sample problem1 : Hello world

Here is a sample line of code that can be executed in Python:

print("Hello, World!")

You can just as easily store a string as a variable and then print it to stdout:

my\_string = "Hello, World!"

print(my\_string)

The above code will print Hello, World! on your screen. Try it yourself in the editor below!

**Input Format**

You do not need to read any input in this challenge.

**Output Format**

Print Hello, World! to stdout.

**Sample Output 0**

Hello, World!

solution:

if \_\_name\_\_ == '\_\_main\_\_':

var = "Hello, World!"

print(var)

Check [Tutorial](https://www.hackerrank.com/challenges/py-if-else/tutorial) tab to know how to to solve.

**Task**   
Given an integer,

, perform the following conditional actions:

* If

 is odd, print Weird

 If is even and in the inclusive range of to

 , print Not Weird

 If is even and in the inclusive range of to

 , print Weird

 If is even and greater than

* , print Not Weird

**Input Format**

A single line containing a positive integer,

.

**Constraints**

**Output Format**

Print Weird if the number is weird; otherwise, print Not Weird.

**Sample Input 0**

3

**Sample Output 0**

Weird

**Explanation 0**

is odd and odd numbers are weird, so we print Weird.

**Sample Input 1**

24

**Sample Output 1**

Not Weird

**Explanation 1**

and is even, so it isn't weird. Thus, we print Not Weird.

#!/bin/python

# Enter your code here. Read input from STDIN. Print output to STDOUT

N = int(raw\_input())

if N%2 != 0 :

print('Weird')

else :

if 2 <= N <= 5:

print('Not Weird')

if 6 <= N <= 20:

print('Weird')

if N > 20 :

print('Not Weird')

**Task**   
Read two integers from STDIN and print three lines where:

1. The first line contains the sum of the two numbers.
2. The second line contains the difference of the two numbers (first - second).
3. The third line contains the product of the two numbers.

**Input Format**

The first line contains the first integer,

. The second line contains the second integer,

.

**Constraints**

**Output Format**

Print the three lines as explained above.

**Sample Input 0**

3

2

**Sample Output 0**

5

1

6

**Explanation 0**

if \_\_name\_\_ == '\_\_main\_\_':

a = int(raw\_input())

b = int(raw\_input())

a1 = a+b

s1 = a - b

m1 = a \* b

print(a1)

print(s1)

print(m1)

**Task**   
Read two integers and print two lines. The first line should contain integer division, // . The second line should contain float division, /

.

You don't need to perform any rounding or formatting operations.

**Input Format**

The first line contains the first integer,

. The second line contains the second integer,

.

**Output Format**

Print the two lines as described above.

**Sample Input 0**

4

3

**Sample Output 0**

1

1.33333333333

from \_\_future\_\_ import division

if \_\_name\_\_ == '\_\_main\_\_':

a = int(raw\_input())

b = int(raw\_input())

print(a//b)

print(a/b)

**Task**   
Read an integer . For all non-negative integers , print

. See the sample for details.

**Input Format**

The first and only line contains the integer,

.

**Constraints**

**Output Format**

Print

lines, one corresponding to each

.

**Sample Input 0**

5

**Sample Output 0**

0

1

4

9

16

if \_\_name\_\_ == '\_\_main\_\_':

n = int(raw\_input())

i = 0

while n > 0 :

print(i\*\*2)

i+=1

n-=1

We add a Leap Day on February 29, almost every four years. The leap day is an extra, or intercalary day and we add it to the shortest month of the year, February.   
In the Gregorian calendar three criteria must be taken into account to identify leap years:

* The year can be evenly divided by 4, is a leap year, unless:
  + The year can be evenly divided by 100, it is NOT a leap year, unless:
    - The year is also evenly divisible by 400. Then it is a leap year.

This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years.[Source](http://www.timeanddate.com/date/leapyear.html)

**Task**   
You are given the year, and you have to write a function to check if the year is leap or not.

Note that you have to complete the function and remaining code is given as template.

**Input Format**

Read y, the year that needs to be checked.

**Constraints**

**Output Format**

Output is taken care of by the template. Your function must return a boolean value (True/False)

**Sample Input 0**

1990

**Sample Output 0**

False

**Explanation 0**

1990 is not a multiple of 4 hence it's not a leap year.

def is\_leap(year):

leap = False

if year%4 == 0 :

if year%100 == 0 :

if year % 400 == 0 :

leap = True

else :

leap = True

# Write your logic here

return leap

year = int(raw\_input())

print is\_leap(year)

Read an integer

.

Without using any string methods, try to print the following:

Note that "

" represents the values in between.

**Input Format**

The first line contains an integer

.

**Output Format**

Output the answer as explained in the task.

**Sample Input 0**

3

**Sample Output 0**

123

from \_\_future\_\_ import print\_function

if \_\_name\_\_ == '\_\_main\_\_':

n = int(raw\_input())

i = 1

j = ''

while i <= n :

j+=str(i)

i+= 1

print(j)

You are given a string and your task is to swap cases. In other words, convert all lowercase letters to uppercase letters and vice versa.

**For Example:**

Www.HackerRank.com → wWW.hACKERrANK.COM

Pythonist 2 → pYTHONIST 2

**Input Format**

A single line containing a string

.

**Constraints**

**Output Format**

Print the modified string

.

**Sample Input 0**

HackerRank.com presents "Pythonist 2".

**Sample Output 0**

hACKERrANK.COM PRESENTS "pYTHONIST 2".

def swap\_case(s):

s = s.swapcase()

return s

if \_\_name\_\_ == '\_\_main\_\_':

s = raw\_input()

result = swap\_case(s)

print result

n Python, a string can be split on a delimiter.

**Example:**

>>> a = "this is a string"

>>> a = a.split(" ") # a is converted to a list of strings.

>>> print a

['this', 'is', 'a', 'string']

Joining a string is simple:

>>> a = "-".join(a)

>>> print a

this-is-a-string

**Task**   
You are given a string. Split the string on a " " (space) delimiter and join using a - hyphen.

**Input Format**   
The first line contains a string consisting of space separated words.

**Output Format**   
Print the formatted string as explained above.

**Sample Input**

this is a string

**Sample Output**

this-is-a-string

def split\_and\_join(line):

s = line.split(" ")

s = '-'.join(s)

return s

if \_\_name\_\_ == '\_\_main\_\_':

line = raw\_input()

result = split\_and\_join(line)

print result

You are given the firstname and lastname of a person on two different lines. Your task is to read them and print the following:

Hello firstname lastname! You just delved into python.

**Input Format**

The first line contains the first name, and the second line contains the last name.

**Constraints**

The length of the first and last name ≤

.

**Output Format**

Print the output as mentioned above.

**Sample Input 0**

Ross

Taylor

**Sample Output 0**

Hello Ross Taylor! You just delved into python.

**Explanation 0**

The input read by the program is stored as a string data type. A string is a collection of characters.

def print\_full\_name(a, b):

print "Hello "+str(a)+" "+str(b)+"! You just delved into python."

if \_\_name\_\_ == '\_\_main\_\_':

first\_name = raw\_input()

last\_name = raw\_input()

print\_full\_name(first\_name, last\_name)

We have seen that lists are mutable (they can be changed), and tuples are immutable (they cannot be changed).

Let's try to understand this with an example.

You are given an immutable string, and you want to make changes to it.

**Example**

>>> string = "abracadabra"

You can access an index by:

>>> print string[5]

a

What if you would like to assign a value?

>>> string[5] = 'k'

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: 'str' object does not support item assignment

How would you approach this?

* One solution is to convert the string to a list and then change the value.

**Example**

>>> string = "abracadabra"

>>> l = list(string)

>>> l[5] = 'k'

>>> string = ''.join(l)

>>> print string

abrackdabra

* Another approach is to slice the string and join it back.

Example

>>> string = string[:5] + "k" + string[6:]

>>> print string

abrackdabra

**Task**   
Read a given string, change the character at a given index and then print the modified string.

**Input Format**   
The first line contains a string,

.   
The next line contains an integer , denoting the index location and a character

separated by a space.

**Output Format**   
Using any of the methods explained above, replace the character at index

with character

.

**Sample Input**

abracadabra

5 k

**Sample Output**

abrackdabra

def mutate\_string(string, position, character):

return string[:position]+character+string[position+1:]

if \_\_name\_\_ == '\_\_main\_\_':

s = raw\_input()

i, c = raw\_input().split()

s\_new = mutate\_string(s, int(i), c)

print s\_new

In this challenge, the user enters a string and a substring. You have to print the number of times that the substring occurs in the given string. String traversal will take place from left to right, not from right to left.

**NOTE:** String letters are case-sensitive.

**Input Format**

The first line of input contains the original string. The next line contains the substring.

**Constraints**

Each character in the string is an *ascii* character.

**Output Format**

Output the integer number indicating the total number of occurrences of the substring in the original string.

**Sample Input**

ABCDCDC

CDC

**Sample Output**

2

**Concept**

Some string processing examples, [such as these](http://www.thelearningpoint.net/computer-science/learning-python-programming-and-data-structures/learning-python-programming-and-data-structures--tutorial-12--string-manipulation), might be useful.   
There are a couple of new concepts:   
In Python, the length of a string is found by the function len(s), where

is the string.   
To traverse through the length of a string, use a *for* loop:

for i in range(0, len(s)):

print (s[i])

A range function is used to loop over some length:

range (0, 5)

Here, the range loops over

to . is excluded.

def count\_substring(string, sub\_string):

count1 = 0

for i in range(0, len(string)):

if string.find(sub\_string,i,i+len(sub\_string))!= -1:

count1+=1

return count1

if \_\_name\_\_ == '\_\_main\_\_':

string = raw\_input().strip()

sub\_string = raw\_input().strip()

count = count\_substring(string, sub\_string)

print count

Python has built-in string validation methods for basic data. It can check if a string is composed of alphabetical characters, alphanumeric characters, digits, etc.

[**str.isalnum()**](https://docs.python.org/2/library/stdtypes.html#str.isalnum)   
This method checks if all the characters of a string are alphanumeric *(a-z, A-Z and 0-9)*.

>>> print 'ab123'.isalnum()

True

>>> print 'ab123#'.isalnum()

False

[**str.isalpha()**](https://docs.python.org/2/library/stdtypes.html#str.isalpha)   
This method checks if all the characters of a string are alphabetical *(a-z and A-Z)*.

>>> print 'abcD'.isalpha()

True

>>> print 'abcd1'.isalpha()

False

[**str.isdigit()**](https://docs.python.org/2/library/stdtypes.html#str.isdigit)   
This method checks if all the characters of a string are digits *(0-9)*.

>>> print '1234'.isdigit()

True

>>> print '123edsd'.isdigit()

False

[**str.islower()**](https://docs.python.org/2/library/stdtypes.html#str.islower)   
This method checks if all the characters of a string are lowercase characters *(a-z)*.

>>> print 'abcd123#'.islower()

True

>>> print 'Abcd123#'.islower()

False

[**str.isupper()**](https://docs.python.org/2/library/stdtypes.html#str.isupper)   
This method checks if all the characters of a string are uppercase characters *(A-Z)*.

>>> print 'ABCD123#'.isupper()

True

>>> print 'Abcd123#'.isupper()

False

**Task**

You are given a string

.   
Your task is to find out if the string

contains: *alphanumeric characters, alphabetical characters, digits, lowercase and uppercase characters*.

**Input Format**

A single line containing a string

.

**Constraints**

**Output Format**

In the first line, print True if

has any *alphanumeric characters*. Otherwise, print False.   
In the second line, print True if has any *alphabetical characters*. Otherwise, print False.   
In the third line, print True if has any *digits*. Otherwise, print False.   
In the fourth line, print True if has any *lowercase characters*. Otherwise, print False.   
In the fifth line, print True if

has any *uppercase characters*. Otherwise, print False.

**Sample Input**

qA2

**Sample Output**

True

True

True

True

True

if \_\_name\_\_ == '\_\_main\_\_':

s = raw\_input()

print(any(s[i].isalnum() for i in range(len(s))))

print(any(s[i].isalpha() for i in range(len(s))))

print(any(s[i].isdigit() for i in range(len(s))))

print(any(s[i].islower() for i in range(len(s))))

print(any(s[i].isupper() for i in range(len(s))))

List comprehension

Let's learn about list comprehensions! You are given three integers and representing the dimensions of a cuboid along with an integer . You have to print a list of all possible coordinates given by on a 3D grid where the sum of is not equal to . Here,

**Input Format**

Four integers

and

each on four separate lines, respectively.

**Constraints**

Print the list in lexicographic increasing order.

**Sample Input 0**

1

1

1

2

**Sample Output 0**

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

**Explanation 0**

***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. This example might help.

***Example:*** You are given two integers x and y . You need to find out the ordered pairs ( i , j ) , such that ( i + j ) is not equal to n and print them in lexicographic order.( 0 <= i <= x ) and ( 0 <= j <= y) This is the code if ***we dont use list comprehensions in Python***.

python x = int ( raw\_input()) y = int ( raw\_input()) n = int ( raw\_input()) ar = [] p = 0 for i in range ( x + 1 ) : for j in range( y + 1): if i+j != n: ar.append([]) ar[p] = [ i , j ] p+=1 print ar   
Other smaller codes may also exist, but using list comprehensions is always a good option. ***Code using list comprehensions:***

python x = int ( raw\_input()) y = int ( raw\_input()) n = int ( raw\_input()) print [ [ i, j] for i in range( x + 1) for j in range( y + 1) if ( ( i + j ) != n )]

**Sample Input 1**

2

2

2

2

**Sample Output 1**

[[0, 0, 0], [0, 0, 1], [0, 1, 0], [0, 1, 2], [0, 2, 1], [0, 2, 2], [1, 0, 0], [1, 0, 2], [1, 1, 1], [1, 1, 2], [1, 2, 0], [1, 2, 1], [1, 2, 2], [2, 0, 1], [2, 0, 2], [2, 1, 0], [2, 1, 1], [2, 1, 2], [2, 2, 0], [2, 2, 1], [2, 2, 2]]

if \_\_name\_\_ == '\_\_main\_\_':

x = int(raw\_input())

y = int(raw\_input())

z = int(raw\_input())

n = int(raw\_input())

print(list([i,j,k] for i in range(0,x+1) for j in range(0,y+1) for k in range(0,z+1) if i+j+k != n))

Given the participants' score sheet for your University Sports Day, you are required to find the runner-up score. You are given

scores. Store them in a list and find the score of the runner-up.

**Input Format**

The first line contains

. The second line contains an array of

integers each separated by a space.

**Constraints**

**Output Format**

Print the runner-up score.

**Sample Input 0**

5

2 3 6 6 5

**Sample Output 0**

5

**Explanation 0**

Given list is

. The maximum score is , second maximum is . Hence, we print as the runner-up score.

if \_\_name\_\_ == '\_\_main\_\_':

n = int(raw\_input())

arr = map(int, raw\_input().split())

max1 = max(arr)

for i in range(0,len(arr)):

if max1 in arr:

arr.pop(arr.index(max1))

print(max(arr))

nested list

Given the names and grades for each student in a Physics class of

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,

, the number of students.   
The subsequent lines describe each student over

lines; the first line contains a student's name, and the second line contains their grade.

**Constraints**

* 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 0**

5

Harry

37.21

Berry

37.21

Tina

37.2

Akriti

41

Harsh

39

**Sample Output 0**

Berry

Harry

**Explanation 0**

There are

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

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

The lowest grade of

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

records = []

for \_ in range(int(raw\_input())):

name = raw\_input()

score = float(raw\_input())

records.append([name,score])

#creating a set of score values in the sorted order

mk\_lst = sorted(set(x[1] for x in records))

#check for the score value and pring the names in the sorted order

for name in sorted(x[0] for x in records if x[1] == mk\_lst[1]):

print(name)

average values :

You have a record of 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 followed by the names and marks for

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

, the number of students. The next

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**

**Output Format**

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

**Sample Input 0**

3

Krishna 67 68 69

Arjun 70 98 63

Malika 52 56 60

Malika

**Sample Output 0**

56.00

**Explanation 0**

Marks for Malika are

whose average is

**Sample Input 1**

2

Harsh 25 26.5 28

Anurag 26 28 30

Harsh

**Sample Output 1**

26.50

if \_\_name\_\_ == '\_\_main\_\_':

n = int(raw\_input())

student\_marks = {}

for \_ in range(n):

line = raw\_input().split()

name, scores = line[0], line[1:]

scores = map(float, scores)

student\_marks[name] = scores

query\_name = raw\_input()

values = student\_marks[query\_name]

print("{0:.2f}".format(sum(values)/len(values)))

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

1. insert i e: Insert integer

at position

 .

 print: Print the list.

 remove e: Delete the first occurrence of integer

 .

 append e: Insert integer

1. at the end of the list.
2. sort: Sort the list.
3. pop: Pop the last element from the list.
4. reverse: Reverse the list.

Initialize your list and read in the value of

followed by lines of commands where each command will be of the

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,

, denoting the number of commands.   
Each line of the

subsequent lines contains one of the commands described above.

**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 0**

12

insert 0 5

insert 1 10

insert 0 6

print

remove 6

append 9

append 1

sort

print

pop

reverse

print

**Sample Output 0**

[6, 5, 10]

[1, 5, 9, 10]

[9, 5, 1]

if \_\_name\_\_ == '\_\_main\_\_':

N = int(raw\_input())

#intializing the list

records = []

def func(list1,command):

if command == 'insert':

records.insert(list1[0],list1[1])

if command == 'print':

print records

if command =='remove':

records.remove(list1[0])

if command == 'append':

records.append(list1[0])

if command =='sort':

records.sort()

if command == 'pop':

records.pop()

if command == 'reverse':

records.reverse()

return

for i in range(N) :

x = []

x = raw\_input().split()

command = x.pop(0)

x = map(int , x)

func(x,command)

Tuple :

**Task**   
Given an integer, , and space-separated integers as input, create a tuple, , of those integers. Then compute and print the result of

.

**Note:** [hash()](https://docs.python.org/3/library/functions.html#hash) is one of the functions in the \_\_builtins\_\_ module, so it need not be imported.

**Input Format**

The first line contains an integer,

, denoting the number of elements in the tuple.   
The second line contains space-separated integers describing the elements in tuple

.

**Output Format**

Print the result of

.

**Sample Input 0**

2

1 2

**Sample Output 0**

3713081631934410656

if \_\_name\_\_ == '\_\_main\_\_':

n = int(raw\_input())

integer\_list = map(int, raw\_input().split())

print hash(tuple(integer\_list))

In Python, a string of text can be aligned *left, right* and *center*.

**.ljust(width)**

This method returns a left aligned string of length *width*.

>>> width = 20

>>> print 'HackerRank'.ljust(width,'-')

HackerRank----------

**.center(width)**

This method returns a centered string of length *width*.

>>> width = 20

>>> print 'HackerRank'.center(width,'-')

-----HackerRank-----

**.rjust(width)**

This method returns a right aligned string of length *width*.

>>> width = 20

>>> print 'HackerRank'.rjust(width,'-')

----------HackerRank

**Task**

You are given a partial code that is used for generating the *HackerRank Logo* of variable *thickness*.   
Your task is to replace the blank (\_\_\_\_\_\_) with *rjust, ljust* or *center*.

**Input Format**

A single line containing the *thickness* value for the logo.

**Constraints**

The *thickness* must be an *odd* number.

**Output Format**

Output the desired logo.

**Sample Input**

5

**Sample Output**

H

HHH

HHHHH

HHHHHHH

HHHHHHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHH

HHHHHHHHHHHHHHHHHHHHHHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHH HHHHH

HHHHHHHHH

HHHHHHH

HHHHH

HHH

H

#Replace all \_\_\_\_\_\_ with rjust, ljust or center.

thickness = int(raw\_input()) #This must be an odd number

c = 'H'

#Top Cone

for i in range(thickness):

print (c\*i).rjust(thickness-1)+c+(c\*i).ljust(thickness-1)

#Top Pillars

for i in range(thickness+1):

print (c\*thickness).center(thickness\*2)+(c\*thickness).center(thickness\*6)

#Middle Belt

for i in range((thickness+1)/2):

print (c\*thickness\*5).center(thickness\*6)

#Bottom Pillars

for i in range(thickness+1):

print (c\*thickness).center(thickness\*2)+(c\*thickness).center(thickness\*6)

#Bottom Cone

for i in range(thickness):

print ((c\*(thickness-i-1)).rjust(thickness)+c+(c\*(thickness-i-1)).ljust(thickness)).rjust(thickness\*6)