12 Comprehensions

"Add punch to your thought..."

Contents

- What are Comprehensions?
- List Comprehension
- Set Comprehension
- Dictionary Comprehension
- Programs
- Exercises

KanNotes

What are comprehensions?

- Comprehensions offer an easy and compact way of creating lists, sets and dictionaries.
- A comprehension works by looping or iterating over items and assigning them to a container like list, set or dictionary.
- This container cannot be a tuple as tuple being immutable is unable to receive assignments.

List Comprehension

- List comprehension consists of brackets containing an expression followed by a **for** clause, and zero or more **for** or **if** clauses.
- So general form of a list comprehension is

  lst = [expression for var in sequence [optional for and/or if]]
- Examples of list comprehension:

```
# generate 20 random numbers in the range 10 to 100
a = [random.randint(10, 100) for n in range(20)]

# generate square and cube of all numbers between 0 and 10
a = [( x, x**2, x**3) for x in range(10)]

# convert a list of strings to a list of integers
a = [int(x) for x in ['10', '20', '30', '40']
```

- Examples of use of if in list comprehension:

```
# generate a list of even numbers in the range 10 to 30
a = [n for n in range(10, 30) if n % 2 == 0]

# from a list delete all numbers having a value between 20 and 50
a = [num for num in a if num < 20 or num > 50]
```

- Example of use of if-else in list comprehension:

```
# when if-else both are used, place them before for
# replace a vowel in a string with !
a = ['!' if alphabet in 'aeiou' else alphabet for alphabet in 'Technical' ]
```

- Example of use of multiple fors and if in list comprehension:

```
# flatten a list of lists
arr = [[1,2,3,4], [5,6,7,8], [10, 11, 12, 13]]
b = [n for ele in arr for n in ele]       # one way

# * can be used to unpack a list
c = [*arr[0], *arr[1], *arr[2]]     # one more way
```

- Note the difference between nested **for** in a list comprehension and a nested comprehension:

```
# produces [4, 5, 6, 5, 6, 7, 6, 7, 8]. Uses nested for
lst = [a + b for a in [1, 2, 3] for b in [3, 4, 5]]

# produces [[4, 5, 6], [5, 6, 7], [6, 7, 8]]. Uses nested comprehension
lst = [[a + b for a in [1, 2, 3]] for b in [3, 4, 5]]
```

Think of first **for** as outer loop and second **for** as inner loop.

- Example of use of multiple fors and if in list comprehension:

```
# generate all unique combinations of 1, 2 and 3
a = [(i, j, k) for i in [1,2,3] for j in [1,2,3] for k in [1, 2, 3] if i != j \
        and j !=k and k != i]
```

Set Comprehension

- Like list comprehensions, set comprehensions offer an easy way of creating sets. It consists of braces containing an expression followed by a **for** clause, and zero or more **for** or **if** clauses.
- So general form of a set comprehension is

  s = {expression for var in sequence [optional for and/or if]}
- Examples of set comprehension:

```
# generate a set containing square of all numbers between 0 and 10
a = {x**2 for x in range(10)}

# from a set delete all numbers between 20 and 50
a = {num for num in a if num > 20 and num < 50}
```

Dictionary Comprehension

- General form of a dictionary comprehension is as follows:

  dict_var = {key:value for (key, value) in dictonary.items( )}

- Examples of dictionary comprehension:

```
d = {'a': 1, 'b': 2, 'c': 3, 'd': 4}

# obtain dictionary with each value cubed
d1 = {k : v ** 3 for (k, v) in d.items( )}
print(d1)          # prints {'a': 1, 'b': 8, 'c': 27, 'd': 64}

# obtain dictionary with each value cubed if value > 3
d2 = {k : v ** 3 for (k, v) in d.items( ) if v > 3}
print(d2)          # prints {'d': 64}

# Identify odd and even entries in the dictionary
d3 = {k : ('Even' if v % 2 == 0 else 'Odd') for (k, v) in d.items( )}
print(d3)          # prints {'a': 'Odd', 'b': 'Even', 'c': 'Odd', 'd': 'Even'}
```

P</> Programs

Problem 12.1

Using list comprehension, write a program to generate a list of numbers in the range 2 to 50 that are divisible by 2 and 4.

Program

```
lst = [num for num in range(2,51) if num % 2 == 0 and num % 4 == 0]
print(lst)
```

Output

```
[4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48]
```

Problem 12.2

Write a program to flatten the following list using list comprehension:

mat = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]

Program

```
mat = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
a = [num for lst in mat for num in lst]
print(a)
```

Output

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

Problem 12.3

Write a program to create a set containing some randomly generated numbers in the range 15 to 45. Count how many of these numbers are less than 30. Delete all numbers which are less than 30.

Program

```
import random
r = {int(15 + 30 * random.random( )) for num in range(10)}
print(r)
count = len({num for num in r if num < 30})
print(count)
s = {num for num in r if num < 30}
r = r - s
print(r)
```

Output

```
{32, 35, 36, 38, 41, 43, 21, 23, 25, 26}
4
{32, 35, 36, 38, 41, 43}
```

Tips

- Deletion of elements cannot be done while iterating the set. Hence a separate set **s** containing elements below 30 is first created and then **r = r - s** is done to delete set s elements from set r.

Problem 12.4

Write a program using list comprehension to eliminate empty tuples from a list of tuples.

Program

```
lst = [( ), ( ), (10), (10, 20), ('',), (10, 20, 30), (40, 50), ( ), (45)]
lst = [tpl for tpl in lst if tpl]
print(lst)
```

Output

```
[10, (10, 20), ('',), (10, 20, 30), (40, 50), 45]
```

Tips

- **if tpl** returns **True** if the tuple is not empty.

Problem 12.5

Given a string, split it on whitespace, capitalize each element of the resulting list and join them back into a string. Your implementation should use a list comprehension.

Program

```
s1 = 'dreams may change, but friends are forever'
s2 = [' '.join(w.capitalize( ) for w in s1.split( ))]
s3 = s2[0]
print(s3)
```

Output

```
'Dreams May Change, But Friends Are Forever'
```

Tips

- To rebuild the list from capitalized elements, start with an empty string.

Problem 12.6

From a dictionary with string keys create a new dictionary with the vowels removed from the keys.

Program

```
words = { 'Tub' : 1, 'Toothbrush' : 2, 'Towel' : 3, 'Nailcutter' : 4}
d = {''.join(alpha for alpha in k if alpha not in 'aeiou'): v for (k, v) in
        words.items( )}
print(d)
```

Output

```
{'Tb': 1, 'Tthbrsh': 2, 'Twl': 3, 'Nlcttr': 4}
```

Tips

- We have use a list comprehension nested inside a dictionary comprehension.
- The list comprehension builds a new key starting with an empty string, adding only those characters from the key which are not vowels.
- The list comprehension is fed with keys by the dictionary comprehension.

Problem 12.7

Write a program to add two 3 x 4 matrices using

(a) lists
(b) list comprehension

Program

```
mat1 = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
mat2 = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
mat3 = [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]

# iterate through rows
for i in range(len(mat1)) :
```

```
    # iterate through columns
    for j in range(len(mat1[0])) :
        mat3[i][j] = mat1[i][j] + mat2[i][j]
print(mat3)
mat3 = [[mat1[i][j] + mat2[i][j] for j in range(len(mat1[0]))]
            for i in range(len(mat1))]
print(mat3)
```

Output

```
[[2, 4, 6, 8], [10, 12, 14, 16], [18, 20, 22, 24]]
[[2, 4, 6, 8], [10, 12, 14, 16], [18, 20, 22, 24]]
```

Tips

- Nested list comprehension is evaluated in the context of the **for** that follows it.

Problem 12.8

Suppose a dictionary contains following information for 5 employees:

```
emp = {
        'A101' : {'name' : 'Ashish', 'age' : 30, 'salary' : 21000},
        'B102' : {'name' : 'Dinesh', 'age' : 25, 'salary' : 12200},
        'A103' : {'name' : 'Ramesh', 'age' : 28, 'salary' : 11000},
        'D104' : {'name' : 'Akheel', 'age' : 30, 'salary' : 18000},
        'A105' : {'name' : 'Akaash', 'age' : 32, 'salary' : 20000}
    }
```

Using dictionary comprehensions, write a program to create:

- Dictionary of all those codes and values, where codes that start with 'A'.
- Dictionary of all codes and names.
- Dictionary of all codes and ages.
- Dictionary of all codes and ages, where age is more than 30.
- Dictionary of all codes and names, where names start with 'A'.
- Dictionary of all codes and salaries, where salary is in the range 13000 to 20000.

Program

```
emp = {
                'A101' : {'name' : 'Ashish', 'age' : 30, 'salary' : 21000},
                'B102' : {'name' : 'Dinesh', 'age' : 25, 'salary' : 12200},
                'A103' : {'name' : 'Ramesh', 'age' : 28, 'salary' : 11000},
                'D104' : {'name' : 'Akheel', 'age' : 30, 'salary' : 18000},
        }
d1 = {k : v for (k, v) in emp.items( ) if k.startswith('A')}
d2 = {k : v['name'] for (k, v) in emp.items( )}
d3 = {k : v['age'] for (k, v) in emp.items( )}
d4 = {k : v['age'] for (k, v) in emp.items( ) if v['age'] > 30}
d5 = {k : v['name'] for (k, v) in emp.items( ) if v['name'].startswith('A')}
d6 = {k : v['salary'] for (k, v) in emp.items( ) if v['salary'] > 13000 and
v['salary'] <= 20000}
print(d1)
print(d2)
print(d3)
print(d4)
print(d5)
print(d6)
```

Output

```
{'A101': {'name': 'Ashish', 'age': 30, 'salary': 21000}, 'A103': {'name':
'Ramesh', 'age': 28, 'salary': 11000}}
{'A101': 'Ashish', 'B102': 'Dinesh', 'A103': 'Ramesh', 'D104': 'Akheel'}
{'A101': 30, 'B102': 25, 'A103': 28, 'D104': 30}
{}
{'A101': 'Ashish', 'D104': 'Akheel'}
{'D104': 18000}
```

Tips

- Note that the data has been organized in nested directories.
- To access 'Ashish' we need to use the syntax **emp['A101']['name']**
- To access 32 we need to use the syntax **emp['A105']['age']**

Exercises

[A] State whether the following statements are True or False:

(a) Tuple comprehension offers a fast and compact way to generate a tuple.

(b) List comprehension and dictionary comprehension can be nested.

(c) A list being used in a list comprehension cannot be modified when it is being iterated.

(d) Sets being immutable cannot be used in comprehension.

(e) Comprehensions can be used create a list, set or a dictionary.

[B] Answer the following questions:

(a) Write a program that generates a list of integer coordinates for all points in the first quadrant from (1, 1) to (5, 5). Use list comprehension.

(b) Using list comprehension, write a program to create a list by multiplying each element in the list by 10.

(c) Write a program to generate first 20 Fibonacci numbers using list comprehension.

(d) Write a program to generate two lists using list comprehension. One list should contain first 20 odd numbers and another should contain first 20 even numbers.

(e) Suppose a list contains positive and negative numbers. Write a program to create two lists—one containing positive numbers and another containing negative numbers.

(f) Suppose a list contains 5 strings. Write a program to convert all these strings to uppercase.

(g) Write a program that converts list of temperatures in Fahrenheit degrees to equivalent Celsius degrees using list comprehension.

(h) Write a program to generate a 2D matrix of size 4 x 5 containing random multiples of 4 in the range 40 to 160.

(i) Write a program that converts words present in a list into uppercase and stores them in a set.

[C] Attempt the following questions:

(a) Consider the following code snippet:

```
s = set([int(n) for n in input('Enter values: ').split( )])
print(s)
```

What will be the output of the above code snippet if input provided to it is 1 2 3 4 5 6 7 2 4 5 0?

(b) How will you convert the following code into a list comprehension?

```
a = [ ]
for n in range(10, 30) :
   if n % 2 == 0 :
      a.append(n)
```

(c) How will you convert the following code into a set comprehension?

```
a = set( )
for n in range(21, 40) :
   if n % 2 == 0 :
      a.add(n)
print(a)
```

(d) What will be the output of the following code snippet?

```
s = [a + b for a in ['They ', 'We '] for b in ['are gone!', 'have come!']]
print(s)
```

(e) From the sentence

```
sent = 'Pack my box with five dozen liquor jugs'
```

how will you generate a set given below?

{'liquor', 'jugs', 'with', 'five', 'dozen', 'Pack'}

(f) Which of the following the correct form of dictionary comprehension?

i. dict_var = {key : value for (key, value) in dictonary.items( )}
ii. dict_var = {key : value for (key, value) in dictonary}
iii. dict_var = {key : value for (key, value) in dictonary.keys( )}

(g) Using comprehension how will you convert

{'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4, 'e' : 5}

into

{'A' : 100, 'B' : 200, 'C' : 300, 'D' : 400, 'E' : 500}?

(h) What will be the output of the following code snippet?

```
lst = [2, 7, 8, 6, 5, 5, 4, 4, 8]
s = {True if n % 2 == 0 else False for n in lst}
print(s)
```

(i) How will you convert

d = {'AMOL' : 20, 'ANIL' : 12, 'SUNIL' : 13, 'RAMESH' : 10}

into

{'Amol' : 400, 'Anil' : 144, 'Sunil' : 169, 'Ramesh' : 100}

(j) How will you convert words present in a list given below into uppercase and store them in a set?

lst = ['Amol', 'Vijay', 'Vinay', 'Rahul', 'Sandeep']