Python Hackerrrank submissions

Say "Hello, World!" With Python
Code:
```
if __name__ == '__main__':
    print("Hello, World!")
```

Python If-Else
Code:
```
#!/bin/python3

import math
import os
import random
import re
import sys



if __name__ == '__main__':
    n = int(input())
    if(n%2 != 0):
        print("Weird")
    else:
        if(n>=2 and n<=5):
            print("Not Weird")
        elif(n>=6 and n<=20):
            print("Weird")
        else:
            print("Not Weird")
```

Arithmetic Operators
Code:
```
f __name__ == '__main__':
    a = int(input())
    b = int(input())

    print(a+b)
    print(a-b)
    print(a*b)
```

Python: Division
Code:
```
if __name__ == '__main__':
    a = int(input())
    b = int(input())

    print(a//b)
    print(a/b)
```

Loops
Code:
```
if __name__ == '__main__':
    n = int(input())
    for i in range (n):
        print(i**2)
```

Write a function

Code:

```
def is_leap(year):
    leap = False
    if(year%4==0 and year%100!=0):
        leap=True
    elif(year%100==0 and year%400!=0):
        leap=False
    elif(year%400==0):
        leap=True
    else:
        leap=False
    # Write your logic here

    return leap

year = int(input())
print(is_leap(year))
```

Print Function

Code:

```
if __name__ == '__main__':
    n = int(input())
    for i in range (1,n+1):
        print(i,end="")
```

List Comprehensions

Code:

```
if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())

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

Find the Runner-Up Score!

Code:

```
if __name__ == '__main__':
    n = int(input())
    arr = map(int, input().split())
    a = list(arr)
    a = list(set(a))
    m = max(a)
    a.remove(m)
    l = []
    for i in range(len(a)):
        l.append(m-a[i])
    least_dif = min(l)
    for i in range(len(l)):
        if(l[i] == least_dif):
            print (a[i])
```

Nested Lists
Code:

```
if __name__ == '__main__':
    stud_lowest = []
    lowest_score = float("inf")
    stud_second_lowest = []
    second_lowest_score = float("inf")

    for i in range(int(input())):
        name, score = input(), float(input())
        if score < lowest_score:
            stud_second_lowest = list(stud_lowest)
            stud_lowest = [name]
            second_lowest_score = lowest_score
            lowest_score = score
        elif score == lowest_score:
            stud_lowest.append(name)
        elif score < second_lowest_score:
            stud_second_lowest = [name]
            second_lowest_score = score
        elif score == second_lowest_score:
            stud_second_lowest.append(name)
    for name in sorted(stud_second_lowest):
        print(name)
```

Finding the percentage
Code:

```
if __name__ == '__main__':
    n = int(input())
    student_marks = {}
    for _ in range(n):
        name, *line = input().split()
        scores = list(map(float, line))
        student_marks[name] = scores
    query_name = input()
    nm=student_marks[query_name]
    avg=sum(nm)/len(nm)
    print("%.2f" % avg)
```

Tuples
Code:

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

sWAP cASE
Code:

```
def swap_case(s):
    return s.swapcase()

if __name__ == '__main__':
    s = input()
    result = swap_case(s)
    print(result)
```

String Split and Join
Code;

```
ef split_and_join(line):
    return "-".join(line.split())

if __name__ == '__main__':
    line = input()
    result = split_and_join(line)
    print(result)
```

What's Your Name?
Code:

```
# Complete the 'print_full_name' function below.
#
# The function is expected to return a STRING.
# The function accepts following parameters:
#  1. STRING first
#  2. STRING last
#

def print_full_name(first, last):
    print("Hello "+ first+" " + last+ "! You just delved into python.")
```

Mutations
Code:

```
def mutate_string(string, position, character):
    return string[:position]+character+string[position+1:]
```

Find a string
Code:

```
def count_substring(string, sub_string):
    counter=0
    while sub_string in string:
        a=string.find(sub_string)
        string=string[a+1:]
        counter=counter+1
    return counter
```

String Validators
Code:

```
if __name__ == '__main__':
    s = input()
    l=list(s)
    print(any(i.isalnum()for i in l))
    print(any(i.isalpha()for i in l))
    print(any(i.isdigit()for i in l))
    print(any(i.islower()for i in l))
    print(any(i.isupper()for i in l))
```

Text Wrap

```
import textwrap

def wrap(string, max_width):
    res=textwrap.fill(string, max_width)
    return res
```

Designer Door Mat

```
ef create_door_mat(height, width):
    pattern = ".|."
    middle_message = "WELCOME"

    for row_num in range(height // 2):
        num_patterns = 2 * row_num + 1
        row_string = (pattern * num_patterns).center(width, "-")
        print(row_string)

    print(middle_message.center(width, "-"))

    for row_num in range(height // 2 - 1, -1, -1):
        num_patterns = 2 * row_num + 1
        row_string = (pattern * num_patterns).center(width, "-")
        print(row_string)



if __name__ == "__main__":
    height, width = map(int, input().split())
    create_door_mat(height, width)
```

String Formatting

```
def print_formatted(number):
    nbin = format(number,'b')
    size = len(nbin)
    for i in range(1,n+1):
        deci=str(i)
        octa = format(i,'o')
        hexa = format(i,'X')
        bina = format(i,'b')

        print(
            deci.rjust(size),
            str(octa).rjust(size),
            str(hexa).rjust(size),
            str(bina).rjust(size))
```

Alphabet Rangoli

```
import string
l=[]
def print_rangoli(size):
    n=size
    width=4*n-3
    char=string.ascii_lowercase

    for i in char:
        l.append(i)


    for i in range (1,n+1):
        print('-'.join(l[n-1:n-i:-1]+l[n-i:n]).center(width,'-'))

    for i in range (n-1,0,-1):
        print('-'.join(l[n-1:n-i:-1]+l[n-i:n]).center(width,'-'))
```

Capitalize!

```
def solve(s):
    return " ".join([name.capitalize() for name in s.split(" ")])
```

The Minion Game

```
ef minion_game(string):
    stuart_score = 0
    kevin_score = 0
    string_length: int = len(string)

    for index, char in enumerate(string):
        points = string_length - index
        if char in {"A", "E", "I", "O", "U"}:
            kevin_score += points
        else:
            stuart_score += points

    if stuart_score > kevin_score:
        print("Stuart", stuart_score)
    elif kevin_score > stuart_score:
        print("Kevin", kevin_score)
    else:
        print("Draw")
```

Merge the Tools!

```
def merge_the_tools(string, sub_length: int):
    n_substrings = len(string) // sub_length

    for i in range(n_substrings):
        start_index = i * sub_length
        end_index = start_index + sub_length
        substring = string[start_index:end_index]

        unique_chars = []
        seen_chars = set()

        for char in substring:
          if char not in seen_chars:
             unique_chars.append(char)
             seen_chars.add(char)
        print("".join(unique_chars))
```

collections.Counter()
Code:
```
def customers_number():
    list1.append(list(map(int,input().split())))

list1=[]
x=int(input())
shoe_sizes=list(map(int,input().split()))
n=int(input())

for i in range(n):
    customers_number()
benefits=0

d=Counter(shoe_sizes)
for i,j in list1:
    if i in d.keys() and d[i]>0:
        benefits= benefits+j
        d[i]=d[i]-1

print(benefits)
```

Introduction to Sets
```
def average(array):
    # your code goes here
    arr=set(array)
    return sum(arr)/len(arr)
```

DefaultDict Tutorial
```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import defaultdict


n,m = list(map(int,input().split()))

group_a = defaultdict(list)

for i in range(n):
    word_a= input()
    group_a[word_a].append(str(i + 1))

for i in range(m):
    word_b = input()
    if word_b in group_a:
        print(' '.join(group_a[word_b]))
    else:
        print(-1)
```

Calendar Module
```
import calendar
import datetime
```

```
month, day, year = map(int, input().split())

date = datetime.date(year, month, day)

day_of_week_index = date.weekday()

weekday_name = calendar.day_name[day_of_week_index].upper()

print(weekday_name)
```

Exceptions

```
T = int(input())
for i in range(T):
    try:
        a, b = map(int, input().split())
        print(a//b)
    except Exception as e:
        print("Error Code:",e)
```

Collections.namedtuple()

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import namedtuple
n=int(input())

data=namedtuple("data",input())
marks_list=[]

for i in range (n):
    marks=int(data(*input().split()).MARKS)
    marks_list.append(marks)

print(sum(marks_list)/n)
```

Time Delta

```
#!/bin/python3
import math
import os
import random
import re
import sys
from datetime import datetime


def time_delta(t_str_1, t_str_2):
    time_format = "%a %d %b %Y %H:%M:%S %z"
    time1 = datetime.strptime(t_str_1, time_format)
    time2 = datetime.strptime(t_str_2, time_format)
    difference_seconds = abs((time1 - time2).total_seconds())
    return str(int(difference_seconds))

if __name__ == '__main__':
    fptr = open(os.environ['OUTPUT_PATH'], 'w')

    t = int(input())
    for t_itr in range(t):
        t1 = input()
        t2 = input()
        delta = time_delta(t1, t2)
        fptr.write(delta + '\n')

    fptr.close()
```

The Captain's Room

```
n = int(input())
rooms = list(map(int, input().split()))

total = sum(rooms)

unique_total = sum(set(rooms))

captain_room = (unique_total * n - total) // (n - 1)

print(captain_room)
```

No Idea!
Code:

```
def calculate_happiness(arr, good_set, bad_set):
    """
    This fuction calculates the "happiness score" based on the existence
of a number in two sets good_set and bad_set. If a number in the list is
also in the 'good' set, the score goes up by 1, and if it is in the 'bad'
set the score decreases by -1.

    Args:
        arr: The list of numbers we're checking.
        good_set: A set containing numbers that are considered 'good'.
        bad_set: A set of numbers that are considered 'bad'.

    Returns:
        The calculated happiness score.
    """

    happiness_score = 0
    for element in arr:
        if element in good_set:
            happiness_score += 1
        elif element in bad_set:
            happiness_score -= 1
    return happiness_score

if __name__ == "__main__":

    input()
    my_list = list(map(int, input().split()))
    set_a = set(map(int, input().split()))
    set_b = set(map(int, input().split()))
    happiness = calculate_happiness(my_list, set_a, set_b)
    print(happiness)
```

Collections.OrderedDict()

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import OrderedDict

num = int(input())
ordered_dict = OrderedDict()

for i in range(num):
    line = input().rsplit(' ', 1)
    price = int(line[-1])
    name = " ".join(line[:-1])
    if name in ordered_dict:
        ordered_dict[name] += price
    else:
        ordered_dict[name] = price

for k in ordered_dict:
    print(k, ordered_dict[k])
```

Symmetric Difference

```
input()
first_set=set(map(int,input().split()))
input()
second_set=set(map(int,input().split()))

symm_difference=sorted(first_set.symmetric_difference(second_set))

for i in symm_difference:
    print(i)
```

Set .add()

```
n = int(input())
setA = set()
for i in range(n):
    setA.add(input())

print (len(setA))
```

Word Order

```
from collections import Counter

N=int(input())
l1=[]
for i in range(N):
    l1.append(input())

counter=Counter(l1)
print(len(counter))
```

```
for i in counter.values():
    print(i,end=" ")
```

Set .discard(), .remove() & .pop()

```
n = int(input())
s = set(map(int, input().split()))
m = int(input())
set1 = []

for i in range(m):
    set1.append(input().split())


for i in set1:
    if i[0] == "remove":
        try:
          s.remove(int(i[1]))
        except KeyError:
          pass
    elif i[0] == "discard":
        s.discard(int(i[1]))
    elif i[0] == "pop":
      if s:
        s.remove(next(iter(s)))


print(sum(s))
```

Collections.deque()

```
from collections import deque

n_ops = int(input())

my_deque = deque()


for i in range(n_ops):
    line = input().split()
    op = line[0]
    if len(line) > 1:
      val = line[1]
    else:
        val = None
     if op == "append":
        my_deque.append(int(val))
    elif op == "appendleft":
        my_deque.appendleft(int(val))
    elif op == "pop":
        my_deque.pop()
    elif op == "popleft":
       my_deque.popleft()

for i in my_deque:
    print(i, end=" ")
```

Company Logo

```
import math
import os
import random
import re
import sys
from collections import Counter

if __name__ == '__main__':

    str = sorted(input())

    count = Counter(list(str))

    for key, val in count.most_common(3):
        print(key, val)
```

Set .union() Operation

```
l=[]
x=int(input())
eng=list(map(int,input().rstrip().split()))

f=int(input())
fr=list(map(int,input().rstrip().split()))

for i in eng:
    l.append(i)

for i in fr:
    l.append(i)

s=set(l)
print(len(s))
```

Piling Up!

```
from collections import deque

def pilling_up(blocks):
    while blocks:
        left_block = blocks[0]
        right_block = blocks[-1]

        if left_block >= right_block:
            current_block = blocks.popleft()
        else:
            current_block = blocks.pop()

        if not blocks:
            return "Yes"

        if blocks[0] > current_block or blocks[-1] > current_block:
            return "No"

    return "Yes"

num_tests = int(input())
for _ in range(num_tests):
    num_blocks = int(input())
    block_val = deque(map(int, input().split()))
    print(pilling_up(block_val))
```

Set .intersection() Operation

```
s1=int(input())
set1=set(map(int,input().split()))
s2=int(input())
set2= set(map(int,input().split()))

summ=0
res=set1.intersection(set2)

for i in res:
    summ+=1
print(summ)
```

Set .difference() Operation

```
s1=int(input())
set1=set(map(int,input().split()))

s2=int(input())
set2=set(map(int,input().split()))

print(len(set1-set2))
```

Set .symmetric_difference() Operation

```
a=int(input())
a1=set(map(int,input().split()))
b=int(input())
b1=set(map(int,input().split()))

print(len(a1.symmetric_difference(b1)))
```

Set Mutations

```
def set_manipulations():

    n = int(input())
    setA = set(map(int, input().split()))
    number_of_actions = int(input()) # Obtain the number of operations

    set_actions = {
        "intersection_update": setA.intersection_update,
        "update": setA.update,
        "symmetric_difference_update": setA.symmetric_difference_update,
        "difference_update": setA.difference_update
    }


    for i in range(number_of_actions):
        action, _ = input().split()
        other_elements = set(map(int, input().split()))
        set_actions[action](other_elements)

    print(sum(setA))

if __name__ == "__main__":
    set_manipulations()
```

Check Subset

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
x = int(input())

for i in range(x):
    n1 = int(input())
    set_A = set(map(int, input().split()))

    n2 = int(input())
    set_B = set(map(int, input().split()))


    print(set_A.issubset(set_B))
```

Check Strict Superset

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
base_set = set(map(int, input().split()))


n = int(input())
is_strict_superset = True


for i in range(n):

    comparison_set = set(map(int, input().split()))

    if not base_set > comparison_set:
        is_strict_superset  = False
        break

print(is_strict_superset)
```

Zipped!

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
def calculate_student_averages():

    n_stud, n_assignments = map(int, input().split())
    assignment_scr = []
    for _ in range(n_assignments):
      scores_per_student = list(map(float, input().split()))
      assignment_scr.append(scores_per_student)

    for stud_scr in zip(*assignment_scr):
        avg_score = sum(stud_scr) / len(stud_scr)
        print(avg_score)

if __name__ == "__main__":
    calculate_student_averages()
```

Input()

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
xk=list(map(int,input().split()))
x=xk[0]
k=xk[1]
p=input()

if eval(p)==k:
    print(True)
else:
    print(False)
```

Python Evaluation

```
eval(input())
```

Athlete Sort

```
#!/bin/python3

import math
import os
import random
import re
import sys



if __name__ == '__main__':
    first_multiple_input = input().rstrip().split()

    n = int(first_multiple_input[0])

    m = int(first_multiple_input[1])

    a = []

    for _ in range(n):
        a.append(list(map(int, input().rstrip().split())))

    k = int(input().strip())
sorted_a=sorted(a, key = lambda x : x[k])

for row in sorted_a:
    print(' '.join(str(x) for x in row))
```

Any or All

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
input()
numbers = input().split()
numbers = [int(num) for num in numbers]

if all(num > 0 for num in numbers) and any(str(num) == str(num)[::-1] for
num in numbers):
    print(True)
else:
    print(False)
```

ginortS

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
s = input()
sorted_chars = sorted(s, key=lambda x: (x.isdigit() and int(x) % 2 == 0,
x.isdigit(), x.isupper(), x.islower(), x))
print("".join(sorted_chars))
```

Map and Lambda Function

```
cube = lambda x: x**3

def fibonacci(n):
    if n <= 0:
      return []
    lis = [0, 1]
    for i in range(2, n):
        lis.append(lis[i-2] + lis[i-1])
    return lis[:n]
```

XML 1 - Find the Score

```
def get_attr_number(node):
    count = len(node.attrib)
    for sub_element in node:
        if sub_element is not None:
            count += get_attr_number(sub_element)
        else:
           count += len(element.attrib)
    return count
```

XML2 - Find the Maximum Depth

```
maxdepth = 0
def depth(el, level):
    level+=1
    global maxdepth
    maxdepth = max(maxdepth, level)
    for sub_el in el:
        depth(sub_el, level)
    return maxdepth
```

Arrays

```
def arrays(arr):
    # complete this function
    # use numpy.array
    return numpy.flip(numpy.array(arr, dtype=float))
```

Shape and Reshape

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import numpy as np

arr=np.array(input().split(),int)
print(arr.reshape(3,3))
```

Transpose and Flatten

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import numpy as np

n, m = map(int, input().split())
rows = []

for i in range(n):
    row = input().strip().split()
    rows.append(row)

array = np.array(rows, dtype=int)
print(array.transpose())
print(array.flatten())
```

Concatenate

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import numpy as np

a, b, c = map(int, input().split())
rows_a = []
rows_b = []
for i in range(a):
    row = input().split()
    rows_a.append(row)

A = np.array(rows_a, dtype=int)

for j in range(b):
    row = input().split()
    rows_b.append(row)

B = np.array(rows_b, dtype=int)

print(np.concatenate((A, B), axis=0))
```

Zeros and Ones

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import numpy as np

nums = list(map(int, input().split()))
print(np.zeros(tuple(nums), dtype=np.int))
print(np.ones(tuple(nums), dtype=np.int))
```

Eye and Identity

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import numpy as np

np.set_printoptions(sign=' ')
n, m = map(int, input().split())
arr = np.eye(n,m)
print(arr)
```

Array Mathematics

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import numpy as np

n, m = map(int, input().split())
a = np.array([input().split() for i in range(n)], dtype=int)
b = np.array([input().split() for i in range(n)], dtype=int)

print(a + b,
 a - b,
 a * b,
 a // b,
 a % b,
 a ** b,
 sep='\n'
 )
```

Floor, Ceil and Rint

```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import numpy as np

np.set_printoptions(sign=' ')

n = input().split()
a = np.array([float(x) for x in n])

print(*map(lambda f: f(a), [np.floor, np.ceil, np.rint]), sep='\n')
```

Sum and Prod

```
import numpy as np
n, m = map(int, input().split())
a = np.array([input().split() for _ in range(n)], dtype=int)

column_sums = np.sum(a, axis=0)
prod_sums = np.prod(column_sums, axis=0)

print(prod_sums)
```

Min and Max
```
import numpy as np

n, m = map(int, input().split())
arr = np.array([input().split() for _ in range(n)], dtype=int)
minn=np.min(arr, axis=1)
res=np.max(minn, axis=0)
print(res)
```

Mean, Var, and Std
```
import numpy as np

n, m = map(int, input().split())
arr = np.array([input().split() for i in range(n)], int)

print(np.mean(arr, axis=1))
print(np.var(arr, axis=0))
print(np.around(np.std(arr), 11))
```

Dot and Cross
```
import numpy as np

N = int(input())
A = np.array([input().split() for i in range(N)], dtype=int)
B = np.array([input().split() for i in range(N)], dtype=int)
print(np.dot(A,B))
```

Inner and Outer
```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import numpy as np

#N=int(input())
A=np.array(input().split(), dtype=int)
B=np.array(input().split(), dtype=int)

print(np.inner(A,B))
print(np.outer(A,B))
```

Polynomials
```
import numpy as np

poly_str = input().split()
poly = [float(x) for x in poly_str]
x = float(input())

res = np.polyval(poly, x)
print(res)
```

Linear Algebra
```
# Enter your code here. Read input from STDIN. Print output to STDOUT
import numpy as np

np.set_printoptions(legacy='1.13')
```

```
N = int(input())
A = np.array([input().split() for _ in range(N)], dtype=float)
print(np.linalg.det(A))
```