Date : 4th - 09 - 2020
Morning Session : 9am – 11.00 PM
By ~ Rohan Kumar

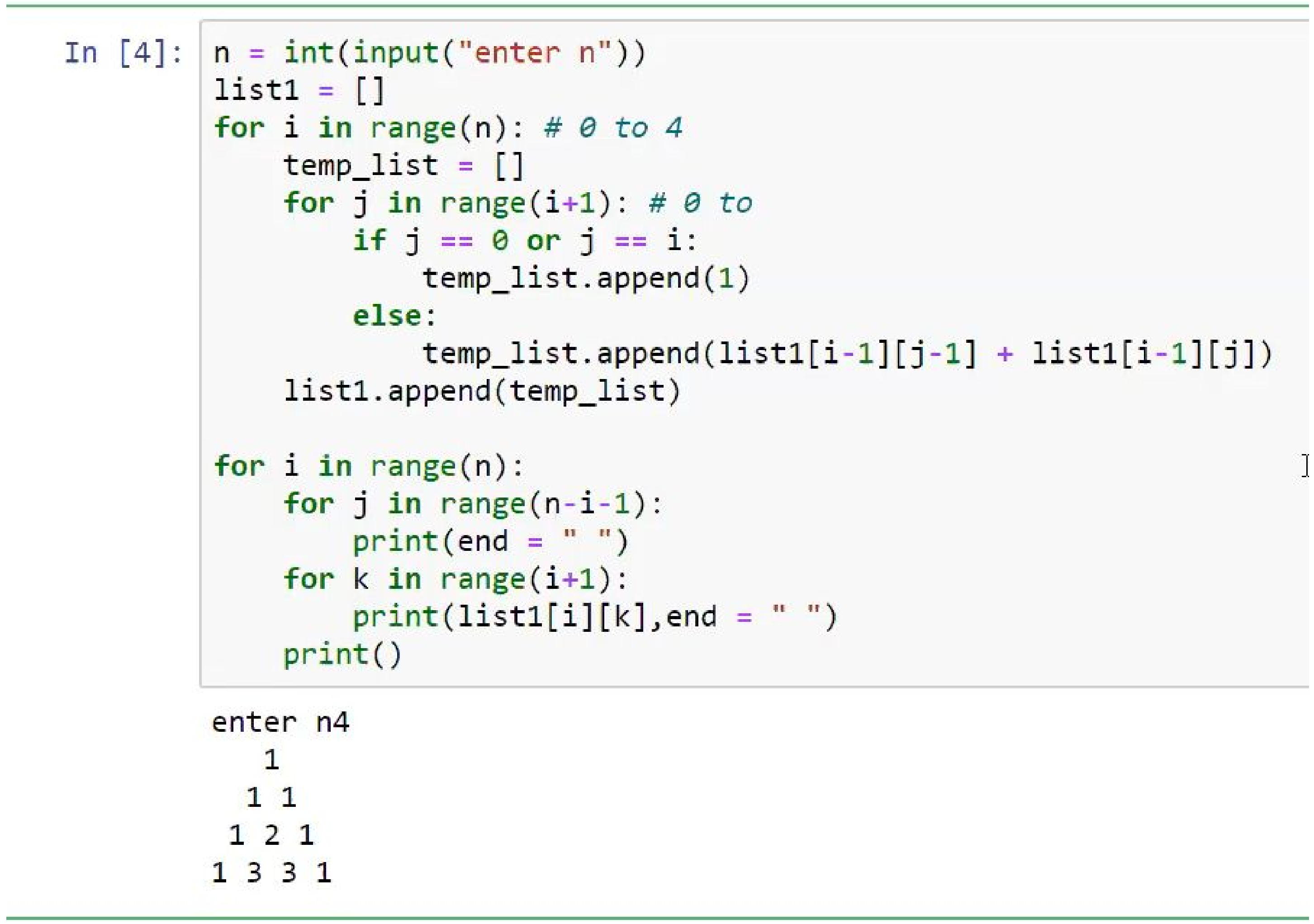
Topics: Python Comprehensions

Pascal Triangle(list): Pascal's triangle is a triangular array of the binomial coefficients. Write a function that takes an integer value n as input and prints first n lines of the Pascal's triangle. Following are the first 6 rows of Pascal's Triangle.

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
```

Follow the pattern:

1) **Start and end with 1**
2) **In between element are sum of above row elements**
3) **N number of rows**



```
In [4]: n = int(input("enter n"))
        list1 = []
        for i in range(n): # 0 to 4
            temp_list = []
            for j in range(i+1): # 0 to
                if j == 0 or j == i:
                    temp_list.append(1)
                else:
                    temp_list.append(list1[i-1][j-1] + list1[i-1][j])
            list1.append(temp_list)

        for i in range(n):
            for j in range(n-i-1):
                print(end = " ")
            for k in range(i+1):
                print(list1[i][k],end = " ")
            print()
```

```
enter n4
   1
  1 1
 1 2 1
1 3 3 1
```

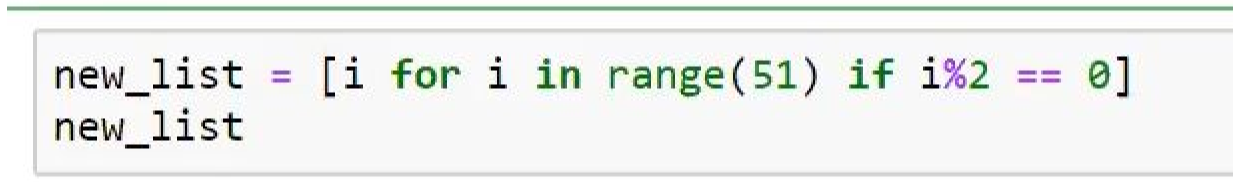
Comprehensions: Comprehensions in Python provide us with a short and concise way to construct new sequences (such as lists, set, dictionary etc.) using sequences which have been already defined. Python supports the following types of comprehensions:

1) List
2) Dictionary
3) Sets

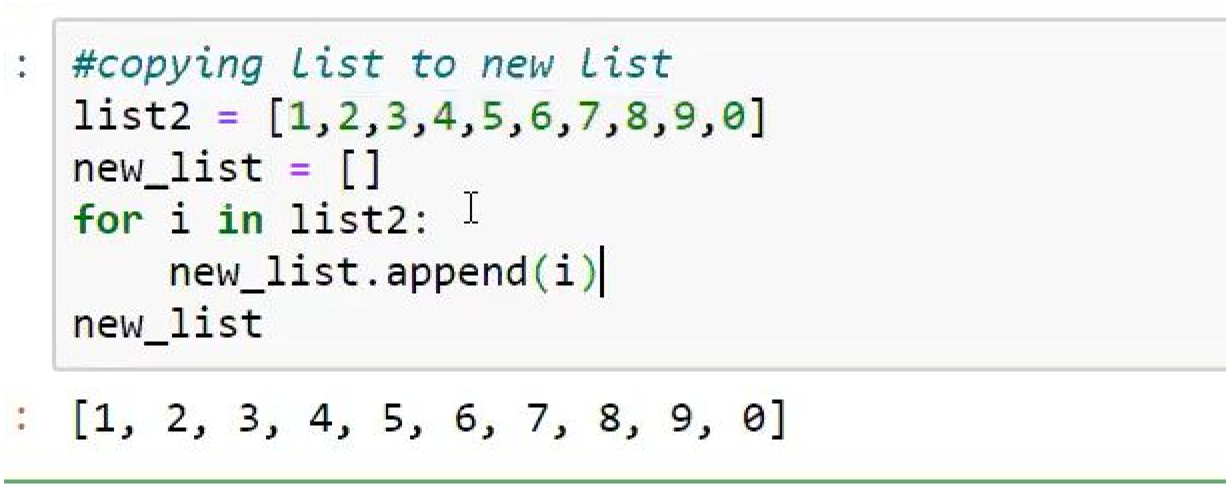
Comprehensions are :

1) Easier & more readable way to create list
2) Code will be short
    - Only in one line

List Comprehensions:

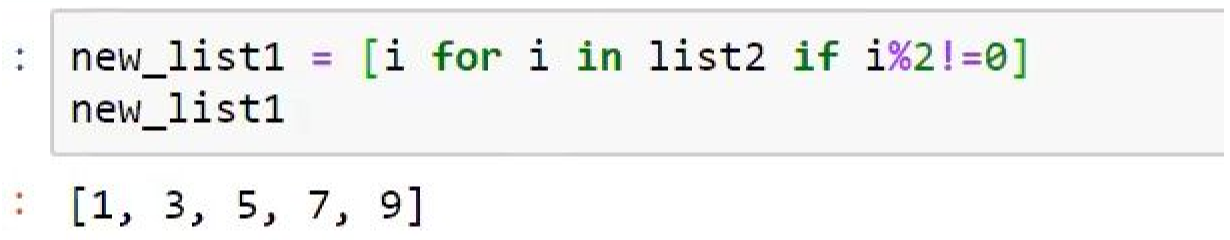


```
new_list = [i for i in range(51) if i%2 == 0]
new_list
```



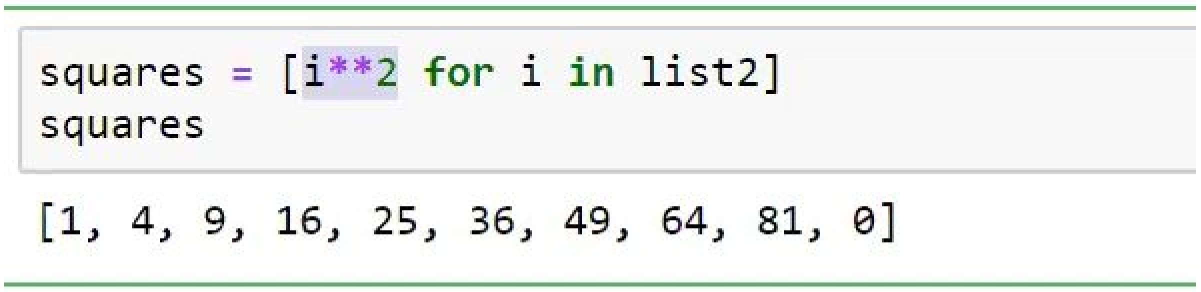
```
#copying list to new list
list2 = [1,2,3,4,5,6,7,8,9,0]
new_list = []
for i in list2:
    new_list.append(i)
new_list
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
```



```
new_list1 = [i for i in list2 if i%2!=0]
new_list1
```

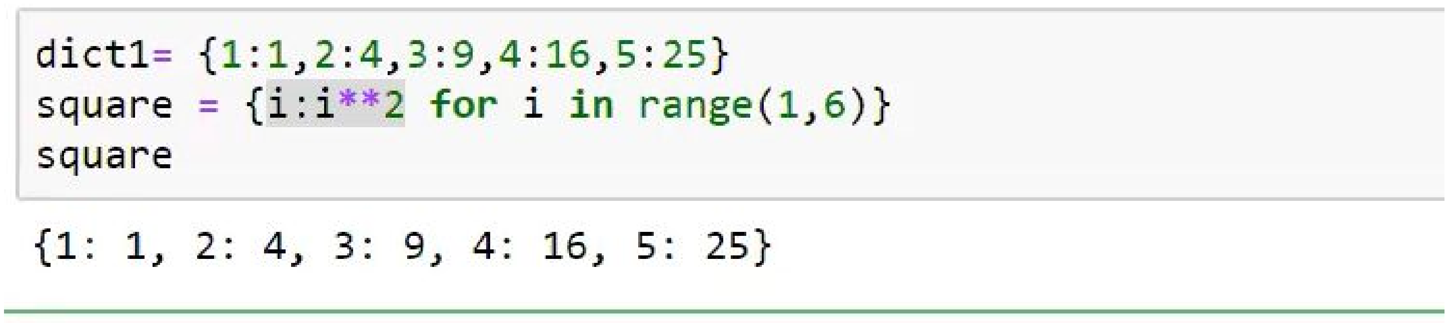
```
[1, 3, 5, 7, 9]
```



```
squares = [i**2 for i in list2]
squares
```

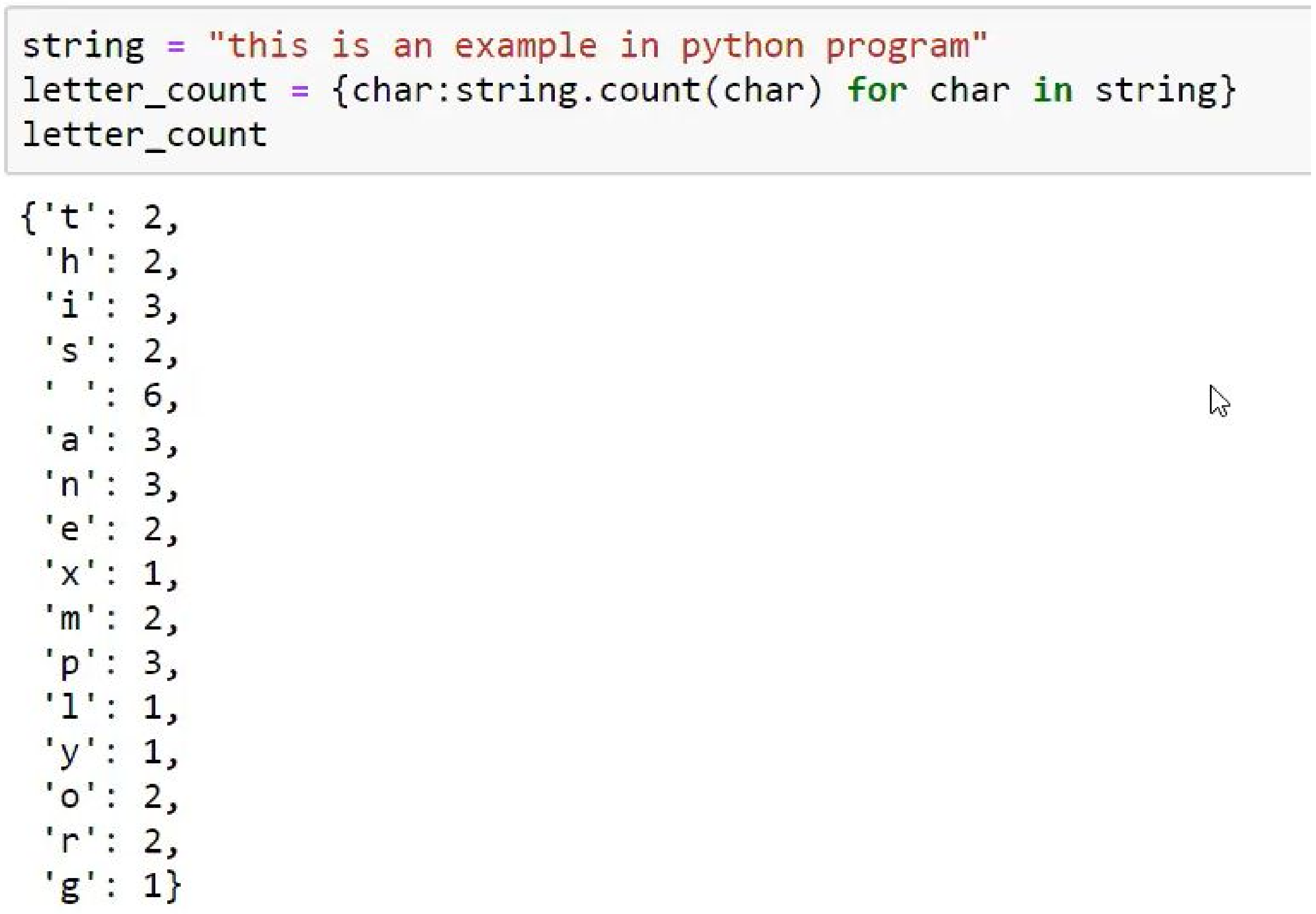
```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 0]
```

Dictionary Comprehension:



```
dict1= {1:1,2:4,3:9,4:16,5:25}
square = {i:i**2 for i in range(1,6)}
square
```

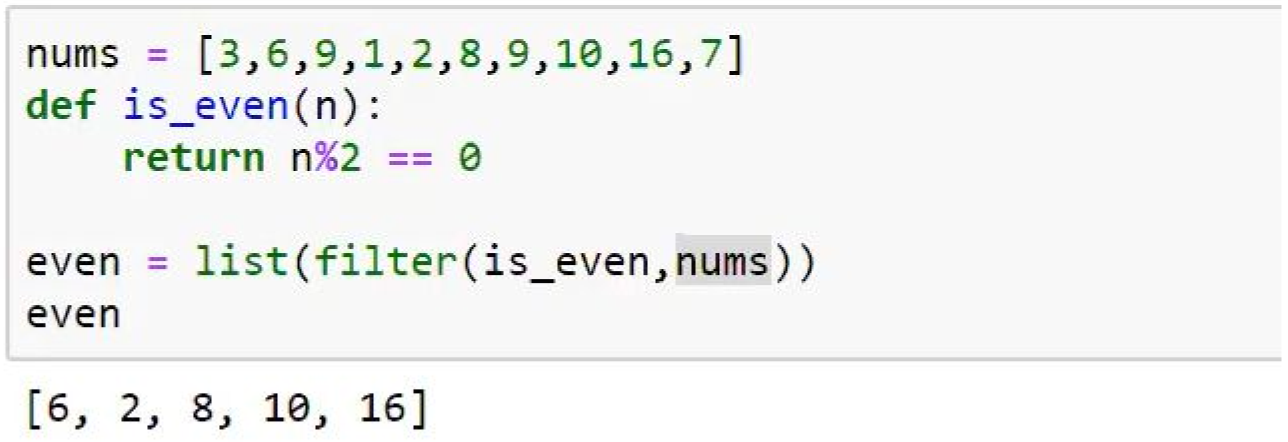
```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```



```
string = "this is an example in python program"
letter_count = {char:string.count(char) for char in string}
letter_count
```

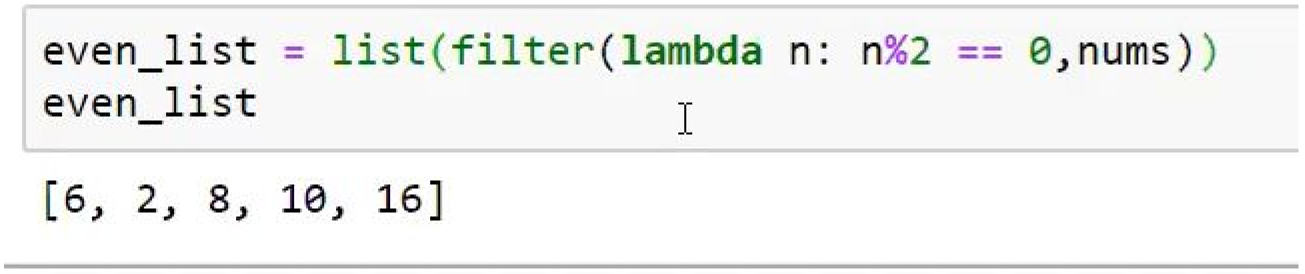
```
{'t': 2,
 'h': 2,
 'i': 3,
 's': 2,
 ' ': 6,
 'a': 3,
 'n': 3,
 'e': 2,
 'x': 1,
 'm': 2,
 'p': 3,
 'l': 1,
 'y': 1,
 'o': 2,
 'r': 2,
 'g': 1}
```

Filter: filters elements based on condition.



```
nums = [3,6,9,1,2,8,9,10,16,7]
def is_even(n):
    return n%2 == 0

even = list(filter(is_even,nums))
even
```

```
[6, 2, 8, 10, 16]
```

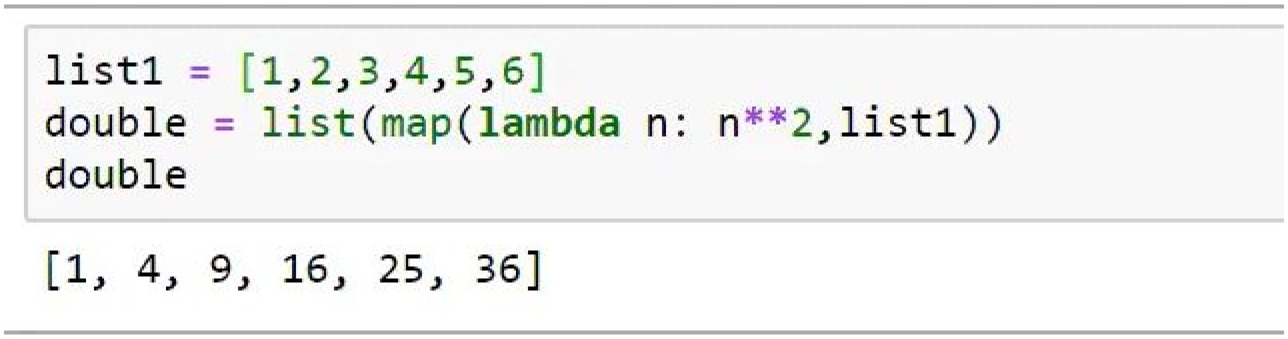


```
even_list = list(filter(lambda n: n%2 == 0,nums))
even_list
```

```
[6, 2, 8, 10, 16]
```

What is Lambda: lambda anonymous function it can take any number of arguments but it can only have one expression

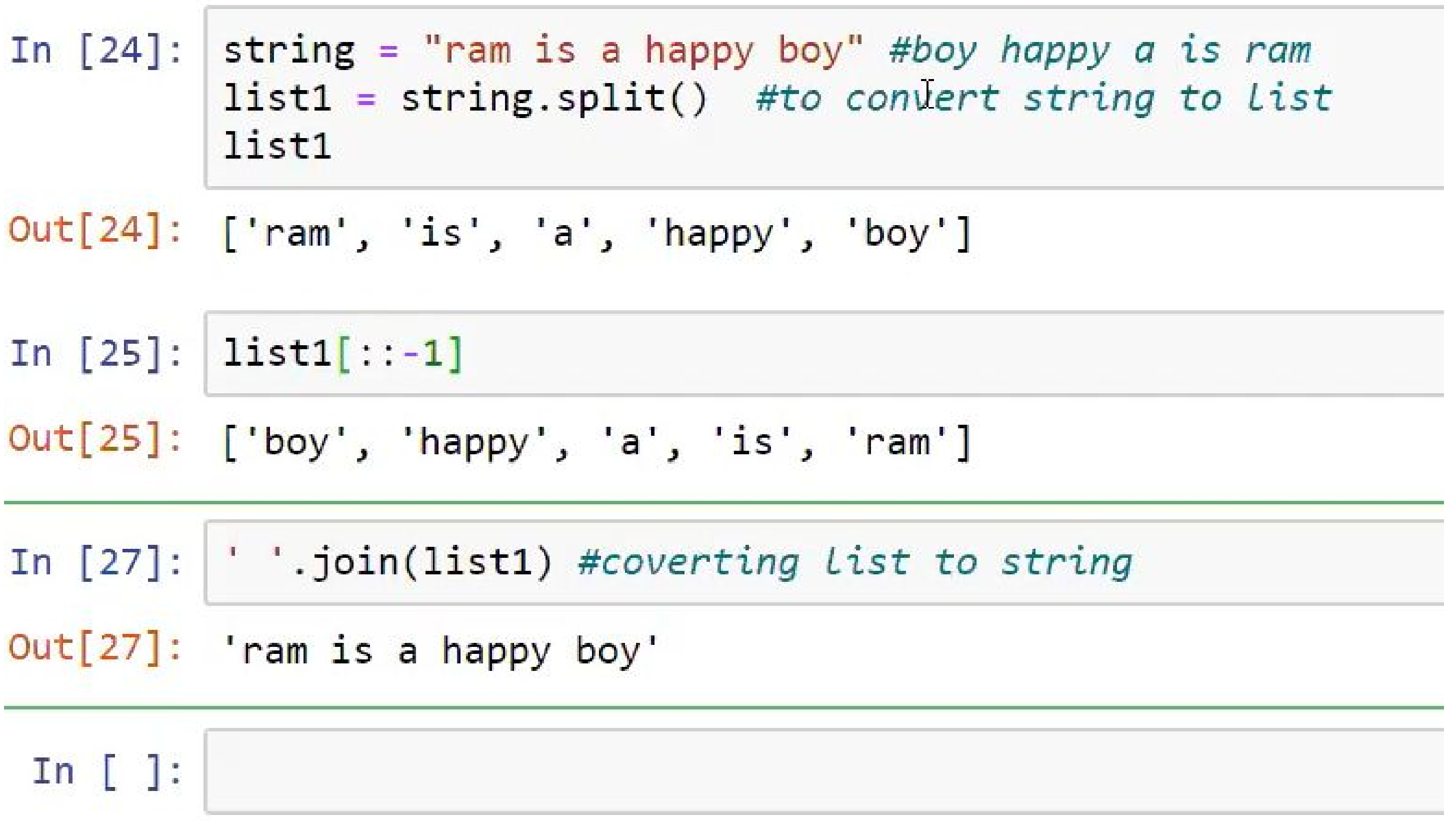
Map: it changes the list or array by applying some operations



```
list1 = [1,2,3,4,5,6]
double = list(map(lambda n: n**2,list1))
double
```

```
[1, 4, 9, 16, 25, 36]
```

Reverse the string:

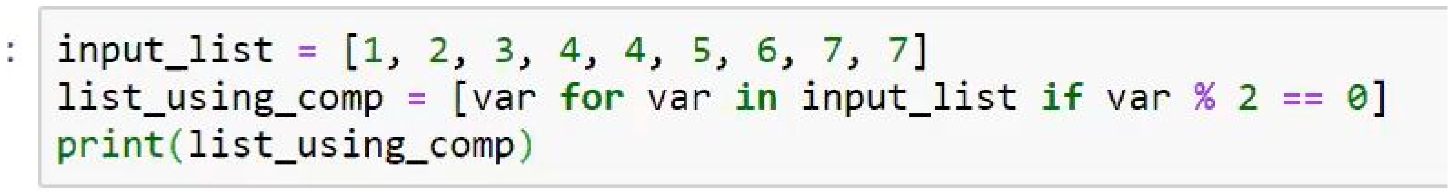


```
In [24]: string = "ram is a happy boy" #boy happy a is ram
         list1 = string.split()  #to convert string to list
         list1

Out[24]: ['ram', 'is', 'a', 'happy', 'boy']

In [25]: list1[::-1]

Out[25]: ['boy', 'happy', 'a', 'is', 'ram']

In [27]: ' '.join(list1) #coverting list to string

Out[27]: 'ram is a happy boy'

In [ ]:
```

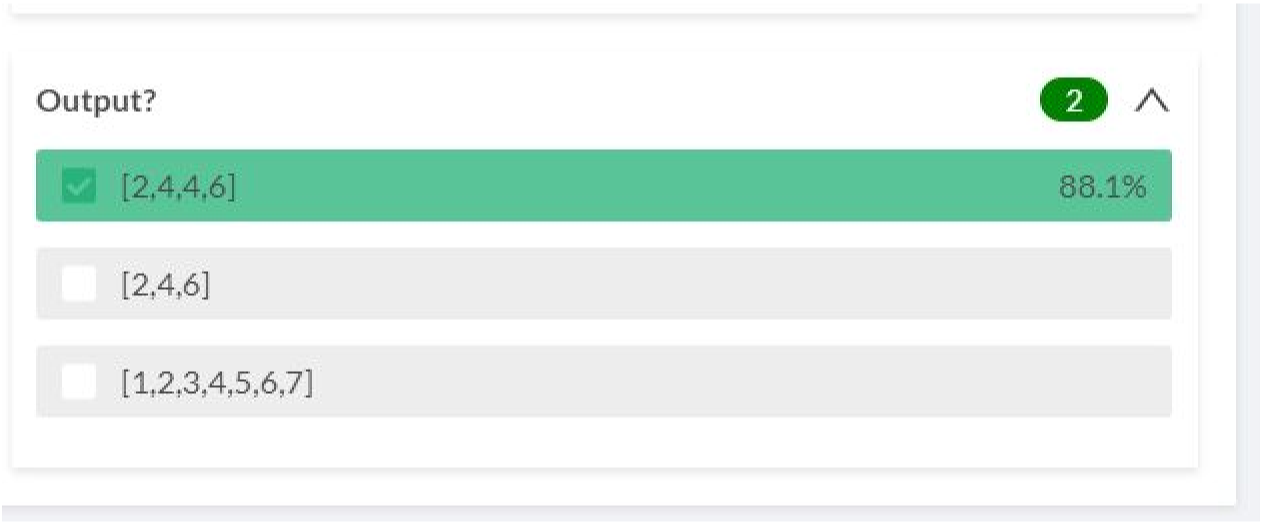
Problems Related to Ds: (list, tuple, dict, set)

For explanation Kindly go through the lecture….

MCQ 1 :



```
input_list = [1, 2, 3, 4, 4, 5, 6, 7, 7]
list_using_comp = [var for var in input_list if var % 2 == 0]
print(list_using_comp)
```



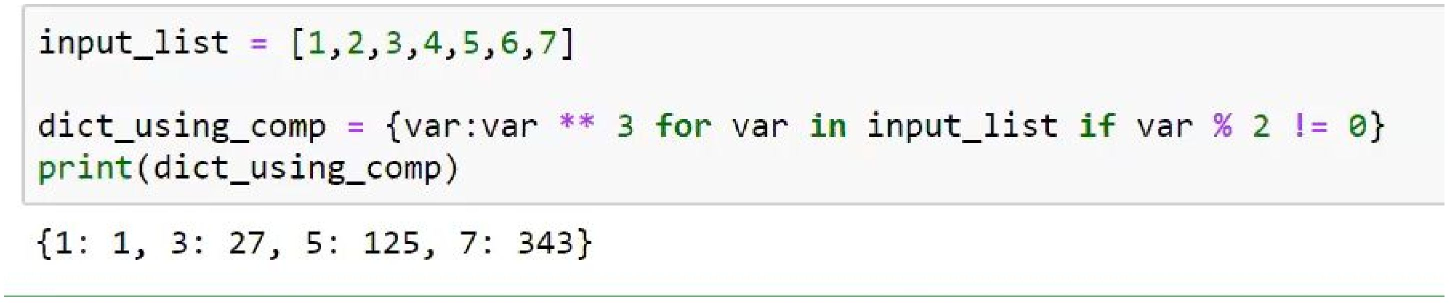
Output? 2

[✓ [2,4,4,6]] 88.1%

[2,4,6]

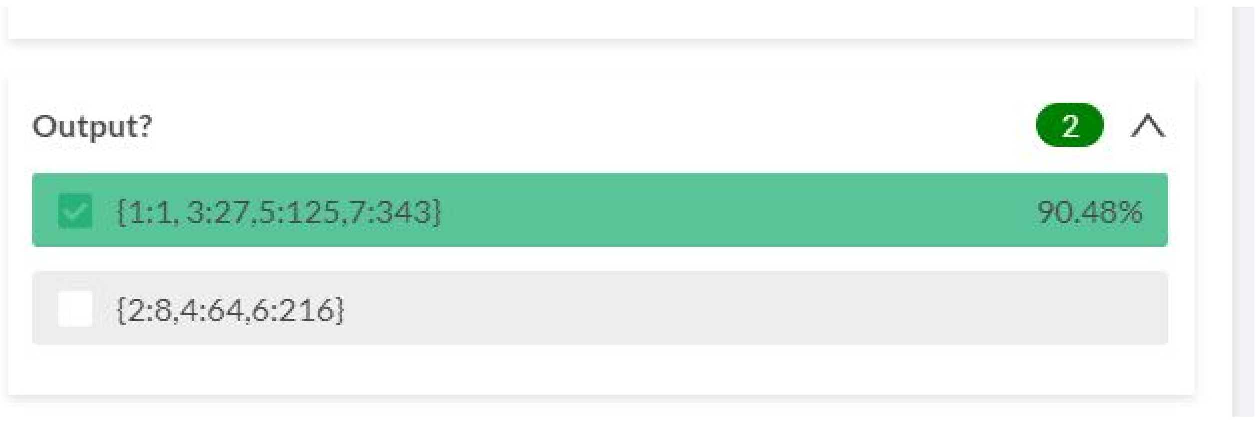
[1,2,3,4,5,6,7]

MCQ 2:



```
input_list = [1,2,3,4,5,6,7]

dict_using_comp = {var:var ** 3 for var in input_list if var % 2 != 0}
print(dict_using_comp)
```

```
{1: 1, 3: 27, 5: 125, 7: 343}
```

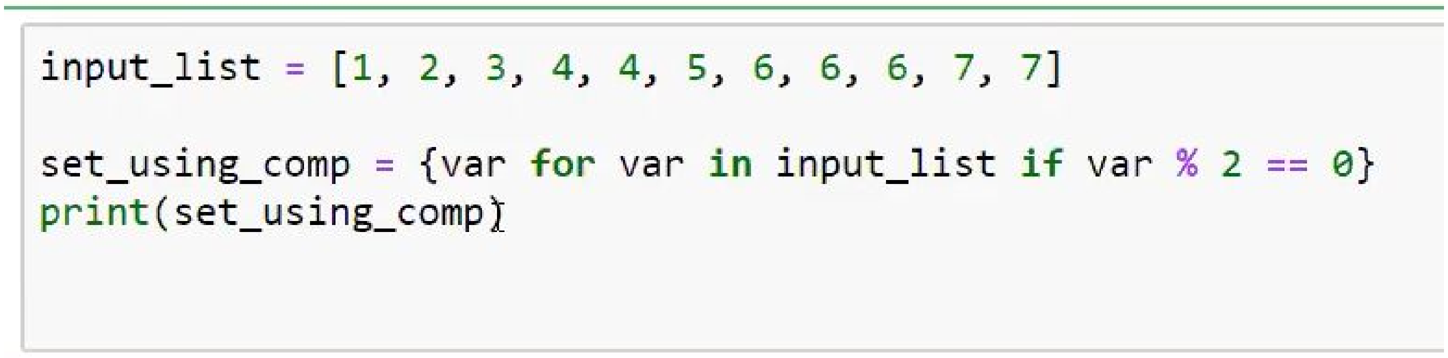


Output? 2

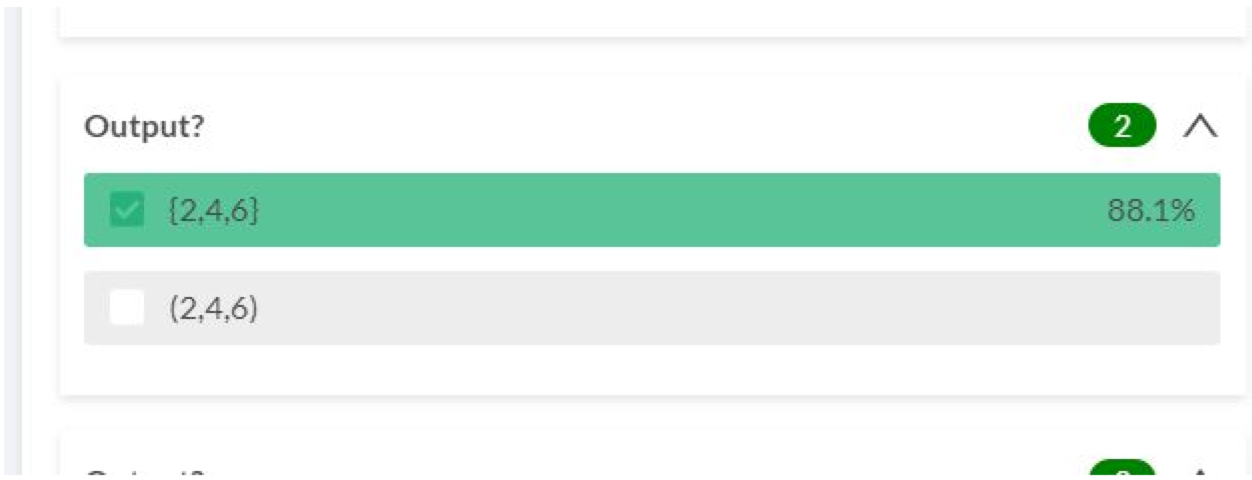
[✓ {1:1, 3:27,5:125,7:343}] 90.48%

{2:8,4:64,6:216}

MCQ 3:



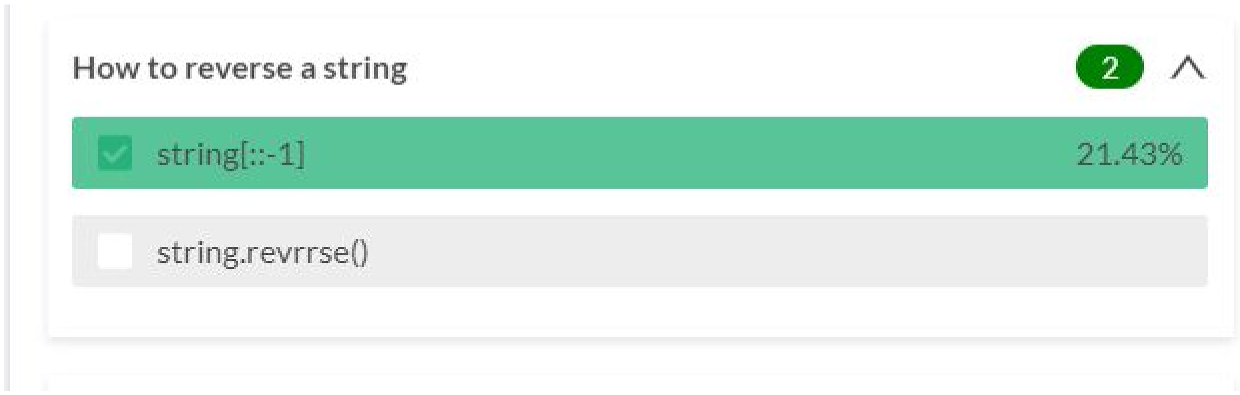
```
input_list = [1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 7]

set_using_comp = {var for var in input_list if var % 2 == 0}
print(set_using_comp)
```



Output? 2

- [✓] {2,4,6} 88.1%
- [ ] (2,4,6)

MCQ 4:



How to reverse a string 2

- [✓] string[::-1] 21.43%
- [ ] string.revrrse()