hw1-1

October 13, 2024

[ ]:
```
#Say,Hello
if __name__ == '__main__':
    print("Hello, World! ")
```

[ ]:
```
#Python If-Else
import math
import os
import random
import re
import sys



if __name__ == '__main__':
    n = int(input().strip())
if n%2!=0:
    print("Weird")
if n%2==0:
   if n in range(2,5):
       print("Not Weird")
   if n in range(6,21):
       print("Weird")
   if n>20:
        print("Not Weird")
```

[ ]:
```
#Arithmetic Operators
if __name__ == '__main__':
    a = int(input())
    b = int(input())
print(a+b)
print(a-b)
print(a*b)
```

[ ]:
```
#Python Division
if __name__ == '__main__':
    a = int(input())
    b = int(input())
```

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

[ ]:
```
#Python loops
if __name__ == '__main__':
    n = int(input())
for i in range(0,n):
    print(i**2)
```

[ ]:
```
#Write a Function
def is_leap(year):
    leap = False
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
    return leap
```

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

[ ]:
```
#List Comprehensions
if __name__ == '__main__':
    x = int(input())
    y = int(input())
    z = int(input())
    n = int(input())
    l=[[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(l)
```

[ ]:
```
#Find the Runner-Up Score!
if __name__ == '__main__':
    n = int(input())
    arr = map(int, input().split())
    s=list(arr)
    s_clean=[]
    for x in s:
       if x not in s_clean:
         s_clean.append(x)
    s_sorted=sorted(s_clean)
    print(s_sorted[-2])
```

```
[ ]: #Nested Lists
if __name__ == '__main__':
    students=[]
    for _ in range(int(input())):
        name = input()
        score = float(input())
        students.append([name, score])
    u_scores=set(student[1] for student in students)
    ordered_scores=sorted(u_scores)
    s_l_score=ordered_scores[1]
    s_l_students = [student[0] for student in students if student[1] ==␣
 ↪s_l_score]
    sorted_s_l=sorted(s_l_students)
    for name in sorted_s_l:
        print(name)
```

```
[ ]: #Finding the percentage
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()
    if query_name in student_marks:
        average=sum(student_marks[query_name])/len(student_marks[query_name])
        print("%.2f" % average)
```

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

```
[ ]: #Lists
if __name__ == '__main__':
    N = int(input())
    l=[]
    for _ in range(N):
       command = input().strip().split()
       if command[0] == "insert":
        l.insert(int(command[1]), int(command[2]))
       elif command[0] == "print":
        print(l)
       elif command[0] == "remove":
        l.remove(int(command[1]))
       elif command[0] == "append":
        l.append(int(command[1]))
       elif command[0] == "sort":
        l.sort()
       elif command[0] == "pop":
        l.pop()
       elif command[0] == "reverse":
        l.reverse()
```

[ ]:
```
#sWAP cASE
def swap_case(s):
    modified = ""
    for x in s:
        if x.islower():
            modified += x.upper()
        elif x.isupper():
            modified += x.lower()
        else:
            modified += x

    return modified

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

[ ]:
```
#String Split and Join
def split_and_join(line):
    l=line.split(" ")
    j="-".join(l)
    return j


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

[ ]:
```
#What's Your Name?
def print_full_name(first, last):
    print(f"Hello {first} {last}! You just delved into python.")

if __name__ == '__main__':
    first_name = input()
    last_name = input()
    print_full_name(first_name, last_name)
```

[ ]:
```
#Mutations
def mutate_string(string, position, character):
    l=list(string)
    l[int(position)]=f"{character}"
    s="".join(l)
    return s

if __name__ == '__main__':
    s = input()
    i, c = input().split()
    s_new = mutate_string(s, int(i), c)
    print(s_new)
```

[ ]:
```
#Find a string
def count_substring(string, sub_string):
    n=0
    for i in range(len(string)-len(sub_string)+1):
        if string[i:i+len(sub_string)]== sub_string:
            n+=1
    return n

if __name__ == '__main__':
    string = input().strip()
    sub_string = input().strip()

    count = count_substring(string, sub_string)
    print(count)
```

[ ]:
```
#String Validators
if __name__ == '__main__':
    s = input()
    print(any(x.isalnum() for x in s))
    print(any(x.isalpha() for x in s))
    print(any(x.isdigit() for x in s))
    print(any(x.islower() for x in s))
    print(any(x.isupper() for x in s))
```

[ ]:
```
#Text Wrap
import textwrap

def wrap(string, max_width):
    wrapped= textwrap.fill(string,int(max_width))
    return wrapped

if __name__ == '__main__':
    string, max_width = input(), int(input())
    result = wrap(string, max_width)
    print(result)
```

[ ]:
```
#String formatting
def print_formatted(number):
    width=len(bin(number)[2:])
    for i in range(1, number + 1):
        d = str(i).rjust(width)
        o = oct(i)[2:].rjust(width)
        h = hex(i)[2:].upper().rjust(width)
        b = bin(i)[2:].rjust(width)
        print(d, o, h, b)


if __name__ == '__main__':
    n = int(input())
    print_formatted(n)
```

[ ]:
```
#Text Alignment
thickness = int(input())
c = 'H'


for i in range(thickness):
    print((c * (2 * i + 1)).center(thickness * 2 - 1,' '))
for i in range(thickness + 1):
    print((c * thickness).center(thickness * 2)+(c * thickness).
 ↪center(thickness *6))
for i in range((thickness + 1) // 2):
    print((c * (thickness * 5)).center(thickness *6))
for i in range(thickness + 1):
    print((c * thickness).center(thickness * 2)+(c * thickness).
 ↪center(thickness * 6))
for i in range(thickness - 1, -1, -1):
    print((c * (2 * i + 1)).center(thickness * 10))
```

[ ]:
```
#Write a function
def is_leap(year):
    leap = False
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return True
    else:
        return False
    return leap
```

[ ]:
```
#Designer Door Mat
N,M=map(int, input().split())
pattern = '.|.'
for i in range(N// 2):
    print((pattern * (2 * i + 1)).center(M, '-'))
print('WELCOME'.center(M, '-'))
for i in range(N // 2 - 1, -1, -1):
    print((pattern * (2 * i + 1)).center(M, '-'))
```

[ ]:
```
#Alphabet Rangoli
import string
def print_rangoli(size):
    import string
    letters = string.ascii_lowercase[:size]
    lines = []
    for i in range(size):
        left=letters[size-1:i:-1]
        middle=letters[i:size]
        line='-'.join(left+middle)
        lines.append(line.center(4*size-3,'-'))
    print('\n'.join(lines[::-1]+ lines[1:]))
```

[ ]:
```
#Capitalize!
def solve(s):
    s = s.split(' ') #creo lista
    for i in range(len(s)): #capitalize ogni elemento della lista
        s[i] = s[i].capitalize()
    return ' '.join(s)
```

[ ]:
```
#The Minion Game
def minion_game(string):

    l = list(string.lower())
    stuart = 0
    kevin = 0

    for position,letter in enumerate(l):
        if letter in ['a','e','i','o','u']:
            n_substrings = len(l) - position
            kevin += n_substrings
            no_substrings =0
        else:
            n_substrings = len(l) - position
            stuart += n_substrings
            n_substrings = 0
```

[ ]:
```
#Merge the Tools!
def merge_the_tools(string, k):
    for i in range(0, len(string), k):
        substring = string[i:i+k]
        seen = set()
        result = []
        for x in substring:
           if x not in seen:
              seen.add(x)
              result.append(x)
        print(''.join(result))
```

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

[ ]:
```
#No Idea!
n, m = map(int, input().split())
arr = list(map(int, input().split()))
A = set(map(int, input().split()))
B = set(map(int, input().split()))
happiness=0
for x in arr:
    if x in A:
        happiness+=1
    if x in B:
        happiness+= -1
print(happiness)
```

[ ]:
```
#Set .add()
N=int(input())
s=set()
for i in range(N):
    s.add(input())
print(len(s))
```

[ ]:
```
#Set .discard(), .remove() & .pop()
n = int(input())
s = set(map(int, input().split()))
N=int(input())

for i in range(N):
    command = list(input().split())
    if command[0]=='pop':
            s.pop()
    if command[0]=='remove':
       if int(command[1]) in s:
          s.remove(int(command[1]))
    if command[0]=='discard':
        s.discard(int(command[1]))

print(sum(s))
```

[ ]:
```
#Set .union() Operation
n = int(input())
English = set(map(int, input().split()))
b = int(input())
French = set(map(int, input().split()))
print(len(English.union(French)))
```

[ ]:
```
#Set .intersection() Operation
n = int(input())
English = set(map(int, input().split()))
b = int(input())
French = set(map(int, input().split()))
print(len(English.intersection(French)))
```

[ ]:
```
#Set .difference() Operation
n = int(input())
English = set(map(int, input().split()))
b = int(input())
French = set(map(int, input().split()))
print(len(English.difference(French)))
```

[ ]:
```
#Set .symmetric_difference() Operation
n = int(input())
English = set(map(int, input().split()))
b = int(input())
French = set(map(int, input().split()))
print(len(English.symmetric_difference(French)))
```

[ ]:
```
#Set Mutations
n = int(input())
A = set(map(int, input().split()))
N = int(input())
for i in range(N):
    command = input().split()
    B = set(map(int, input().split()))
    if command[0] == "intersection_update":
        A.intersection_update(B)
    elif command[0] == "update":
        A.update(B)
    elif command[0] == "symmetric_difference_update":
        A.symmetric_difference_update(B)
    elif command[0] == "difference_update":
        A.difference_update(B)

print(sum(A))
```

[ ]:
```
#The Captain's Room
K = int(input())
rooms = list(map(int,input().split()))
rooms_set = set(rooms)
captain_room = (sum(rooms_set) * K - sum(rooms)) // (K - 1)
print(captain_room)
```

[ ]:
```
#Check Subset
T= int(input())

for i in range(T):
    n = int(input())
    A = set(map(int, input().split()))
    m = int(input())
    B = set(map(int, input().split()))
    print(A.intersection(B) == A)
```

[ ]:
```
#Check Strict Superset
A = set(map(int, input().split()))
n = int(input())
result = True

for i in range(n):
    s = set(map(int, input().split()))
    if not (A.issuperset(s) and A != s):
        result = False
        break
print(result)
```

[ ]:
```
#collections.Counter()
from collections import Counter
X=int(input())
s_counter = Counter(map(int, input().split()))
N = int(input())
price = 0
for i in range(N):
    s_size, p = map(int, input().split())
    if s_counter[s_size]:
        s_counter[s_size] -=1
        price += p
print(price)
```

```
[ ]: #DefaultDict Tutorial
     from collections import defaultdict
     d = defaultdict(list)
     n, m = map(int, input().split())
     A = [str(input()) for i in range(n)]
     B = [str(input()) for i in range(m)]
     for a in A:
         if a not in d.keys():
             d[a].extend([index + 1 for index,value in enumerate(A) if value ==a])
     for b in B:
         if b in d.keys():
             print(' '.join(map(str, d[b])))
         else:
             print('-1')
```

```
[ ]: #Collections.namedtuple()
     from collections import namedtuple
     N=int(input())
     Student=namedtuple('Student', input().split())
     marks=0
     for x in range(N):
         s=Student(*input().split())
         marks+=int(s.MARKS)
     print(f'{marks/N:.2f}')
```

```
[ ]: #Collections.OrderedDict()
     from collections import OrderedDict
     N=int(input())
     d=OrderedDict()
     for i in range(N):
          L=input().rsplit(maxsplit=1)
          item=L[0]
          price=int(L[1])
          if item in d:
              d[item]= d[item]+price
          else: d.update({item: price})
     for x,y in d.items(): print(x,y)
```

```
[ ]: #Word Order
     from collections import OrderedDict
     n=int(input())
     words=OrderedDict()
     for i in range(n):
         w=input()
         if w in words:
             words[w]=words[w]+1
         else: words[w]=1
```

```
print(len(words))
print(*words.values())
```

[ ]:
```
#Collections.deque()
from collections import deque
d=deque()
N=int(input())
for i in range(N):
    l=input().split()
    method=l[0]
    if method == 'append':
            d.append(l[1])
    elif method == 'pop':
              d.pop()
    elif method == 'popleft':
              d.popleft()
    else:
              d.appendleft(l[1])
print(' '.join(d))
```

[ ]:
```
#Company Logo
import math
import os
import random
import re
import sys
from collections import Counter
s= sorted(input())
s_count = Counter(s)
most_commons = s_count.most_common(3)
for i in range(len(most_commons)):
    print(most_commons[i][0], most_commons[i][1])
```

[ ]:
```
#Piling Up!
T=int(input())
for i in range(T):
    cubes = int(input())
    Lengths = list(map(int, input().split()))
    flag = "Yes"
    rl = list(reversed(Lengths))
    for i in range(1, len(Lengths)):
        if (Lengths[0] < Lengths[i] and rl[0] < rl[i]):
            flag = "No"
            break
    print(flag)
```

[ ]:
```
#Calendar Module
import calendar
m, d, y = input().split()
year = int(y)
month = int(m)
day = int(d)
week_day = calendar.weekday(year, month, day)
week_name = calendar.day_name[week_day].upper()
print(week_name)
```

[ ]:
```
#Time Delta
import math
import os
import random
import re
import sys

from datetime import datetime
def time_delta(t1, t2):
    dt1= datetime.strptime(t1, '%a %d %b %Y %H:%M:%S %z')
    dt2= datetime.strptime(t2, '%a %d %b %Y %H:%M:%S %z')
    difference=abs((dt1 - dt2).total_seconds())
    return int(difference)
```

[ ]:
```
#Exceptions
T=int(input())
for i in range(T):
    a,b= input().split()
    try:
          print(int(a)//int(b))
    except ValueError as e:
           print("Error Code:",e)
    except ZeroDivisionError as e:
           print("Error Code:",e)
```

[ ]:
```
#Zipped!
N,X=input().split()
subjects = []

for i in range(int(X)):
    subjects.append(list(map(float, input().split())))

for grade in zip(*subjects):
    print(sum(grade)/int(X))
```

[ ]:
```
#Athlete Sort
import math
import os
import random
import re
import sys

if __name__ == '__main__':
    nm = input().split()

    n = int(nm[0])

    m = int(nm[1])

    arr = []

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

    k = int(input())
    arr.sort(key=lambda x: x[k])
    for i in arr:
        print(*i)
```

[ ]:
```
#ginortS
S=input()
upper= sorted([x for x in S if x.isupper()])
lower = sorted([x for x in S if x.islower()])
even= sorted([x for x in S if x.isdigit() if int(x) % 2 == 0])
odd = sorted([x for x in S if x.isdigit() if int(x) % 2 == 1])
print("".join(lower+upper+odd+even))
```

[ ]:
```
#Map and Lambda Function
cube = lambda x: x**3 # complete the lambda function

def fibonacci(n):
    # return a list of fibonacci numbers
    if n == 0:
        return []
    if n == 1:
        return [0]
    result = [0, 1]
    for i in range(2, n):
        N = result[-1] + result[-2]
        result.append(N)
    return result
```

[ ]:
```
#Detect Floating Point Number
T=int(input())
```

```
for i in range(T):
    N = input()
    if '0' in N:
        print('False')
    elif '.' in N:
        try:
            float(N)
            print('True')
        except:
               print('False')
    else: print('False')
```

[ ]:
```
#Re.split()
regex_pattern = r"[,.]"        # Do not delete 'r'.

import re
print("\n".join(re.split(regex_pattern, input())))
```

[ ]:
```
#Group(), Groups() & Groupdict()
import re
S=input()
pattern = r'([a-zA-Z0-9])\1'
m = re.search(pattern, S)
if m:
    print(m.group(1))
else:
    print(-1)
```

[ ]:
```
#Re.findall() & Re.finditer()
import re
S=input()
vowels = "AEIOUaeiou"
consonants = "QWRTYPSDFGHJKLZXCVBNMqwrtypsdfghjklzxcvbnm"
pattern = rf'([{vowels}]{{2,}})(?=[{consonants}])'
result = re.findall(pattern,S)
if result == []:
    print (-1)
else:
    for i in result:
        print(i)
```

[ ]:
```
#Re.start() & Re.end()
import re
S=input()
k = input()

pattern = re.compile(f'(?=({k}))')
```

```
result = list(pattern.finditer(S))

if len(result)>1:
   for m in result:
        print((m.start(1), m.end(1)-1))
else:
    print((-1,-1))
```

[ ]:
```
#Regex Substitution
import re
N=int(input())
for n in range(N):
    i = input()
    while " && " in i:
        i = i.replace(" && ", " and ")
    while " || " in i:
        i = i.replace(" || ", " or ")
    print(i)
```

[ ]:
```
#Validating Roman Numerals
regex_pattern = r"M{0,3}(CM|CD|D?C{0,3})(XC|XL|L?X{0,3})(IX|IV|V?
↪I{0,3})$"        # Do not delete 'r'.

import re
print(str(bool(re.match(regex_pattern, input()))))
```

[ ]:
```
#Validating phone numbers
import re
N=int(input())
pattern=r'^[789]\d{9}$'
for i in range(N):
    number = input().strip()
    if re.match(pattern, number):
        print("YES")
    else:
        print("NO")
```

[ ]:
```
#Validating and Parsing Email Addresses
import re
import email.utils
n = int(input())
pattern = r'^[a-z][\w\-_.]*[@][a-z]+[.][a-z]{1,3}$'
for i in range(n):
    name, address = email.utils.parseaddr(input())
    if re.match(pattern,address):
        print(email.utils.formataddr((name, address)))
    else:
        continue
```

[ ]:
```
#Hex Color Code
import re
N=int(input())
for i in range(N):
    m = re.findall(r'#[0-9A-Fa-f]{6}(?=\S)|#[0-9A-Fa-f]{3}(?=\S)',input())
    if m:
        for i in range(len(m)):
            print(m[i])
```

[ ]:
```
#HTML Parser - Part 1
from html.parser import HTMLParser

class MyParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print(f'Start : {tag}')
        self.print_attrs(attrs)

    def handle_endtag(self, tag):
        print(f'End   : {tag}')

    def handle_startendtag(self, tag, attrs):
        print(f'Empty : {tag}')
        self.print_attrs(attrs)

    def print_attrs(self, attrs):
        for name , value in attrs:
            print(f'-> {name} > {value}')

parser = MyParser()
N=int(input())
parser.feed(''.join(input() for i in range(N)))
```

[ ]:
```
#HTML Parser - Part 2
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_comment(self, data):
        if '\n' in data:
            print(">>> Multi-line Comment")
            print(data)
        else:
            print(">>> Single-line Comment")
            print(data)

    def handle_data(self, data):
        if '\n' not in data:
            print(">>> Data")
            print(data)


html = ""
for i in range(int(input())):
    html += input().rstrip()
    html += '\n'

parser = MyHTMLParser()
parser.feed(html)
parser.close()
```

[ ]:
```
#Detect HTML Tags, Attributes and Attribute Values
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_starttag(self, tag, attrs):
        print(tag)
        self.print_attrs(attrs)

    def print_attrs(self, attrs):
        for item in attrs:
            name , value = item
            print(f'-> {name} > {value}')

my_parser  = MyHTMLParser()
N=int(input())
html = "".join(input() for i in range(N))
my_parser.feed(html)
```

[ ]:
```
#Validating UID
import re
pattern = r'^(?=(.*[A-Z]){2})(?=(.*\d){3})(?!.*(.).*\3).{10}$'
T=int(input())
for i in range(T):
    m = re.match(pattern, input())
    if m:
        print('Valid')
    else:
        print('Invalid')
```

[ ]:
```
#Validating Credit Card Numbers
import re
N = int(input())
pattern_1 =  r"^[4-6]\d{3}(-?\d{4}){3}$"
```

```
pattern_2 =  r"(\d)(-?\1){3,}"
def is_valid(credit_card):
    return bool(re.match(pattern_1,credit_card))

def is_invalid(credit_card):
    return re.findall(pattern_2,credit_card)
for i in range(N):
    credit_card = input()
    if is_valid(credit_card) and  len(is_invalid(credit_card))==0:
        print("Valid")
    else:
        print("Invalid")
```

[ ]:
```
#Validating Postal Codes
regex_integer_in_range = r"^[0-9]{6}$"
regex_alternating_repetitive_digit_pair = r"([0-9]{1})(?=[0-9]{1}\1)"

import re
P = input()

print (bool(re.match(regex_integer_in_range, P))
and len(re.findall(regex_alternating_repetitive_digit_pair, P)) < 2)
```

[ ]:
```
#Matrix Script
import math
import os
import random
import re
import sys

first_multiple_input = input().rstrip().split()

n = int(first_multiple_input[0])

m = int(first_multiple_input[1])

matrix = []

for _ in range(n):
    matrix_item = input()
    matrix.append(matrix_item)
print(re.sub(r'(?<=[A-Za-z0-9])([ !@#$%&]+)(?=[A-Za-z0-9])',' ', ''.join(s[i]
      for i in range(m) for s in matrix)))
```

[ ]:
```
#XML 1 - Find the Score
import sys
import xml.etree.ElementTree as etree
```

```
def get_attr_number(node):
    attrs = len(node.attrib)
    if len(node) == 0:
        return attrs
    return attrs + sum(get_attr_number(child) for child in node)


if __name__ == '__main__':
    sys.stdin.readline()
    xml = sys.stdin.read()
    tree = etree.ElementTree(etree.fromstring(xml))
    root = tree.getroot()
    print(get_attr_number(root))
```

[ ]:
```
#XML2 - Find the Maximum Depth
import xml.etree.ElementTree as etree

maxdepth = 0
def depth(elem, level):
    global maxdepth
    level += 1
    for child in elem:
        depth(child, level)
    maxdepth = max(maxdepth, level)

if __name__ == '__main__':
    n = int(input())
    xml = ""
    for i in range(n):
        xml =  xml + input() + "\n"
    tree = etree.ElementTree(etree.fromstring(xml))
    depth(tree.getroot(), -1)
    print(maxdepth)
```

[ ]:
```
#Standardize Mobile Number Using Decorators
def wrapper(f):
    def fun(l):
        l = ['+91 '+num[-10:-5]+' '+num[-5:] for num in l]
        return f(l)
    return fun
```

[ ]:
```
#Decorators 2 - Name Directory
def person_lister(f):
    def inner(people):
        sort = sorted(people, key=lambda x: int(x[2]))
        return [f(x) for x in sort]
    return inner
```

[ ]:
```
#Arrays
def arrays(arr):
    a=numpy.array(arr,float)
    return a[::-1]
```

[ ]:
```
#Shape and Reshape
import numpy as np
y = input().split()
x = np.array(y,int)
x = x.reshape(3,3)
print(x)
```

[ ]:
```
#Transpose and Flatten
import numpy as np
N,M = list(map(int,input().split()))
NM = np.array([list(map(int,input().split())) for i in range(N)])
print(np.transpose(NM))
print(NM.flatten())
```

[ ]:
```
#Concatenate
import numpy as np
n,m,P = map(int, input().split())
N = np.array([list(map(int, input().split())) for i in range(n)])
M = np.array([list(map(int, input().split())) for i in range(m)])

print(np.concatenate((N, M)))
```

[ ]:
```
#Zeros and Ones
import numpy as np
l = list(map(int, input().split()))
print(np.zeros(l,dtype=int))
print(np.ones(l,dtype=int))
```

[ ]:
```
#Eye and Identity
import numpy as np
np.set_printoptions(legacy='1.13')
N,M=map(int,input().split())
print(np.eye(N,M))
```

[ ]:
```
#Array Mathematics
import numpy as np
M, N = map(int, input().split())
A = np.array([[x for x in input().split()] for i in range(M)], int)
B = np.array([[x for x in input().split()] for i in range(M)], int)
print(A+B)
print(A-B)
print(A*B)
print(A//B)
print(A%B)
print(A**B)
```

[ ]:
```
#Floor, Ceil and Rint
import numpy as np
np.set_printoptions(legacy='1.13')
A=list(map(float,input().split()))
arr=np.array(A)
print(np.floor(arr))
print(np.ceil(arr))
print(np.rint(arr))
```

[ ]:
```
#Sum and Prod
import numpy as np
N, M = map(int, input().split())
A = np.array([list(map(int, input().split())) for i in range(N)])
print(np.prod(np.sum(A, axis=0)))
```

[ ]:
```
#Min and Max
import numpy as np
N,M=map(int,input().split())
arr = np.array([input().split() for i in range(N)], int)
print(np.max(np.min(arr,axis=1),axis= None))
```

[ ]:
```
#Mean, Var, and Std
import numpy as np
N, M = map(int, input().split())
A = np.array([list(map(int, input().split())) for i in range(N)])
print(np.mean(A,axis=1))
print(np.var(A,axis=0))
print(round(np.std(A), 11))
```

[ ]:
```
#Dot and Cross
import numpy as np
N = int(input())
A = np.array([list(map(int, input().split())) for i in range(N)])
B = np.array([list(map(int, input().split())) for i in range(N)])
product = np.matmul(A, B)
print(product)
```

[ ]:
```
#Inner and Outer
import numpy as np
A = np.array(list(map(int, input().split())))
B = np.array(list(map(int, input().split())))
```

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

[ ]:
```
#Polynomials
import numpy as np
P = list(map(float,input().split()))
x = int(input())
arr = np.polyval(P,x)
print(arr)
```

[ ]:
```
#Linear Algebra
import numpy as np
N = int(input())
A = np.array([list(map(float, input().split())) for i in range(N)])
rounded = round(np.linalg.det(A),2)
print(rounded)
```