Chapter 2

Basic Data Types

Chapter 2
Basic Data Types

In This Chapter:

1. Lists
2. Tuples
3. List Comprehensions
4. Find the Second Largest Number
5. Nested Lists
6. Finding the percentage

2.1 Lists

Consider a list (list = []). You can perform the following commands:

1. insert i e: Insert integer at position .
2. print: Print the list.
3. remove e: Delete the first occurrence of integer .
4. append e: Insert integer at the end of the list.
5. sort: Sort the list.
6. pop: Pop the last element from the list.
7. reverse: Reverse the list.

Initialize your list and read in the value of n followed by n lines of commands where each command will be of the 7 types listed above. Iterate through each command in order and perform the corresponding operation on your list.

Input Format

The first line contains an integer,n , denoting the number of commands.

Each line of subsequent lines contains one of the above commands.

Constraints

- The elements added to the list must be *integers*.

Output Format

For each command of type print, print the list on a new line.

Sample Input

```
12
insert 0 5
insert 1 10
insert 0 6
print
remove 6
append 9
append 1
sort
print
pop
reverse
print
```

Sample Output

```
[6, 5, 10]
[1, 5, 9, 10]
[9, 5, 1]
```

Code

```
if __name__ == '__main__':
    N = int(raw_input())
    L = []

    for i in range(0, N):
        tokens = raw_input().split()

        if tokens[0] == 'insert':
            L.insert(int(tokens[1]), int(tokens[2]))
        elif tokens[0] == 'print':
            print L
        elif tokens[0] == 'remove':
            L.remove(int(tokens[1]))
        elif tokens[0] == 'append':
            L.append(int(tokens[1]))
        elif tokens[0] == 'sort':
            L.sort()
        elif tokens[0] == 'pop':
            L.pop()
        elif tokens[0] == 'reverse':
            L.reverse()
```

2.2 Tuples

Given an integer,n , and n space-separated integers as input, create a tuple,t , of those integers. Then compute and print the result of hash(t) .

Note: hash() is one of the functions in the __builtins__ module, so it need not be imported.

Input Format

The first line contains an integer, n, denoting the number of elements in the tuple.

The second line contains n space-separated integers describing the elements in tuple t.

Output Format

Print the result of hash(t).

Sample Input

```
2
1 2
```

Sample Output

```
3713081631934410656
```

Code

```
if __name__ == '__main__':
    n = int(raw_input())
    integer_list = map(int, raw_input().split())
    t= tuple(integer_list)
    #print(t)
    print(hash(t))
```

2.3 List Comprehensions

You are given three integers X, Y and Z representing the dimensions of a cuboid along with an integer N. You have to print a list of all possible coordinates given by (i,j,k) on a 3D grid where the sum of i+j+k is not equal to N. Here,

$$0 \leq i \leq X; 0 \leq j \leq Y; 0 \leq k \leq Z$$

Input Format

Integers X, Y, Z and N each on four separate lines, respectively.

Constraints

Print the list in lexicographic increasing order.

Sample Input

```
1
1
1
2
```

Sample Output

```
[[0, 0, 0], [0, 0, 1], [0, 1, 0], [1, 0, 0], [1, 1, 1]]
```

Concept

You have already used lists in previous hacks. List comprehensions are an elegant way to build a list without having to use different *for* loops to append values one by one.
The simplest form of a list comprehension is:

[expression-involving-loop-variable
for loop-variable in sequence]

This will step over every element in a sequence, successively setting the loop-variable equal to every element one at a time. It will then build up a list by evaluating the expression-involving-loop-variable for each one. This eliminates the need to use lambda forms and generally produces a much more readable code than using *map()* and a more compact code than using a *for* loop.

```
>> ListOfNumbers = [ x for x in range(10) ] # List of integers from 0 to 9
>> ListOfNumbers
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

List comprehensions can be nested where they take the following form:

[expression-involving-loop-variables
for outer-loop-variable in outer-sequence
for inner-loop-variable in inner-sequence]

This is equivalent to writing:

```
results = []
for outer_loop_variable in outer_sequence:
    for inner_loop_variable in inner_sequence:
        results.append( expression_involving_loop_variables )
```

The final form of list comprehension involves creating a list and filtering it similar to using the *filter()* method. The filtering form of list comprehension takes the following form:

[expression-involving-loop-variable
for loop-variable in sequence
if boolean-expression-involving-loop-variable]

This form is similar to the simple form of list comprehension, but it evaluates boolean-expression-involving-loop-variable for every item. It also only keeps those members for which the boolean expression is *True*.

```
>> ListOfThreeMultiples = [x for x in range(10) if x % 3 == 0]
# Multiples of 3 below 10
>> ListOfThreeMultiples
[0, 3, 6, 9]
```

Code

```
if __name__ == '__main__':
    X = int(raw_input())
    Y = int(raw_input())
    Z = int(raw_input())
    N = int(raw_input())

    print [[x, y, z]
           for x in range(X + 1)
           for y in range(Y + 1)
           for z in range(Z + 1)
           if x + y + z != N]
```

2.4 Find the Second Largest Number

You are given N numbers. Store them in a list and find the second largest number.

Input Format

The first line contains N. The second line contains an array A[] of N integers each separated by a space.

Constraints

$2 \leq N \leq 10$

$-100 \leq A[i] \leq 100$

Output Format

Output the value of the second largest number.

Sample Input

```
5
2 3 6 6 5
```

Sample Output

```
5
```

Code

```
if __name__ == '__main__':
    n = int(raw_input())
    arr = map(int, raw_input().split())

    print(sorted(set(arr))[-2])
```

2.5 Nested Lists

Given the names and grades for each student in a Physics class of N students, store them in a nested list and print the name(s) of any student(s) having the second lowest grade.

Note: If there are multiple students with the same grade, order their names alphabetically and print each name on a new line.

Input Format

The first line contains an integer, N, the number of students.

The 2N subsequent lines describe each student over 2 lines; the first line contains a student's name, and the second line contains their grade.

Constraints

$2 \leq N \leq 5$

There will always be one or more students having the second lowest grade.

Output Format

Print the name(s) of any student(s) having the second lowest grade in Physics; if there are multiple students, order their names alphabetically and print each one on a new line.

Sample Input

```
5
Harry
37.21
Berry
37.21
Tina
37.2
Akriti
41
Harsh
39
```

Sample Output

```
Berry
Harry
```

Explanation

There are 5 students in this class whose names and grades are assembled to build the following list:

```
students = [['Harry', 37.21], ['Berry', 37.21], ['Tina', 37.2], ['Akriti', 41], ['Harsh', 39]]
```

The lowest grade of 37.2 belongs to *Tina*. The second lowest grade of 37.21 belongs to both *Harry* and *Berry*, so we order their names alphabetically and print each name on a new line.

Code

```
students = list()
N = int(raw_input())

for _ in range(N):
    name = raw_input()
    score = float(raw_input())
    students.append([name, score])

scores = set([students[x][1] for x in range(N)])
scores = list(scores)
scores.sort()

students = [x[0] for x in students if x[1] == scores[1]]
students.sort()

for s in students:
    print s
```

2.6 Finding the percentage

You have a record of N students. Each record contains the student's name, and their percent marks in Maths, Physics and Chemistry. The marks can be floating values. The user enters some integer N followed by the names and marks for N students. You are required to save the record in a dictionary data type. The user then enters a student's name. Output the average percentage marks obtained by that student, correct to two decimal places.

Input Format

The first line contains the integer N, the number of students. The next N lines contains the name and marks obtained by that student separated by a space. The final line contains the name of a particular student previously listed.

Constraints

$$2 \leq N \leq 10$$

$$0 \leq Marks \leq 100$$

Output Format

Print one line: The average of the marks obtained by the particular student correct to 2 decimal places.

Sample Input

```
3
Krishna 67 68 69
Arjun 70 98 63
Malika 52 56 60
Malika
```

Sample Output

```
56.00
```

Explanation

Marks for Malika are {52, 56, 60} whose average is $\frac{52+56+60}{3} \Rightarrow 56$

Code

```
# Enter your code here. Read input from STDIN. Print

N=int(raw_input())
d={}
for i in range(0,N):
    iin=raw_input().split(" ")
    name=iin[0]

    sum=float(iin[1])+float(iin[2])+float(iin[3])
    d[name]=sum*100/300

name=raw_input()

print '%.2f' %d[name]
```