NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

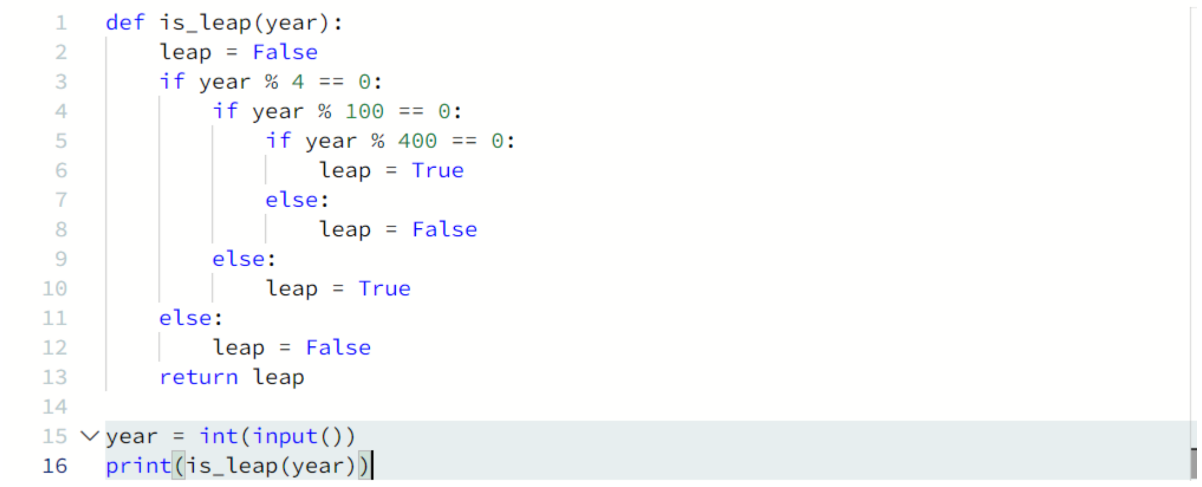
Robotics and Intelligent Machine Engineering

Artificial Intelligence (CSE-860)
ASSIGNMENT # 3

Submitted To:	**Dr. Yasar Ayaz**
Submitted By:	**Saad Ahmed Khan**
Registration Number:	**451360**
Date of Submission:	**January 01, 2024**

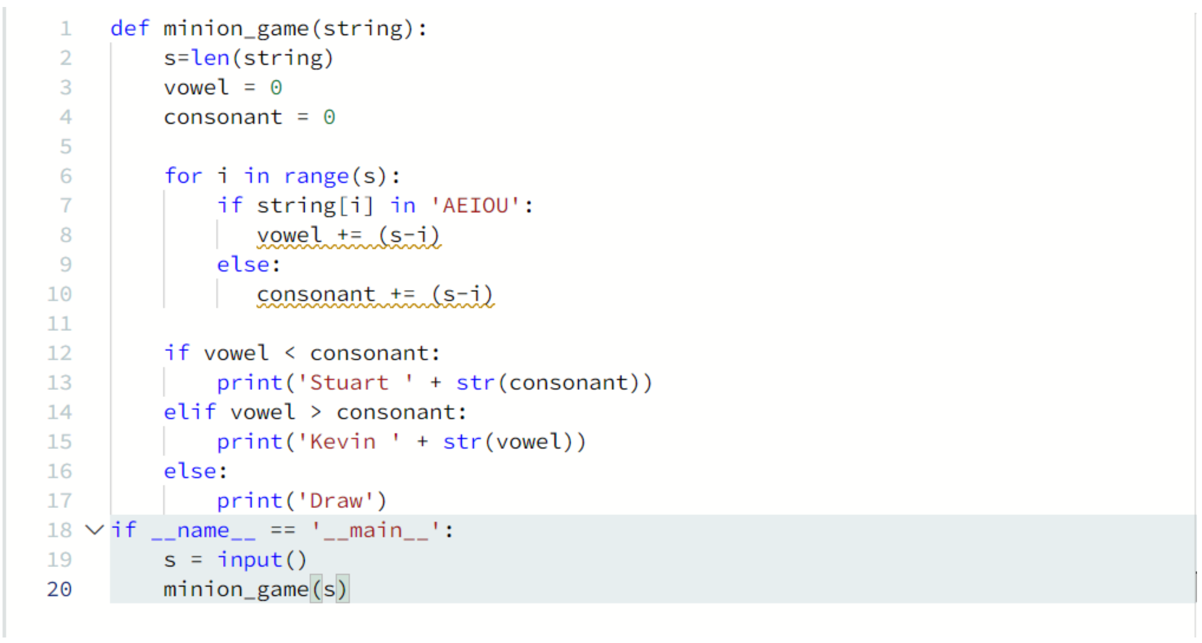
Medium Level Tasks

Write a Function



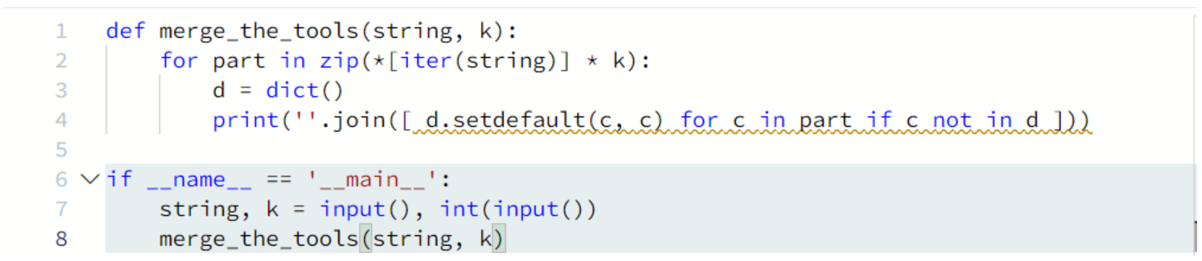
```python
def is_leap(year):
    leap = False
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                leap = True
            else:
                leap = False
        else:
            leap = True
    else:
        leap = False
    return leap

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

The Minion Game



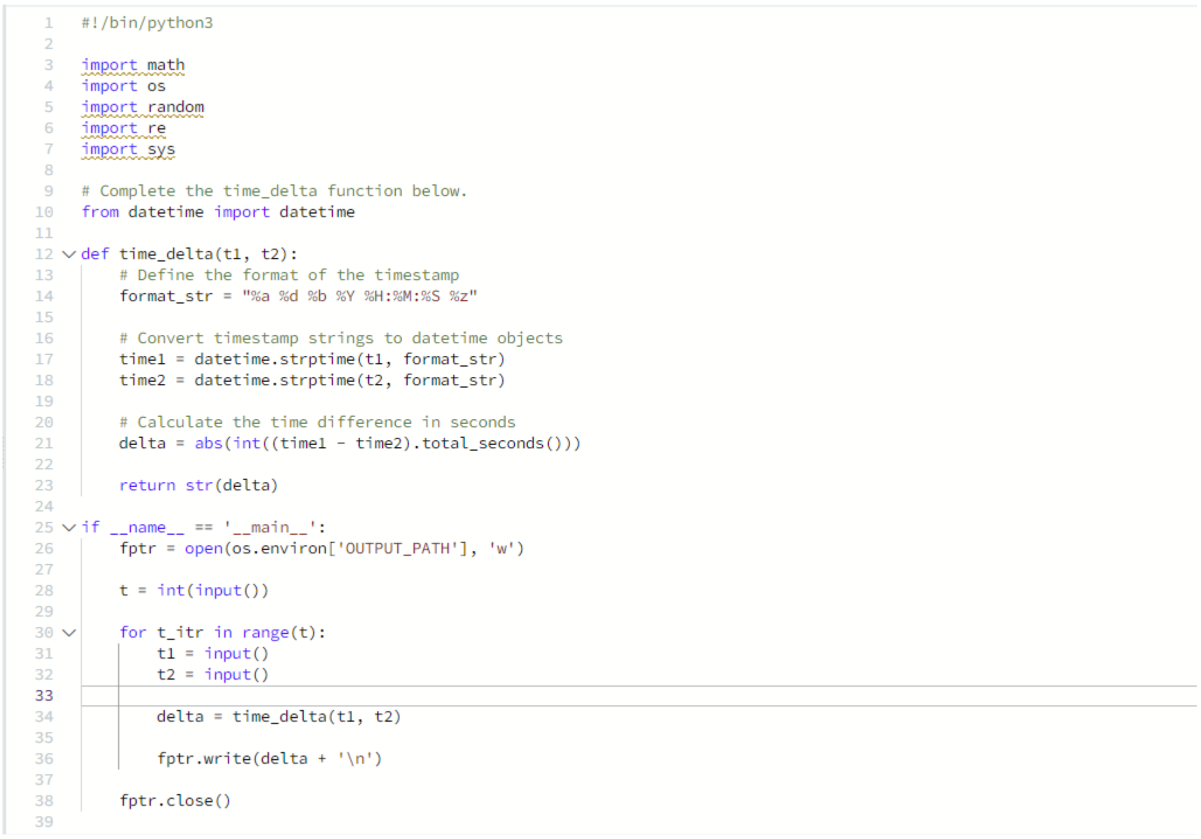
```python
def minion_game(string):
    s=len(string)
    vowel = 0
    consonant = 0

    for i in range(s):
        if string[i] in 'AEIOU':
            vowel += (s-i)
        else:
            consonant += (s-i)

    if vowel < consonant:
        print('Stuart ' + str(consonant))
    elif vowel > consonant:
        print('Kevin ' + str(vowel))
    else:
        print('Draw')
if __name__ == '__main__':
    s = input()
    minion_game(s)
```

Merge the Tools!



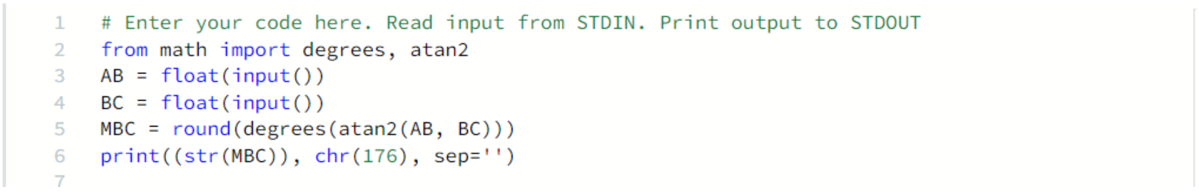
```python
def merge_the_tools(string, k):
    for part in zip(*[iter(string)] * k):
        d = dict()
        print(''.join([ d.setdefault(c, c) for c in part if c not in d ]))

if __name__ == '__main__':
    string, k = input(), int(input())
    merge_the_tools(string, k)
```

Time Delta



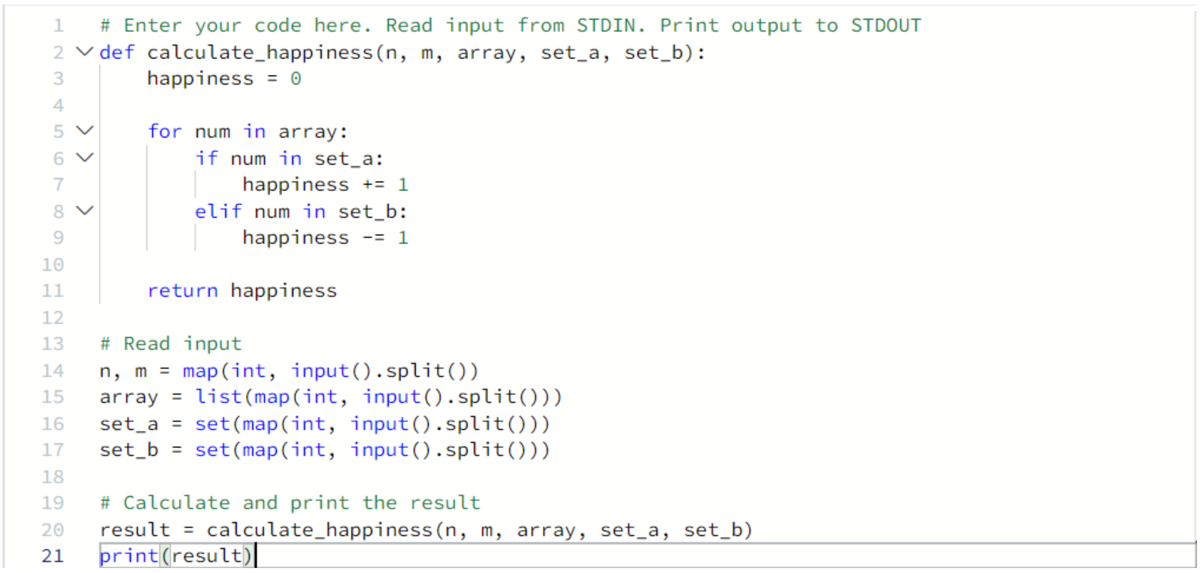
```python
#!/bin/python3

import math
import os
import random
import re
import sys

# Complete the time_delta function below.
from datetime import datetime

def time_delta(t1, t2):
    # Define the format of the timestamp
    format_str = "%a %d %b %Y %H:%M:%S %z"

    # Convert timestamp strings to datetime objects
    time1 = datetime.strptime(t1, format_str)
    time2 = datetime.strptime(t2, format_str)

    # Calculate the time difference in seconds
    delta = abs(int((time1 - time2).total_seconds()))

    return str(delta)

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()

```

Find Angle MBC



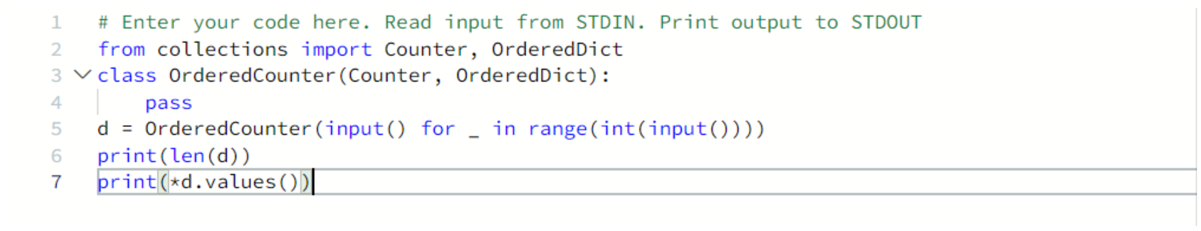
```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
from math import degrees, atan2
AB = float(input())
BC = float(input())
MBC = round(degrees(atan2(AB, BC)))
print((str(MBC)), chr(176), sep='')

```

No Idea!



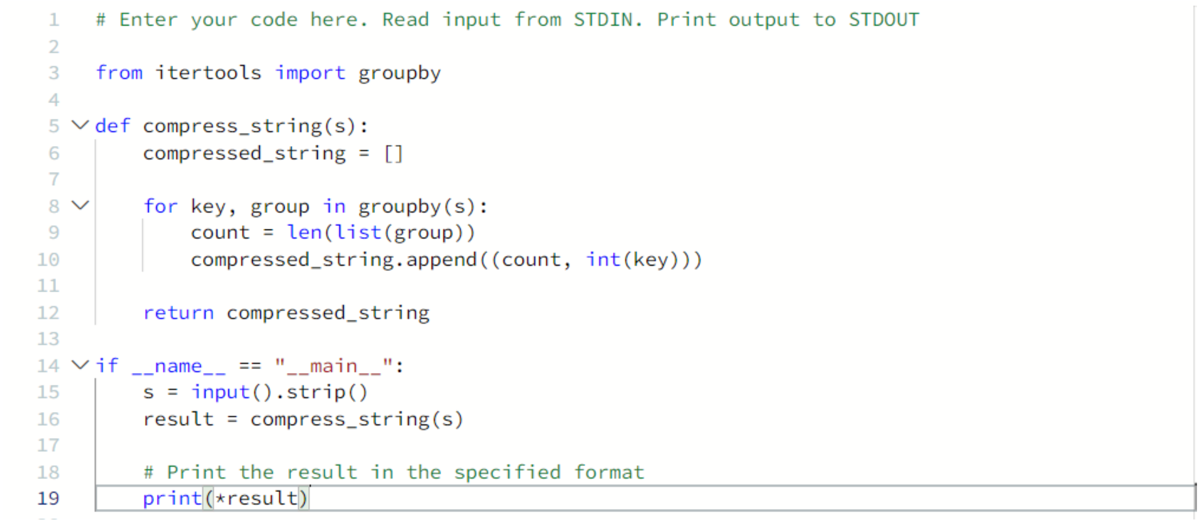
```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
def calculate_happiness(n, m, array, set_a, set_b):
    happiness = 0

    for num in array:
        if num in set_a:
            happiness += 1
        elif num in set_b:
            happiness -= 1

    return happiness

# Read input
n, m = map(int, input().split())
array = list(map(int, input().split()))
set_a = set(map(int, input().split()))
set_b = set(map(int, input().split()))

# Calculate and print the result
result = calculate_happiness(n, m, array, set_a, set_b)
print(result)
```

Word Order



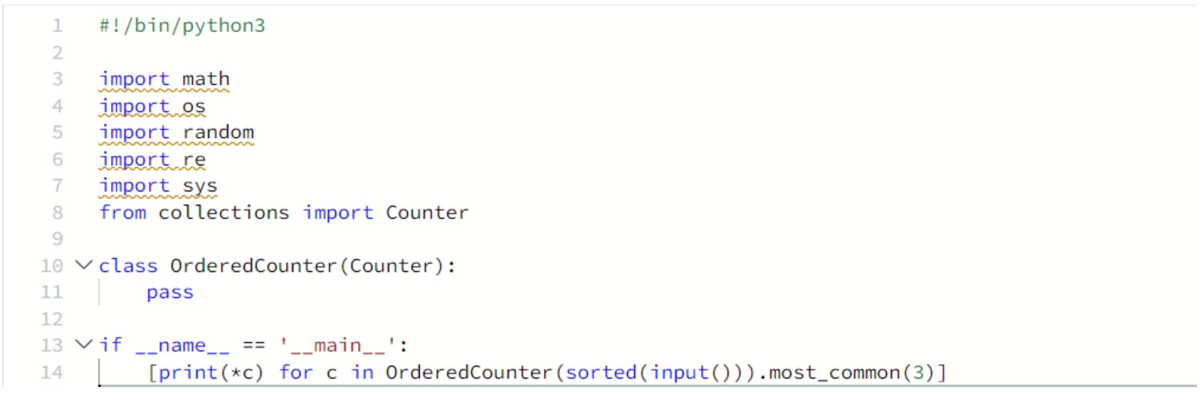
```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
from collections import Counter, OrderedDict
class OrderedCounter(Counter, OrderedDict):
    pass
d = OrderedCounter(input() for _ in range(int(input())))
print(len(d))
print(*d.values())
```

Compress the String!



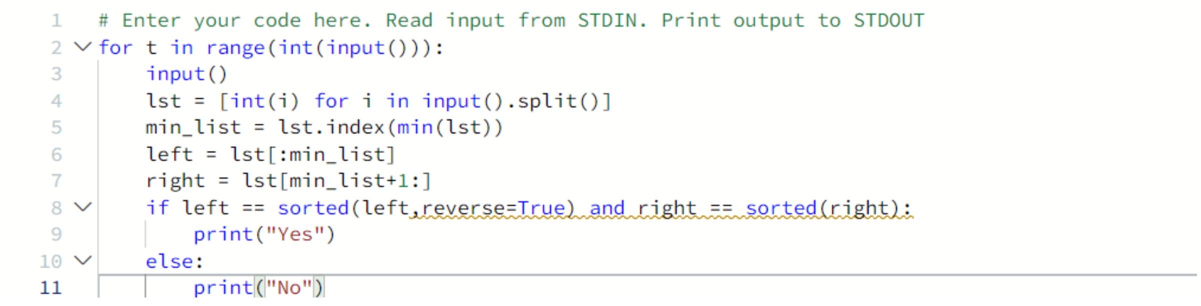
```python
# Enter your code here. Read input from STDIN. Print output to STDOUT

from itertools import groupby

def compress_string(s):
    compressed_string = []

    for key, group in groupby(s):
        count = len(list(group))
        compressed_string.append((count, int(key)))

    return compressed_string

if __name__ == "__main__":
    s = input().strip()
    result = compress_string(s)

    # Print the result in the specified format
    print(*result)
```

Company Logo



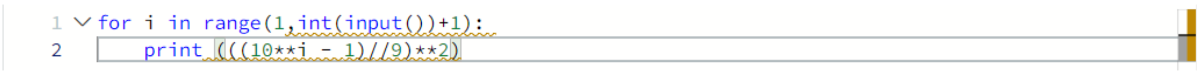
```python
#!/bin/python3

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

class OrderedCounter(Counter):
    pass

if __name__ == '__main__':
    [print(*c) for c in OrderedCounter(sorted(input())).most_common(3)]
```

Piling Up!



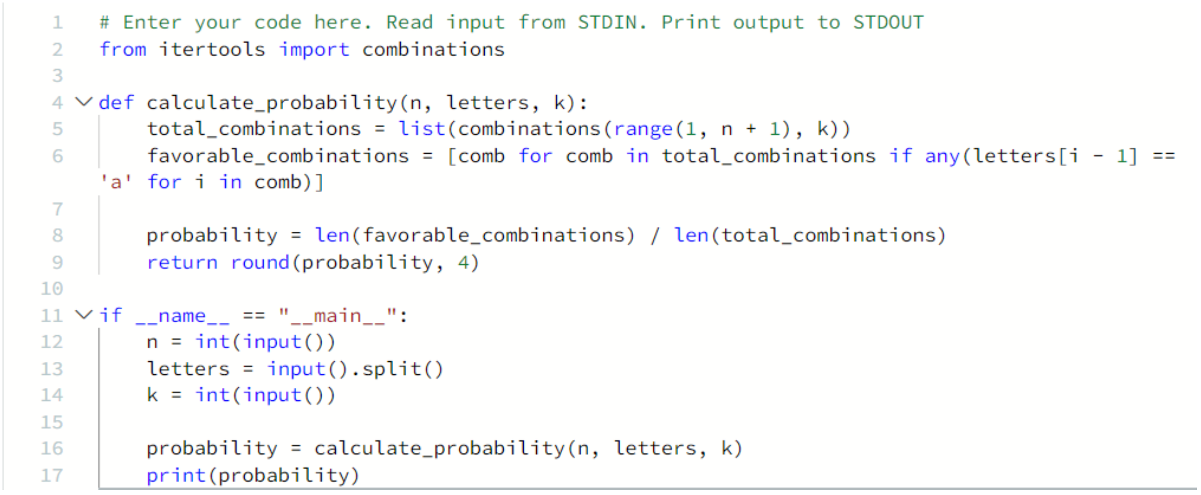
```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
for t in range(int(input())):
    input()
    lst = [int(i) for i in input().split()]
    min_list = lst.index(min(lst))
    left = lst[:min_list]
    right = lst[min_list+1:]
    if left == sorted(left,reverse=True) and right == sorted(right):
        print("Yes")
    else:
        print("No")
```

Triangle Quest 2



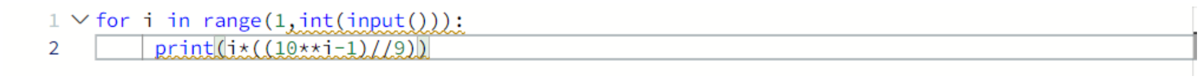
```python
for i in range(1,int(input())+1):
    print (((10**i - 1)//9)**2)
```

Iterables and Iterators



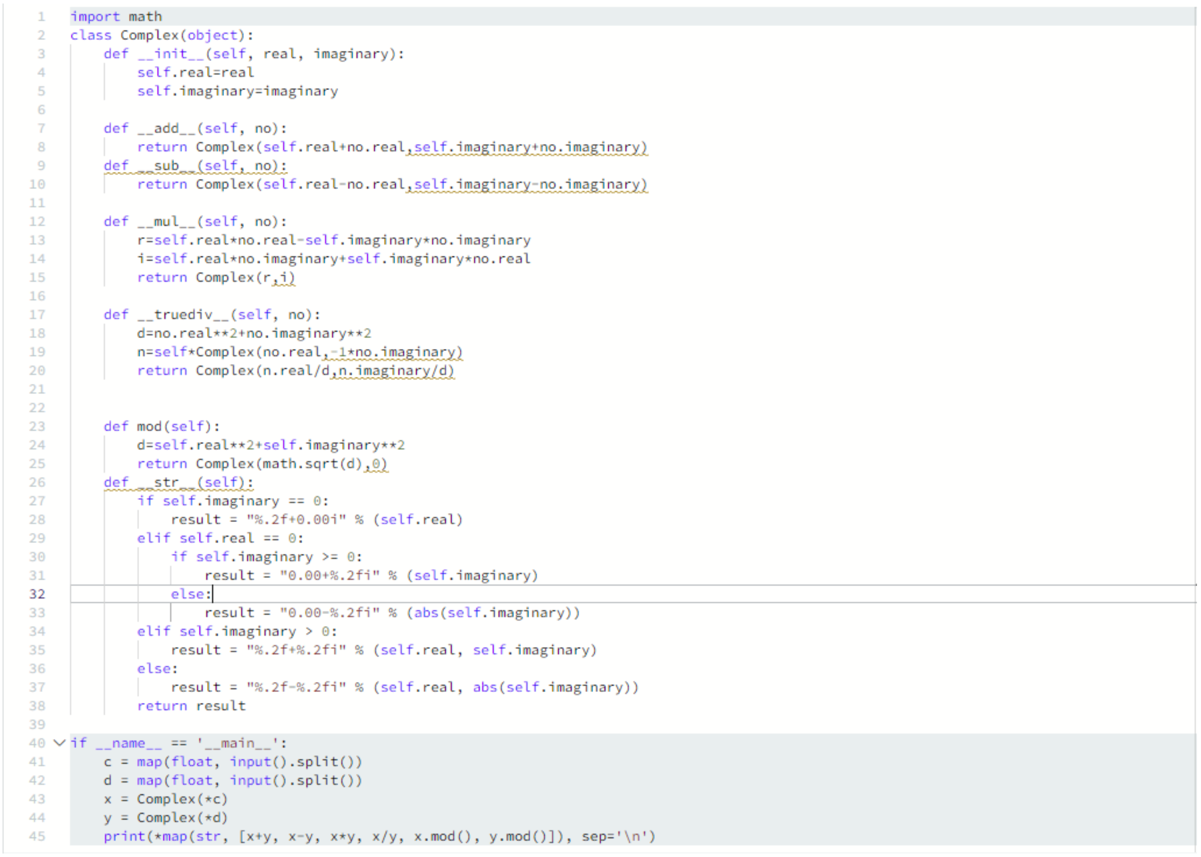
```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import combinations

def calculate_probability(n, letters, k):
    total_combinations = list(combinations(range(1, n + 1), k))
    favorable_combinations = [comb for comb in total_combinations if any(letters[i - 1] == 'a' for i in comb)]

    probability = len(favorable_combinations) / len(total_combinations)
    return round(probability, 4)

if __name__ == "__main__":
    n = int(input())
    letters = input().split()
    k = int(input())

    probability = calculate_probability(n, letters, k)
    print(probability)
```

Triangle Quest



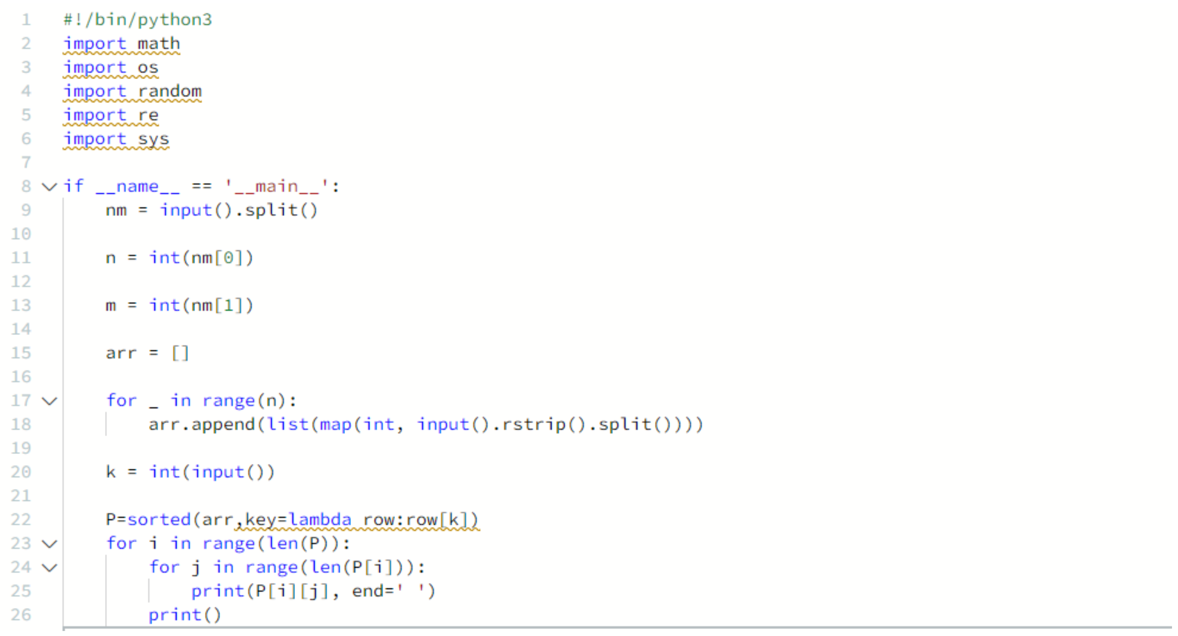
```python
for i in range(1,int(input())):
    print(i*((10**i-1)//9))
```

Classes: Dealing with Complex Numbers



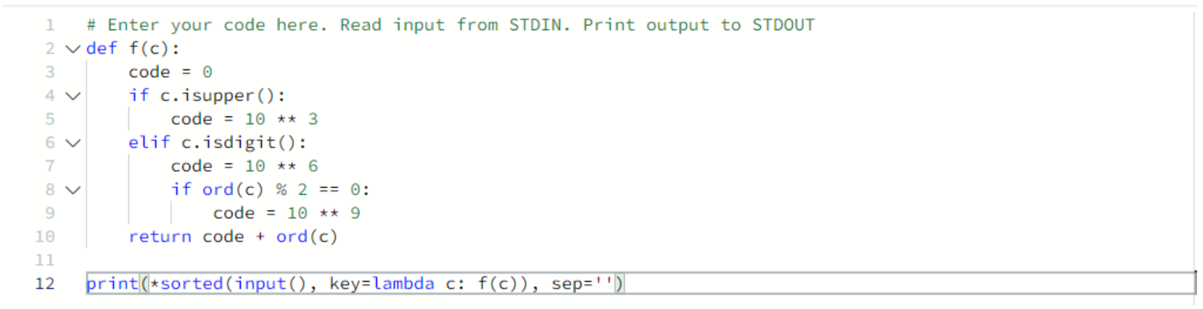
```python
import math
class Complex(object):
    def __init__(self, real, imaginary):
        self.real=real
        self.imaginary=imaginary

    def __add__(self, no):
        return Complex(self.real+no.real,self.imaginary+no.imaginary)
    def __sub__(self, no):
        return Complex(self.real-no.real,self.imaginary-no.imaginary)

    def __mul__(self, no):
        r=self.real*no.real-self.imaginary*no.imaginary
        i=self.real*no.imaginary+self.imaginary*no.real
        return Complex(r,i)

    def __truediv__(self, no):
        d=no.real**2+no.imaginary**2
        n=self*Complex(no.real,-1*no.imaginary)
        return Complex(n.real/d,n.imaginary/d)


    def mod(self):
        d=self.real**2+self.imaginary**2
        return Complex(math.sqrt(d),0)
    def __str__(self):
        if self.imaginary == 0:
            result = "%.2f+0.00i" % (self.real)
        elif self.real == 0:
            if self.imaginary >= 0:
                result = "0.00+%.2fi" % (self.imaginary)
            else:
                result = "0.00-%.2fi" % (abs(self.imaginary))
        elif self.imaginary > 0:
            result = "%.2f+%.2fi" % (self.real, self.imaginary)
        else:
            result = "%.2f-%.2fi" % (self.real, abs(self.imaginary))
        return result

if __name__ == '__main__':
    c = map(float, input().split())
    d = map(float, input().split())
    x = Complex(*c)
    y = Complex(*d)
    print(*map(str, [x+y, x-y, x*y, x/y, x.mod(), y.mod()]), sep='\n')
```

Athlete Sort



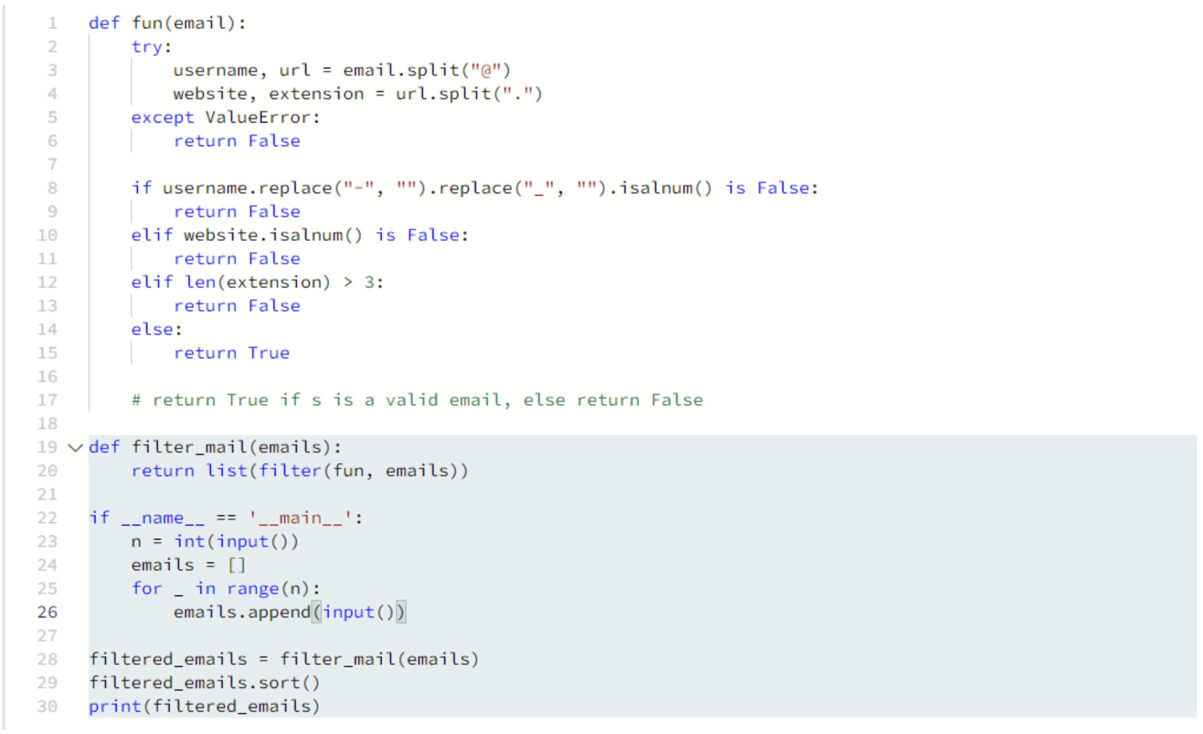
```python
#!/bin/python3
import math
import os
import random
import re
import sys

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())

    P=sorted(arr,key=lambda row:row[k])
    for i in range(len(P)):
        for j in range(len(P[i])):
            print(P[i][j], end=' ')
        print()
```

Ginorts



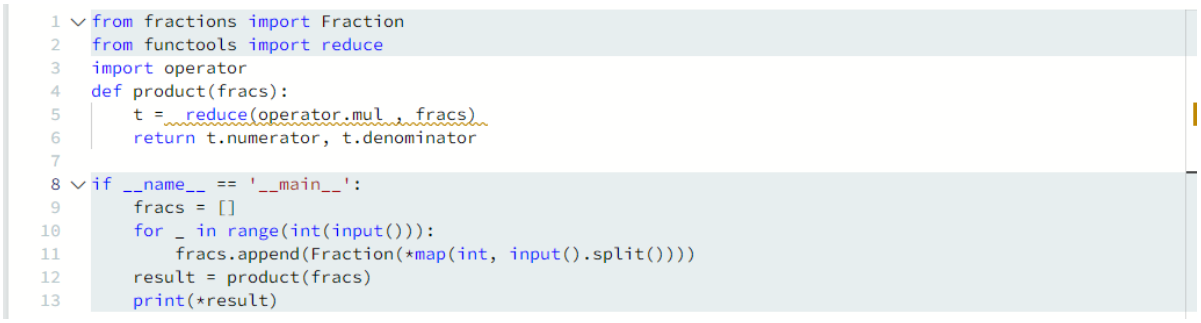
```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
def f(c):
    code = 0
    if c.isupper():
        code = 10 ** 3
    elif c.isdigit():
        code = 10 ** 6
        if ord(c) % 2 == 0:
            code = 10 ** 9
    return code + ord(c)

print(*sorted(input(), key=lambda c: f(c)), sep='')
```

Validating email address with a filter



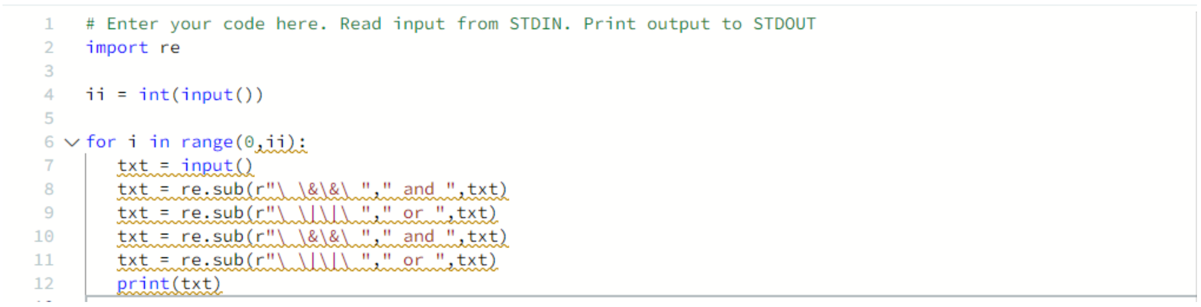
```python
def fun(email):
    try:
        username, url = email.split("@")
        website, extension = url.split(".")
    except ValueError:
        return False

    if username.replace("-", "").replace("_", "").isalnum() is False:
        return False
    elif website.isalnum() is False:
        return False
    elif len(extension) > 3:
        return False
    else:
        return True

    # return True if s is a valid email, else return False

def filter_mail(emails):
    return list(filter(fun, emails))

if __name__ == '__main__':
    n = int(input())
    emails = []
    for _ in range(n):
        emails.append(input())

filtered_emails = filter_mail(emails)
filtered_emails.sort()
print(filtered_emails)
```

Reduce Function



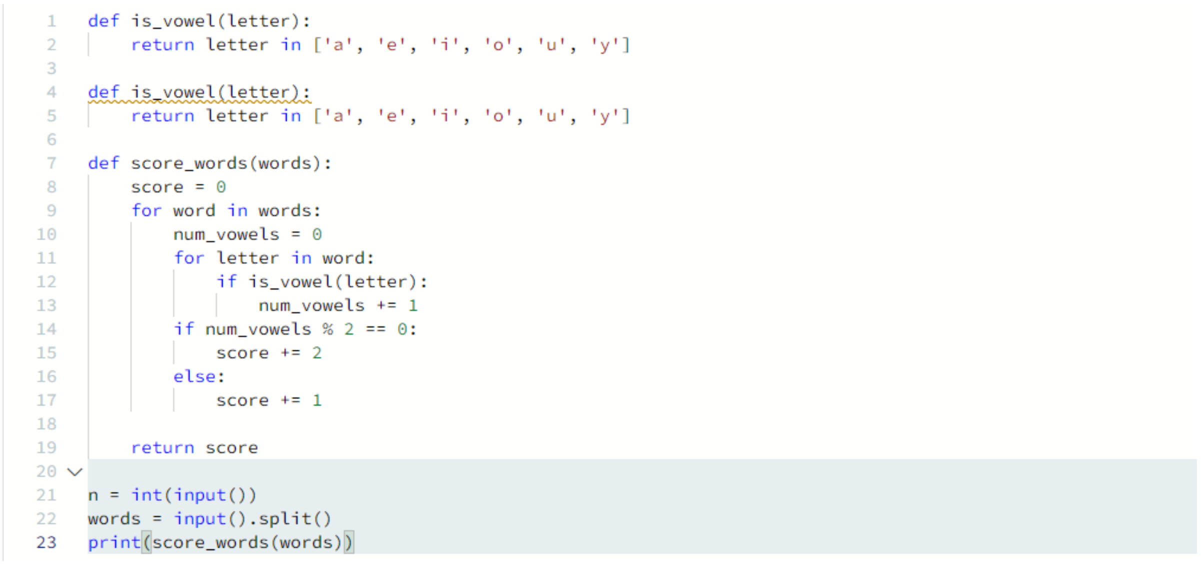
```python
from fractions import Fraction
from functools import reduce
import operator
def product(fracs):
    t =  reduce(operator.mul , fracs)
    return t.numerator, t.denominator

if __name__ == '__main__':
    fracs = []
    for _ in range(int(input())):
        fracs.append(Fraction(*map(int, input().split())))
    result = product(fracs)
    print(*result)
```

Regex Substitution



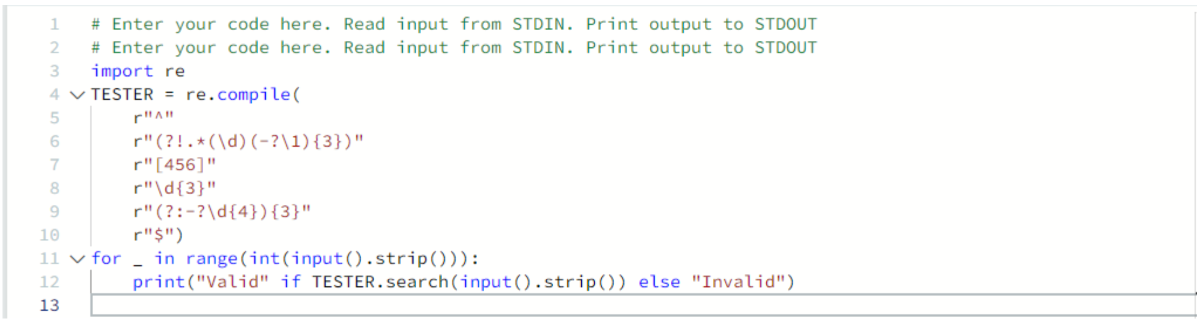
```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re

ii = int(input())

for i in range(0,ii):
    txt = input()
    txt = re.sub(r"\ \&\&\ "," and ",txt)
    txt = re.sub(r"\ \|\|\ "," or ",txt)
    txt = re.sub(r"\ \&\&\ "," and ",txt)
    txt = re.sub(r"\ \|\|\ "," or ",txt)
    print(txt)
```

Words Score



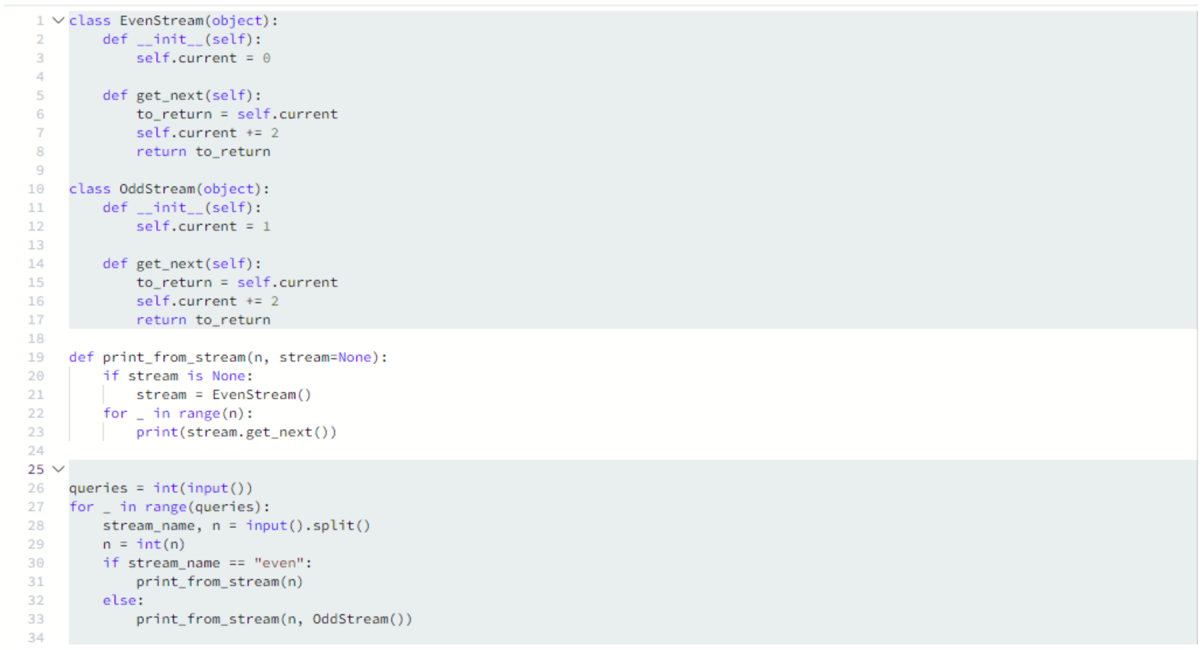
```python
def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u', 'y']

def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u', 'y']

def score_words(words):
    score = 0
    for word in words:
        num_vowels = 0
        for letter in word:
            if is_vowel(letter):
                num_vowels += 1
        if num_vowels % 2 == 0:
            score += 2
        else:
            score += 1

    return score

n = int(input())
words = input().split()
print(score_words(words))
```

Validating Credit Card Numbers



```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
# Enter your code here. Read input from STDIN. Print output to STDOUT
import re
TESTER = re.compile(
    r"^"
    r"(?!.*(\d)(-?\1){3})"
    r"[456]"
    r"\d{3}"
    r"(?:-?\d{4}){3}"
    r"$")
for _ in range(int(input().strip())):
    print("Valid" if TESTER.search(input().strip()) else "Invalid")

```

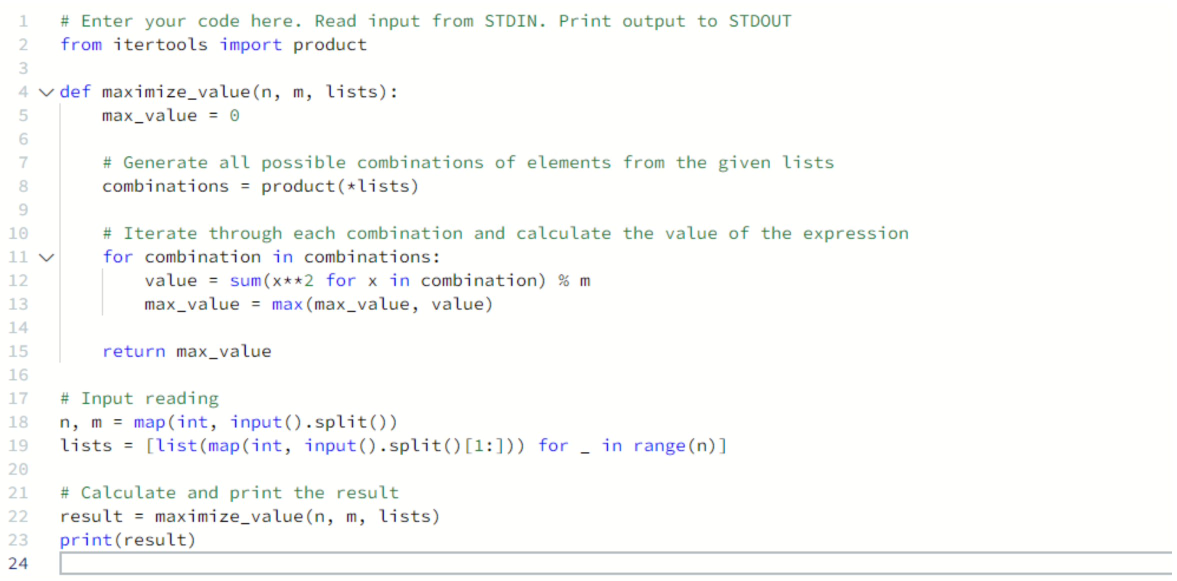
Default Arguments



```python
class EvenStream(object):
    def __init__(self):
        self.current = 0

    def get_next(self):
        to_return = self.current
        self.current += 2
        return to_return

class OddStream(object):
    def __init__(self):
        self.current = 1

    def get_next(self):
        to_return = self.current
        self.current += 2
        return to_return

def print_from_stream(n, stream=None):
    if stream is None:
        stream = EvenStream()
    for _ in range(n):
        print(stream.get_next())


queries = int(input())
for _ in range(queries):
    stream_name, n = input().split()
    n = int(n)
    if stream_name == "even":
        print_from_stream(n)
    else:
        print_from_stream(n, OddStream())

```

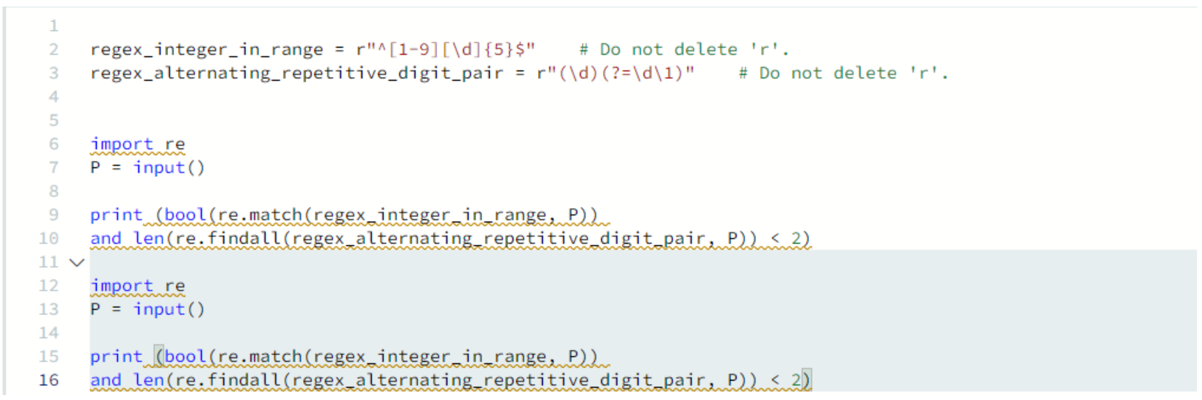
Hard Level Tasks

Maximize It!



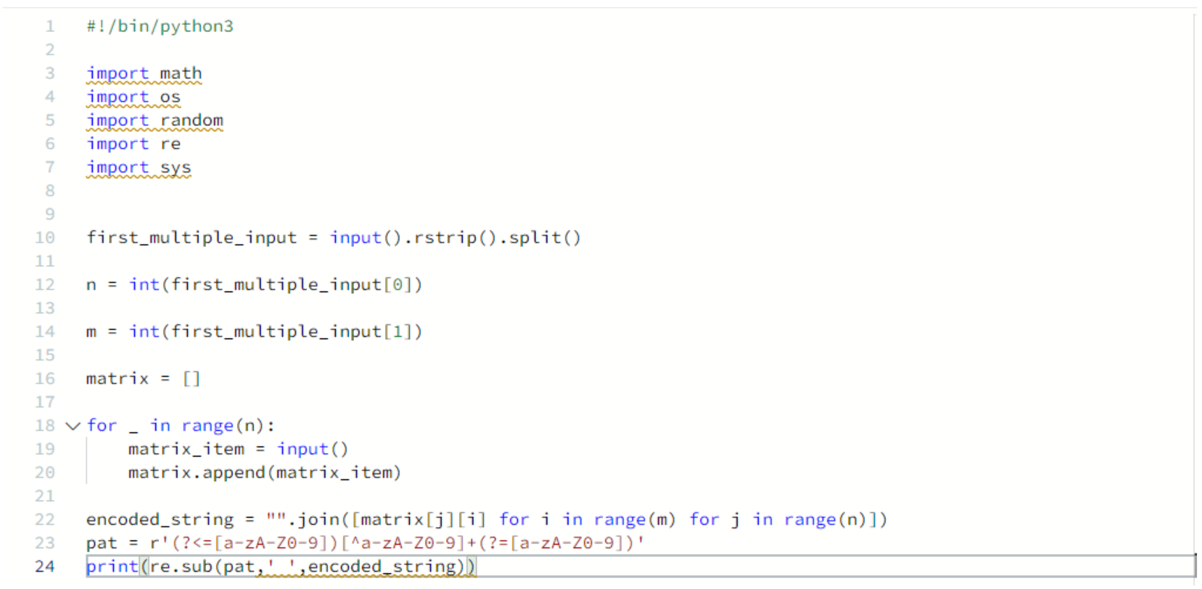
```python
# Enter your code here. Read input from STDIN. Print output to STDOUT
from itertools import product

def maximize_value(n, m, lists):
    max_value = 0

    # Generate all possible combinations of elements from the given lists
    combinations = product(*lists)

    # Iterate through each combination and calculate the value of the expression
    for combination in combinations:
        value = sum(x**2 for x in combination) % m
        max_value = max(max_value, value)

    return max_value

# Input reading
n, m = map(int, input().split())
lists = [list(map(int, input().split()[1:])) for _ in range(n)]

# Calculate and print the result
result = maximize_value(n, m, lists)
print(result)

```

Validating Postal Codes



```python

regex_integer_in_range = r"^[1-9][\d]{5}$"    # Do not delete 'r'.
regex_alternating_repetitive_digit_pair = r"(\d)(?=\d\1)"    # Do not delete 'r'.


import re
P = input()

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

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



```python
#!/bin/python3

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)

encoded_string = "".join([matrix[j][i] for i in range(m) for j in range(n)])
pat = r'(?<=[a-zA-Z0-9])[^a-zA-Z0-9]+(?=[a-zA-Z0-9])'
print(re.sub(pat,' ',encoded_string))
```