



gauravsharma727545@gmail.com ~

NPTEL (https://swayam.gov.in/explorer?ncCode=NPTEL) » Programming, Data Structures And

Algorithms Using Python (course)

Announcements (announcements)

About the Course (https://swayam.gov.in/nd1 noc19 cs40/preview) Ask a Question (forum)

Progress (student/home) Mentor (student/mentor)

Course outline

How to access the portal

Week 1: Introduction

Week 1 Quiz

Week 2: Basics of Python

Week 2 Quiz

Week 2 Programming Assignment

Week 3: Lists, inductive function

Week 4 Programming Assignment

Due on 2019-08-29, 23:59 IST

Write two Python functions as specified below. Paste the text for both functions together into the submission window. Your function will be called automatically with various inputs and should return values as specified. Do not write commands to read any input or print any output.

- You may define additional auxiliary functions as needed.
- In all cases you may assume that the value passed to the function is of the expected type, so your function does not have to check for malformed inputs.
- For each function, there are normally some public test cases and some (hidden) private test cases.
- "Compile and run" will evaluate your submission against the public test cases.
- "Submit" will evaluate your submission against the hidden private test cases. There are 10 private test cases, with equal weightage. You will get feedback about which private test cases pass or fail, though you cannot see the actual test cases.
- Ignore warnings about "Presentation errors".
- 1. Write a Python function frequency(I) that takes as input a list of integers and returns a pair of the form (minfreqlist,maxfreqlist) where
 - minfreqlist is a list of numbers with minimum frequency in I, sorted in ascending order

definitions, sorting

Week 3 **Programming Assignment**

Week 4: Sorting, Tuples, Dictionaries, **Passing Functions, List** Comprehension maxfreglist is a list of numbers with maximum frequency in I, sorted in ascending

For instance

```
>>> frequency([13,12,11,13,14,13,7,11,13,14,12])
([7], [13])
>>> frequency([13,12,11,13,14,13,7,11,13,14,12,14,14])
([7], [13, 14])
>>> frequency([13,12,11,13,14,13,7,11,13,14,12,14,14,7])
([7, 11, 12], [13, 14])
```

Week 4 Quiz

Week 4 **Programming Assignment**

Week 4 **Programming Assignment** name=93)

Week 5: **Exception** handling, input/output, file handling, string processing

Week 5 **Programming Assignment**

Week 5 **Programming** Assignment name=95)

Week 6: Backtracking, scope, data structures: stacks. queues and heaps

2. An airline has assigned each city that it serves a unique numeric code. It has collected information about all the direct flights it operates, represented as a list of pairs of the form (i,i), where i is the code of the starting city and i is the code of the destination.

It now wants to compute all pairs of cities connected by one intermediate hop --- city i is connected to city j by one intermediate hop if there are direct flights of the form (i,k) and (k,j) for some other city k. The airline is only interested in one hop flights (/noc19_cs40/progassignmebetween different cities --- pairs of the form (i,i) are not useful.

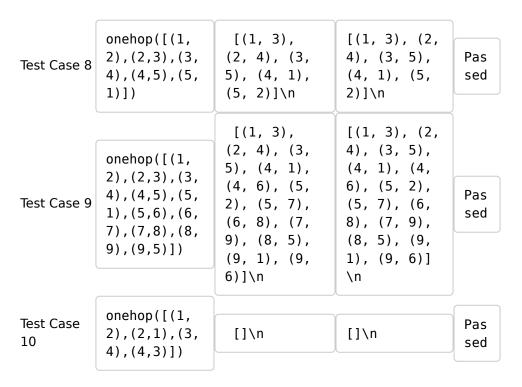
> Write a Python function onehop(I) that takes as input a list of pairs representing direct flights, as described above, and returns a list of all pairs (i,j), where i != j, such that i and j are connected by one hop. Note that it may already be the case that there is a direct flight from i to j. So long as there is an intermediate k with a flight from i to k and from k to j, the list returned by the function should include (i,j). The input list may be in any order. The pairs in the output list should be in lexicographic (dictionary) order. Each pair should be listed exactly once.

For instance

```
>>> onehop([(2,3),(1,2)])
                        [(1, 3)]
                        >>> onehop([(2,3),(1,2),(3,1),(1,3),(3,2),(2,4),(4,1)])
(\frac{1}{2}\cos 40) prograssignment? [(1, 2), (1, 3), (1, 4), (2, 1), (3, 2), (3, 4), (4, 2), (4, 3)]
                        >>> onehop([(1,2),(3,4),(5,6)])
                        []
```

Week 6 Quiz Week 7: Classes, objects and user defined datatypes Week 7 Quiz	Private Test cases Input used for evaluation		Expected Output Actual Output		Status
	Test Case 1	frequency([1 7322,271898, 374,374,374, 423432423,42 3432423,4234 32423,423432 423,5325,532 5,5325,5325, 5325])		Υ	
			([17322, 27 1898], [532 5])\n	([17322, 271 898], [532 5])\n	Pas sed
Week 8: Dynamic programming, wrap-up					
Week 8 Programming Assignment	Test Case 2	frequency([1 7322,374,173 22,374,1732 2,374])	([374, 1732 2], [374, 17 322])\n	([374, 1732 2], [374, 17 322])\n	Pas sed
 Week 8 Programming Assignment (/noc19_cs40/progate 	Test Case 3	frequency([9 842])	([9842], [9 842])\n	([9842], [98 42])\n	Pas sed
name=98) Download		frequency([- 17322,-27189 8,-374,-374, -374,-423432 423,-42343242 3,-42343242 3,-5325,-532 5,-5325,-532 5,-5325])			
Text Transcripts	Test Case 4		([-271898, -17322], [-5 325])\n	([-271898, - 17322], [-53 25])\n	Pas sed
Online Programming Test - Sample					
Online Programming Test 1, 26 Sep 2019, 09:30- 11:30	Test Case 5	frequency([- 17322,-374,- 17322,-374,- 17322,-374])	([-17322, - 374], [-1732 2, -374])\n	([-17322, -3 74], [-1732 2, -374])\n	Pas sed
	Test Case 6	onehop([(1, 3),(1,2),(2, 3),(2,1),(3, 2),(3,1)])	[(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)]\n	[(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)] \n	Pas sed
	Test Case 7	onehop([(1, 2), (1, 3), (1, 4), (1, 5), (1, 6),	[(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1,	[(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7),	Pas sed

```
(1, 8), (1,
(1, 7), (1,
                 7), (1, 8),
8), (1, 9),
                 (1, 9), (2,
                                 9), (2, 1),
                 1), (2, 3),
(2, 1), (2,
                                  (2, 3), (2,
3), (2, 4),
                 (2, 4), (2,
                                 4), (2, 5),
(2, 5), (2,
                 5), (2, 6),
                                  (2, 6), (2,
6), (2, 7),
                 (2, 7), (2,
                                 7), (2, 8),
(2, 8), (2,
                 8), (2, 9),
                                  (2, 9), (3,
9), (3, 1),
                 (3, 1), (3,
                                 1), (3, 2),
(3, 2), (3,
                 2), (3, 4),
                                  (3, 4), (3,
4), (3, 5),
                 (3, 5), (3,
                                 5), (3, 6),
(3, 6), (3,
                 6), (3, 7),
                                  (3, 7), (3,
7), (3, 8),
                 (3, 8), (3,
                                 8), (3, 9),
(3, 9), (4,
                 9), (4, 1),
                                  (4, 1), (4,
                                 2), (4, 3),
1), (4, 2),
                 (4, 2), (4,
(4, 3), (4,
                 3), (4, 5),
                                  (4, 5), (4,
                 (4, 6), (4,
                                 6), (4, 7),
5), (4, 6),
(4, 7),
                 7), (4, 8),
                                  (4, 8), (4,
        (4,
8), (4, 9),
                 (4, 9), (5,
                                 9), (5, 1),
(5, 1), (5,
                 1), (5, 2),
                                  (5, 2), (5,
2), (5, 3),
                 (5, 3), (5,
                                 3), (5, 4),
(5, 4), (5,
                 4), (5, 6),
                                  (5, 6), (5,
6), (5, 7),
                 (5, 7), (5,
                                 7), (5, 8),
(5, 8), (5,
                 8), (5, 9),
                                  (5, 9), (6,
9), (6, 1),
                 (6, 1), (6,
                                 1), (6, 2),
(6, 2), (6,
                 2), (6, 3),
                                  (6, 3), (6,
3), (6, 4),
                 (6, 4), (6,
                                 4), (6, 5),
(6, 5), (6,
                 5), (6, 7),
                                  (6, 7), (6,
                                 8), (6, 9),
7), (6, 8),
                 (6, 8), (6,
(6, 9), (7,
                9), (7, 1),
                                  (7, 1), (7,
1), (7, 2),
                 (7, 2), (7,
                                 2), (7, 3),
(7, 3), (7,
                 3), (7, 4),
                                  (7, 4), (7,
4), (7, 5),
                 (7, 5), (7,
                                 5), (7, 6),
(7, 6), (7,
                 6), (7, 8),
                                  (7, 8), (7,
                 (7, 9), (8,
                                 9), (8, 1),
8), (7, 9),
                                  (8, 2), (8,
(8, 1),
                 1), (8, 2),
        (8,
2), (8, 3),
                 (8, 3), (8,
                                 3), (8, 4),
(8, 4), (8,
                4), (8, 5),
                                  (8, 5), (8,
5), (8, 6),
                 (8, 6), (8,
                                 6), (8, 7),
                                  (8, 9), (9,
(8, 7), (8,
                 7), (8, 9),
9), (9, 1),
                 (9, 1), (9,
                                 1), (9, 2),
(9, 2), (9,
                 2), (9, 3),
                                  (9, 3), (9,
3), (9, 4),
                 (9, 4), (9,
                                 4), (9, 5),
(9, 5), (9,
                 5), (9, 6),
                                  (9, 6), (9,
                 (9, 7), (9,
6), (9, 7),
                                 7), (9, 8)]
(9, 8)])
                 8)]\n
                                 \n
```



Due Date Exceeded. 10 out of 10 tests passed. You scored 100.0/100.

Your last recorded submission was :

```
def frequency(l):
    new_l = list(set(l))
    freq_list = [l.count(x) for x in new_l]
 12345678
           min_freq_list = [new_l[x] for x in range(len(freq_list)) if freq_list
max_freq_list = [new_l[x] for x in range(len(freq_list)) if freq_list
min_freq_list.sort()
max_freq_list.sort()
            return (min freq list, max freq list)
 9
10
    def onehop(l):
11
12
13
14
15
16
17
18
19
20
21
22
24
25
26
27
28
29
30
            data = l
           data.sort(key=lambda tup: tup[0])
all_pairs = []
            for e in l:
                  xx, yy = e1
                                if y == xx and x != yy and (x,yy) not in all_pairs:
    all_pairs.append((x, yy))
            all pairs = sorted(\overline{all pairs}, key=lambda tup: (tup[0], tup[1]))
            return all pairs
     import ast
    def parse(inp):
   inp = ast.literal_eval(inp)
   return (inp)
    fncall = input()
lparen = fncall.find("(")
rparen = fncall.rfind(")")
31
32
33
34
    fname = fncall[:lparen]
farg = fncall[lparen+1:rparen]
35
    if fname == "frequency":
        arg = parse(farg)
```

```
37
38
39
40
41
42
43
44
45
print(frequency(arg))
if fname == "onehop":
    arg = parse(farg)
    print(onehop(arg))
```