ui22cs03

October 8, 2023

Write Python programs using 1. List Comprehension

[ ]:
```
#List Comprehnesion :-  A Python list comprehension consists of brackets␣
 ↪containing the expression, which is executed for each element along with the␣
 ↪for loop to iterate over each element in the Python list. \
# List comprehension offers a shorter syntax when you want to create a new list␣
 ↪based on the values of an existing list.
# Here is the method it writes:> newlist = [expression for item in iterable if␣
 ↪condition == True]
```

[ ]:
```
#Let's start with basic Syntax of List Comprehension
#Here we used List Comprehension to Search the list with given condition
cars= ["Toyota","Tata","Tesla","Mahindra","Suzuki"]
new_cars = [x for x in cars if "T" in x]
only_Suzuki = [car for car in cars if car!="Suzuki"]
all_capital = [car.upper() for car in cars]
sort = []
#tata_means = [car if car !="Tata" else "Reliable" for car in cars]
# newlist = [x if x != "banana" else "orange" for x in fruits]
print(new_cars)
print(only_Suzuki)
print(all_capital)
#print(tata_means)
```

[ ]:
```
# Iterating through a string Using List Comprehension
iterate = [letter for letter in "ADITYA"]
print(iterate)
```

```
['A', 'D', 'I', 'T', 'Y', 'A']
```

[ ]:
```
range_newlist = [x for x in range(10)]
print(range_newlist)
```

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

[ ]:
```
#Conditionals in List Comprehension
even_no = [num for num in range(100) if num%2==0]
print(even_no)
```

```
num_list = [y for y in range(100) if y%2==0 if y%5==0]
print(num_list)
obj = ["Even" if i%2==0 else "Odd" for i in num_list]
print(obj)
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40,
42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80,
82, 84, 86, 88, 90, 92, 94, 96, 98]
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
['Even', 'Even', 'Even', 'Even', 'Even', 'Even', 'Even', 'Even', 'Even', 'Even']
```

[ ]:
```
#Transpose of matrix in list comprehension
matrix_create = [[j for j in range(5)] for i in range(3)]

print(matrix_create)
matrix = [[1, 2], [3,4], [5,6], [7,8]]
transpose = [[row[i] for row in matrix] for i in range(2)]
print(matrix_create)
print(matrix)
print (transpose)
```

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

2. Lambda Operator : an anonymous function means that a function is without a name. As we already know that def keyword is used to define the normal functions and the lambda keyword is used to create anonymous functions.

---

lambda arguments : expression

[ ]:
```
x = lambda a: a + 10
print(x(5))
```

```
15
```

[ ]:
```
calc = lambda num: "Even number" if num % 2 == 0 else "Odd number"
print(calc(21))
```

```
Odd number
```

[ ]:
```
# using lambda to print table of 10
numbers = list(map(lambda i: i*10, [i for i in range(1, 6)]))
print(numbers)
```

[1]:
```
#One-line function to square a number using lambda operator:
x=lambda a: a**5
print(x(10))
```

```
100000
```

[2]:
```
m=lambda a,b: a+b
print(m([1,2,3],[7,8,9]))
```

```
[1, 2, 3, 7, 8, 9]
```

Map : map() function returns a map object(which is an iterator) of the results after applying the given function to each item of a given iterable (list, tuple etc.)

[4]:
```
#Doubling each element in a list using lamda and map:
a=[1,2,4.4,7.2213,312]
b=map(lambda i:i*2,a)
print(b)
print(list(b))
```

```
<map object at 0x7b8461918760>
[2, 4, 8.8, 14.4426, 624]
```

[7]:
```
#Making a list containing the size of all items of a list:
a=['PPFS','Letcountit','DSA','DBMS']
b=map(lambda i:len(i),a)
print(list(b))
```

```
[4, 10, 3, 4]
```

Reduce The reduce(fun,seq) function is used to apply a particular function passed in its argument to all of the list elements mentioned in the sequence passed along.This function is defined in “functools” module.

[9]:
```
#Summing all the elements of a list using reduce:
from functools import reduce
e=[1,2,4,6,2]
sum=reduce(lambda a,b : a+b,e)
print(sum)
```

```
15
```

[12]:
```
#Concatenating all strings in a list
d=["Indian ","Institute ", "of ","Information ","Technology",", Surat","␣
 ↪Gujarat"]
college_name=reduce(lambda a,b : a+b,d)
print(college_name)
```

```
Indian Institute of Information Technology, Surat Gujarat
```

Filter The filter() method filters the given sequence with the help of a function that tests each element in the sequence to be true or not.

[ ]:
```
#Returning a list of numbers more than 18 from a given list:
ages=[34,5,32,5,88,32,65,23,6,87,15]
ad=filter(lambda a:a>18,ages)
print(ad)
print(list(ad))
```

[13]:
```
#Returning a list of even numbers more than 45 from a given
num=[1,5,7,13,73,68,238,124,32,1156,23,719]
at=(filter(lambda a: a%2==0 and a>45,num))
print(list(at))
```

```
[68, 238, 124, 1156]
```

Eneumerate The enumerate() function takes a collection (e.g. a tuple) and returns it as an enumerate object. The enumerate() function adds a counter as the key of the enumerate object.

[15]:
```
#Converting a tuple into an enumerate object:
x=("A","D","I","T","Y","A")
y=enumerate(x)
print(y)
print(list(y))
```

```
<enumerate object at 0x7b84492fa300>
[(0, 'A'), (1, 'D'), (2, 'I'), (3, 'T'), (4, 'Y'), (5, 'A')]
```

[16]:
```
#The zip() function takes iterables (can be zero or more), aggregates them in a␣
 ↪tuple, and returns it.
# Making a dictionary with car names as key and colors as value:
cars=["Mecedez","Volkswagon","BMW"]
colors=["red","blue","black"]
d=dict(zip(cars,colors))
print(d)
```

```
{'Mecedez': 'red', 'Volkswagon': 'blue', 'BMW': 'black'}
```

[ ]:
```
#Making a dictionary with a vbalue as index and a tuple as value:
 cars=["Mecedez","Volkswagon","BMW"]
colors=[("red","blue"),("black","yellow"),("green","grey")]
d=dict(zip(cars,colors))
print(d)
```