second call starts off by creating a new list in a new block of memory. l then refers to this new list and appends 0 and 1 to this new list. The third function call uses the original list stored in the original memory block. That is why it starts off with 0.

# Guess The Output



Language: Python

```
class A(object):
    val = 1
class B(A):
    pass
class C(A):
    pass
B.val = 2
A.val = 3
print (A.val, B.val, C.val)
```

# Guess The Output



**Output:** 3 2 3

**Explanation:** In Python, class variables are internally handled as dictionaries. If a variable name is not found in the dictionary of the current class, the class hierarchy (i.e., its parent classes) are searched until the referenced variable name is found, if the variable is not found error is being thrown. The output 3 2 3 instead of 3 3 3 is because here B.val reflects 2 instead of 3 since it is overridden earlier.

# Guess The Output



Language: Python

```
def mock():
    "CS Mock provides Mock Interview and Mentorship"
    print (mock.__doc__[17:31])
```

**cs** mock

# Guess The Output



## Output: Mock Interview

**Explanation:** There is a docstring defined for this method, by putting a string on the first line after the start of the function definition. The docstring can be referenced using the `__doc__` attribute of the function. And hence it prints the indexed string.

# Guess The Output



Language: Python

```
list1 = ['CS', 'Mock', 'Interview', 'Mentorship']
print (list1[-2])
```

cs mock

# Guess The Output



**Output:** Interview

**Explanation:** This will print an item located at index -2 i.e. second last element in Output.

# Guess The Output



Language: Python

```
list1 = range(100, 110)
print (list1.index(105))
```

cs mock

# Guess The Output



**Output:** 5

**Explanation:**

Statement 1: will generate numbers from 100 to 110 and append all these numbers in the list.

Statement 2: will give the index value of 105 in the list list1.

# Guess The Output



Language: Python

```
list = ['a', 'b', 'c', 'd', 'e']
print (list[10:])
```

cs mock

# Guess The Output



**Output:** `[]`

**Explanation:**

A expression `list[listelements]*N` where `N` is an integer appends `N` copies of list elements in the original list. If `N` is a negative integer or 0 output will be an empty list else if `N` is positive list elements will be added `N` times to the original list.

# Guess The Output



Language: Python

```
T1 = 1
T2 = (2)
print(T1 + T2)
```

# Guess The Output



**Output:** 3

**Explanation:** T2 acts like integer and not tuple unless a comma is added in the end. To create a tuple with only one item, you have to add a comma after the item, unless Python will not recognize the variable as a tuple.

# Guess The Output



Language: Python

```
temp = ['Computer', 'Science']
arr = [i[0].upper() for i in temp]
print(arr)
```

# Guess The Output



**Output:** ['C','S']

**Explanation:** The variable `i` is used to iterate over each element in list `temp`. `i[0]` represents the character at the 0th index of `i` and the `.upper()` function is used to capitalize the character present at `i[0]`.

# Guess The Output



Language: Python

```
L1 = []
L1.append([1, [2, 3], 4])
L1.extend([7, 8, 9])
print(L1[0][1][1] + L1[2])
```

# Guess The Output



**Output:** 11

**Explanation:** In the `print()`, indexing is used. `L1[0]` denotes `[1, [2, 3], 4]`, `L1[0][1]` denotes `[2, 3]`, `L1[0][1][1] = 3` and `L1[2] = 8`. Thus, the two integers are added,  $3 + 8 = 11$  and output comes as 11.

# Guess The Output



Language: Python

```
List = [True, 50, 10]  
List.insert(2, 5)  
print(sum(List))
```

# Guess The Output



**Output:** 66

**Explanation:** The List initially has 3 elements. The `insert()` adds element 5 at index 2, moving element 10 at index 3 and the List becomes [True, 50, 5, 10]. Boolean has an integer value of 1, thus sum becomes  $1 + 50 + 5 + 10 = 66$ .

# Guess The Output



Language: Python

```
x = ['ab', 'cd']
for i in x:
    x.append(i.upper())
print(x)
```

cs mock

# Guess The Output



**Output:** No output

**Explanation:** The loop does not terminate as new elements are being added to the list in each iteration. So our program will stuck in infinite loop.

# Guess The Output



Language: Python

```
my_tuple = (0, 1)
my_tuple.append( (2, 3) )
print (len(my_tuple))
```

# Guess The Output



**Output:** Error

**Explanation:** Tuples are immutable and don't have an append method as in case of Lists. Hence an error is thrown in this case.

# Guess The Output



Language: Python

```
T = (2e-04, True, False, 8, 1.001, True)
val = 0
for x in T:
    val += int(x)
print(val)
```

cs mock

# Guess The Output



**Output:** 11

**Explanation:** Integer value of `2e-04`(0.0002) is 0, True holds a value 1 and False a 0, integer value of 1.001 is 1. Thus total  $0 + 1 + 0 + 8 + 1 + 1 = 11$ .

# Guess The Output



Language: Python

```
tuple1 = (1, 2, 4, 3)
tuple2 = (1, 2, 3, 4)
print(tuple1 < tuple2)
```

cs mock

# Guess The Output



**Output: False**

**Explanation:** In this case elements will be compared one by one. So, when it compares 4 with 3 it will return False.

# Guess The Output



Language: Python

```
tuple = {}  
tuple[(1,2)] = 3  
tuple[(2,1)] = 3  
tuple[(1)] = 4  
sum = 0  
for k in tuple:  
    sum += tuple[k]  
print(sum)
```

# Guess The Output



**Output:** 10

**Explanation:** Tuples can be used for keys into dictionary. The tuples can have mixed lengths and the order of the items in the tuple is considered when comparing the equality of the keys.

# Guess The Output



Language: Python

```
print(2**3 + (5 + 6)**(1 + 1))
```

# Guess The Output



**Output:** 129

**Explanation:**

The above code will print 129 by following the BEDMAS rule of operator precedence.

# Guess The Output



Language: Python

```
var = 10
print(type(var))
var = "Hello"
print(type(var))
```

# Guess The Output



**Output:** int and str

**Explanation:**

Since we convert a to a tuple and then try to change its content, we will get an error since tuples are immutable.

# Guess The Output



Language: Python

```
a = [1, 2, 3]
a = tuple(a)
a[0] = 2
print(a)
```

# Guess The Output



**Output:** error

**Explanation:**

Since we convert a to a tuple and then try to change its content, we will get an error since tuples are immutable.

# Guess The Output



Language: Python

```
a = [1, 2, 3, 4, 5]
sum = 0
for ele in a:
    sum += ele
print(sum)
```

# Guess The Output



**Output:** 15

**Explanation:**

The above code calculates the sum of all elements in the list.

# Guess The Output



Language: Python

```
count = 0
while(True):
    if count % 3 == 0:
        print(count, end = " ")
    if(count > 15):
        break;
    count += 1
```

# Guess The Output



**Output:** 0,3,6,9,12,15

**Explanation:**

The above code prints the multiples of 3 not greater than 15, and then breaks off.

# Guess The Output



Language: Python

```
def solve(a, b):  
    return b if a == 0 else solve(b % a, a)  
print(solve(20, 50))
```

# Guess The Output



**Output:** 10

**Explanation:**

The above function basically calculates the gcd of 2 numbers recursively. The gcd of 20 and 50 is 10, so the answer is 10.

# Guess The Output



Language: Python

```
def func():
    global value
    value = "Local"
```

```
value = "Global"
func()
print(value)
```

# Guess The Output



**Output:** Local

**Explanation:**

We set the value of “value” as Global. To change its value from inside the function, we use the `global` keyword along with “value” to change its value to local, and then print it.

# Guess The Output



Language: Python

```
def check(a):  
    print("Even" if a % 2 == 0 else "Odd")  
  
check(12)
```

# Guess The Output



**Output:** Even

**Explanation:**

The program uses ternary operators to check if a given number is even or not.

# Guess The Output



Language: Python

```
a = [1, 2]
print(a * 3)
```

# Guess The Output



Language: Python

```
numbers = (4, 7, 19, 2, 89, 45, 72, 22)
sorted_numbers = sorted(numbers)
odd_numbers = [x for x in sorted_numbers
if x % 2 != 0]
print(odd_numbers)
```

# Guess The Output



**Output:** [7,19,45,89]

**Explanation:**

The above code basically forms a list containing the odd numbers in the numbers list, in sorted order.

# Guess The Output



Language: Python

```
word = "Python Programming"
n = len(word)
word1 = word.upper()
word2 = word.lower()
converted_word = ""
for i in range(n):
    if i % 2 == 0:
        converted_word += word2[i]
    else:
        converted_word += word1[i]
print(converted_word)
```

cs mock

# Guess The Output



**Output:** pYThoN prOgRaMmInG

**Explanation:**

In this code snippet, we convert every element in odd index to lower case and every element in even index to uppercase.

# Guess The Output



Language: Python

```
a = "4, 5"  
nums = a.split(',')  
x, y = nums  
int_prod = int(x) * int(y)  
print(int_prod)
```

# Guess The Output



**Output:** 20

**Explanation:**

In this code snippet, we break the given string into its 2 integer components, and then find their product, considering them as integer types.

# Guess The Output



Language: Python

```
square = lambda x: x ** 2
a = []
for i in range(5):
    a.append(square(i))

print(a)
```

# Guess The Output



**Output:** [0,1,4,9,16]

**Explanation:**

The above code snippet stores the 1st 5 perfect squares into a list. The perfect squares are evaluated using a lambda function.

# Guess The Output



Language: Python

```
set1 = {1, 3, 5}  
set2 = {2, 4, 6}  
print(len(set1 + set2))
```

# Guess The Output



**Output:** Error

**Explanation:**

The code will give an error since + operator is not overloaded for sets in Python.

# Guess The Output



Language: Python

```
s1 = {1, 2, 3, 4, 5}  
s2 = {2, 4, 6}  
print(s1 ^ s2)
```

# Guess The Output



**Output:** {1,3,5,6}

**Explanation:**

The `^` operator in sets will return a set containing common of elements of its operand sets.

# Guess The Output



Language: Python

```
a = [1, 2, 3, 4]
b = [3, 4, 5, 6]
c = [x for x in a if x not in b]
print(c)
```

# Guess The Output



**Output:** [1,2]

**Explanation:**

Above code snippet prints the values in a, which are not present in b.

# Guess The Output



Language: Python

```
a = [[], "abc", [0], 1, 0]
print(list(filter(bool, a)))
```

# Guess The Output



**Output:** ['abc',[0],1]

**Explanation:**

The above code filters all the elements from list a, which evaluates to boolean value true, i.e. any non empty string, list or non-zero value is accepted.

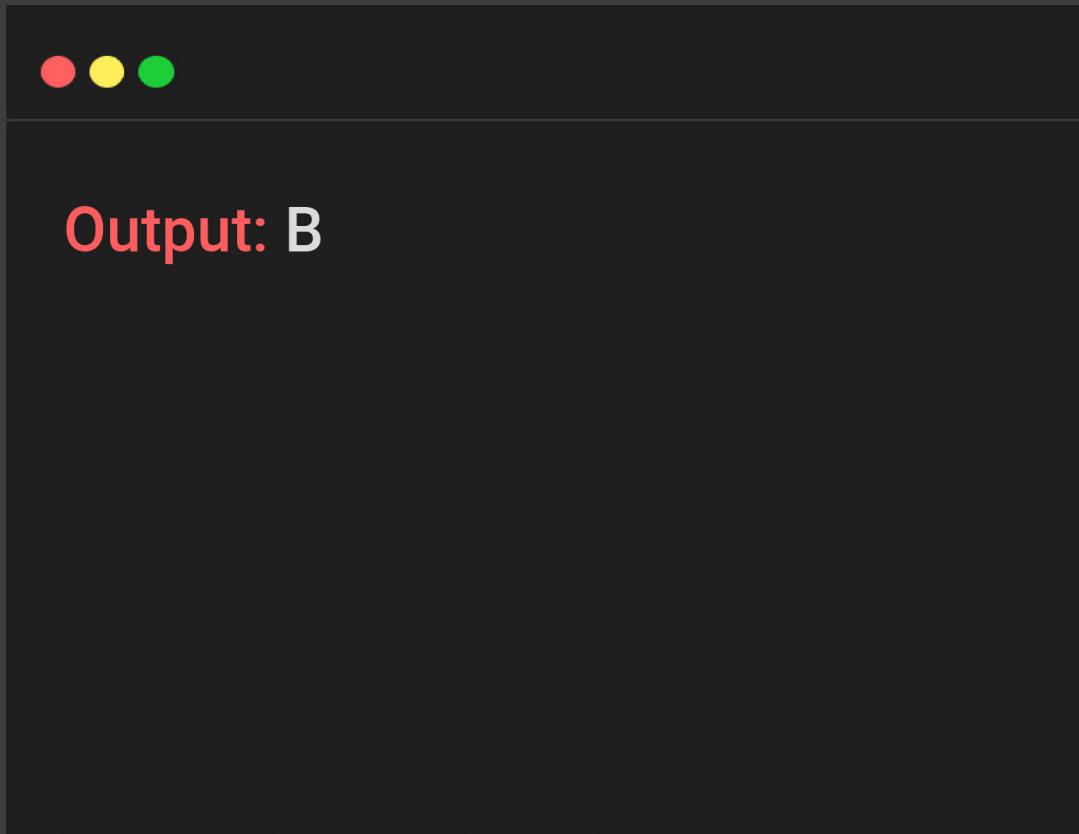
# Guess The Output



Language: Python

```
print(chr(ord(chr(66))))
```

# Guess The Output



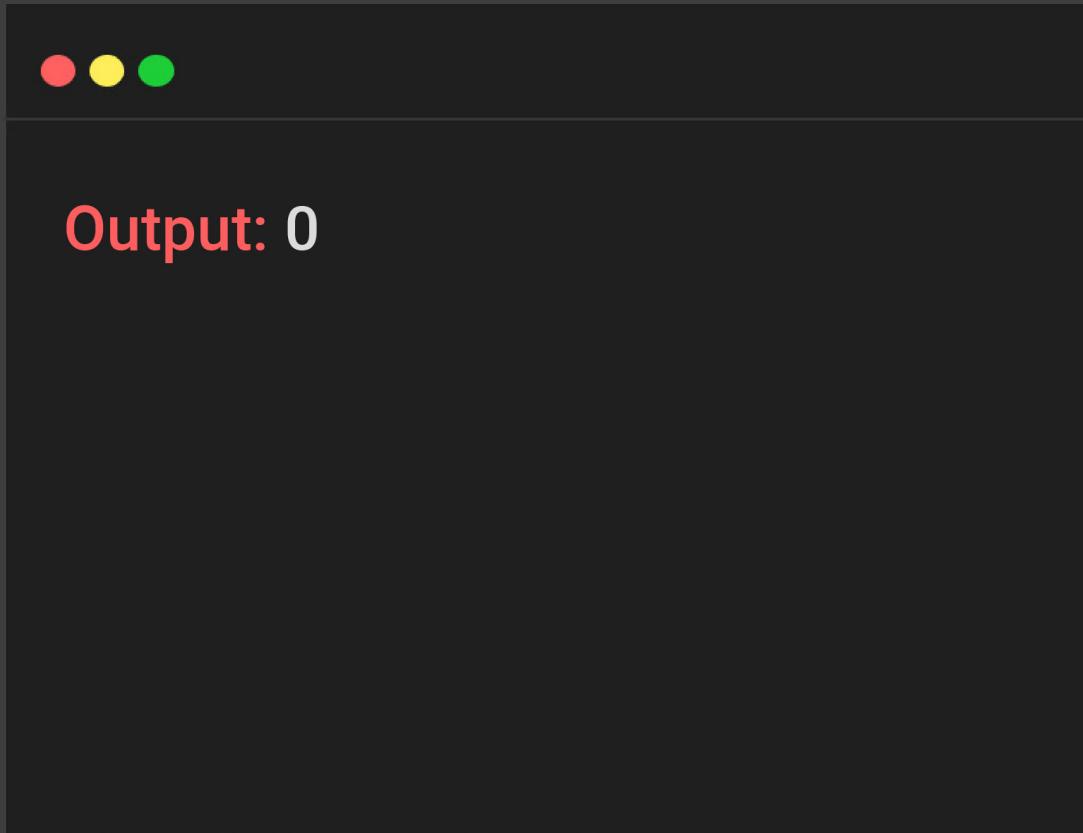
# Guess The Output



Language: Python

```
a,b=1,0
a=a^b
b=a^b
a=a^b
print(a)
```

# Guess The Output



# Guess The Output



Language: Python

`print(2**2**3**1)`

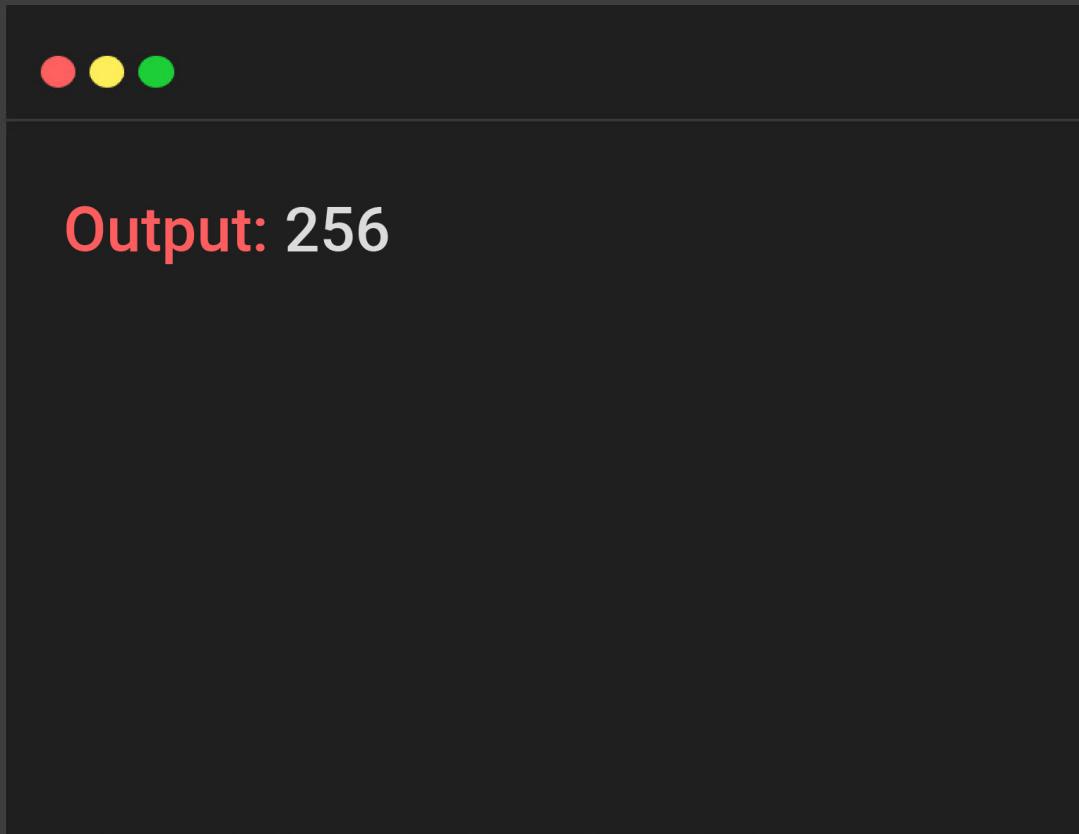


256



64

# Guess The Output



# Guess The Output



Language: Python

```
class A:  
    print("Inside class")  
A()  
A()  
obj=A()
```

# Guess The Output



**Output:** 3 and 1

**Explanation:**

obj is the reference variable here and an object will be created each time A() is called. So there will be 3 objects created.

# Guess The Output



Language: Python

```
class A:  
    def __init__(self):  
        self.count=5  
        self.count=count+1  
    a=A()  
    print(a.count)
```

# Guess The Output



**Output:** error

**Explanation:**

It will throw an error as inside constructor, "count" is not defined.

# Guess The Output



Language: Python

```
class A:  
    def __init__(self, x= 1):  
        self.x = x  
class der(A):  
    def __init__(self,y = 2):  
        super().__init__()  
        self.y = y  
def main():  
    obj = der()  
    print(obj.x, obj.y)  
main()
```



1,2



1,0

# Guess The Output



**Output:** 1,2

**Explanation:**

In the above piece of code, the invoking method has been properly implemented and hence  $x=1$  and  $y=2$ .

# Guess The Output



Language: Python

```
d1={"abc":5,"def":6,"ghi":7}
```

```
print(d1[0])
```

# Guess The Output



**Output:** Error

**Explanation:**

The above code will contain an error. Because 0 is not a key abc, def and ghi are keys to the dictionary.

# Guess The Output



Language: Python

```
dict={"Joey":1,"Rachel":2}
```

```
dict.update({"Phoebe":2})
```

```
print(dict)
```

# Guess The Output



**Output:** {"Joey":1,"Rachel":2,"Phoebe":2}

**Explanation:**

It will simply print the dictionary as  
dict={"Joey":1,"Rachel":2,"Phoebe":2}

# Guess The Output



Language: Python

```
dict1={"a":10,"b":2,"c":3}
str1=""
for i in dict1:
    str1=str1+str(dict1[i])+" "
    str2=str1[:-1]
print(str2[::-1])
```

# Guess The Output



**Output:** 3,2,01

**Explanation:**

3, 2, 01 will be the following Python code output.

# Guess The Output



Language: Python

```
str1="Information"
```

```
print(str1[2:8])
```

# Guess The Output



**Output:** format

**Explanation:**

Concept of slicing is used in this question. In string slicing, the output is the substring starting from the first given index position i.e 2 to one less than the second given index position i.e.  $(8-1=7)$  of the given string str1. Hence, the output will be "format".

# Guess The Output



Language: Python

```
str1="Aplication"
```

```
str2=str1.replace('a','A')
```

```
print(str2)
```

# Guess The Output



**Output:** ApplicAion

**Explanation:**

Replace() function in string is used here to replace all the existing "a" by "A" in the given string.

# Guess The Output



Language: Python

```
str1="poWer"
```

```
str1.upper()
```

```
print(str1)
```

# Guess The Output



Language: Python

```
list1=[0,2,5,1]
str1="7"
for i in list1:
    str1=str1+i
print(str1)
```

# Guess The Output



**Output:** error

**Explanation:**

list1 contains integers as its elements. Hence these cannot be concatenated to string str1 by simple "+" operand. These should be converted to string first by use of str() function, then only these will get concatenated.

# Guess The Output



Language: Python

```
str1="Stack of books"  
print(len(str1))
```

# Guess The Output



**Output:** 14

**Explanation:**

`len()` returns the length of the given string `str1`, including spaces and considering " " as a single character.

# Guess The Output



Language: Python

```
'p' + 'q' if '12'.isdigit() else 'r' + 's'
```

# Guess The Output



**Output:** pq

**Explanation:**

If condition is true so pq will be the output. So, option A is correct.

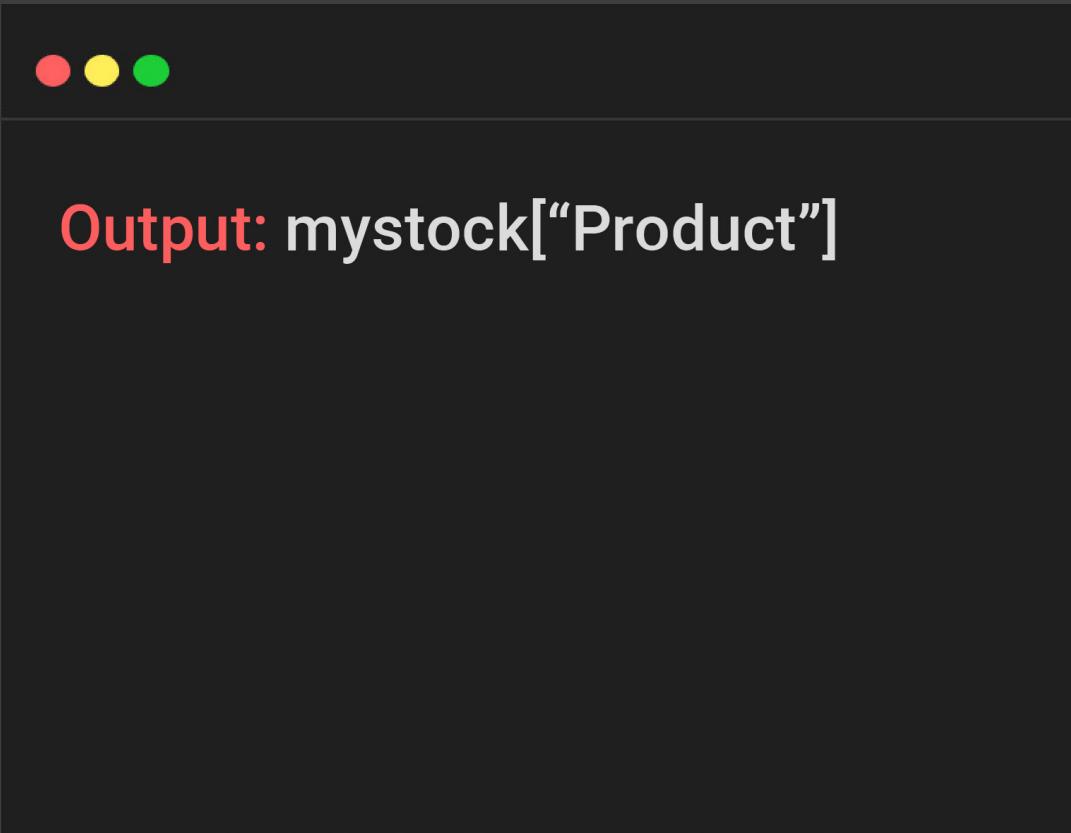
# Guess The Output



Language: Python

```
mystock = {  
    "Product": "Earphone",  
    "Price": 800,  
    "Quantity": 50,  
    "InStock" : "Yes"  
}
```

# Guess The Output



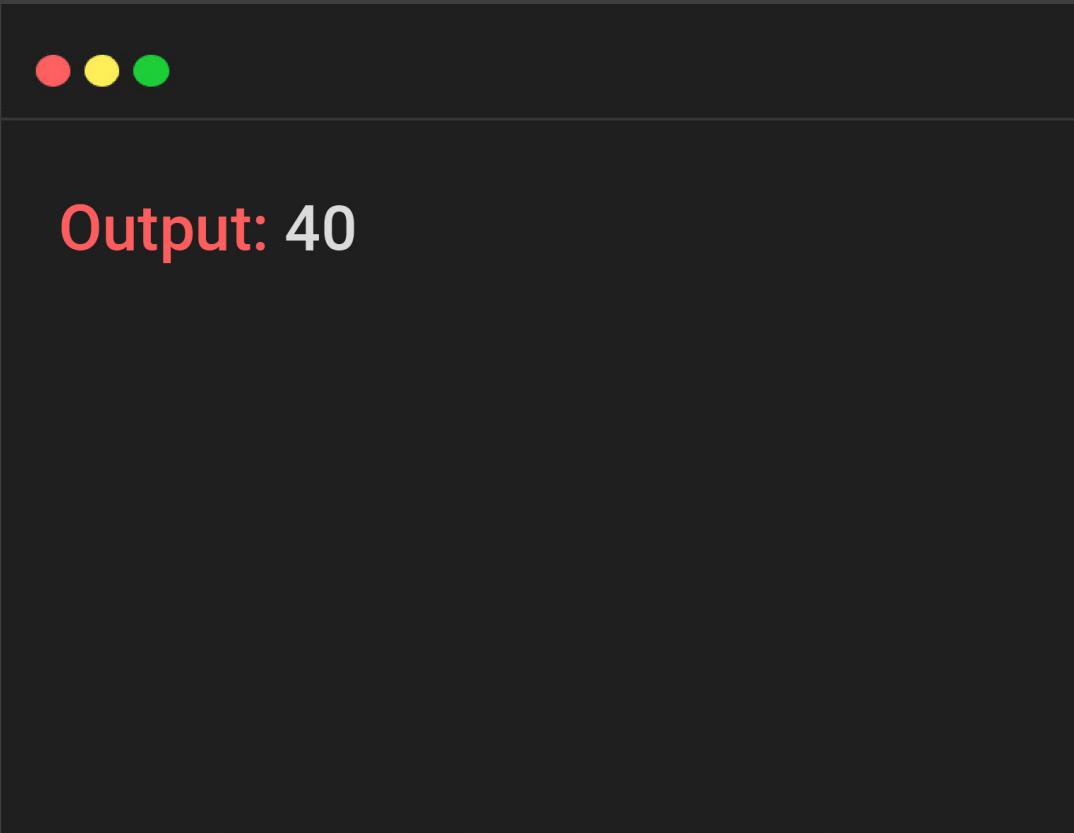
# Guess The Output



Language: Python

```
d = {"john":40, "peter":45}  
d["john"]
```

# Guess The Output



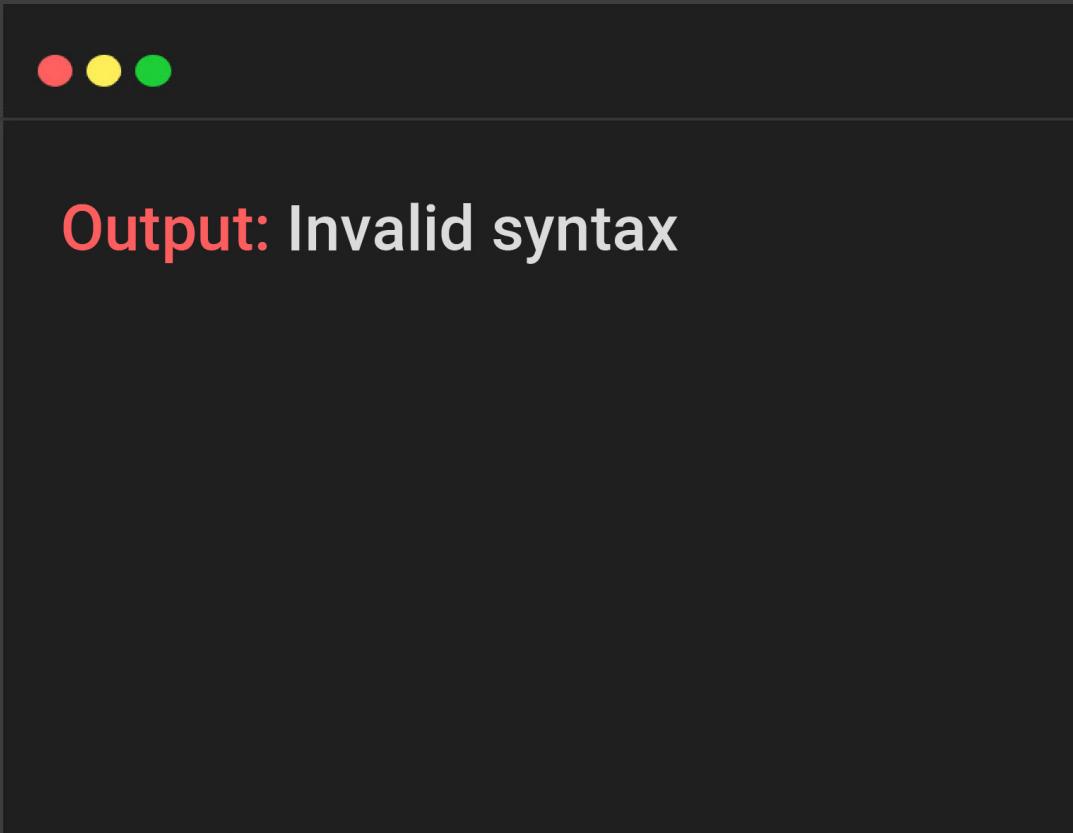
# Guess The Output



Language: Python

```
i = 1:  
while True:  
    if i%3 == 0:  
        break  
    print(i)
```

# Guess The Output



# Guess The Output



Language: Python

```
exp="5%2+3*7"  
print(eval(exp))
```

# Guess The Output



**Output:** 22

**Explanation:**

The `eval()` function evaluates the specified expression, if the expression is a legal Python statement, it will be executed. `eval()` function used to evaluate infix expression.

# Guess The Output



Language: Python

```
mylist = iter(["Bumrah", "Virat", "Hardik"])
x = next(mylist)
print(x)
```

# Guess The Output



**Output:** Bumrah

**Explanation:**

Next() gives the next iterator in given list .

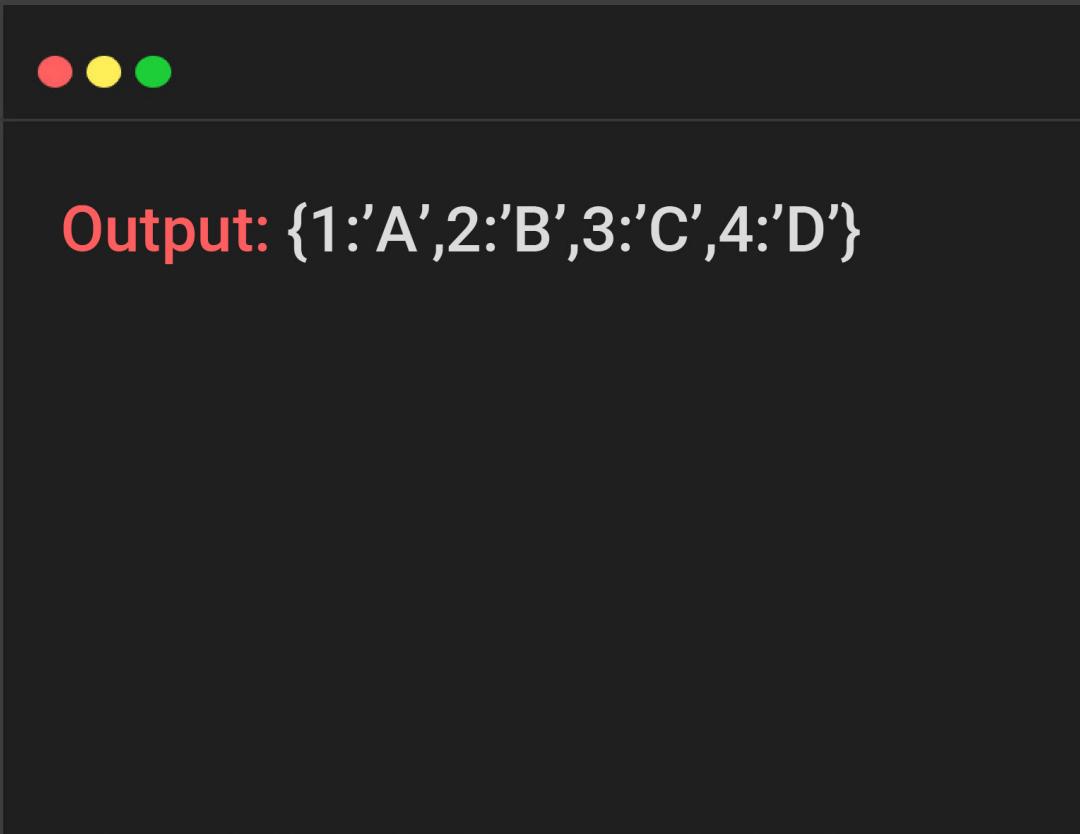
# Guess The Output



Language: Python

```
a={"1":"A","2":"B","3":"C"}  
a.setdefault(4,"D")  
print(a)
```

# Guess The Output



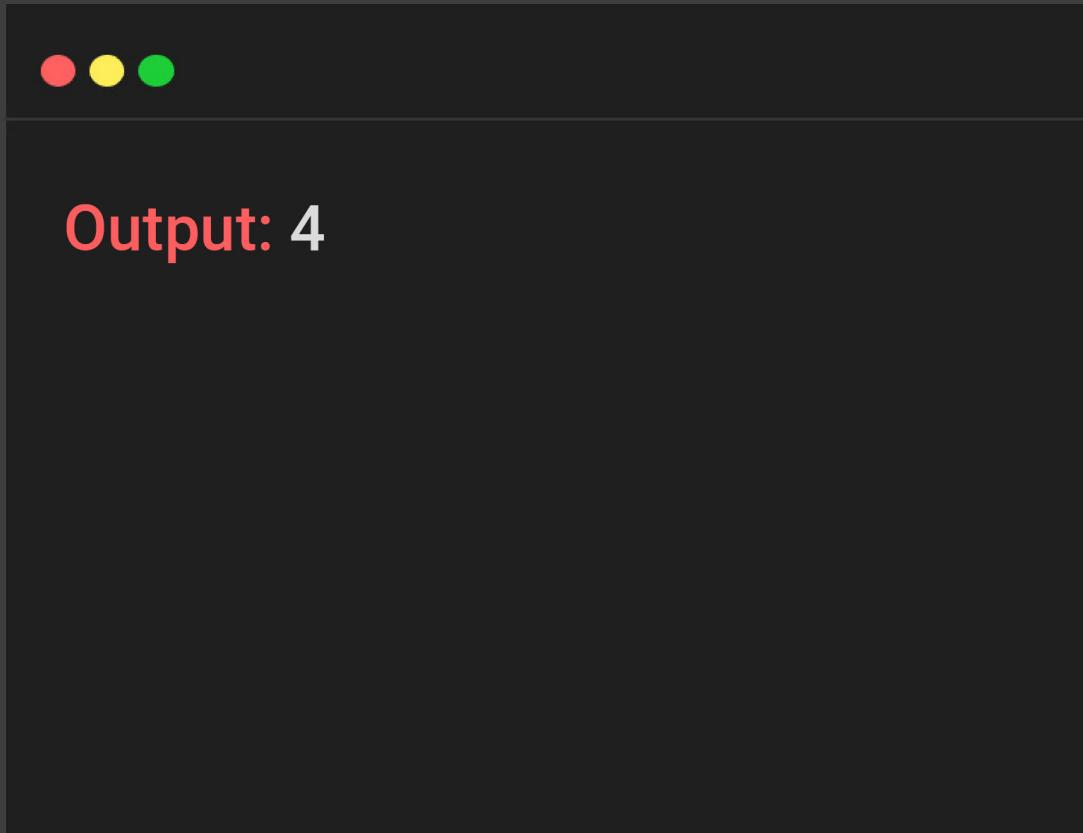
# Guess The Output



Language: Python

```
a={"1":"A","2":"B","3":"C"}  
print(a.get(5,4))
```

# Guess The Output



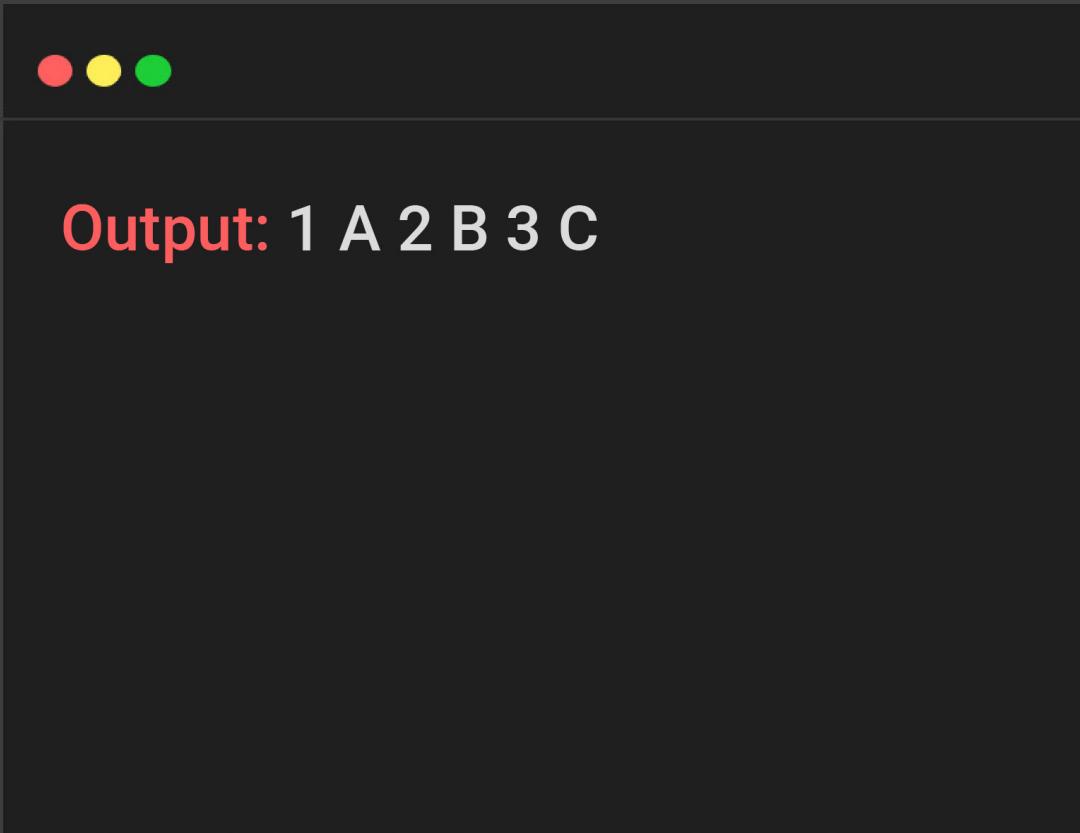
# Guess The Output



Language: Python

```
a={"A":1,"B":2,"C":3}
for i,j in a.items():
    print(i,j,end=" ")
```

# Guess The Output



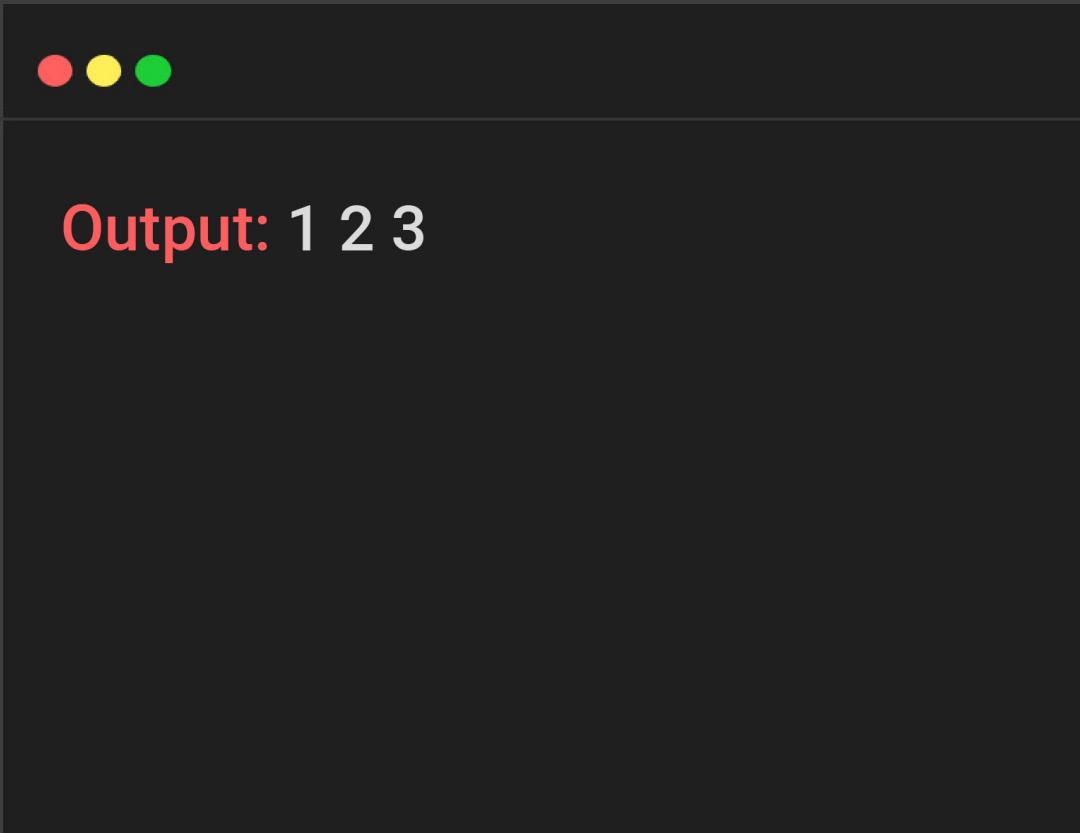
# Guess The Output



Language: Python

```
a={"A":1,"B":2,"C":3}
for i in a:
    print(i,end=" ")
```

# Guess The Output



# Guess The Output



Language: Python

```
d={"john":40,"peter":45}  
print(list(d.keys()))
```

# Guess The Output



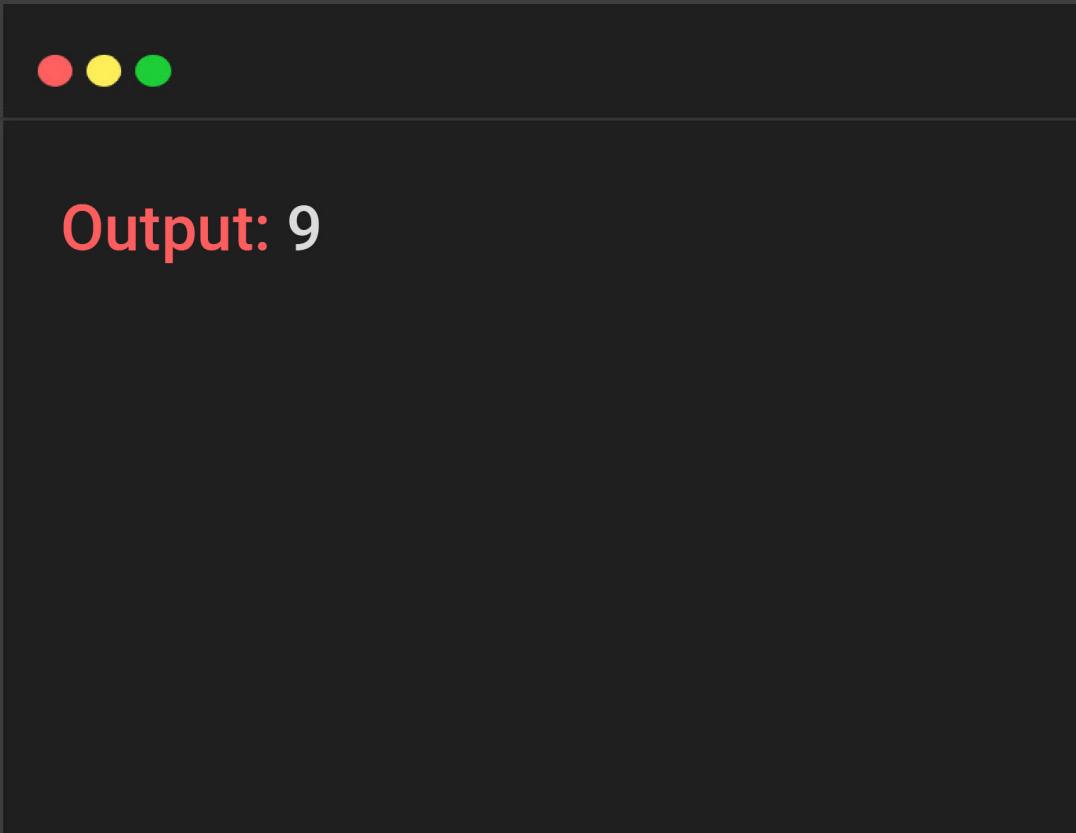
# Guess The Output



Language: Python

```
a={1:5,2:3,3:4}  
print(a.pop(4,9))
```

# Guess The Output



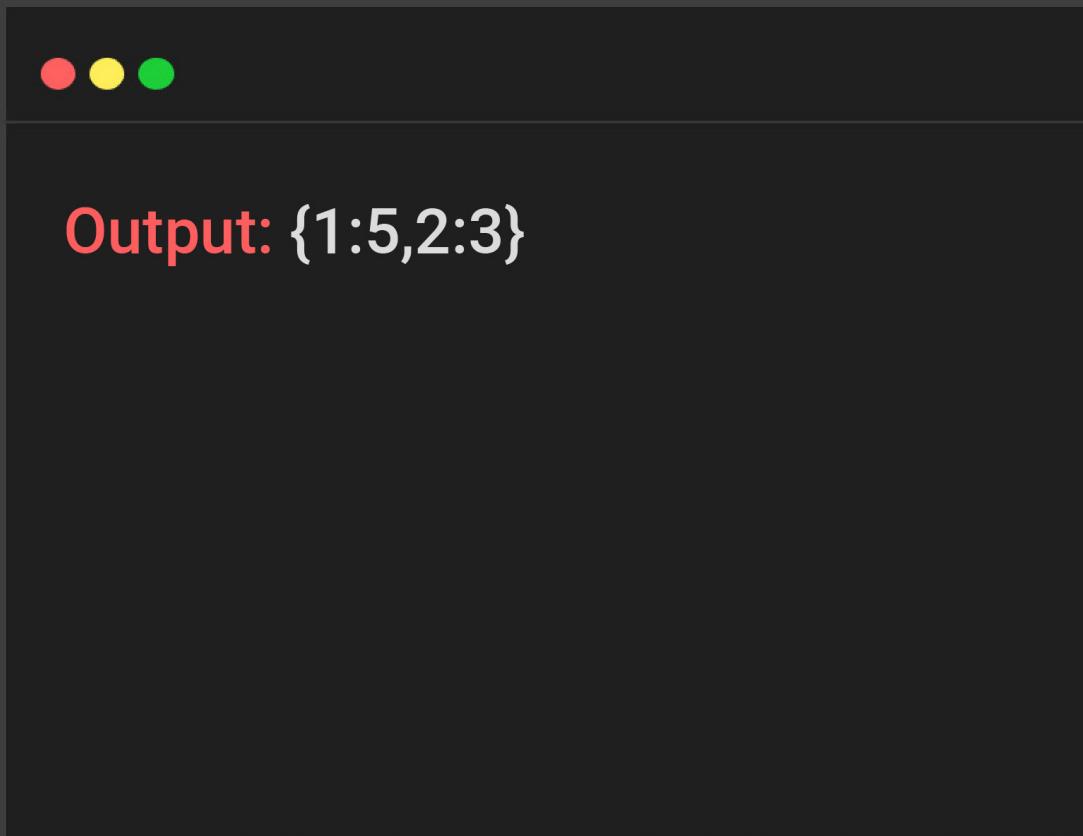
# Guess The Output



Language: Python

```
a={1:5,2:3,3:4}  
a.pop(3)  
print(a)
```

# Guess The Output



# Guess The Output

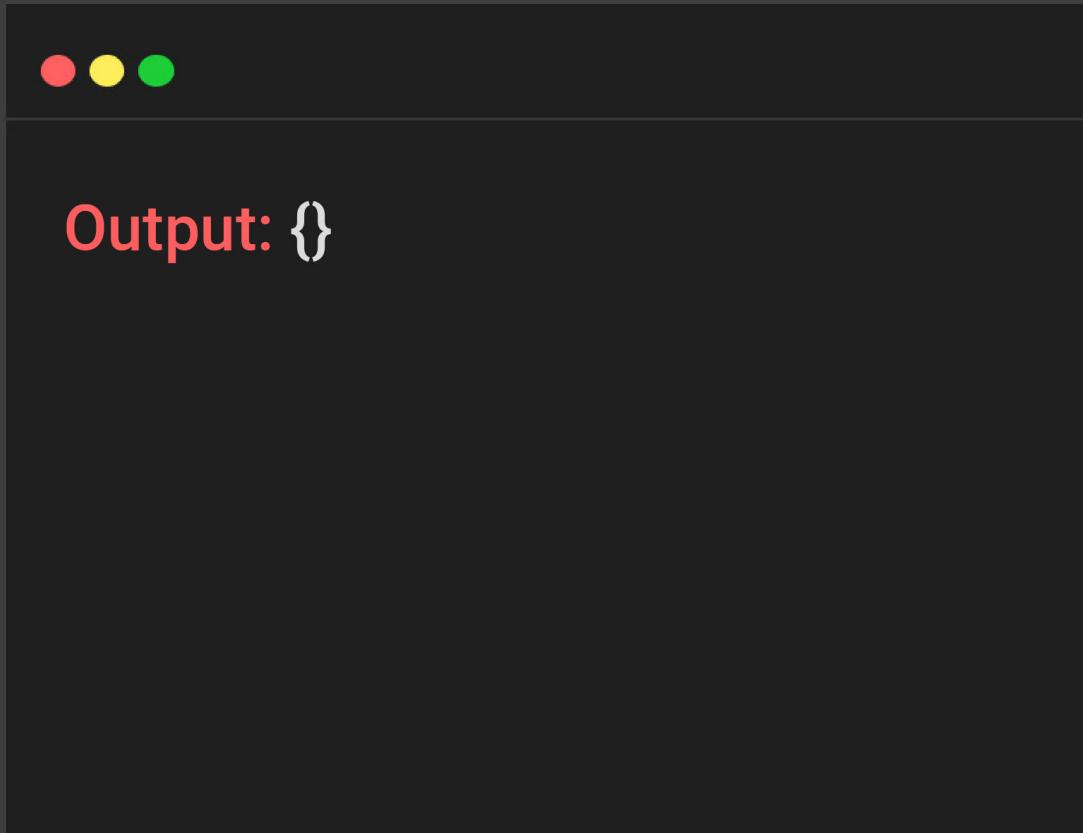


Language: Python

```
a={"A":1,"A":2,"C":3}  
a.clear()  
print(a)
```



# Guess The Output



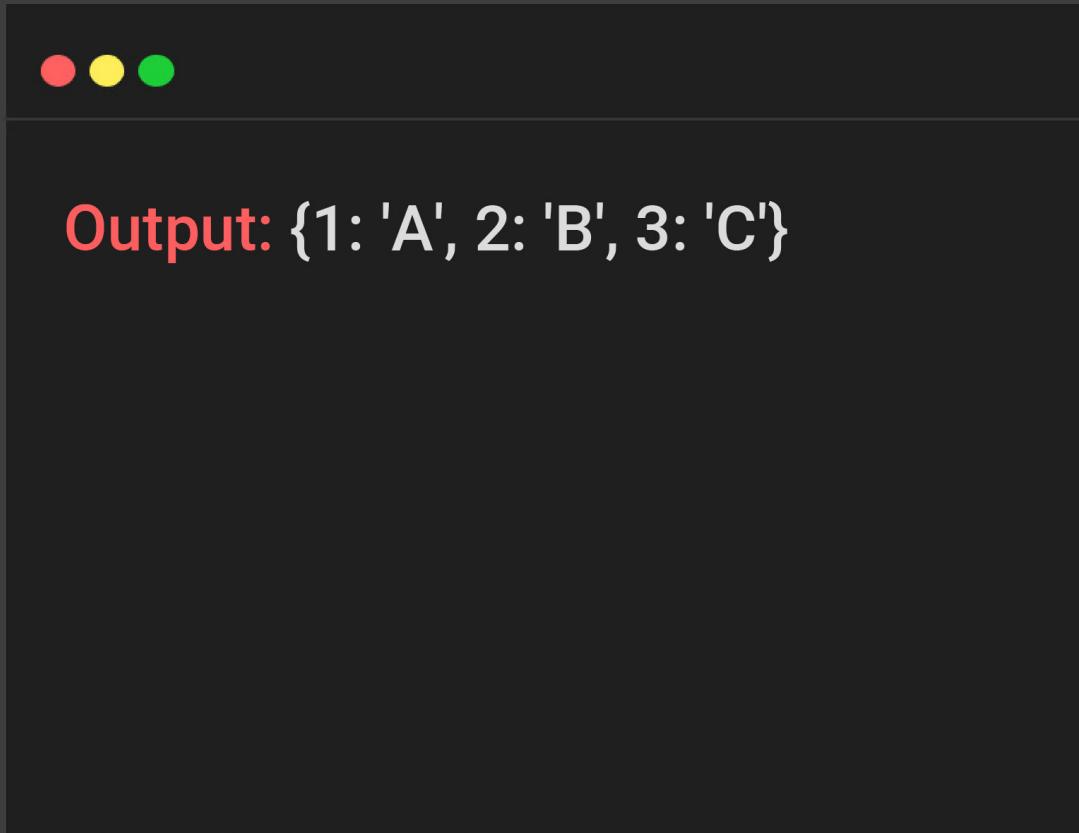
# Guess The Output



Language: Python

```
a={"1":"A","2":"B","3":"C"}  
b=a.copy()  
b[2]="D"  
print(a)
```

# Guess The Output



# Guess The Output



Language: Python

```
myint = 10
```

```
def myfunction():
```

```
    myint = 20
    myfunction()
```

```
    print(myint)
```

# Guess The Output



**Output:** 10

**Explanation:**

Whenever we reallocate a global variable in the local scope of a function, the relocation only holds inside the local scope. The variable goes back to the global value when the code returns the global scope.

# Guess The Output



Language: Python

```
myint = 21
if False:
    myint = 34
def myfunction():
    if True:
        myint = 65
myfunction()
print(myint)
```

# Guess The Output



**Output:** 21

**Explanation:**

The variable, `myint`, is initially assigned to 21. The `if False` statement evaluates to `False`, so the `myint = 34` reassignment never occurs. The reassignment within the `myfunction()` function does occur inside the local scope, not the global one. When we try to access the variable `myint` in the global scope, the variable's value remains unchanged at 21.

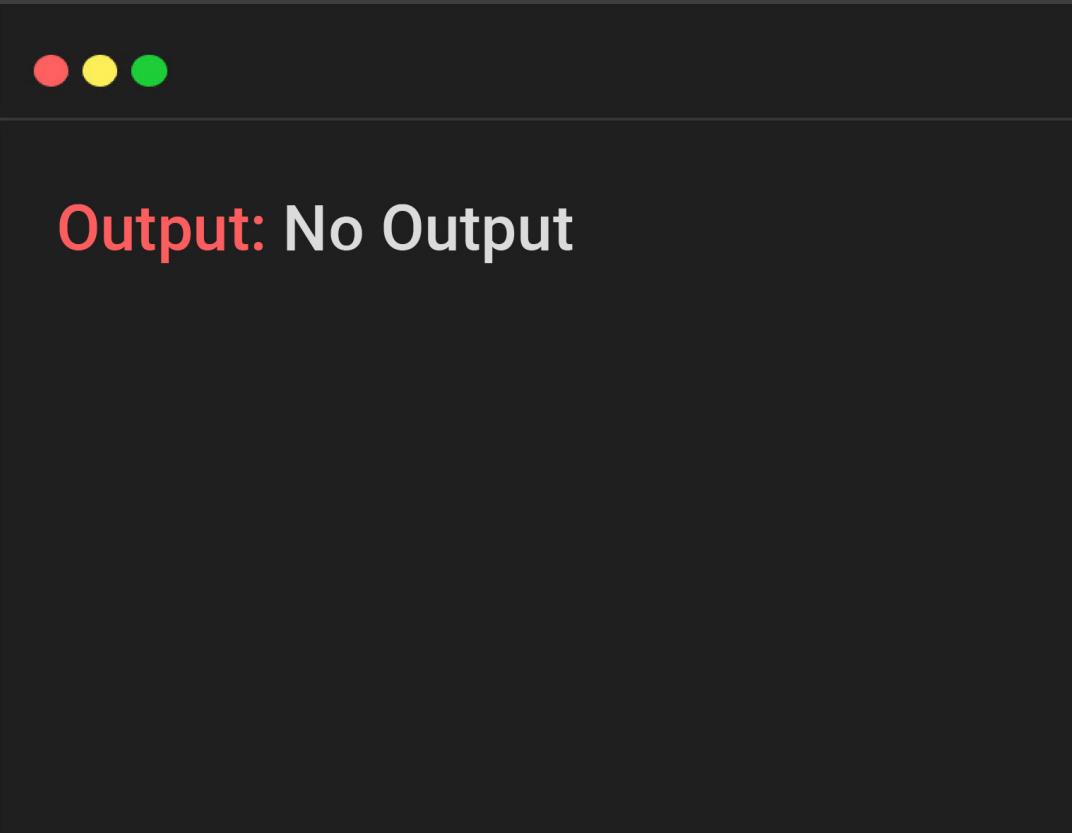
# Guess The Output



Language: Python

```
class A:  
    @staticmethod  
    def a(x):  
        print(x)  
        A.a(100)
```

# Guess The Output



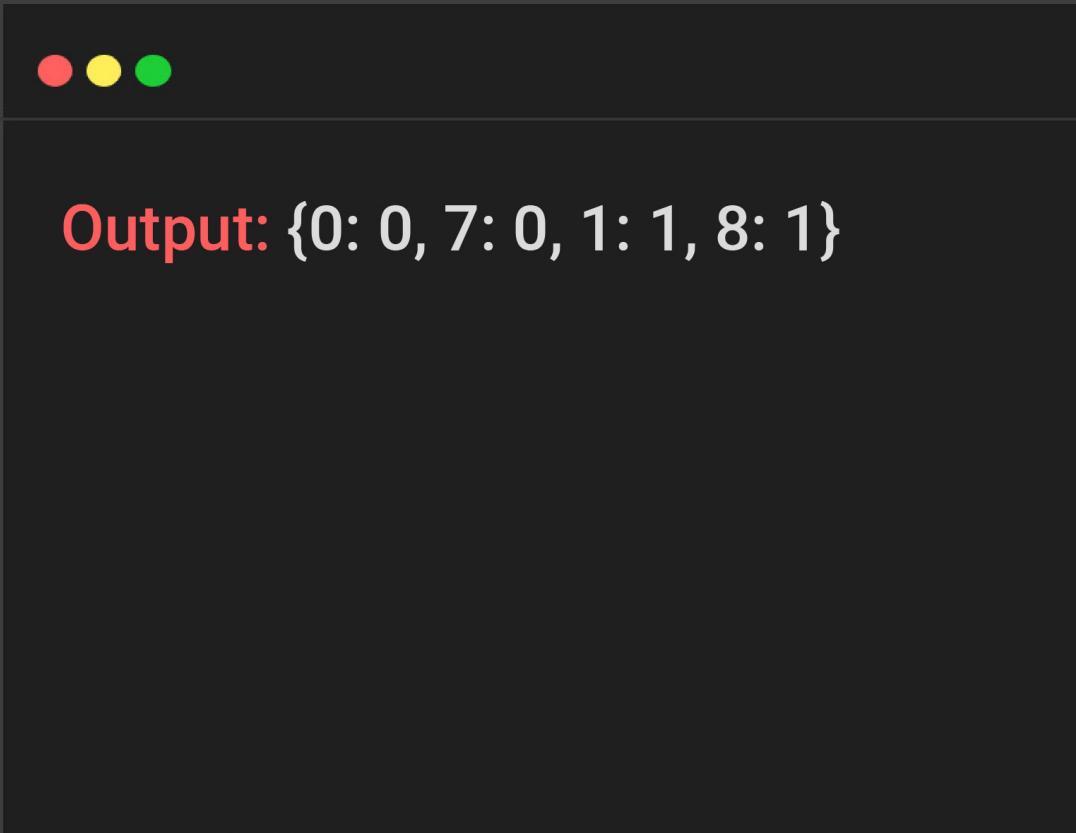
# Guess The Output



Language: Python

```
D = dict()  
for x in enumerate(range(2)):  
    D[x[0]] = x[1]  
    D[x[1]+7] = x[0]  
print(D)
```

# Guess The Output



# Guess The Output

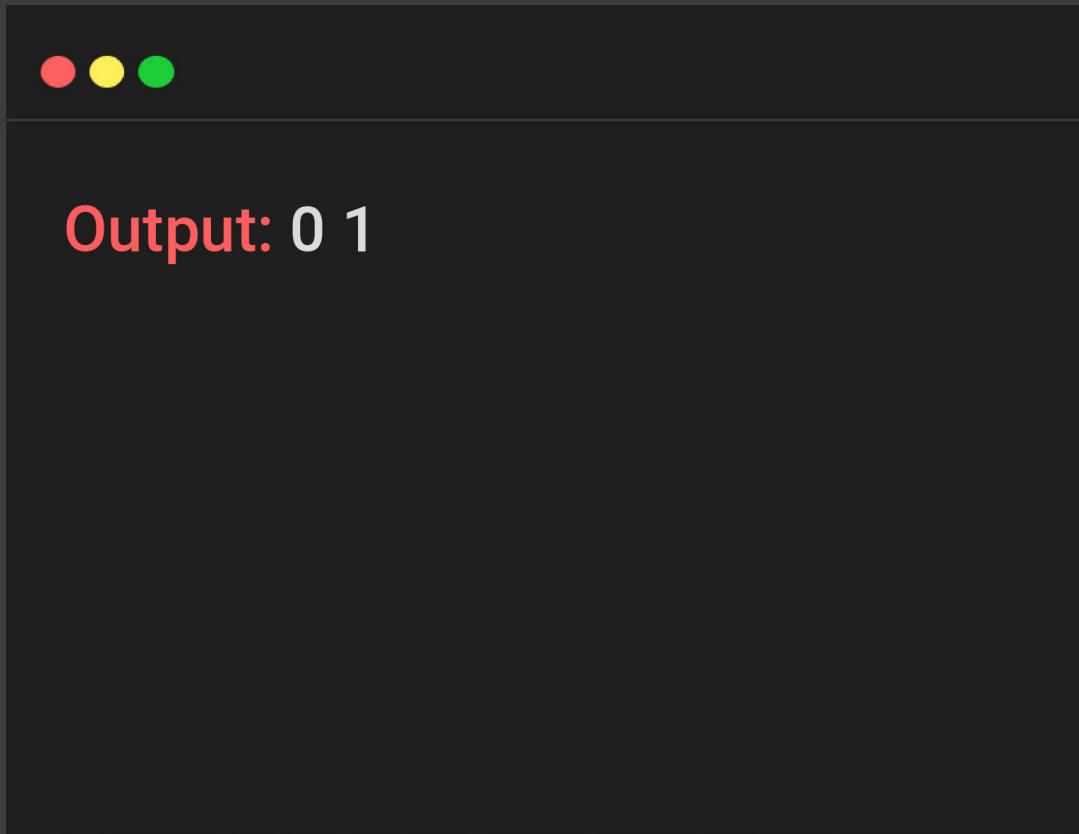


Language: Python

```
class Test:  
    def __init__(self):  
        self.x = 0  
class Derived_Test(Test):  
    def __init__(self):  
        Test.__init__(self)  
        self.y = 1  
def main():  
    b = Derived_Test()  
    print(b.x,b.y)  
main()
```

cs mock

# Guess The Output



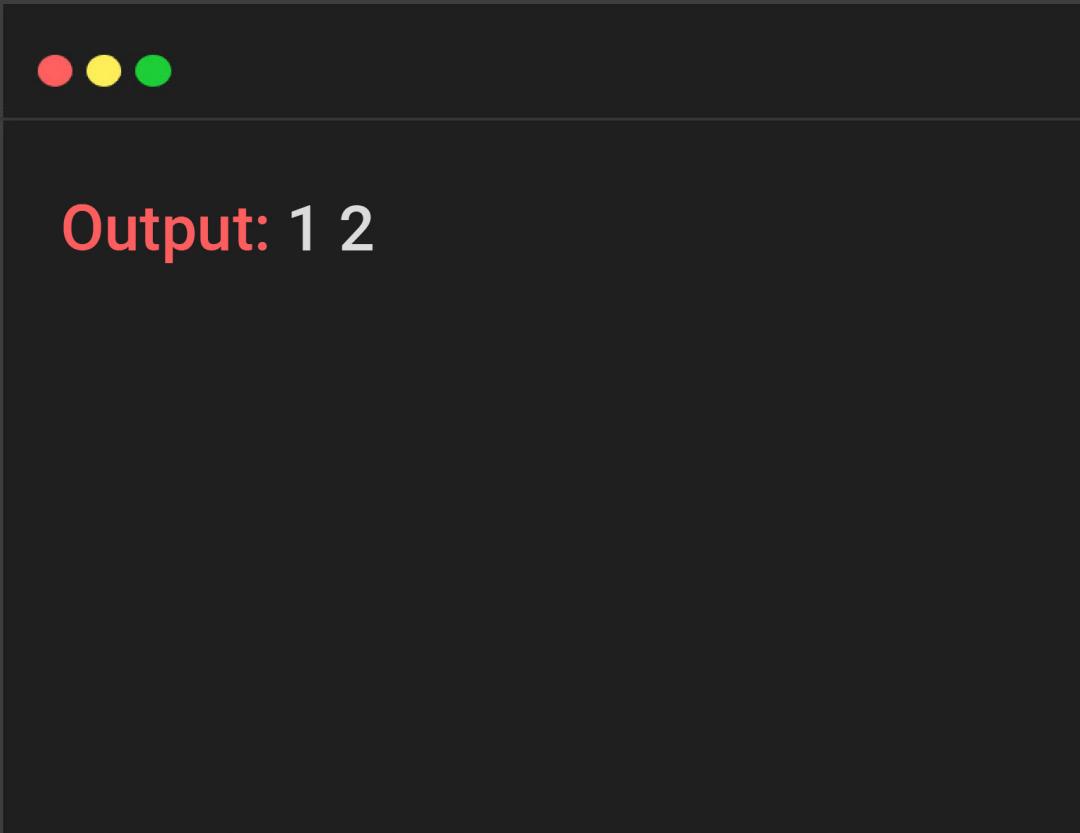
# Guess The Output



Language: Python

```
class A:  
    def __init__(self, x= 1):  
        self.x = x  
class der(A):  
    def __init__(self,y = 2):  
        super().__init__()  
        self.y = y  
def main():  
    obj = der()  
    print(obj.x, obj.y)  
main()
```

# Guess The Output



# Guess The Output

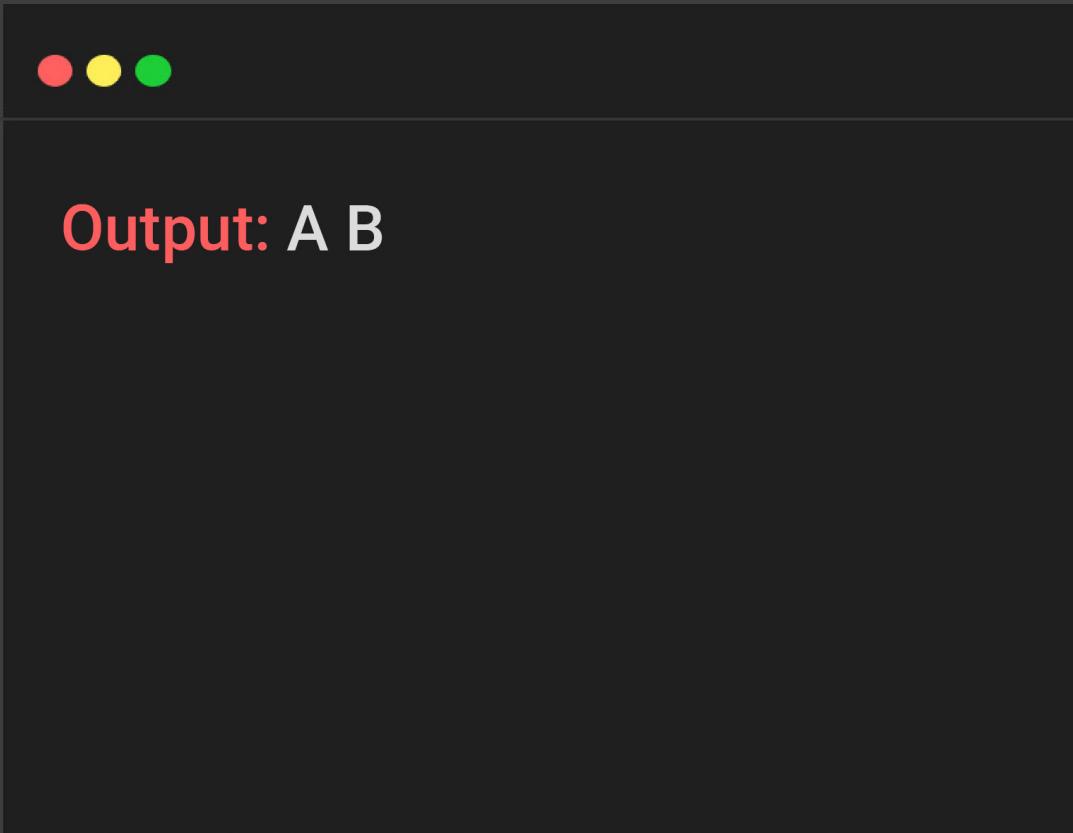


Language: Python

```
class A:  
    def one(self):  
        return self.two()  
    def two(self):  
        return 'A'  
class B(A):  
    def two(self):  
        return 'B'  
obj1=A()  
obj2=B()  
print(obj1.two(),obj2.two())
```

cs mock

# Guess The Output



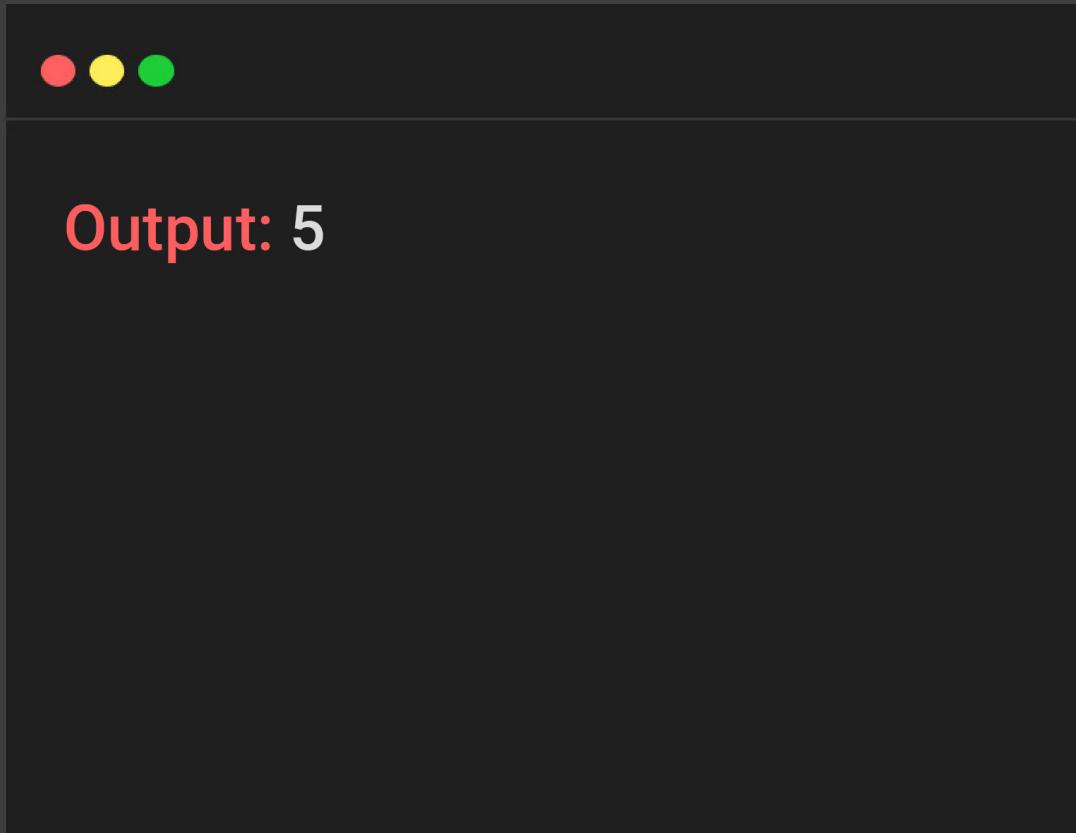
# Guess The Output



Language: Python

```
class A:  
    def __init__(self):  
        self._x = 5  
class B(A):  
    def display(self):  
        print(self._x)  
def main():  
    obj = B()  
    obj.display()  
main()
```

# Guess The Output



# Guess The Output

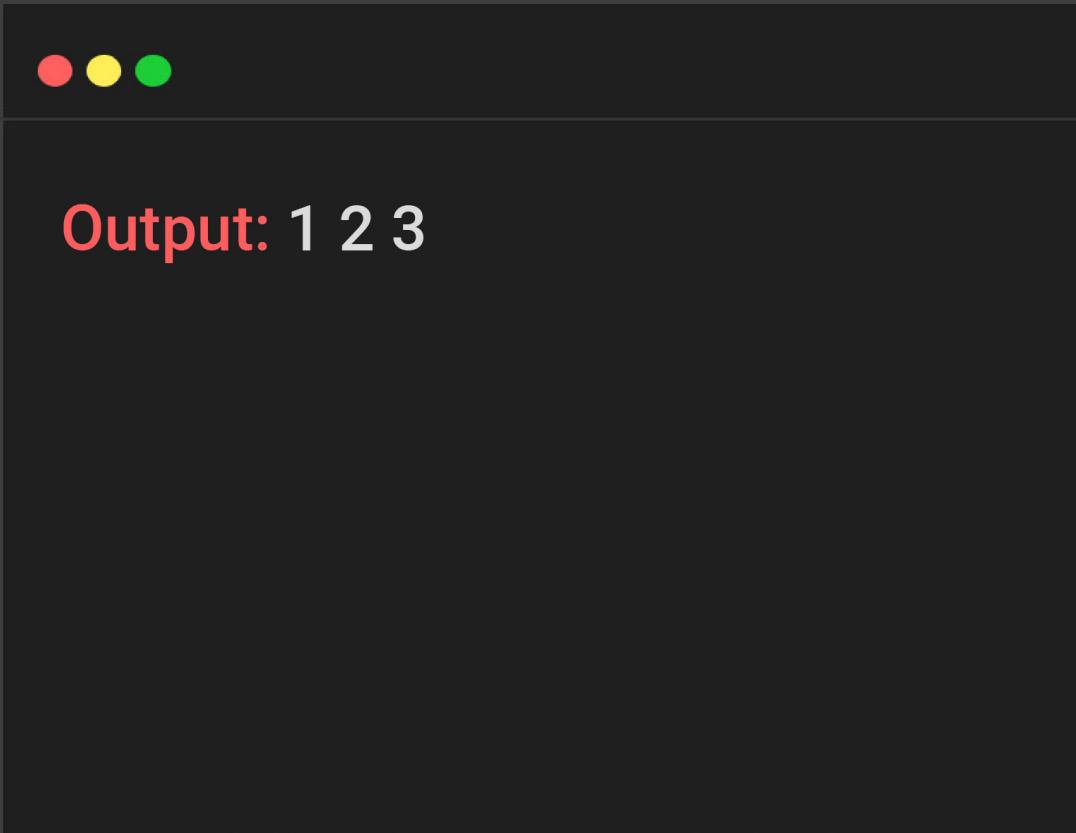


Language: Python

```
class A:  
    def __repr__(self):  
        return "1"  
class B(A):  
    def __repr__(self):  
        return "2"  
class C(B):  
    def __repr__(self):  
        return "3"  
o1 = A()  
o2 = B()  
o3 = C()  
print(obj1, obj2, obj3)
```

cs mock

# Guess The Output



# Guess The Output

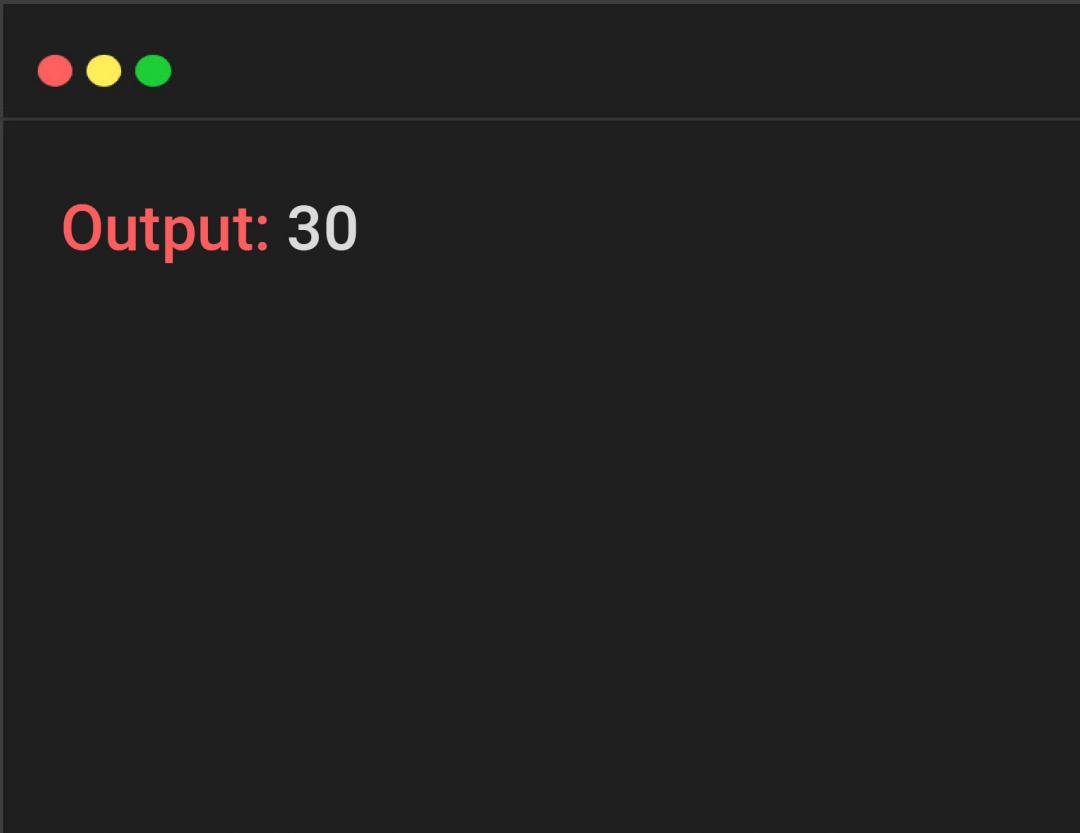


Language: Python

```
class A:  
    def __init__(self):  
        self.multiply(15)  
        print(self.i)  
    def multiply(self, i):  
        self.i = 4 * i;  
class B(A):  
    def __init__(self):  
        super().__init__()  
    def multiply(self, i):  
        self.i = 2 * i;  
obj = B()
```

cs mock

# Guess The Output



# Guess The Output

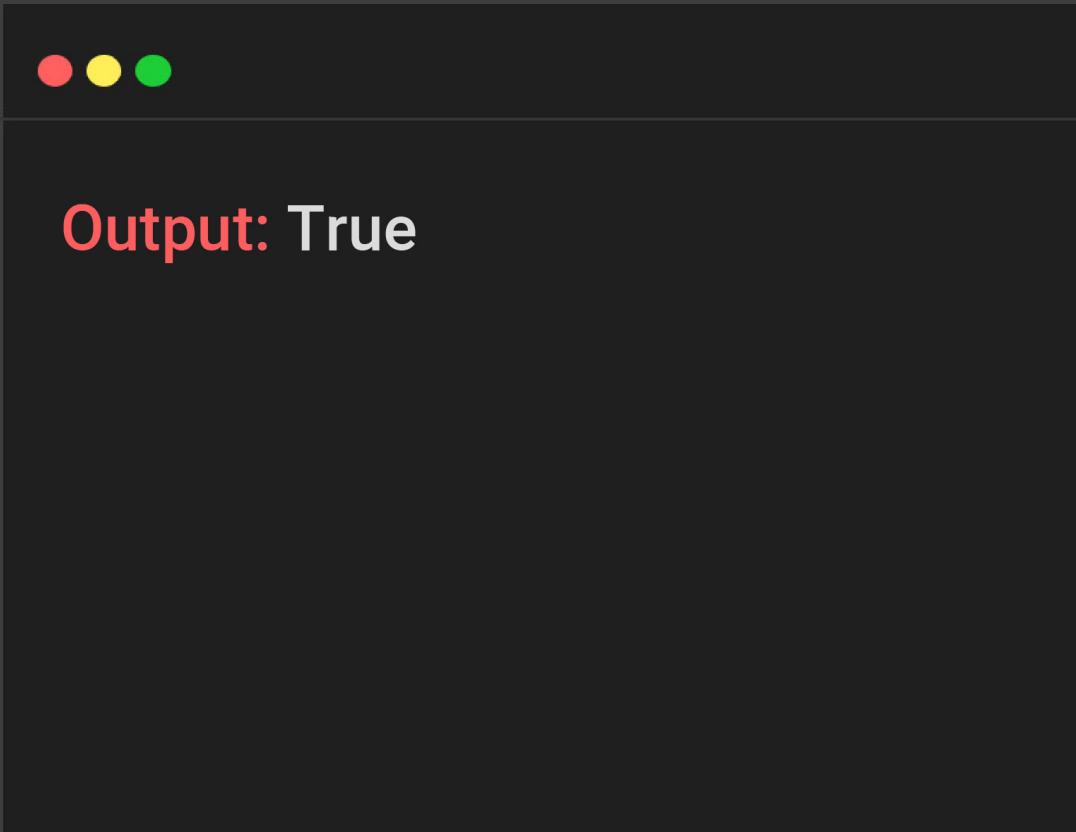


Language: Python

```
class A:  
    def __init__(self, x, y):  
        self.x = x  
        self.y = y  
    def __str__(self):  
        return 1  
    def __eq__(self, other):  
        return self.x * self.y == other.x * other.y  
obj1 = A(5, 2)  
obj2 = A(2, 5)  
print(obj1 == obj2)
```

cs mock

# Guess The Output



# Guess The Output



Language: Python

```
class A:  
    def one(self):  
        return self.two()  
    def two(self):  
        return 'A'  
class B(A):  
    def two(self):  
        return 'B'  
obj2=B()  
print(obj2.two())
```

cs mock

# Guess The Output



**Output:** An exception is thrown

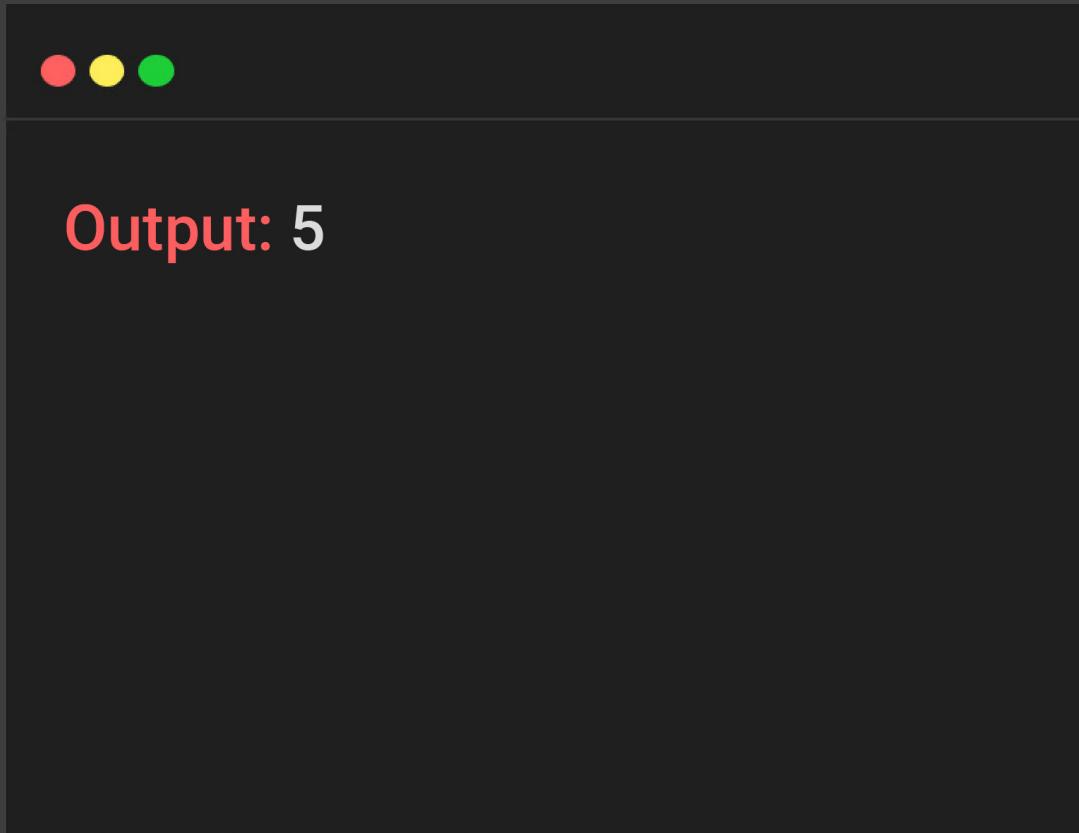
# Guess The Output



Language: Python

```
class fruits:  
    def __init__(self):  
        self.price = 100  
        self.__bags = 5  
    def display(self):  
        print(self.__bags)  
obj=fruits()  
obj.display()
```

# Guess The Output



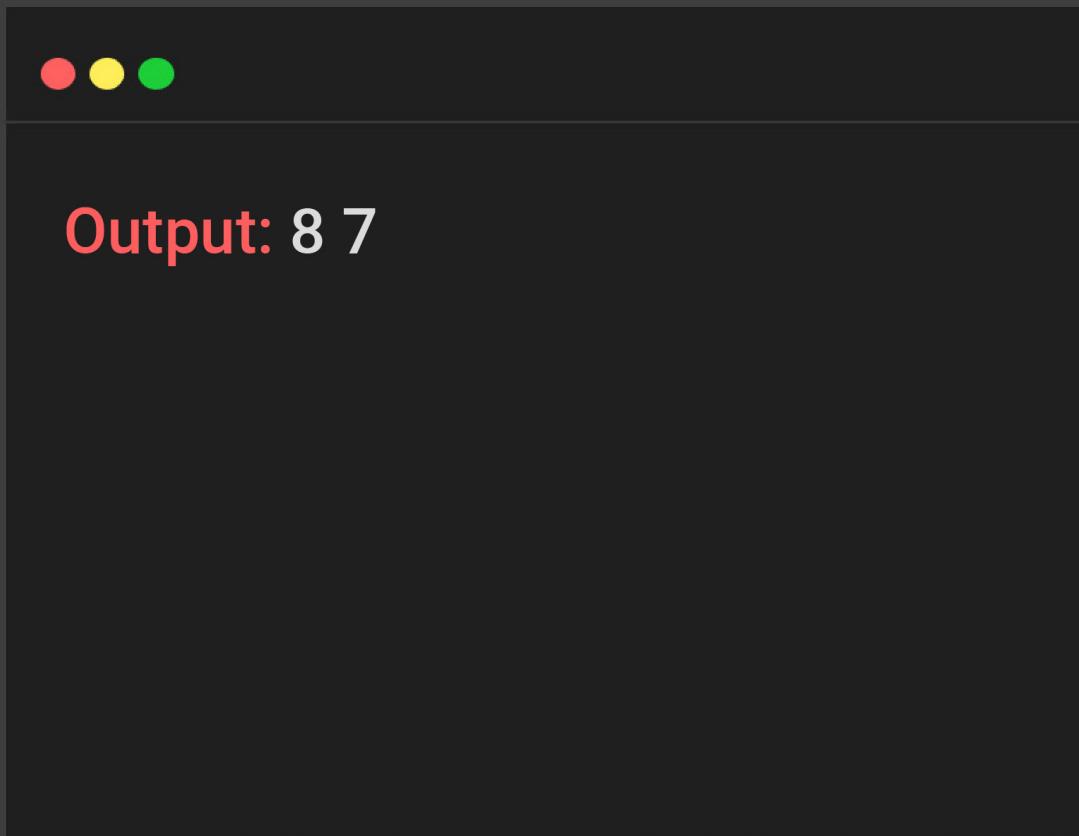
# Guess The Output



Language: Python

```
class student:  
    def __init__(self):  
        self.marks = 97  
        self.__cgpa = 8.7  
    def display(self):  
        print(self.marks)  
obj=student()  
print(obj._student__cgpa)
```

# Guess The Output



# Guess The Output



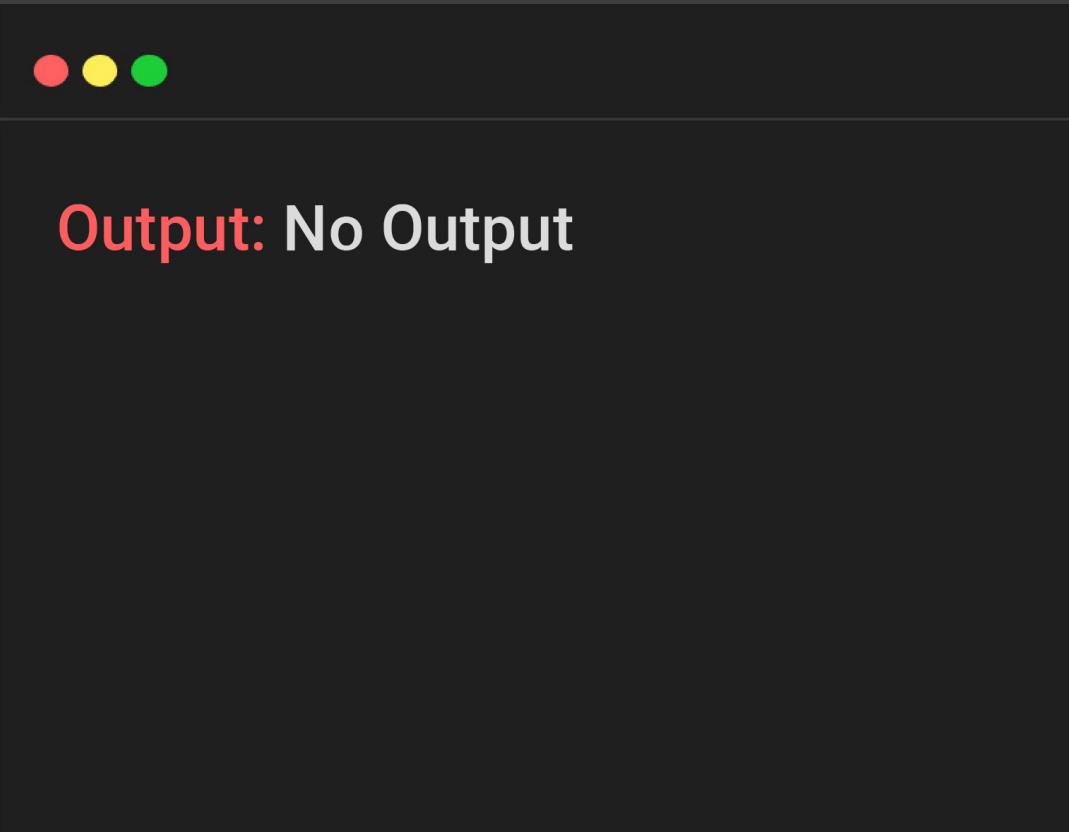
Language: Python

x=10

y=8

assert x>y, 'X too small'

# Guess The Output



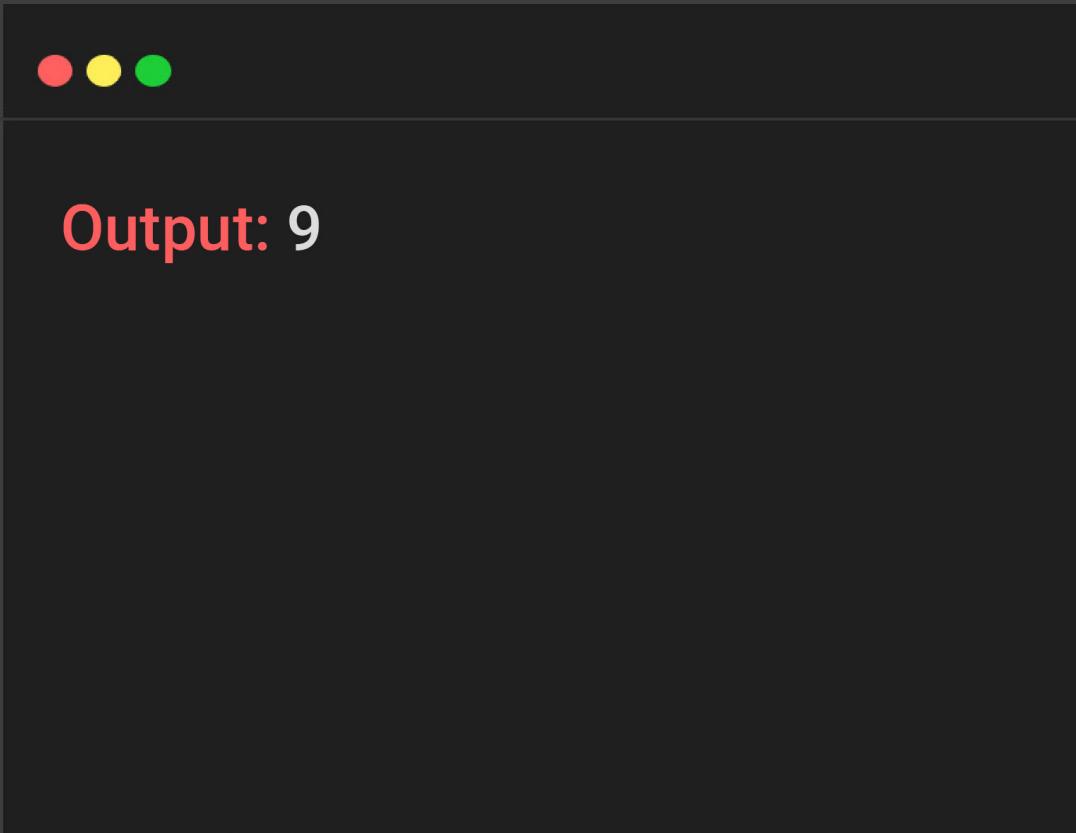
# Guess The Output



Language: Python

```
#generator
def f(x):
    yield x+1
g=f(8)
print(next(g))
```

# Guess The Output



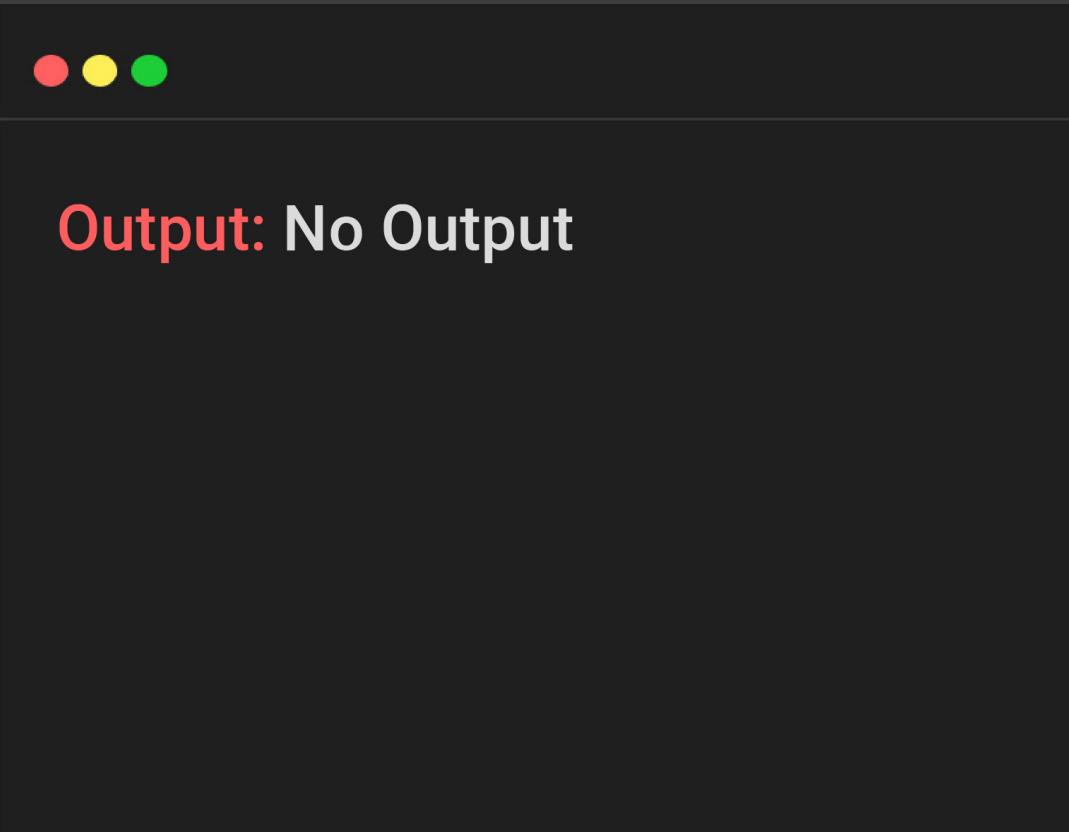
# Guess The Output



Language: Python

```
#generator
def f(x):
    yield x+1
g=f(8)
print(next(g))
```

# Guess The Output



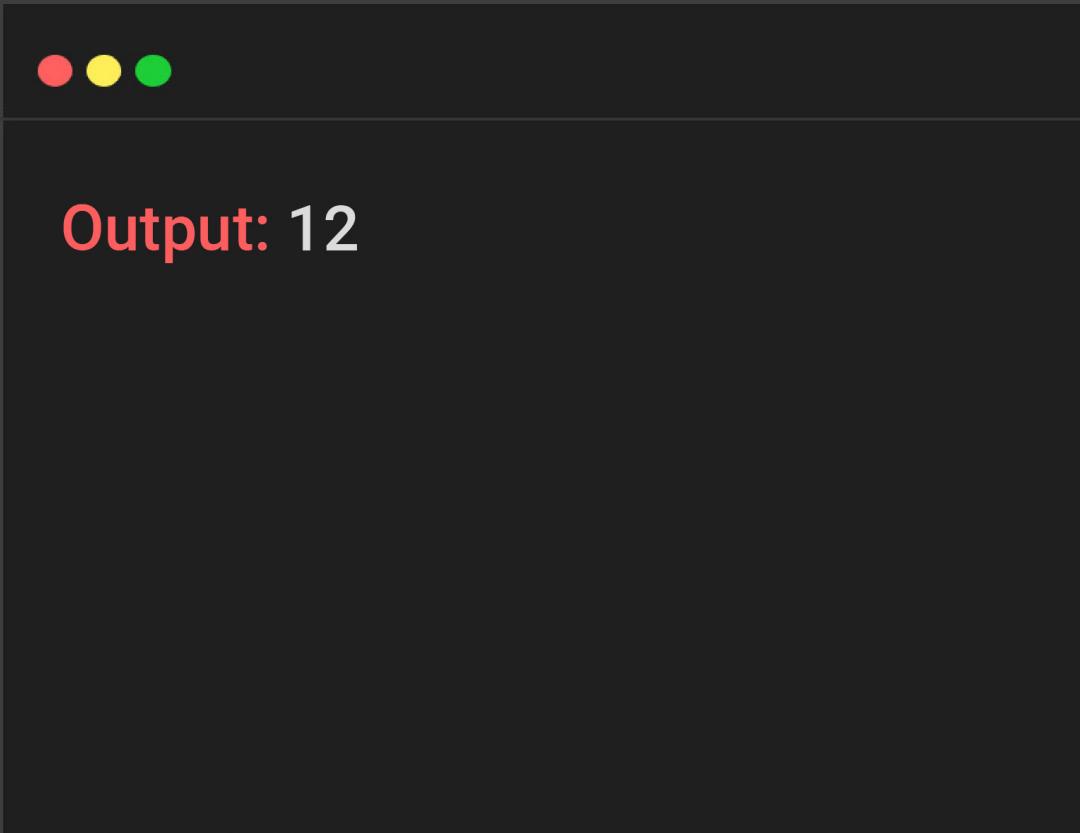
# Guess The Output



Language: Python

```
def f(x):  
    yield x+1  
    print("test")  
    yield x+2  
g=f(10)  
print(next(g))  
print(next(g))
```

# Guess The Output



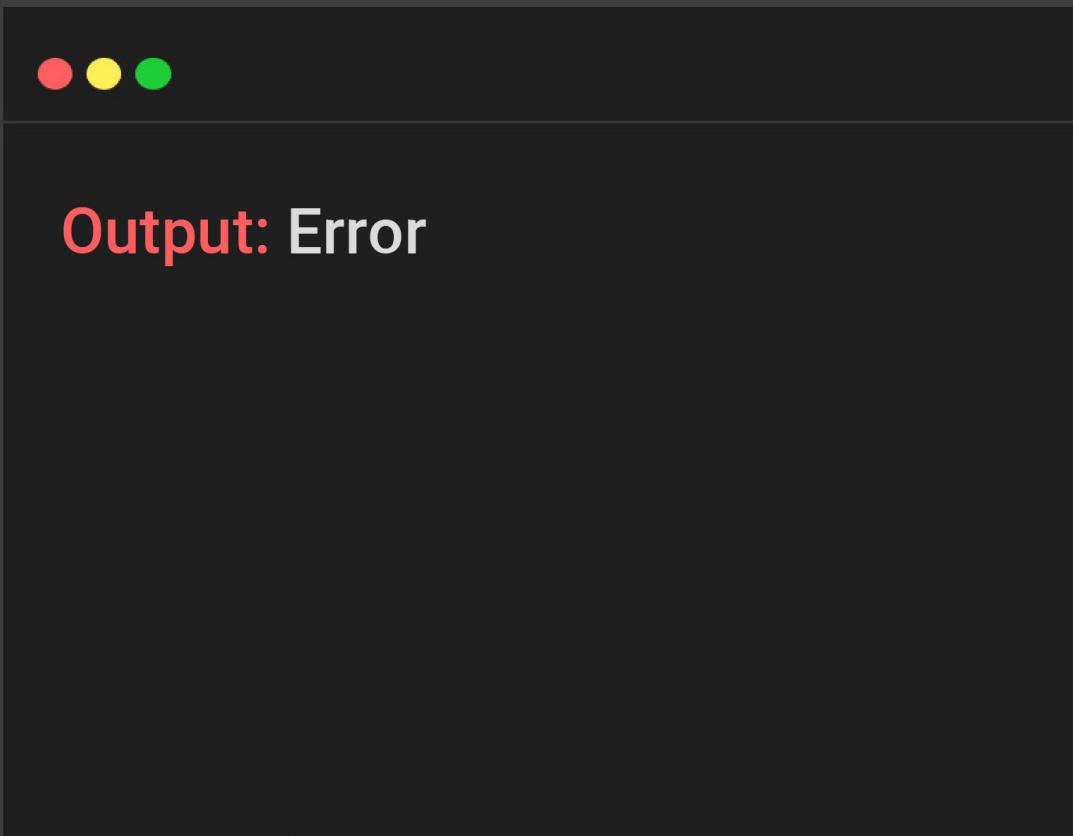
# Guess The Output



Language: Python

```
def a():
    try:
        f(x, 4)
    finally:
        print('after f')
        print('after f?')
a()
```

# Guess The Output



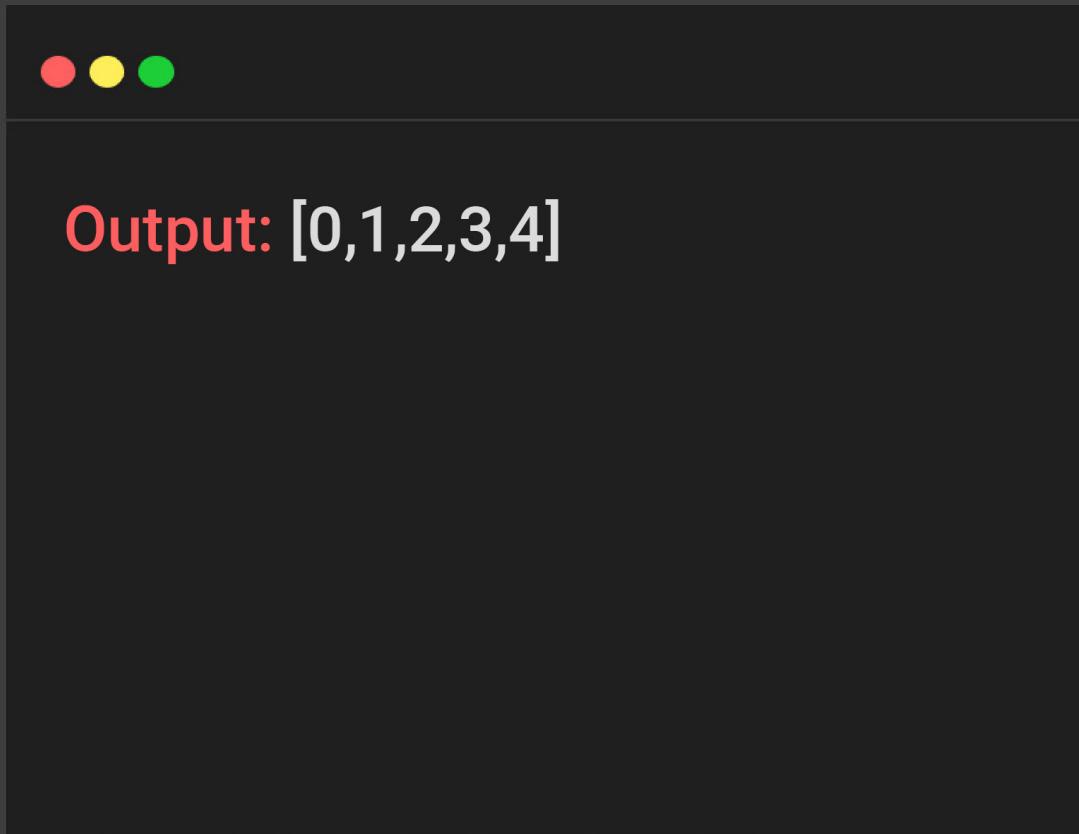
# Guess The Output



Language: Python

```
def f(x):  
    for i in range(5):  
        yield i  
g=f(8)  
print(list(g))
```

# Guess The Output



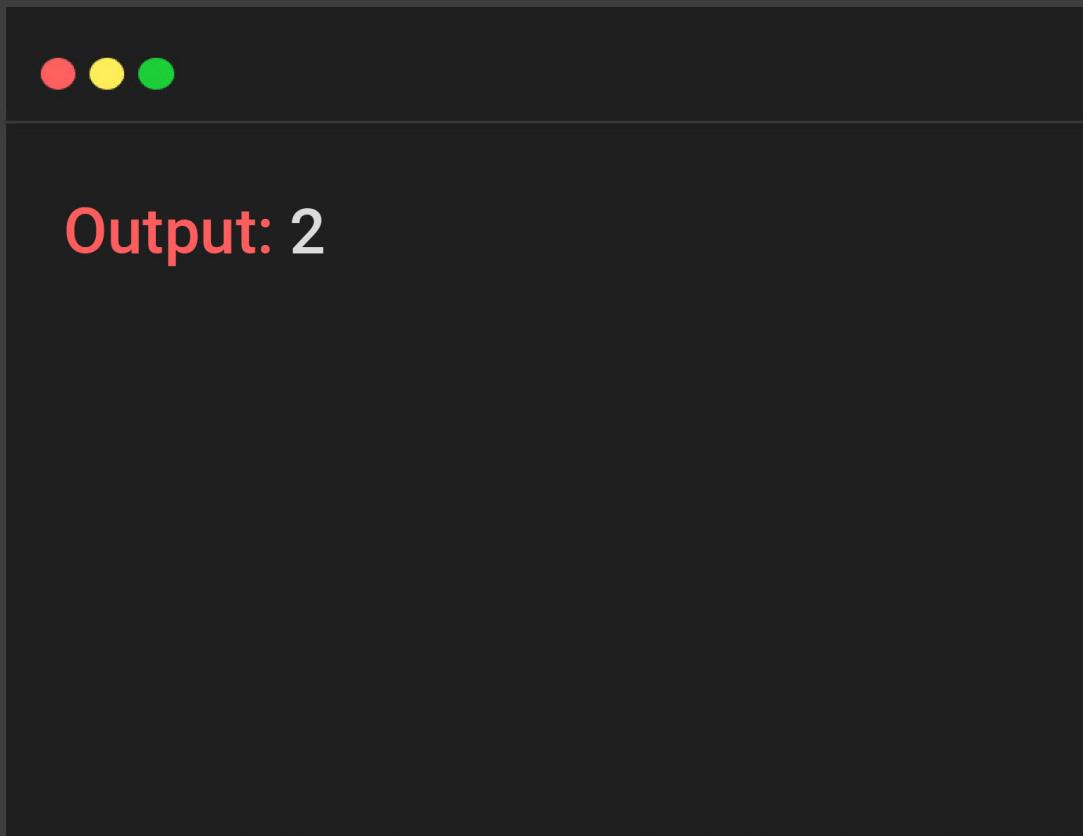
# Guess The Output



Language: Python

```
def foo():
    try:
        return 1
    finally:
        return 2
k = foo()
print(k)
```

# Guess The Output



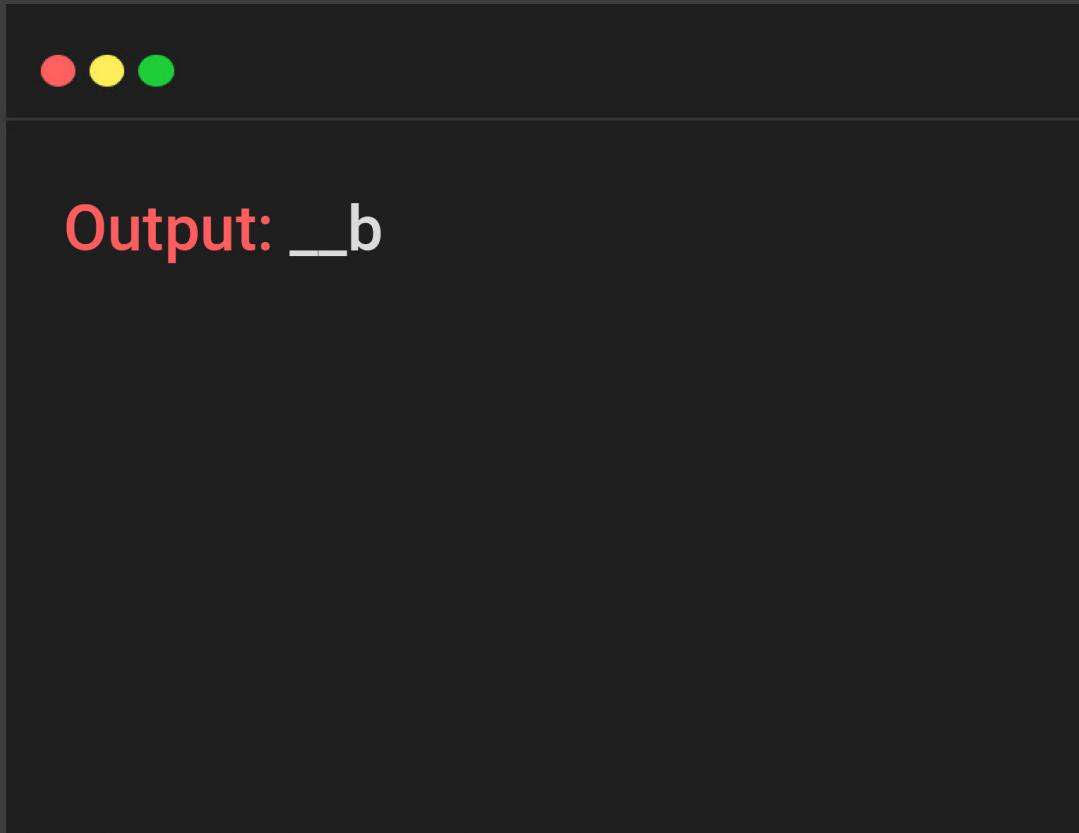
# Guess The Output



Language: Python

```
def Demo:  
def __init__(self):  
    __a = 1  
    self.__b = 1  
    self.__c__ = 1  
    __d__ = 1
```

# Guess The Output



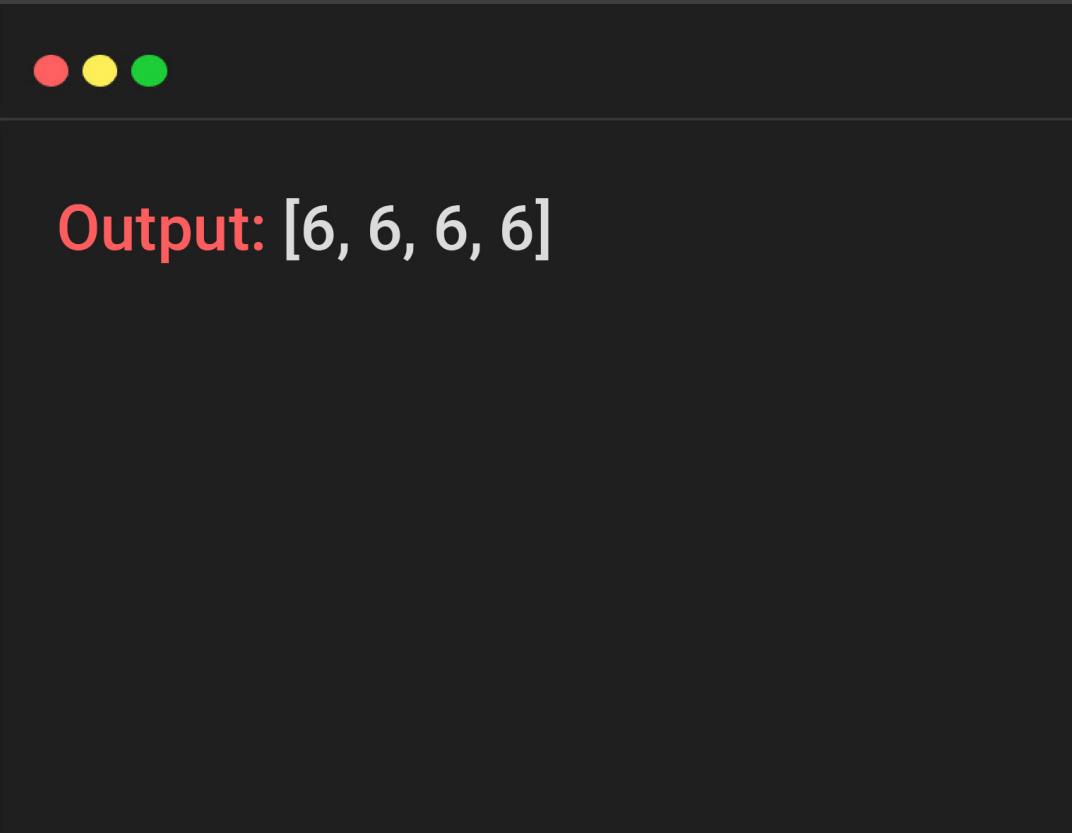
# Guess The Output



Language: Python

```
def multipliers():
    return [lambda x : i * x for i in range(4)]
print [m(2) for m in multipliers()]
```

# Guess The Output



# Guess The Output

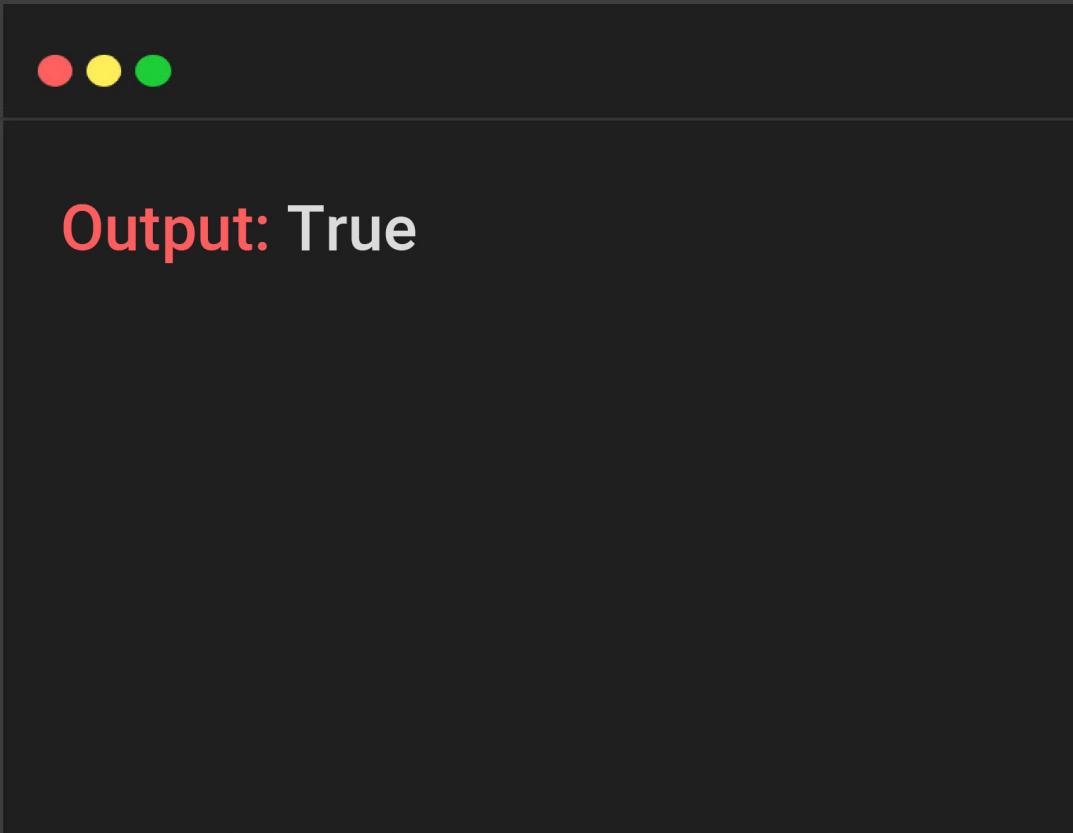


Language: Python

```
def foo(x):  
    x[0] = ['def']  
    x[1] = ['abc']  
    return id(x)  
q = ['abc', 'def']  
print(id(q) == foo(q))
```

**cs** mock

# Guess The Output



# Guess The Output



Language: Python

```
z=set('abc')
z.add('san')
z.update(set(['p', 'q']))
```

# Guess The Output



**Output:** {'a', 'b', 'c', 'p', 'q', 'san'}

**Explanation:**

The code shown first adds the element 'san' to the set z. The set z is then updated and two more elements, namely, 'p' and 'q' are added to it. Hence the output is: {'a', 'b', 'c', 'p', 'q', 'san'}

# Guess The Output



Language: Python

```
print("abc. DEF".capitalize())
```

# Guess The Output



# Guess The Output



Language: Python

```
print('*', "abcde".center(6), '*', sep="")
```



\*abcde \*



\* abcde \*

# Guess The Output



**Output:** \*abcde \*

**Explanation:**

Padding is done towards the right-hand-side first when the final string is of even length.

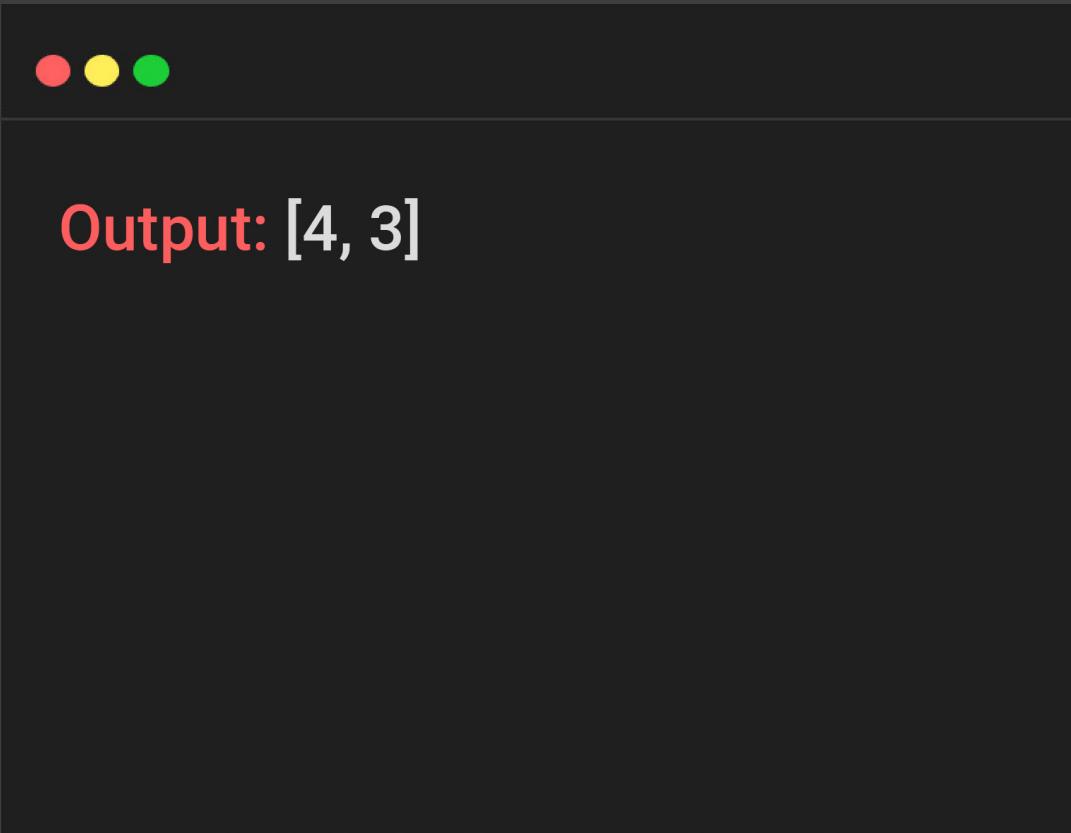
# Guess The Output



Language: Python

```
>>>list1 = [1, 3]
>>>list2 = list1
>>>list1[0] = 4
>>>print(list2)
```

# Guess The Output



# Guess The Output



Language: Python

```
i = 0
while i < 5:
    print(i)
    i += 1
    if i == 3:
        break
    else:
        print(0)
```

# Guess The Output



**Output:** 0 1 2

**Explanation:**

The else part is not executed if control breaks out of the loop.

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in range(len(x)):  
    print(i)
```

# Guess The Output



**Output:** 1 2 3 4

**Explanation:**

i takes values 0, 1, 2 and 3

# Guess The Output



Language: Python

```
def addItem(listParam):  
    listParam += [1]  
mylist = [1, 2, 3, 4]  
addItem(mylist)  
print(len(mylist))
```

# Guess The Output



**Output:** 5

**Explanation:**

+ will append the element in the list

# Guess The Output



Language: Python

```
z=set('abc$de')  
'a' in z
```



True



False

# Guess The Output



**Output:** True

**Explanation:**

The code shown above is used to check whether a particular item is a part of a given set or not. Since 'a' is a part of the set z, the output is true. Note that this code would result in an error in the absence of the quotes.

# Guess The Output

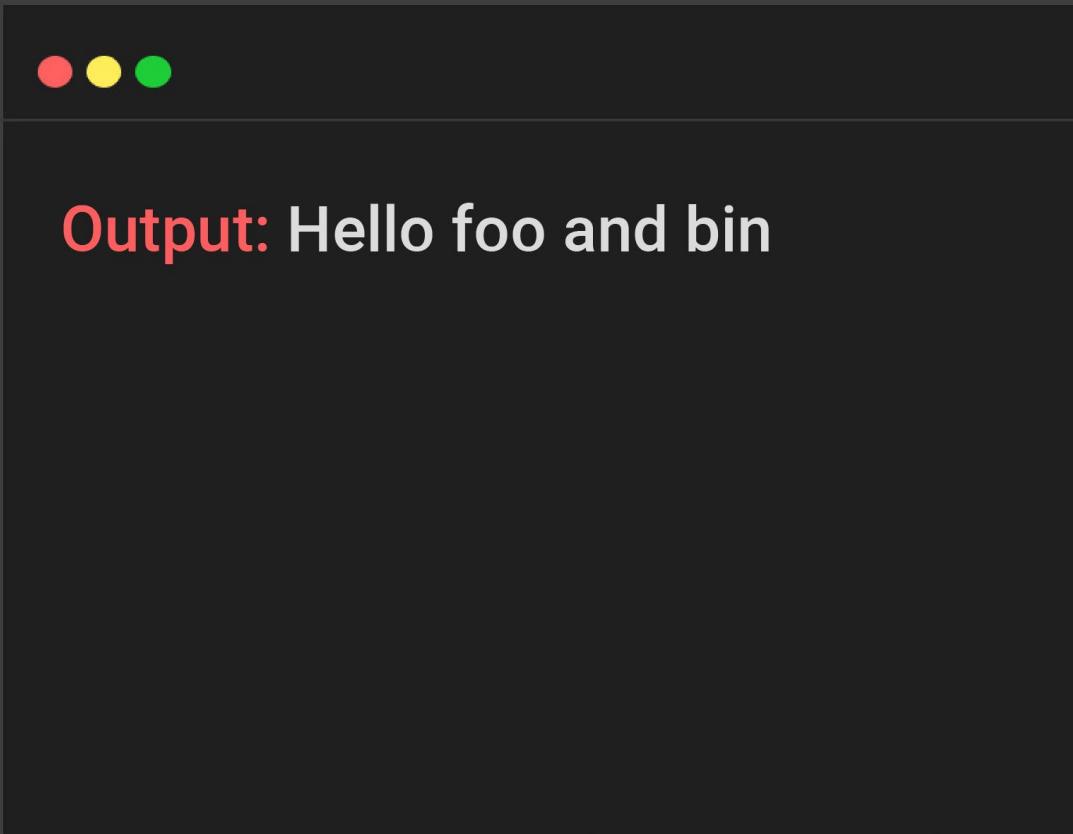


Language: Python

```
print("Hello {0[0]} and {0[1]}".format(('foo', 'bin')))
```

 mock

# Guess The Output



# Guess The Output



Language: Python

```
print("Hello {0[0]} and {0[1]}"  
.format(('foo', 'bin')))x = [[0], [1]]  
  
print(''.join(list(map(str, x))))
```

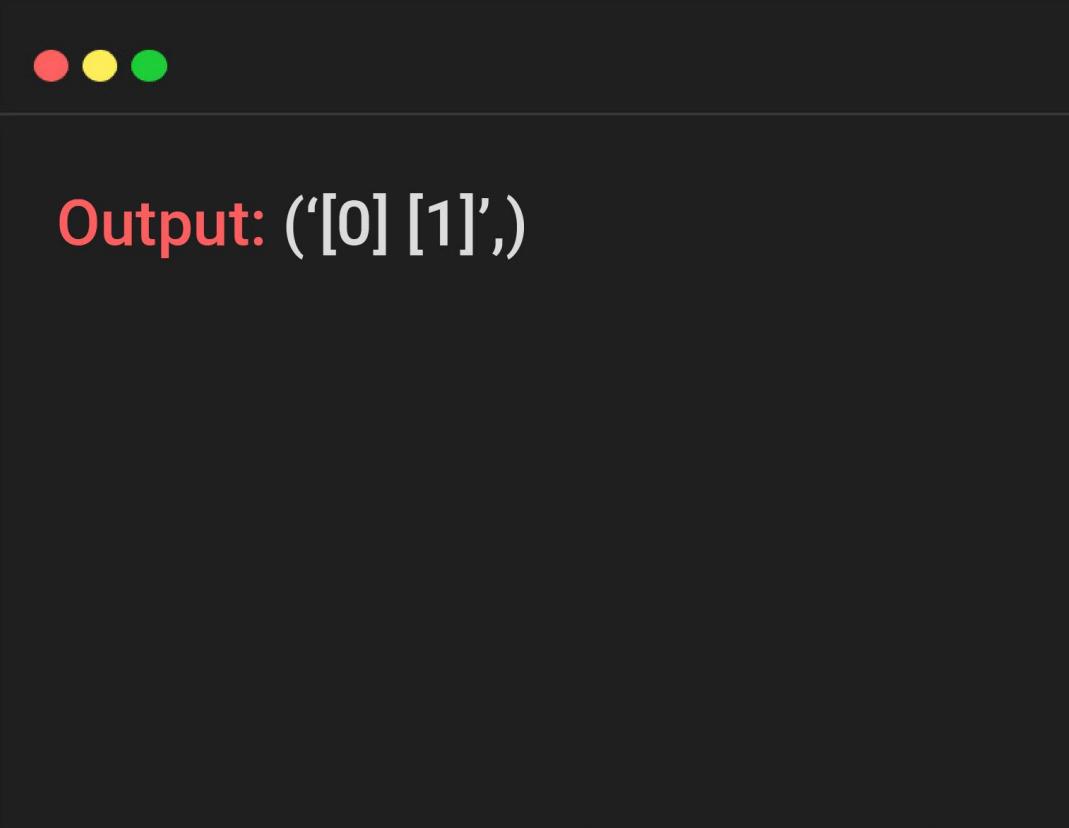


('[0] [1]',)



[0] [1]

# Guess The Output



# Guess The Output



Language: Python

```
def foo():
    try:
        return 1
    finally:
        return 2
k = foo()
print(k)
```

**cs** mock

# Guess The Output



**Output:** 2

**Explanation:**

The finally block is executed even there is a return statement in the try block.

# Guess The Output



Language: Python

```
print('{0:.2}'.format(1/3))
```

# Guess The Output



**Output:** 0.33

**Explanation:**

2 specifies the precision

# Guess The Output



Language: Python

```
print('ab12'.isalnum())
```

# Guess The Output



**Output:** True

**Explanation:**

The string has only letters and digits.

# Guess The Output



Language: Python

```
print('__foo__'.isidentifier())
```

 mock

# Guess The Output



**Output:** True

**Explanation:**

It is a valid identifier.

# Guess The Output



**Output:** foo

**Explanation:**

All leading whitespace is removed.

# Guess The Output



Language: Python

```
x = [i**+1 for i in range(3)]; print(x);
```

# Guess The Output



**Output:** [0, 1, 2]

**Explanation:**

$i^{**+1}$  is evaluated as  $(i)^{**(+1)}$ .

# Guess The Output



Language: Python

```
print([if i%2==0: i; else: i+1;  
      for i in range(4)])
```

# Guess The Output



**Output:** Error

**Explanation:**  
Syntax error.

# Guess The Output



Language: Python

```
l=[1,2,3,4,5]  
[x&1 for x in l]
```

# Guess The Output



**Output:** [1, 0, 1, 0, 1]

**Explanation:**

In the code shown above, each of the numbers of the list, that is, 1, 2, 3, 4 and 5 are AND-ed with 1 and the result is printed in the form of a list. Hence the output is [1, 0, 1, 0, 1].

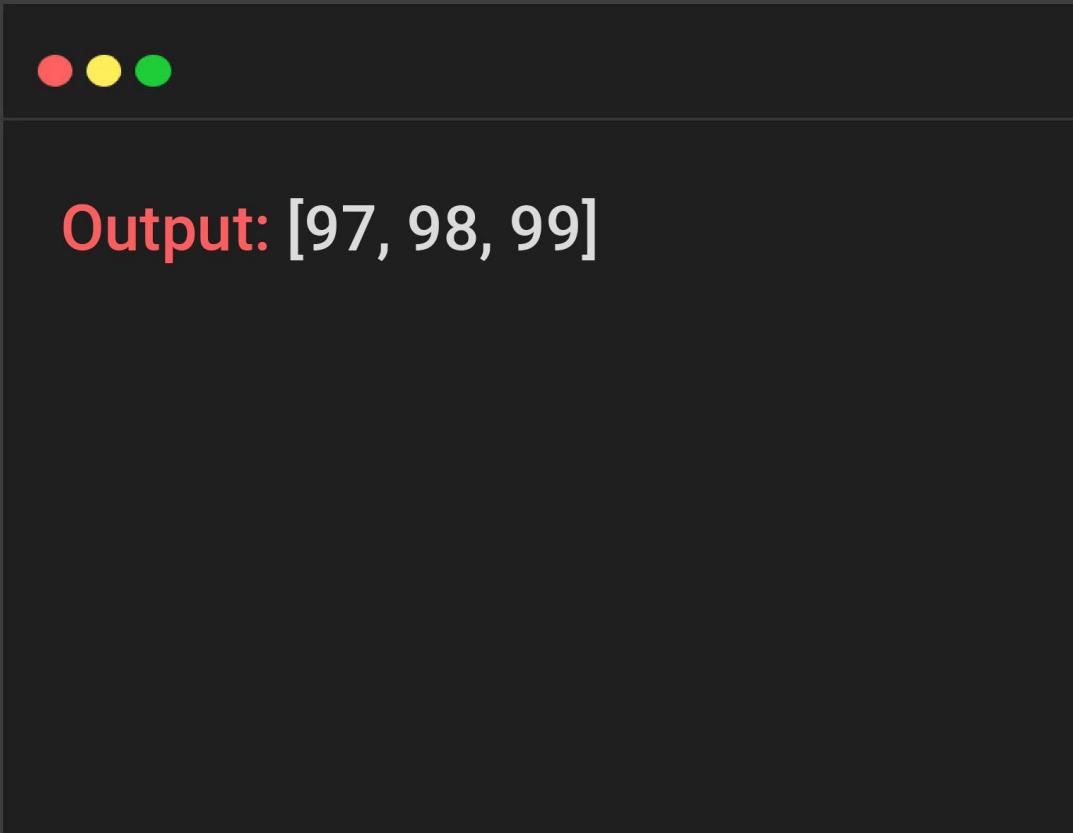
# Guess The Output



Language: Python

`[ord(ch) for ch in 'abc']`

# Guess The Output



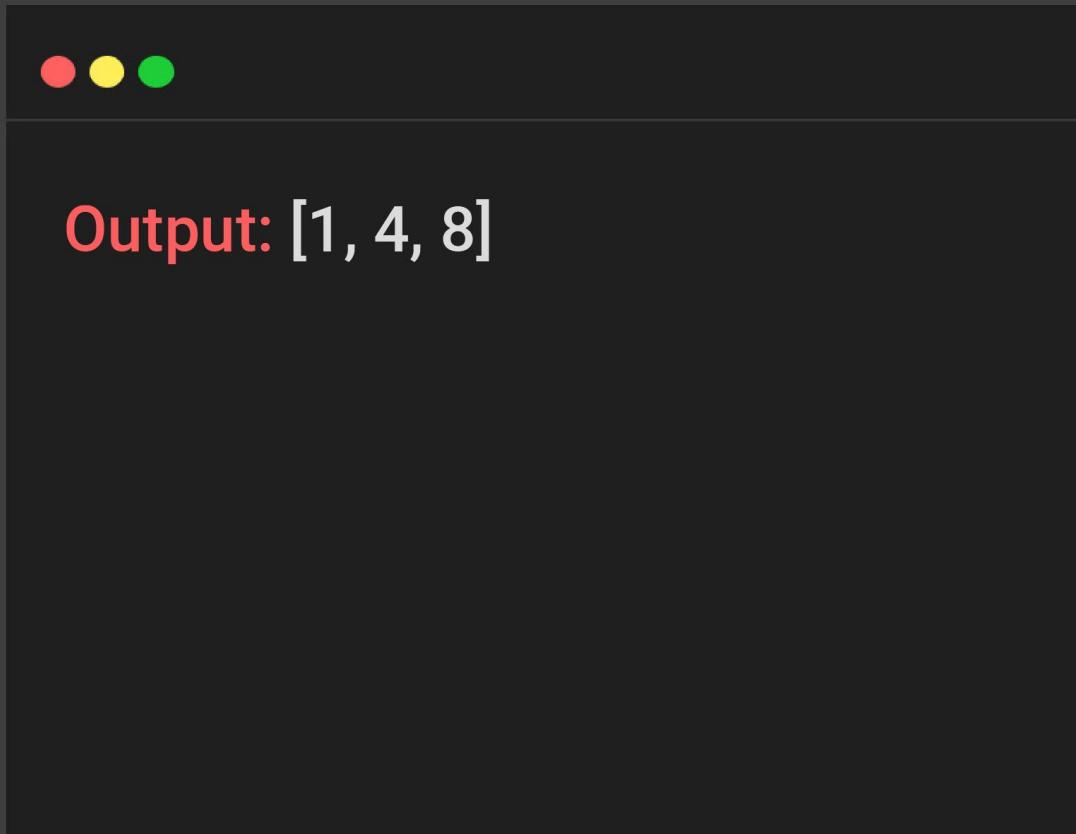
# Guess The Output



Language: Python

```
>>>t = (1, 2, 4, 3, 8, 9)
>>>[t[i] for i in range(0, len(t), 2)]
```

# Guess The Output



# Guess The Output



Language: Python

```
>>>my_tuple = (1, 2, 3, 4)
>>>my_tuple.append( (5, 6, 7) )
>>>print len(my_tuple)
```

# Guess The Output



**Output:** Error

**Explanation:**

Tuples are immutable and don't have an append method. An exception is thrown in this case.

# Guess The Output



Language: Python

```
print('abcd'.translate('a'.
maketrans('abc', 'bcd')))
```

# Guess The Output



**Output:** bcdd

**Explanation:**

The output is bcdd since no translation is provided for d.

# Guess The Output



Language: Python

```
print('+99'.zfill(5))
```

 mock

# Guess The Output



**Output: +0099**

**Explanation:**

zeros are filled in between the first sign and the rest of the string.

# Guess The Output



Language: Python

```
print("xyyzxyzxzxyy"  
.count('yy', 1))
```

# Guess The Output



**Output:** 2

**Explanation:**

Counts the number of times the substring 'yy' is present in the given string, starting from position 1.

# Guess The Output



Language: Python

```
print("xyyzxyzxzxxy"  
.count('xyy', 0, 100))
```

# Guess The Output



**Output:** 2

**Explanation:**

An error will not occur if the end value is greater than the length of the string itself.

# Guess The Output



Language: Python

```
print('abc'.encode())
```

# Guess The Output



**Output:** b'abc'

**Explanation:**

A bytes object is returned by encode.

# Guess The Output



Language: Python

```
class change:  
    def __init__(self, x, y, z):  
        self.a = x + y + z  
    x = change(1,2,3)  
    y = getattr(x, 'a')  
    setattr(x, 'a', y+1)  
    print(x.a)
```

# Guess The Output



**Output:** 7

**Explanation:**

First,  $a=1+2+3=6$ . Then, after `setattr()` is invoked,  
 $x.a=6+1=7$ .

# Guess The Output



Language: Python

```
>>> class A:  
def __init__(self,b):  
    self.b=b  
def display(self):  
    print(self.b)  
>>> obj=A("Hello")  
>>> del obj
```

# Guess The Output



**Output:** True

**Explanation:**

It is possible to delete an object of the class. On further typing obj in the python shell, it throws an error because the defined object has now been deleted.

# Guess The Output



Language: Python

```
class test:  
    def __init__(self):  
        self.variable = 'Old'  
        self.Change(self.variable)  
    def Change(self, var):  
        var = 'New'  
obj=test()  
print(obj.variable)
```



Old



New

# Guess The Output



**Output:** Old

**Explanation:**

This is because strings are immutable. Hence any change made isn't reflected in the original string.

# Guess The Output



Language: Python

```
class fruits:  
    def __init__(self, price):  
        self.price = price  
    obj=fruits(50)  
    obj.quantity=10  
    obj.bags=2  
    print(obj.quantity+len(obj.__dict__))
```

# Guess The Output



**Output:** 13

**Explanation:**

In the above code, `obj.quantity` has been initialised to 10. There are a total of three items in the dictionary, `price`, `quantity` and `bags`. Hence, `len(obj.__dict__)` is 3.

# Guess The Output



Language: Python

```
class Demo:  
    def __init__(self):  
        pass  
    def test(self):  
        print(__name__)  
obj = Demo()  
obj.test()
```

# Guess The Output



**Output: `__main__`**

**Explanation:**

Since the above code is being run not as a result of an import from another module, the variable will have value “`__main__`”.

# Guess The Output



Language: Python

```
if (9 < 0) and (0 < -9):  
    print("hello")  
elif (9 > 0) or False:  
    print("good")  
else:  
    print("bad")
```

# Guess The Output



**Output:** good

**Explanation:**

The code shown above prints the appropriate option depending on the conditions given. The condition which matches is  $(9>0)$ , and hence the output is: good.

# Guess The Output



Language: Python

`not(10<20) and not(10>30)`

# Guess The Output



**Output:** False

**Explanation:**

The expression `not(10<20)` returns false. The expression `not(10>30)` returns true. The `and` operation between false and true returns false. Hence the output is false.

# Guess The Output



Language: Python

```
print("%06d"%X)
```

**cs** mock

# Guess The Output



**Output:** 000345

**Explanation:**

The above expression returns the output 000345. It adds the required number of zeroes before the given number in order to make the number of digits 6.

# Guess The Output



Language: Python

```
'%(qty)d more %(food)s' %{'qty':1,  
'food': 'spam'}
```

**cs** mock

# Guess The Output



**Output:** '1 more spam'

**Explanation:**

In the code shown above, (qty) and (food) in the format string on the left refers to keys in the dictionary literal on the right and fetch their assorted values. Hence the output of the code shown above is: 1 more spam.

# Guess The Output



Language: Python

```
t = '%(a)s, %(b)s, %(c)s'  
t % dict(a='hello', b='world', c='universe')
```

# Guess The Output



**Output:** 'hello, world, universe'

**Explanation:**

Within the subject string, curly braces represent substitution targets and arguments to be inserted.

Hence the output of the code shown above:  
'hello, world, universe'.

# Guess The Output



Language: Python

```
'{a}, {0}, {abc}'.format(10, a=2.5, abc=[1, 2])
```

# Guess The Output



**Output:** '2.5, 10, [1, 2]'

## Explanation:

Since we have specified that the order of the output be: {a}, {0}, {abc}, hence the value of associated with {a} is printed first followed by that of {0} and {abc}. Hence the output of the code shown above is: '2.5, 10, [1, 2]'.

# Guess The Output



Language: Python

```
l=list('HELLO')
'first={0[0]}, third={0[2]}'.format(l)
```

**cs** mock

# Guess The Output



**Output:** 'first=H, third=L'

**Explanation:**

In the code shown above, the value for first is substituted by `I[0]`, that is H and the value for third is substituted by `I[2]`, that is L. Hence the output of the code shown above is: 'first=H, third=L'. The list `I= ['H', 'E', 'L', 'L', 'O']`.

# Guess The Output



Language: Python

```
l=list('HELLO')
p=l[0], l[-1], l[1:3]
'a={0}, b={1}, c={2}'.format(*p)
```

# Guess The Output



**Output:** “a=H, b=O, c=['E', 'L']”

## Explanation:

In the code shown above, the value for a is substituted by l[0], that is 'H', the value of b is substituted by l[-1], that is 'O' and the value for c is substituted by l[1:3]. Here the use of \*p is to unpack a tuple items into individual function arguments.

# Guess The Output



Language: Python

```
'{}'.format(bin((2**16)-1))
```

cs mock

# Guess The Output



**Output:** True

**Explanation:**

The output of the codes shown above is '0b1111111111111111'. Hence the statement is true.

# Guess The Output



**Output:** CSMock

**Explanation:**

The code shown above prints the values substituted for p and q in the same order. Note that there is no blank space between p and q.

# Guess The Output



Language: Python

```
'The {} side {} {}'.format('bright', 'of', 'life')
```

# Guess The Output



**Output:** Error

**Explanation:**

The code shown above results in an error. This is because we have switched from automatic field numbering to manual field numbering, that is, from {} to {1}. Hence this code results in an error.

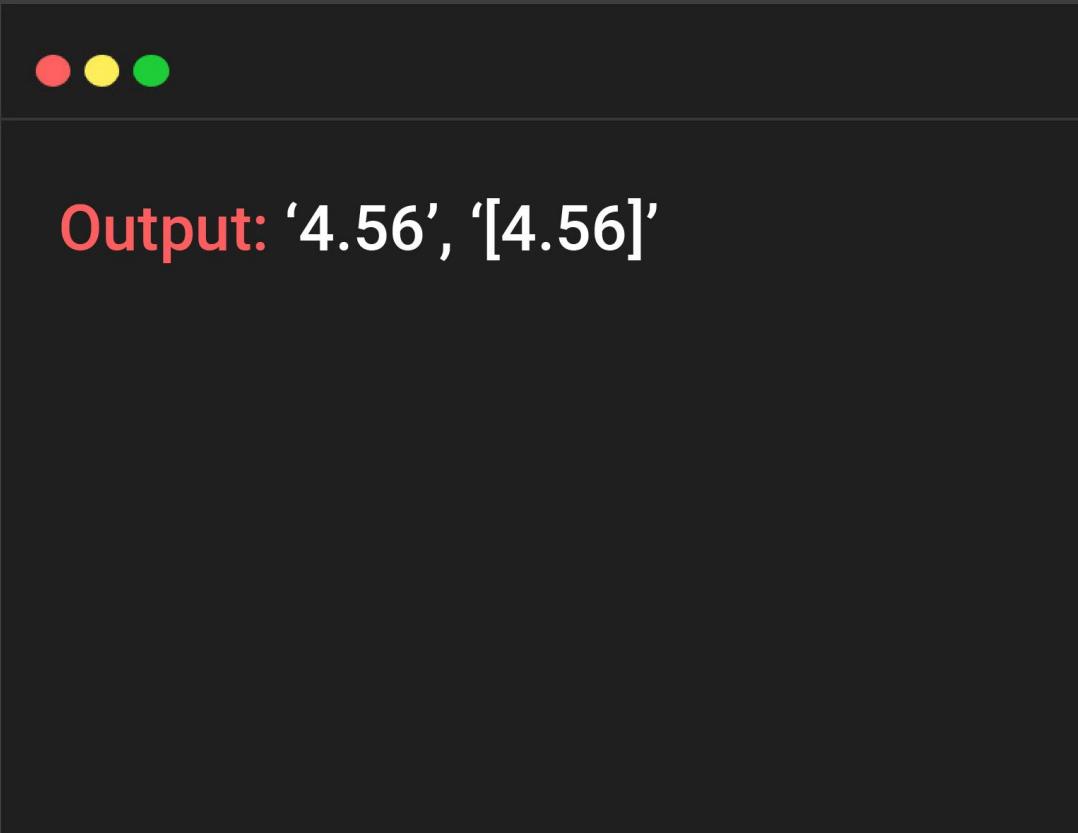
# Guess The Output



Language: Python

`'{0}'.format([4.56,])`

# Guess The Output



# Guess The Output



Language: Python

```
def mk(x):
    def mk1():
        print("Decorated")
        x()
    return mk1
def mk2():
    print("Ordinary")
p = mk(mk2)
p()
```

# Guess The Output



**Output:** `mk()`

**Explanation:**

In the code shown above, the function `mk()` is the decorator. The function which is getting decorated is `mk2()`. The return function is given the name `p()`.

# Guess The Output



Language: Python

```
def f1():
    print("Hello")
```

```
def f1():
    print("Hello")
f1 = f(f1)
```

# Guess The Output



**Output:** True

**Explanation:**

The @ symbol can be used as an alternate way to specify a function that needs to be decorated. The output of the codes shown above is the same. Hence they are equivalent. Therefore this statement is true.

# Guess The Output



Language: Python

```
def f(x):
    def f1(a, b):
        print("hello")
        if b==0:
            print("NO")
            return
        return f(a, b)
    return f1
@f
def f(a, b):
    return a%b
f(4,0)
```

cs mock

# Guess The Output



**Output:** hello No

**Explanation:**

In the code shown above, we have used a decorator in order to avoid the Zero Division Error.

# Guess The Output



Language: Python

```
def sf():
    pass
sf = mk(sf)
@f
def sf():
    return
```

**cs** mock

# Guess The Output



**Output:** mk

**Explanation:**

In the code shown above, @sf is not a decorator but only a decorator line. The '@' symbol represents the application of a decorator. The decorator here is the function mk.

# Guess The Output



Language: Python

```
class A:  
    @staticmethod  
    def a(x):  
        print(x)  
A.a(100)
```



100



Error

# Guess The Output



**Output:** 100

**Explanation:**

The code shown above demonstrates rebinding using a static method. This can be done with or without a decorator. The output of this code will be 100.

# Guess The Output



Language: Python

```
def d(f):  
    def n(*args):  
        return '$' + str(f(*args))  
    return n  
@d  
def p(a, t):  
    return a + a*t  
print(p(100,0))
```

cs mock

# Guess The Output



**Output:** \$100

**Explanation:**

In the code shown above, the decorator helps us to prefix the dollar sign along with the value. Since the second argument is zero, the output of the code is: \$100.

# Guess The Output



Language: Python

```
x = ['ab', 'cd']
for i in x:
    i.upper()
print(x)
```

# Guess The Output



**Output:** ['ab', 'cd']

**Explanation:**

The function `upper()` does not modify a string in place, it returns a new string which isn't being stored anywhere.

# Guess The Output



Language: Python

```
i = 1
while True:
    if i%3 == 0:
        break
    print(i)
    i += 1
```

# Guess The Output



**Output:** Error

**Explanation:**

SyntaxError, there shouldn't be a space  
between + and = in +=.

# Guess The Output



Language: Python

```
i = 1
while True:
    if i%007 == 0:
        break
    print(i)
    i+=1
```

# Guess The Output



**Output:** 1 2 3 4 5 6

**Explanation:**

Control exits the loop when i becomes 7.

# Guess The Output



Language: Python

```
i = 5
while True:
    if i%0011 == 0:
        break
    print(i)
    i += 1
```

# Guess The Output



**Output:** 5 6 7 8

**Explanation:**

0011 is an octal number.

# Guess The Output



Language: Python

```
i = 5
while True:
    if i%009 == 0:
        break
    print(i)
    i += 1
```

# Guess The Output



**Output:** Error

**Explanation:**

9 isn't allowed in an octal number.

# Guess The Output



Language: Python

```
i = 1
while True:
    if i%2 == 0:
        break
    print(i)
    i += 2
```

# Guess The Output



**Output:** 1 3 5...

**Explanation:**

The loop does not terminate since i is never an even number.

# Guess The Output



Language: Python

```
True = False
while True:
    print(True)
    break
```

# Guess The Output



**Output:** Error

**Explanation:**

SyntaxError, True is a keyword and it's value cannot be changed.

# Guess The Output



Language: Python

```
i = 0
while i < 5:
    print(i)
    i += 1
    if i == 3:
        break
else:
    print(0)
```

# Guess The Output



**Output:** 0 1 2

**Explanation:**

The else part is not executed if control breaks out of the loop.

# Guess The Output



Language: Python

```
i = 0
while i < 3:
    print(i)
    i += 1
else:
    print(0)
```

# Guess The Output



**Output:** 0 1 2 0

**Explanation:**

The else part is executed when the condition in the while statement is false.

# Guess The Output



Language: Python

```
x = "abcdef"  
while i in x:  
    print(i, end=" ")
```

# Guess The Output



**Output:** abcdef

**Explanation:**

NameError, i is not defined

# Guess The Output



Language: Python

```
x = "abcdef"  
i = "a"  
while i in x:  
    x = x[:-1]  
    print(i, end = " ")
```

# Guess The Output



**Output:** a

**Explanation:**

The string x is being shortened by one character in each iteration.

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in x:  
    print(i)  
    x.upper()
```

# Guess The Output



**Output:** a b c d

**Explanation:**

Changes do not happen in-place, rather a new instance of the string is returned.

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in x:  
    print(i.upper())
```

**cs** mock

# Guess The Output



**Output:** A B C D

**Explanation:**

The instance of the string returned by  
upper() is being printed.

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in range(x):  
    print(i)
```

# Guess The Output



**Output:** error

**Explanation:**

range(str) is not allowed.

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in range(len(x)):  
    print(i)
```

# Guess The Output



**Output:** 0 1 2 3

**Explanation:**

i takes values 0, 1, 2 and 3

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in range(len(x)):  
    print(i.upper())
```

# Guess The Output



**Output:** error

**Explanation:**

Objects of type int have no attribute  
upper().

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in range(len(x)):  
    i.upper()  
print (x)
```

# Guess The Output



**Output:** abcd

**Explanation:**

Changes do not happen in-place, rather a new instance of the string is returned.

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in range(len(x)):  
    i[x].upper()  
print (x)
```

# Guess The Output



**Output:** error

**Explanation:**

Objects of type int aren't subscriptable.  
However, if the statement was `x[i]`, an  
error would not have been thrown.

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in range(len(x)):  
    x = 'a'  
    print(x)
```

# Guess The Output



**Output:** a a a a

**Explanation:**

range() is computed only at the time of  
entering the loop.

# Guess The Output



Language: Python

```
x = 'abcd'  
for i in range(len(x)):  
    print(x)  
    x = 'a'
```

# Guess The Output



**Output:** abcd a a a

**Explanation:**

abcd a a a is the output as x is modified only after 'abcd' has been printed once.

# Guess The Output



Language: Python

```
d = {0: 'a', 1: 'b', 2: 'c'}  
for i in d:  
    print(i)
```

# Guess The Output



**Output:** 0 1 2

**Explanation:**

Loops over the keys of the dictionary.

# Guess The Output



Language: Python

```
d = {0: 'a', 1: 'b', 2: 'c'}  
for x, y in d.items():  
    print(x, y)
```

# Guess The Output



# Guess The Output



Language: Python

```
d = {0: 'a', 1: 'b', 2: 'c'}  
for x in d.keys():  
    print(d[x])
```

# Guess The Output



**Output:** abc

**Explanation:**

Loops over the keys and prints the values.

# Guess The Output



Language: Python

```
d = {0: 'a', 1: 'b', 2: 'c'}  
for x in d.values():  
    print(x)
```

# Guess The Output



**Output:** abc

**Explanation:**

Loops over the values..

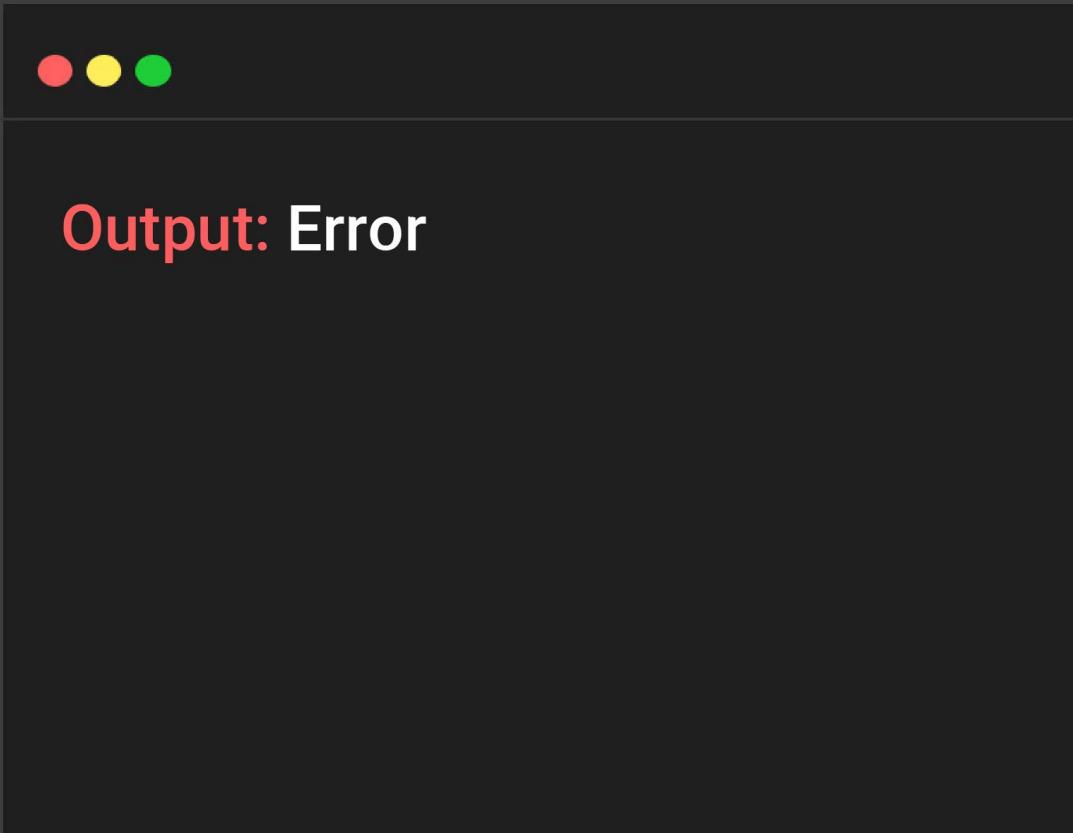
# Guess The Output



Language: Python

```
d = {0: 'a', 1: 'b', 2: 'c'}  
for x in d.values():  
    print(d[x])
```

# Guess The Output



# Guess The Output



Language: Python

```
d = {0, 1, 2}  
for x in d.values():  
    print(x)
```

# Guess The Output



**Output:** Error

**Explanation:**

Objects of type set have no attribute values.

# Guess The Output



Language: Python

```
d = {0, 1, 2}  
for x in d.:  
    print(x)
```

# Guess The Output



**Output:** 0 1 2

**Explanation:**

Loops over the elements of the set and prints them.

# Guess The Output



Language: Python

```
d = {0, 1, 2}  
for x in d:  
    print(d.add(x))
```

**cs** mock

# Guess The Output



**Output:** None None None

**Explanation:**

Variable x takes the values 0, 1 and 2.  
set.add() returns None which is printed.

# Guess The Output



Language: Python

```
for i in range(0):  
    print(i)
```

# Guess The Output



**Output:** no output

**Explanation:**  
range(0) is empty.

# Guess The Output



Language: Python

```
for i in range(int(2.0)):  
    print(i)
```

**cs** mock

# Guess The Output



**Output:** 0 1

**Explanation:**

`range(int(2.0))` is the same as `range(2)`

# Guess The Output



Language: Python

```
for i in "":  
    print (i)
```

**cs** mock

# Guess The Output



**Output:** no output

**Explanation:**

The string does not have any character to loop over.

# Guess The Output



Language: Python

```
x = 2
for i in range(x):
    x += 1
    print (x)
```

# Guess The Output



**Output:** 3 4

**Explanation:**

Variable x is incremented and printed twice.

# Guess The Output



Language: Python

```
for i in range(10):
    if i == 5:
        break
    else:
        print(i)
else:
    print("Here")
```

# Guess The Output



**Output:** 0 1 2 3 4

**Explanation:**

The else part is executed if control  
doesn't break out of the loop.

# Guess The Output



Language: Python

```
x = (i for i in range(3))
for i in x:
    print(i)
for i in x:
    print(i)
```

# Guess The Output



**Output:** 0 1 2

**Explanation:**

We can loop over a generator object  
only once.

# Guess The Output



Language: Python

```
a = [0, 1, 2, 3]
for a[0] in a:
    print(a[0])
```

# Guess The Output



**Output:** 0 1 2 3

**Explanation:**

The value of `a[0]` changes in each iteration. Since the first value that it takes is itself, there is no visible error in the current example.

# Guess The Output



Language: Python

```
a = [0, 1, 2, 3]
i = -2
for i not in a:
    print(i)
    i += 1
```

# Guess The Output



**Output:** error

**Explanation:**

SyntaxError, not in isn't allowed in for loops.

# Guess The Output



**Output:** ('abef', "", "")

**Explanation:**

The separator is not present in the string hence the second and the third elements of the tuple are null strings.

# Guess The Output



Language: Python

```
print('Ab!2'.swapcase())
```

# Guess The Output



**Output:** aB!2

**Explanation:**

Lowercase letters are converted to uppercase and vice-versa.

# Guess The Output



Language: Python

```
print('abcd'.translate('a'  
.maketrans('abc', 'bcd')))
```

# Guess The Output



**Output:** bcdd

**Explanation:**

The output is bcdd since no translation is provided for d.

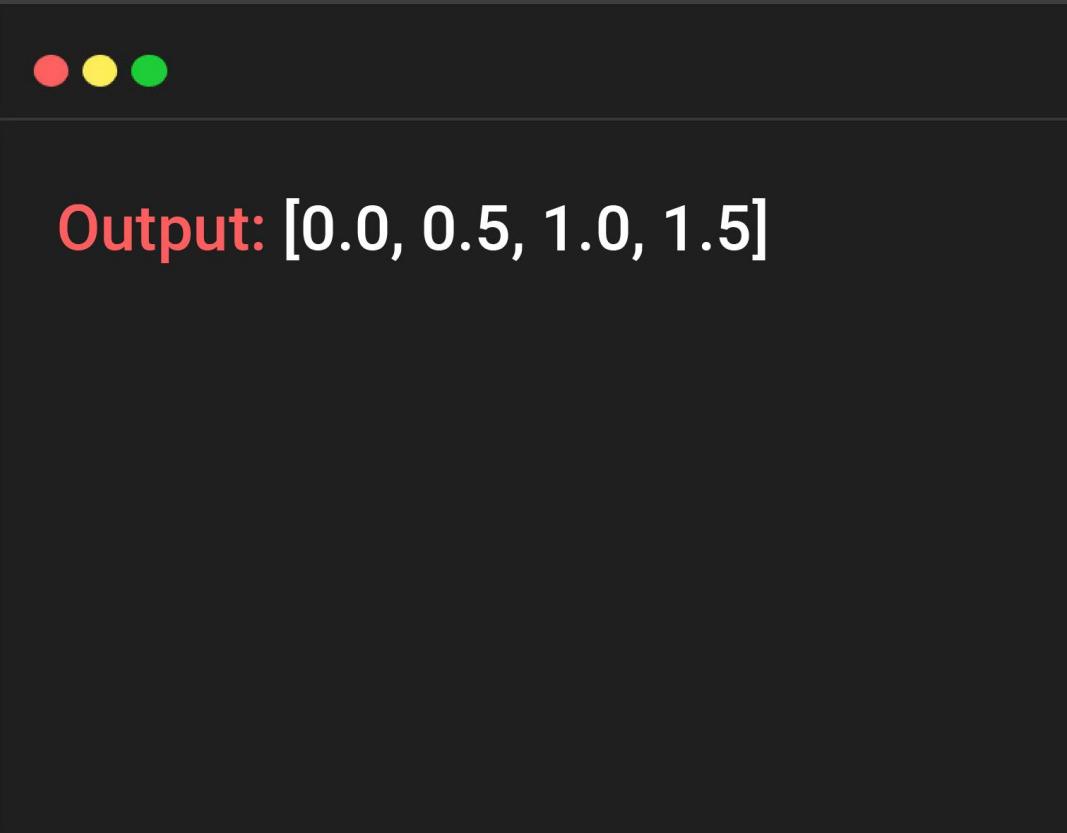
# Guess The Output



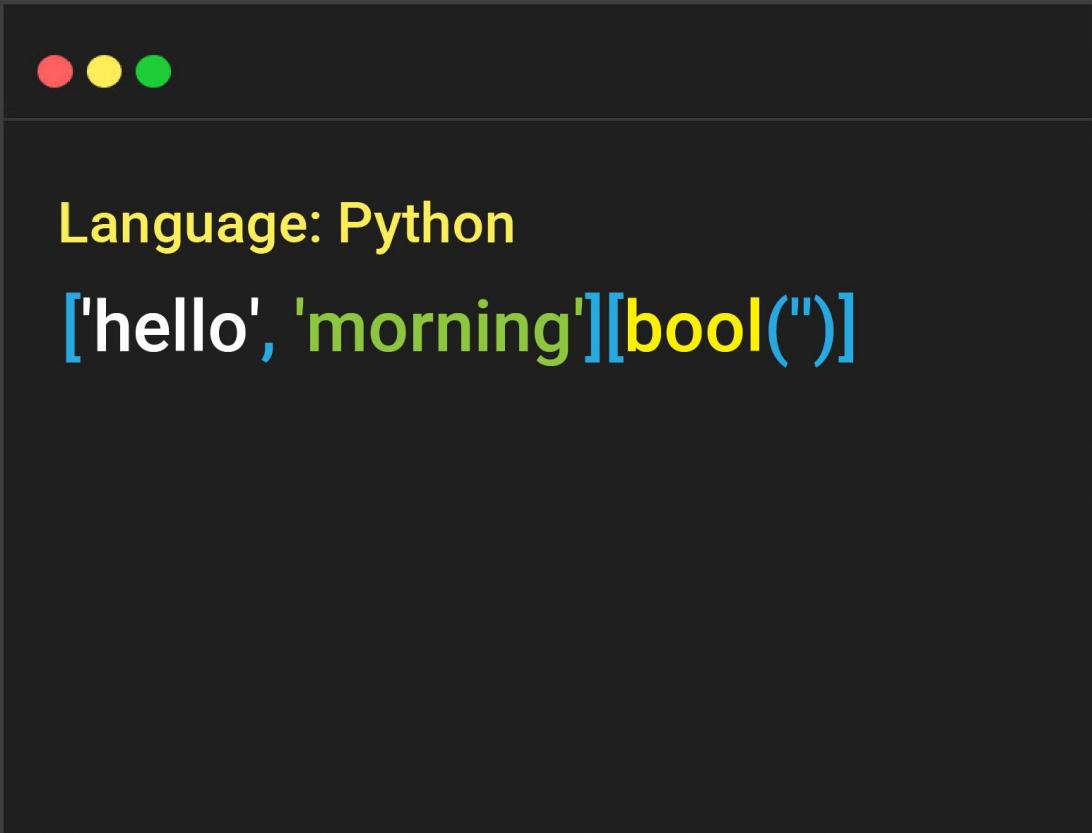
Language: Python

Suppose `list1 = [0.5 * x for x in range(0, 4)]`, `list1` is:

# Guess The Output



# Guess The Output



# Guess The Output

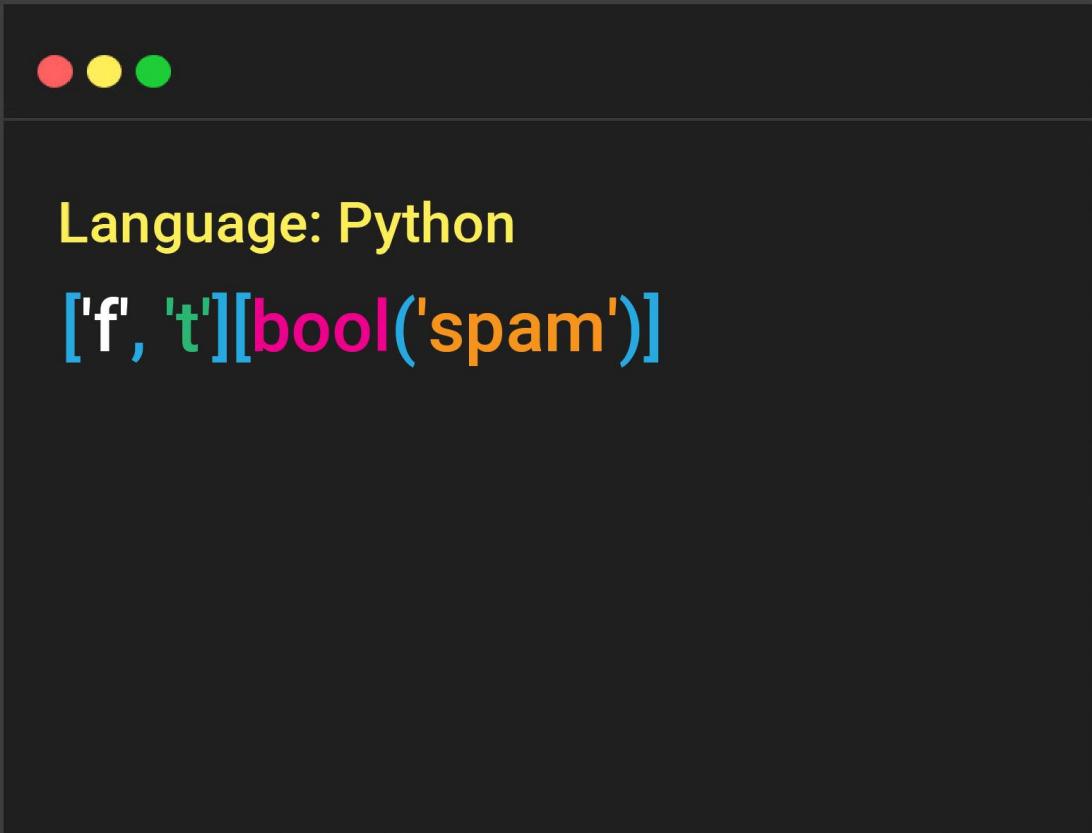


**Output:** hello

**Explanation:**

The line of code shown above can be simplified to state that 'hello' should be printed if the argument passed to the Boolean function amounts to zero, else 'morning' will be printed.

# Guess The Output



Language: Python

```
['f', 't'][bool('spam')]
```

# Guess The Output



**Output:** t

**Explanation:**

The line of code can be translated to state that 'f' is printed if the argument passed to the Boolean function amount to zero. Else 't' is printed. The argument given to the Boolean function in the above case is 'spam', which does not amount to zero. Hence the output is t.

# Guess The Output



Language: Python

```
l=[1, 0, 2, 0, 'hello', "", []]  
list(filter(bool, l))
```

# Guess The Output



**Output:** [1, 2, 'hello']

**Explanation:**

The code shown above returns a new list containing only those elements of the list `l` which do not amount to zero. Hence the output is: [1, 2, 'hello'].

# Guess The Output



Language: Python

`bin(10-2)+bin(12^4)`

# Guess The Output



**Output:** 0b10000b1000

**Explanation:**

The output of  $\text{bin}(10-2) = 0b1000$  and that of  $\text{bin}(12^4)$  is  $0b1000$ . Hence the output of the above expression is: 0b10000b1000.

# Guess The Output



Language: Python

```
>>>t = (1, 2, 4, 3)
>>>t2 = (1, 2, 3, 4)
>>>t1 < t2
```

# Guess The Output



**Output:** false

**Explanation:**

Elements are compared one by one in this case.

# Guess The Output



Language: Python

```
>>>my_tuple = (1, 2, 3, 4)
>>>my_tuple.append( (5, 6, 7) )
>>>print len(my_tuple)
```

# Guess The Output



**Output:** error

**Explanation:**

Tuples are immutable and don't have an append method. An exception is thrown in this case.

# Guess The Output



Language: Python

```
numberGames = {}  
numberGames[(1,2,4)] = 8  
numberGames[(4,2,1)] = 10  
numberGames[(1,2)] = 12  
sum = 0  
for k in numberGames:  
    sum += numberGames[k]  
print len(numberGames) + sum
```

# Guess The Output



**Output:** 33

**Explanation:**

Tuples can be used for keys into dictionary.  
The tuples can have mixed length and the  
order of the items in the tuple is considered  
when comparing the equality of the keys.

# Guess The Output



Language: Python

```
>>> a=(1,2,3)
>>> b=('A','B','C')
>>> c=tuple(zip(a,b))
```

# Guess The Output



**Output:** `((1, 'A'), (2, 'B'), (3, 'C'))`

# Guess The Output



Language: Python

```
import collections
a=collections.namedtuple('a',[i,j])
obj=a(i=4,j=7)
obj
```

# Guess The Output



**Output:** a(i=4, j=7)

**Explanation:**

The above piece of code illustrates the concept of named tuples.

# Guess The Output



Language: Python

```
a = [5,5,6,7,7,7]
b = set(a)
def test(lst):
    if lst in b:
        return 1
    else:
        return 0
for i in filter(test, a):
    print(i, end=" ")
```

# Guess The Output



**Output:** 5 5 6 7 7 7

## Explanation:

The filter function will return all the values from list a which are true when passed to function test. Since all the members of the set are non-duplicate members of the list, all of the values will return true. Hence all the values in the list are printed.

# Guess The Output



Language: Python

```
>>> a={3,4,5}  
>>> b={5,6,7}  
>>> a|b
```

# Guess The Output



**Output:** {3, 4, 5, 6, 7}

**Explanation:**

The operation in the above piece of code is union operation. This operation produces a set of elements in both set a and set b.

# Guess The Output



Language: Python

```
a={3,4,{7,5}}
print(a[2][0])
```

# Guess The Output



**Output:** error

**Explanation:**

In python, elements of a set must not be mutable and sets are mutable. Thus, subsets can't exist.

# Guess The Output



Language: Python

```
>>> a=frozenset(set([5,6,7]))  
>>> a
```

# Guess The Output



**Output:** frozenset({5,6,7})

**Explanation:**

The above piece of code is the correct syntax for creating a frozenset.

# Guess The Output



Language: Python

```
>>> a=frozenset([5,6,7])
>>> a
>>> a.add(5)
```

# Guess The Output



**Output:** immutable

**Explanation:**

Since a frozen set is immutable, add method doesn't exist for frozen method.

# Guess The Output



Language: Python

`a={1,2,3}`

`a.intersection_update({2,3,4,5})`

`a`

# Guess The Output



**Output:** {2,3}

**Explanation:**

The method `intersection_update` returns a set which is an intersection of both the sets.

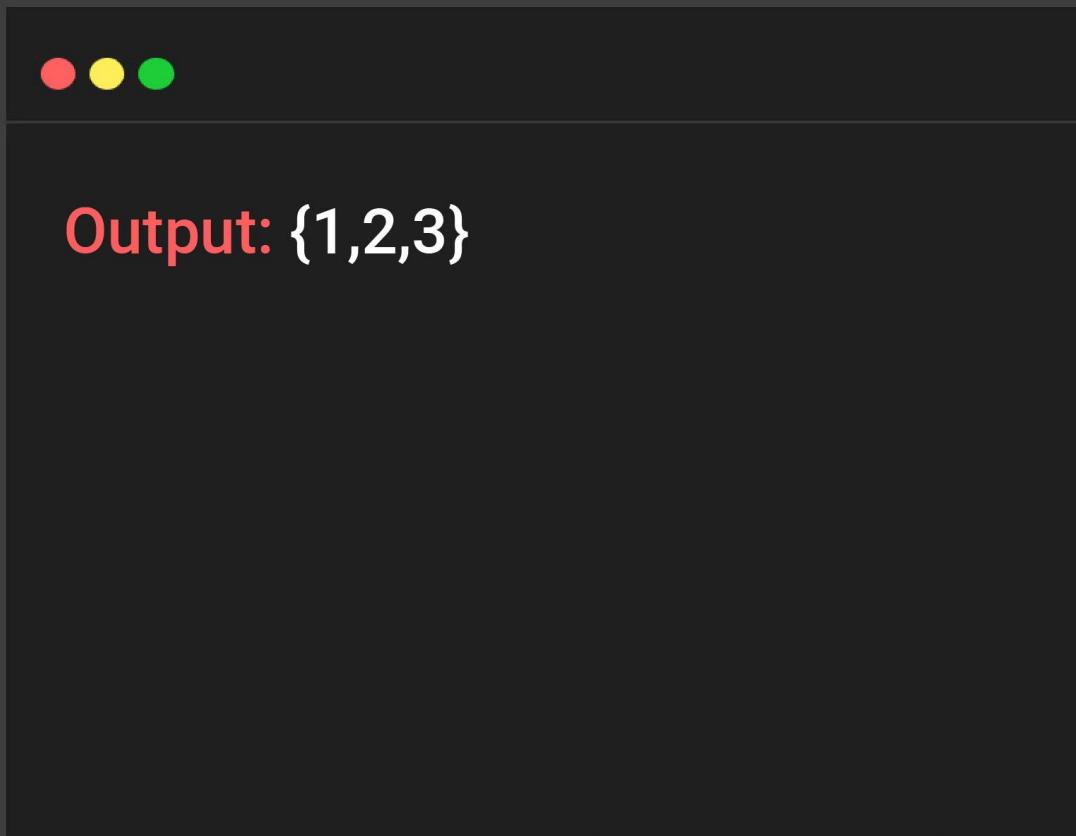
# Guess The Output



Language: Python

```
a={1,2,3}  
b=a.copy()  
b.add(4)  
a
```

# Guess The Output



# Guess The Output



Language: Python

```
>>> a={1,2,3}  
>>> {x*2 for x in a|{4,5}}
```

# Guess The Output



**Output:** {8, 2, 10, 4, 6}

**Explanation:**

Set comprehensions are allowed.

# Guess The Output



Language: Python

```
a={1,2,3}  
b={1,2,3}  
c=a.issubset(b)  
print(c)
```

# Guess The Output



**Output:** true

**Explanation:**

The method `issubset()` returns True if b is a proper subset of a.

# Guess The Output



Language: Python

```
a={1,2,3}  
b={1,2,3}  
c=a.issuperset(b)  
print(c)
```

# Guess The Output



**Output:** False

**Explanation:**

The method `issubset()` returns True if b is a proper subset of a.

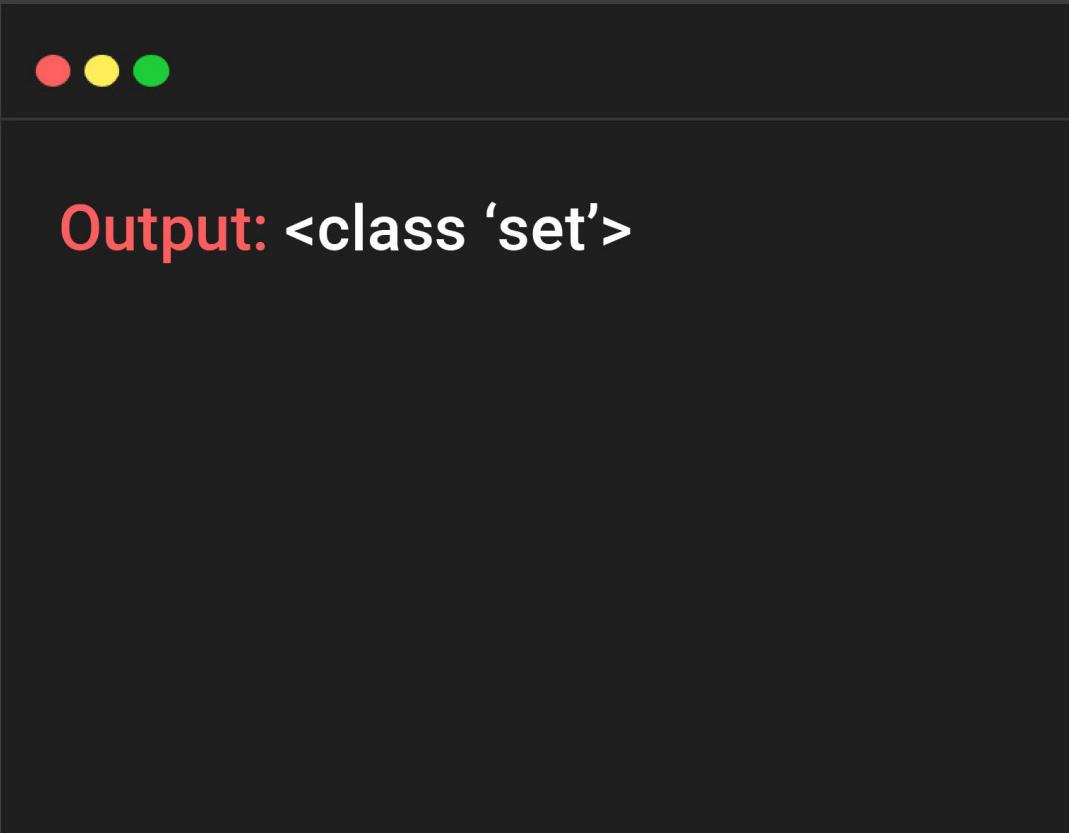
# Guess The Output



Language: Python

```
s=set()  
type(s)
```

# Guess The Output



# Guess The Output



Language: Python

```
s={4>3, 0, 3-3}  
all(s)
```

# Guess The Output



**Output:** False

**Explanation:**

In the example shown above, we have 0 as a value. Hence false is returned.

# Guess The Output



Language: Python

```
z=set('abc')
z.add('san')
z.update(set(['p', 'q']))
z
```

# Guess The Output



**Output:** {'a', 'b', 'c', 'p', 'q', 'san'}

## Explanation:

The code shown first adds the element 'san' to the set z. The set z is then updated and two more elements, namely, 'p' and 'q' are added to it. Hence the output is: {'a', 'b', 'c', 'p', 'q', 'san'}

# Guess The Output



Language: Python

```
A = [[1, 2, 3],  
     [4, 5, 6],  
     [7, 8, 9]]
```

# Guess The Output



**Output:** A[1][2]

**Explanation:**

The output that is required is 6, that is, row 2, item 3. This position is represented by the statement: A[1][2].

# Guess The Output



Language: Python

```
A = [[1, 2, 3],  
     [4, 5, 6],  
     [7, 8, 9]]  
[A[i][i] for i in range(len(A))]
```

# Guess The Output



**Output:** [1, 5, 9]

## Explanation:

We can also perform tasks like pulling out a diagonal. The expression shown above uses range to generate the list of offsets and the indices with the row and column the same, picking out  $A[0][0]$ , then  $A[1][1]$  and so on. Hence the output of the code is: [1, 5, 9].

# Guess The Output



Language: Python

```
l=[[1, 2, 3], [4, 5, 6]]  
for i in range(len(l)):  
    for j in range(len(l[i])):  
        l[i][j]+=10  
l
```

# Guess The Output



**Output:** [11, 12, 13], [14, 15, 16]

**Explanation:**

We use range twice if the shapes differ.  
Each element of list l is increased by 10.  
Hence the output is: [[11, 12, 13],  
[14, 15, 16]]

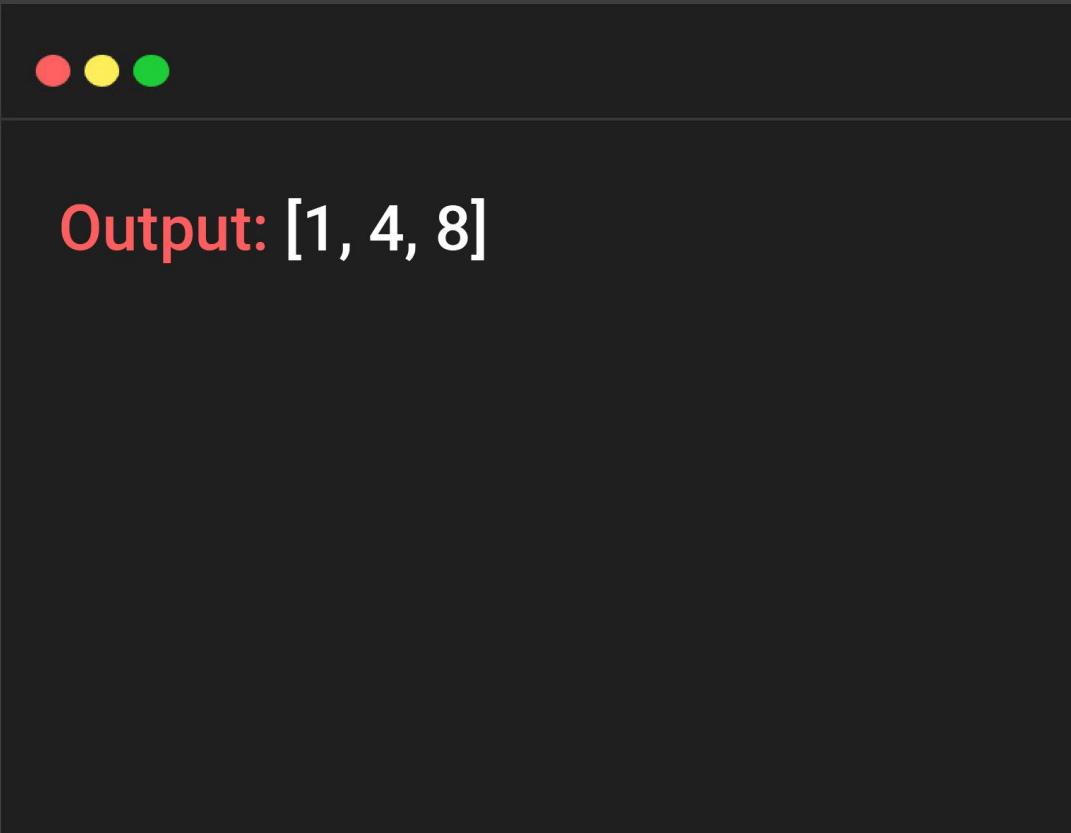
# Guess The Output



Language: Python

```
>>>t = (1, 2, 4, 3, 8, 9)
>>>[t[i] for i in range(0, len(t), 2)]
```

# Guess The Output



# Guess The Output



Language: Python

```
def change(a):
    b=[x*2 for x in a]
    print(b)
def change(a):
    b=[x*x for x in a]
    print(b)
from mod1 import change
from mod2 import change
s=[1,2,3]
change(s)
```

# Guess The Output



**Output:** There is name clause.

**Explanation:**

A name clash is when two different entities with the same identifier become part of the same scope. Since both the modules have the same function name, there is a name clash.

# Guess The Output



Language: Python

```
from math import factorial  
print(math.factorial(5))
```

# Guess The Output



**Output:** Error

**Explanation:**

There is name clause.

# Guess The Output



Language: Python

```
def printMax(a, b):
    if a > b:
        print(a, 'is maximum')
    elif a == b:
        print(a, 'is equal to', b)
    else:
        print(b, 'is maximum')
printMax(3, 4)
```

cs mock

# Guess The Output



**Output:** 4 is maximum

**Explanation:**

Here, we define a function called `printMax` that uses two parameters called `a` and `b`. We find out the greater number using a simple `if..else` statement and then print the bigger number.

# Guess The Output



Language: Python

```
def maximum(x, y):
    if x > y:
        return x
    elif x == y:
        return 'The numbers are equal'
    else:
        return y
print(maximum(2, 3))
```

# Guess The Output



**Output:** 3

**Explanation:**

The maximum function returns the maximum of the parameters, in this case the numbers supplied to the function. It uses a simple if..else statement to find the greater value and then returns that value.

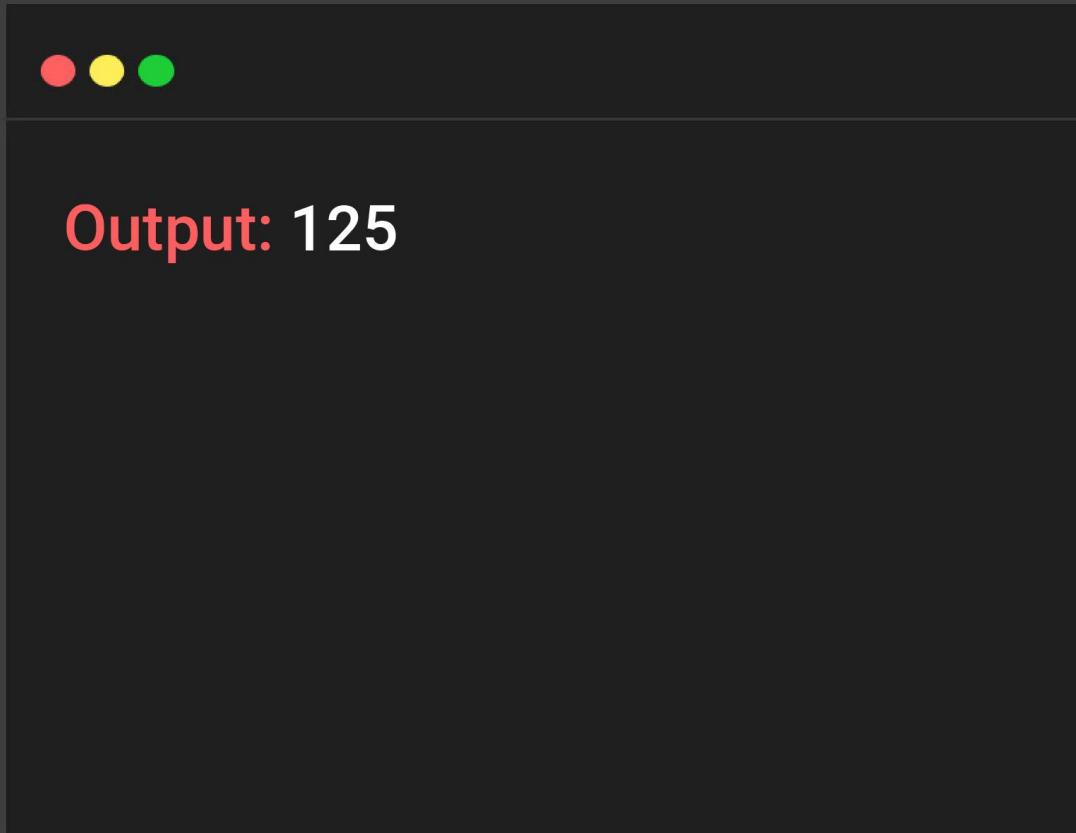
# Guess The Output



Language: Python

```
lamb = lambda x: x ** 3
print(lamb(5))
```

# Guess The Output



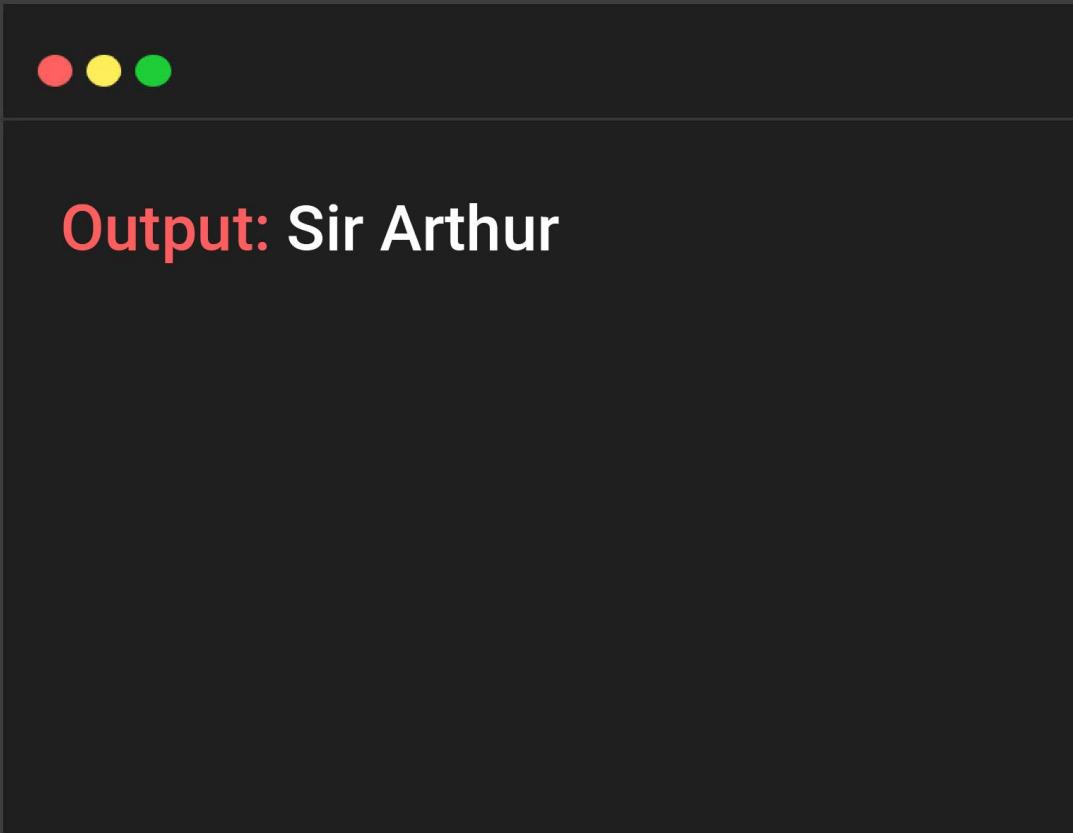
# Guess The Output



Language: Python

```
def writer():
    title = 'Sir'
    name = (lambda x:title + ' ' + x)
    return name
who = writer()
who('Arthur')
```

# Guess The Output



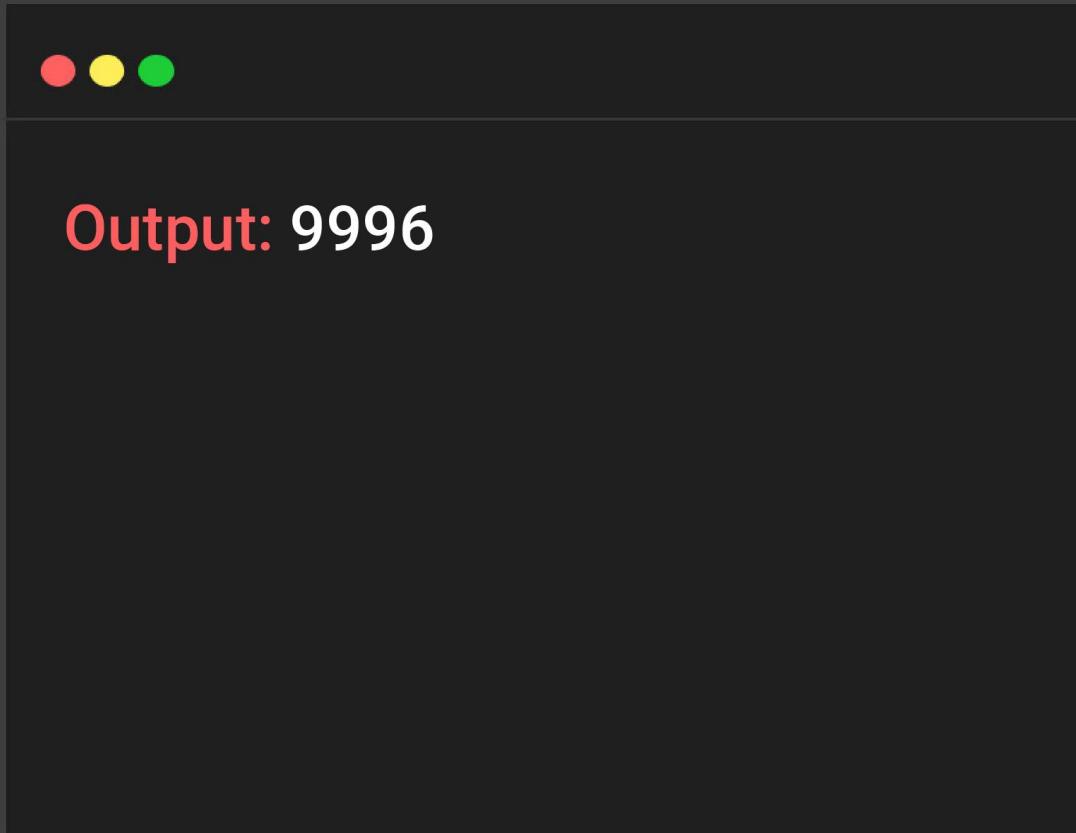
# Guess The Output



Language: Python

```
min = (lambda x, y: x if x < y else y)
min(101*99, 102*98)
```

# Guess The Output



# Guess The Output



Language: Python

```
i=0
def change(i):
    i=i+1
    return i
change(1)
print(i)
```

# Guess The Output



**Output:** 0

**Explanation:**

Any change made in to an immutable data type in a function isn't reflected outside the function..



## About us :

CS Mock aims to smoothen the placement journey of college graduates by polishing their skills, highlighting the points of improvement and boosting their confidence. We provide 1-on-1 live mock interview and mentorship with industry professionals who are part of hiring team.



+91 9956127183



support@csmock.com



www.csmock.com

