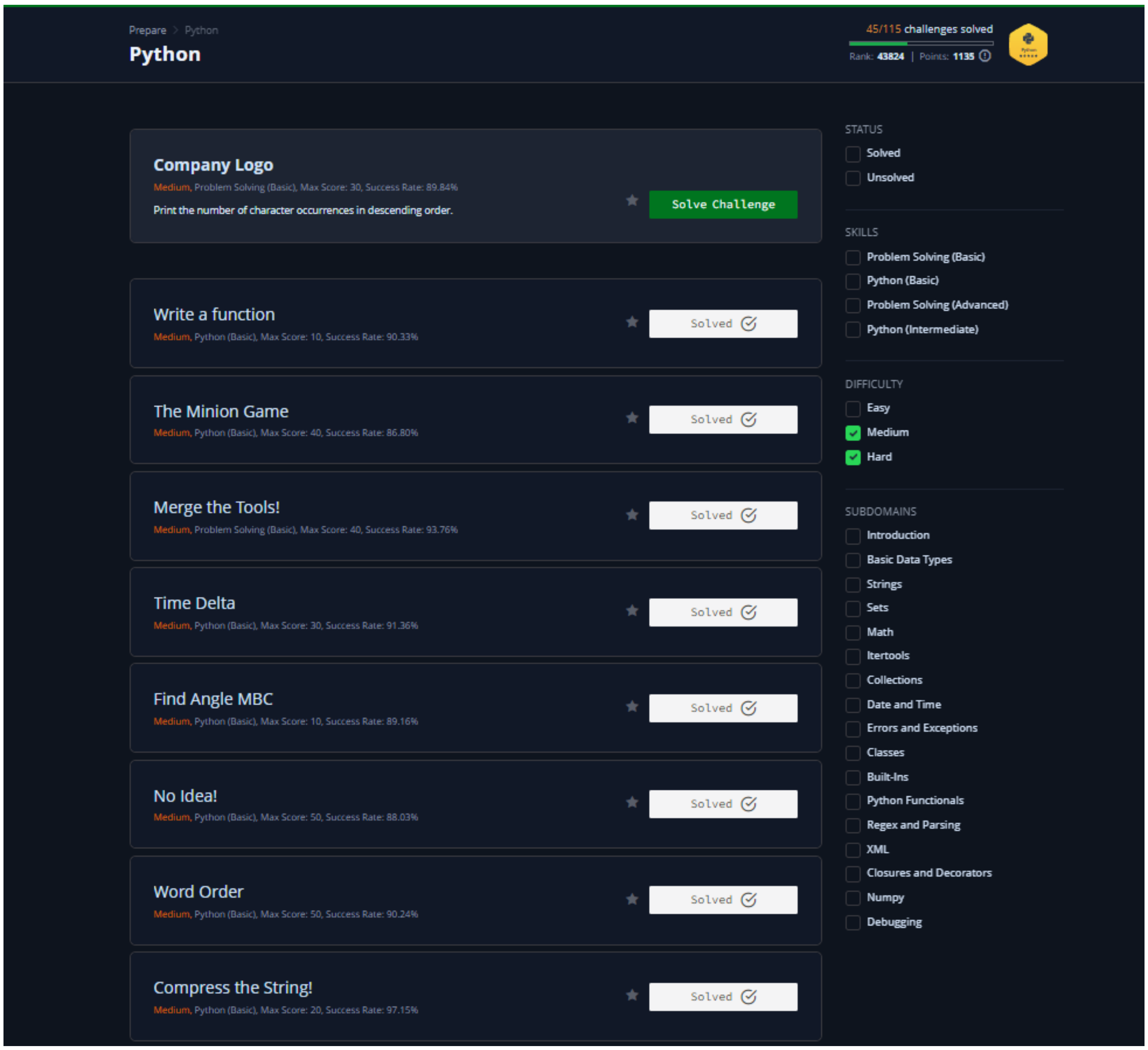
AI ASSIGNMENT

(MEDIUM AND HARD LEVEL TASKS AT HACKERRANK)



Prepare > Python

Python

45/115 challenges solved

Rank: 43824 | Points: 1135

Company Logo
Medium, Problem Solving (Basic), Max Score: 30, Success Rate: 89.84%
Print the number of character occurrences in descending order.

Write a function
Medium, Python (Basic), Max Score: 10, Success Rate: 90.33%
Solved

The Minion Game
Medium, Python (Basic), Max Score: 40, Success Rate: 86.80%
Solved

Merge the Tools!
Medium, Problem Solving (Basic), Max Score: 40, Success Rate: 93.76%
Solved

Time Delta
Medium, Python (Basic), Max Score: 30, Success Rate: 91.36%
Solved

Find Angle MBC
Medium, Python (Basic), Max Score: 10, Success Rate: 89.16%
Solved

No Idea!
Medium, Python (Basic), Max Score: 50, Success Rate: 88.03%
Solved

Word Order
Medium, Python (Basic), Max Score: 50, Success Rate: 90.24%
Solved

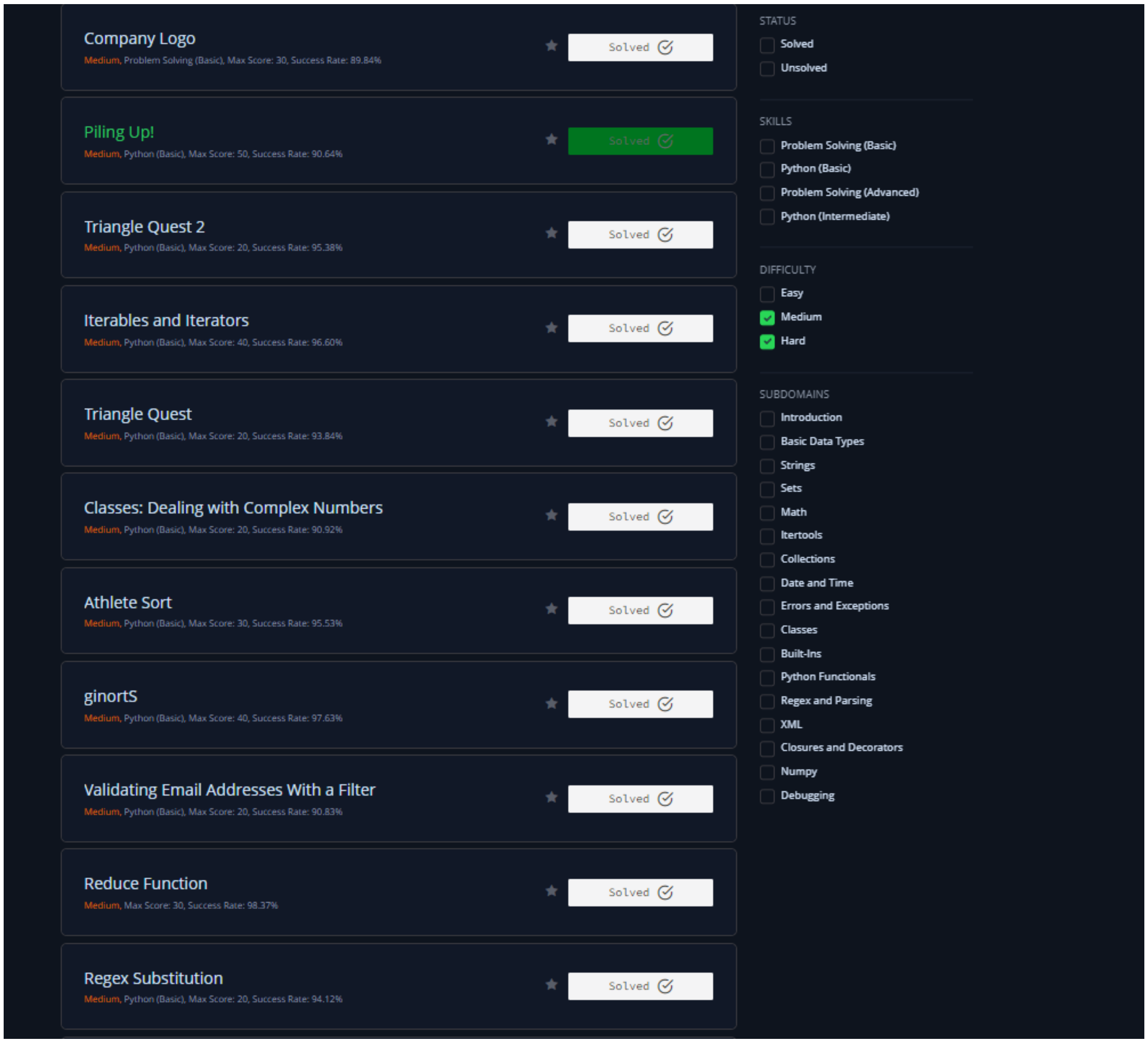
Compress the String!
Medium, Python (Basic), Max Score: 20, Success Rate: 97.15%
Solved

STATUS
- [ ] Solved
- [ ] Unsolved

SKILLS
- [ ] Problem Solving (Basic)
- [ ] Python (Basic)
- [ ] Problem Solving (Advanced)
- [ ] Python (Intermediate)

DIFFICULTY
- [ ] Easy
- [✓] Medium
- [✓] Hard

SUBDOMAINS
- [ ] Introduction
- [ ] Basic Data Types
- [ ] Strings
- [ ] Sets
- [ ] Math
- [ ] Itertools
- [ ] Collections
- [ ] Date and Time
- [ ] Errors and Exceptions
- [ ] Classes
- [ ] Built-Ins
- [ ] Python Functionals
- [ ] Regex and Parsing
- [ ] XML
- [ ] Closures and Decorators
- [ ] Numpy
- [ ] Debugging



Company Logo
Medium, Problem Solving (Basic), Max Score: 30, Success Rate: 89.84%
Solved

Piling Up!
Medium, Python (Basic), Max Score: 50, Success Rate: 90.64%
Solved

Triangle Quest 2
Medium, Python (Basic), Max Score: 20, Success Rate: 95.38%
Solved

Iterables and Iterators
Medium, Python (Basic), Max Score: 40, Success Rate: 96.60%
Solved

Triangle Quest
Medium, Python (Basic), Max Score: 20, Success Rate: 93.84%
Solved

Classes: Dealing with Complex Numbers
Medium, Python (Basic), Max Score: 20, Success Rate: 90.92%
Solved

Athlete Sort
Medium, Python (Basic), Max Score: 30, Success Rate: 95.53%
Solved

ginortS
Medium, Python (Basic), Max Score: 40, Success Rate: 97.63%
Solved

Validating Email Addresses With a Filter
Medium, Python (Basic), Max Score: 20, Success Rate: 90.83%
Solved

Reduce Function
Medium, Max Score: 30, Success Rate: 98.37%
Solved

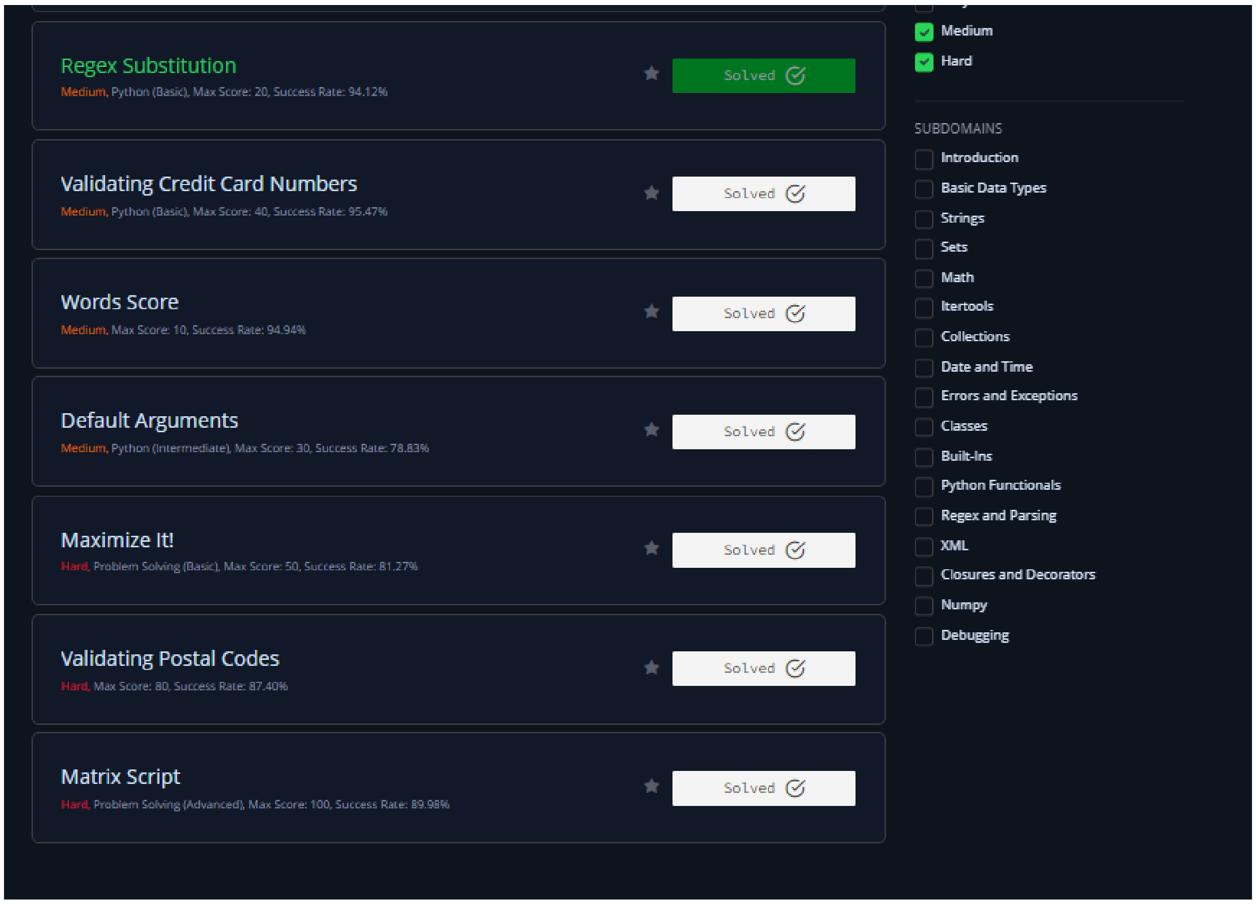
Regex Substitution
Medium, Python (Basic), Max Score: 20, Success Rate: 94.12%
Solved

STATUS
- [ ] Solved
- [ ] Unsolved

SKILLS
- [ ] Problem Solving (Basic)
- [ ] Python (Basic)
- [ ] Problem Solving (Advanced)
- [ ] Python (Intermediate)

DIFFICULTY
- [ ] Easy
- [x] Medium
- [x] Hard

SUBDOMAINS
- [ ] Introduction
- [ ] Basic Data Types
- [ ] Strings
- [ ] Sets
- [ ] Math
- [ ] Itertools
- [ ] Collections
- [ ] Date and Time
- [ ] Errors and Exceptions
- [ ] Classes
- [ ] Built-Ins
- [ ] Python Functionals
- [ ] Regex and Parsing
- [ ] XML
- [ ] Closures and Decorators
- [ ] Numpy
- [ ] Debugging



Regex Substitution
Medium, Python (Basic), Max Score: 20, Success Rate: 94.12%
Solved

Validating Credit Card Numbers
Medium, Python (Basic), Max Score: 40, Success Rate: 95.47%
Solved

Words Score
Medium, Max Score: 10, Success Rate: 94.94%
Solved

Default Arguments
Medium, Python (Intermediate), Max Score: 30, Success Rate: 78.83%
Solved

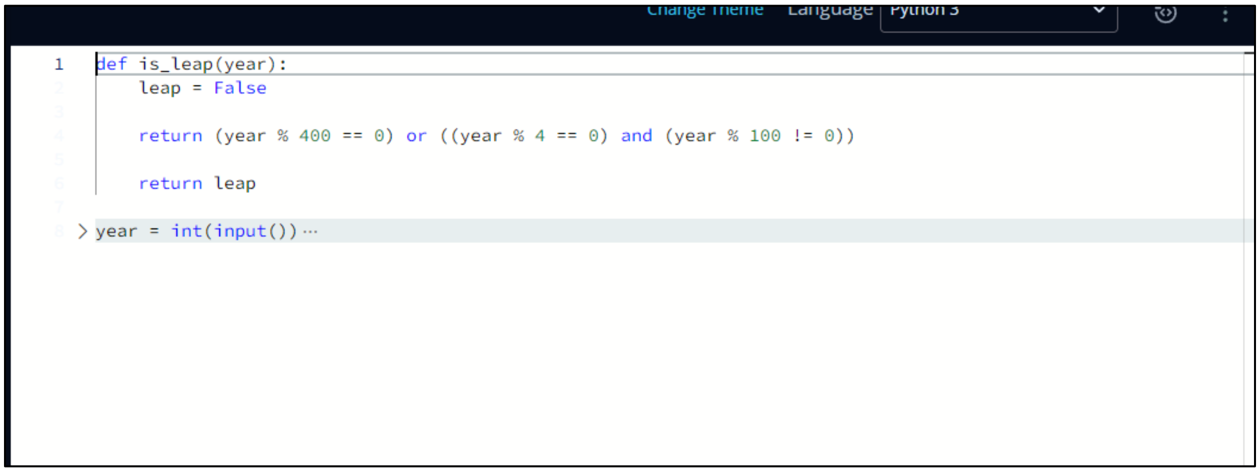
Maximize It!
Hard, Problem Solving (Basic), Max Score: 50, Success Rate: 81.27%
Solved

Validating Postal Codes
Hard, Max Score: 80, Success Rate: 87.40%
Solved

Matrix Script
Hard, Problem Solving (Advanced), Max Score: 100, Success Rate: 89.98%
Solved

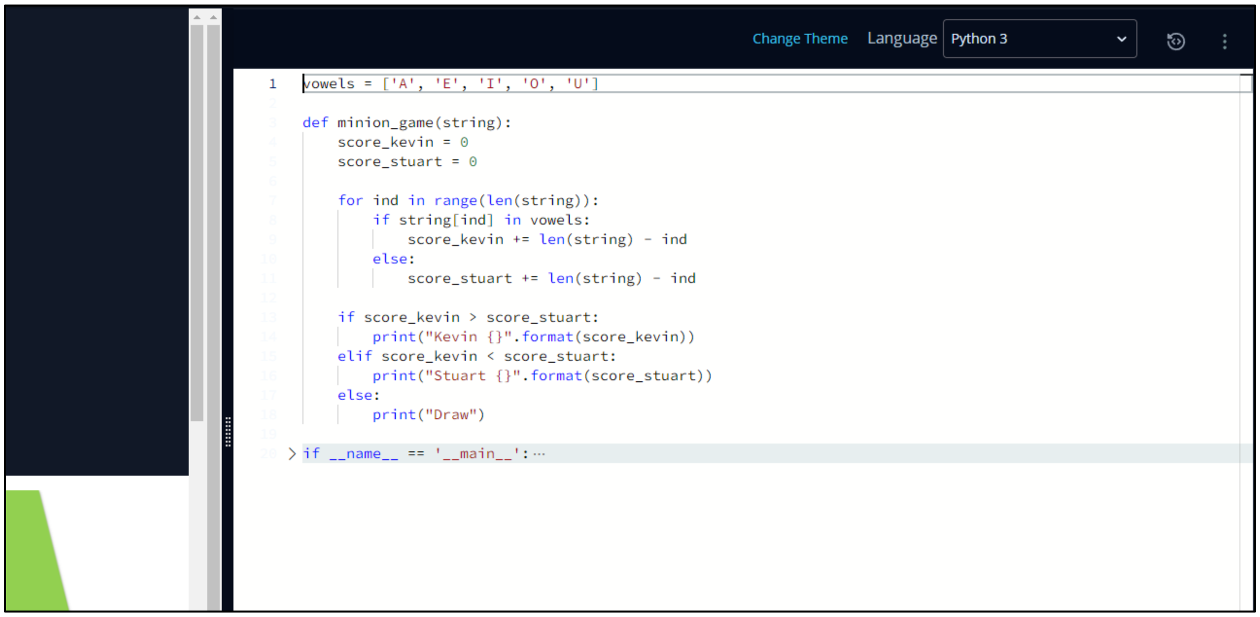
MEDIUM LEVEL TASKS

1. Write a function



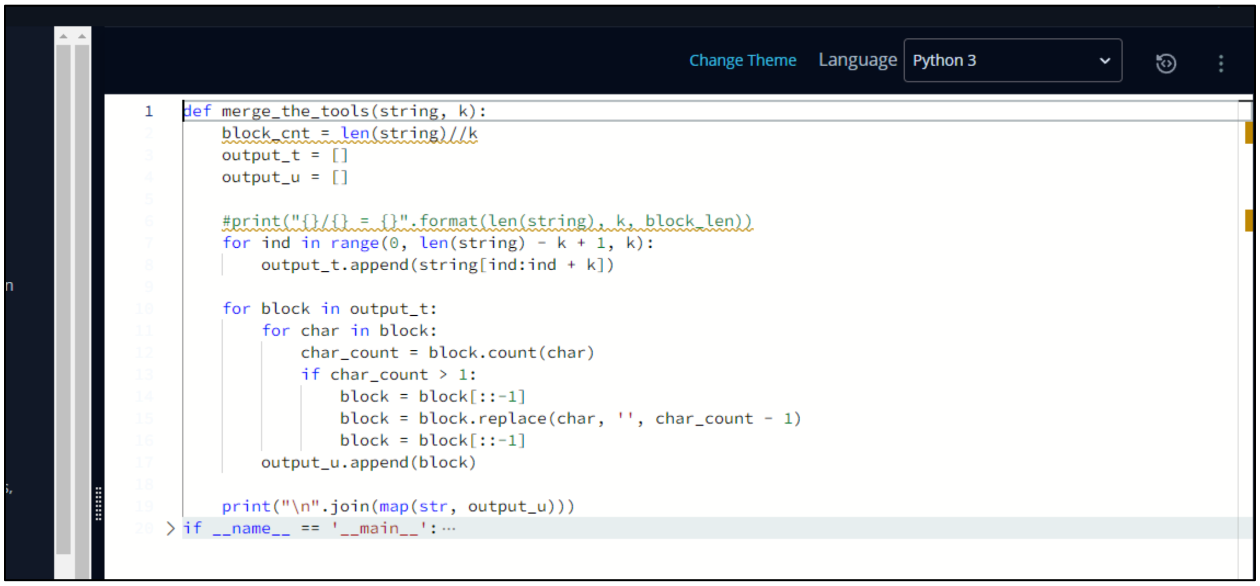
```
def is_leap(year):
    leap = False

    return (year % 400 == 0) or ((year % 4 == 0) and (year % 100 != 0))

    return leap

> year = int(input())···
```

2. The minion game



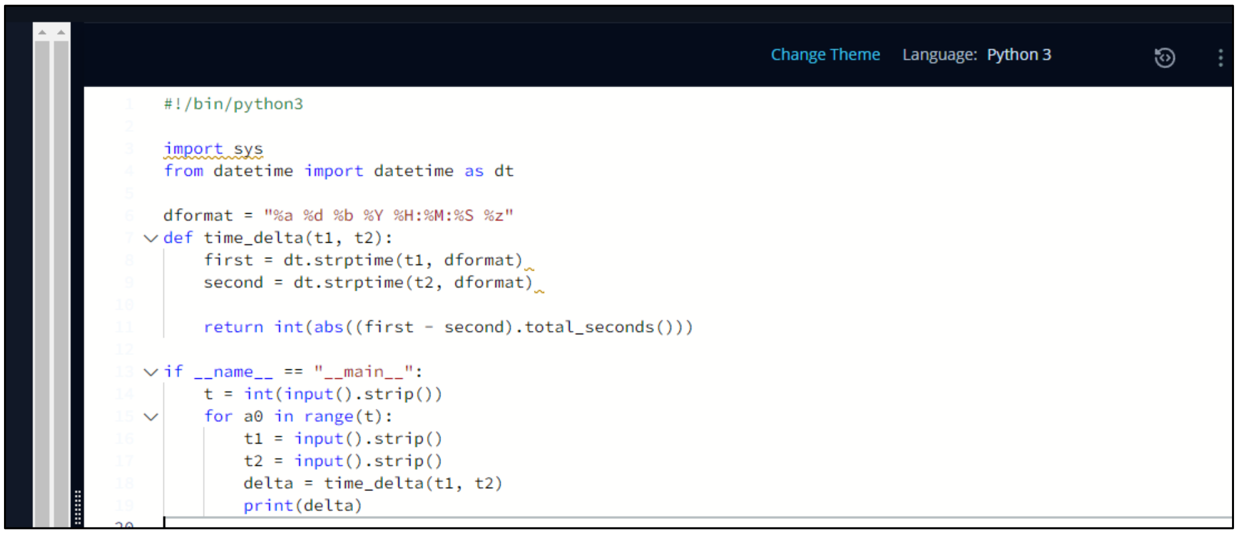
```python
vowels = ['A', 'E', 'I', 'O', 'U']

def minion_game(string):
    score_kevin = 0
    score_stuart = 0

    for ind in range(len(string)):
        if string[ind] in vowels:
            score_kevin += len(string) - ind
        else:
            score_stuart += len(string) - ind

    if score_kevin > score_stuart:
        print("Kevin {}".format(score_kevin))
    elif score_kevin < score_stuart:
        print("Stuart {}".format(score_stuart))
    else:
        print("Draw")

if __name__ == '__main__':…
```

3. Merge the Tool



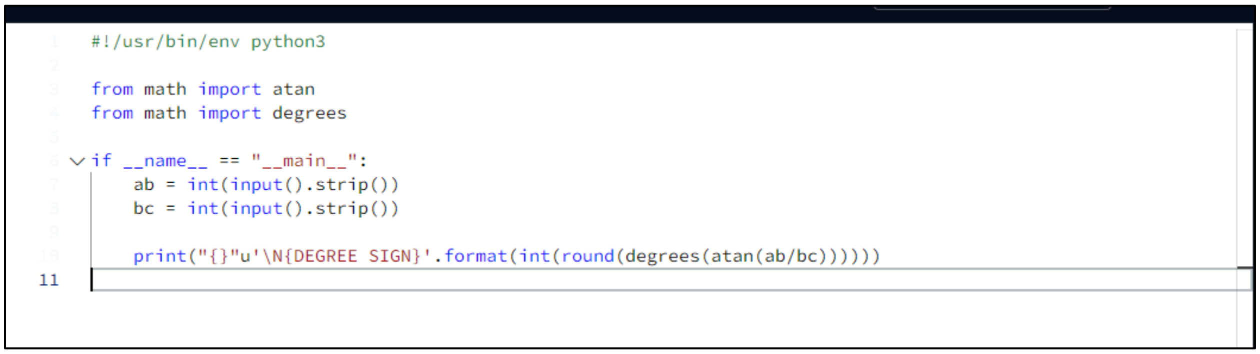
```python
def merge_the_tools(string, k):
    block_cnt = len(string)//k
    output_t = []
    output_u = []

    #print("{}/{} = {}".format(len(string), k, block_len))
    for ind in range(0, len(string) - k + 1, k):
        output_t.append(string[ind:ind + k])

    for block in output_t:
        for char in block:
            char_count = block.count(char)
            if char_count > 1:
                block = block[::-1]
                block = block.replace(char, '', char_count - 1)
                block = block[::-1]
        output_u.append(block)

    print("\n".join(map(str, output_u)))
if __name__ == '__main__':…
```

4. Time Delta



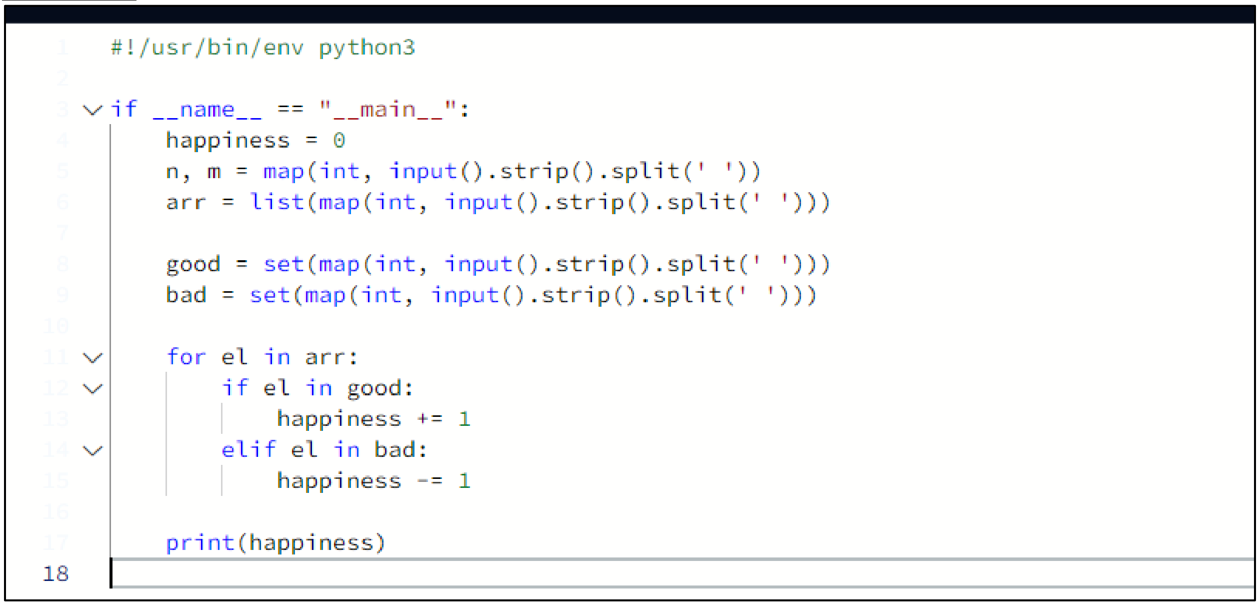
```python
#!/bin/python3

import sys
from datetime import datetime as dt

dformat = "%a %d %b %Y %H:%M:%S %z"
def time_delta(t1, t2):
    first = dt.strptime(t1, dformat)
    second = dt.strptime(t2, dformat)

    return int(abs((first - second).total_seconds()))

if __name__ == "__main__":
    t = int(input().strip())
    for a0 in range(t):
        t1 = input().strip()
        t2 = input().strip()
        delta = time_delta(t1, t2)
        print(delta)
```

5. Find angle MBC



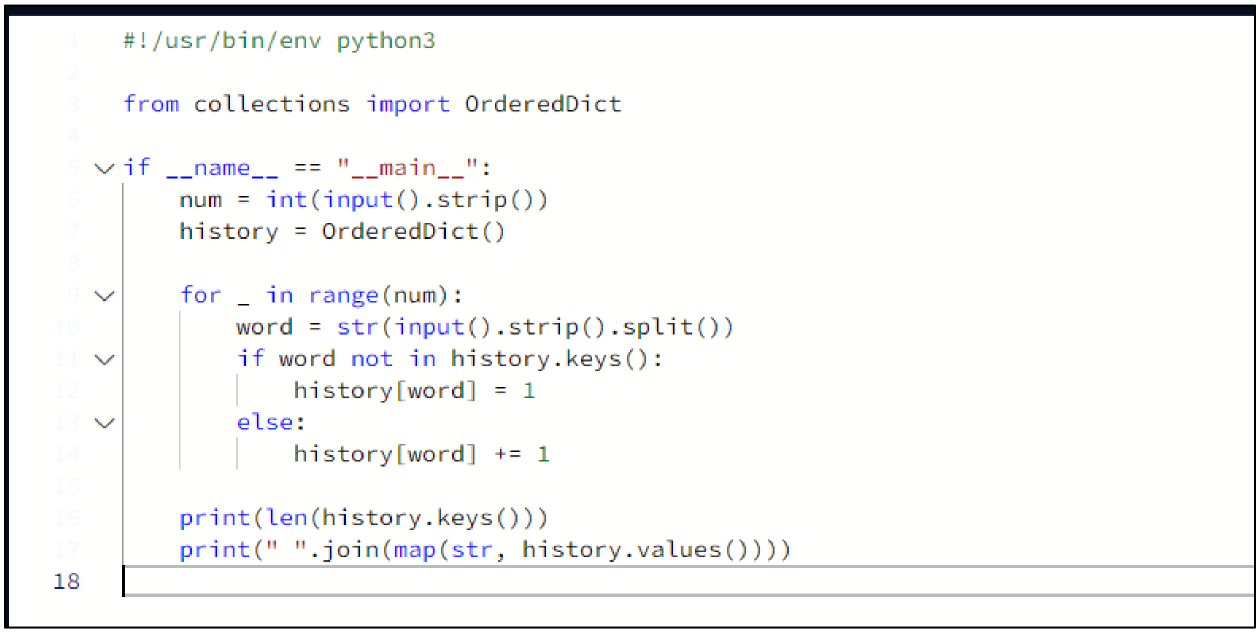
```python
#!/usr/bin/env python3

from math import atan
from math import degrees

if __name__ == "__main__":
    ab = int(input().strip())
    bc = int(input().strip())

    print("{}"u'\N{DEGREE SIGN}'.format(int(round(degrees(atan(ab/bc))))))

```

6. No idea



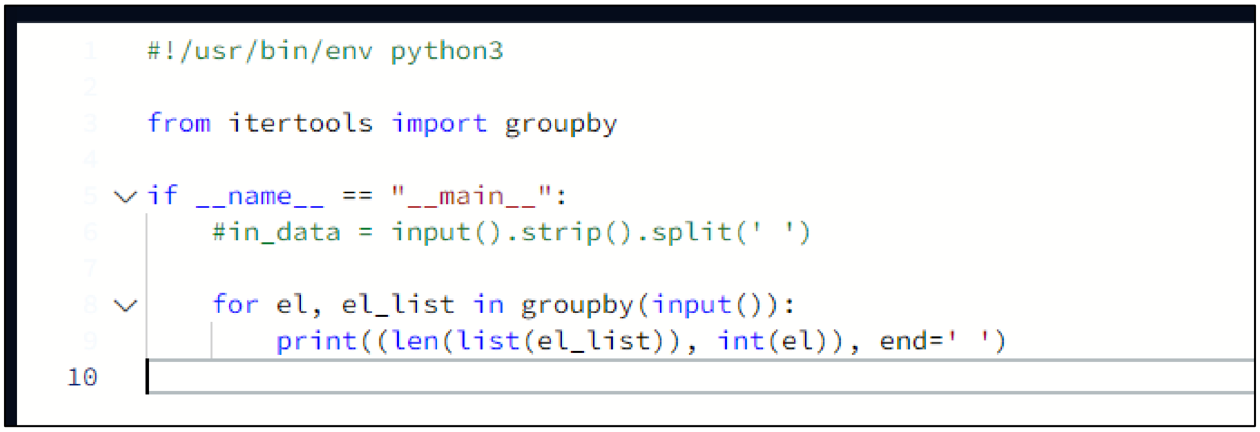
```python
#!/usr/bin/env python3

if __name__ == "__main__":
    happiness = 0
    n, m = map(int, input().strip().split(' '))
    arr = list(map(int, input().strip().split(' ')))

    good = set(map(int, input().strip().split(' ')))
    bad = set(map(int, input().strip().split(' ')))

    for el in arr:
        if el in good:
            happiness += 1
        elif el in bad:
            happiness -= 1

    print(happiness)

```

7. Word order



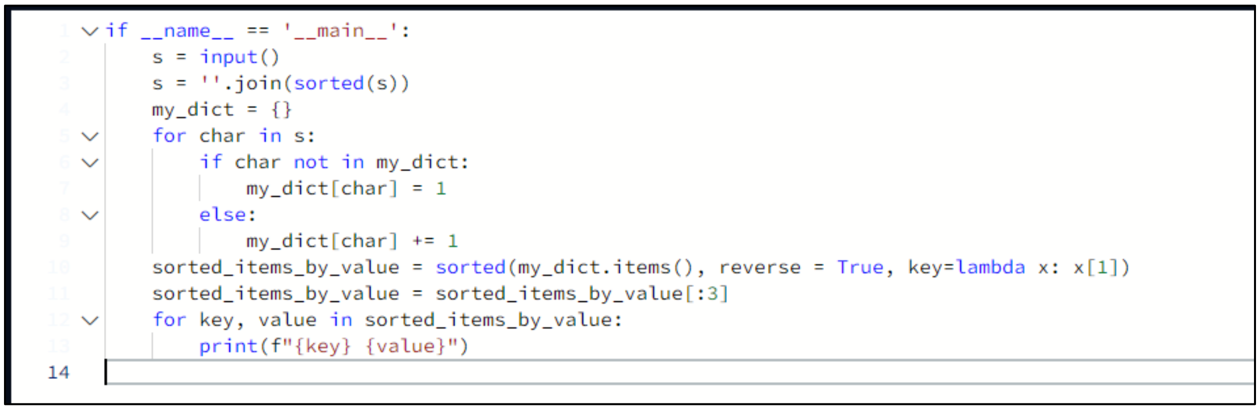
```python
#!/usr/bin/env python3

from collections import OrderedDict

if __name__ == "__main__":
    num = int(input().strip())
    history = OrderedDict()

    for _ in range(num):
        word = str(input().strip().split())
        if word not in history.keys():
            history[word] = 1
        else:
            history[word] += 1

    print(len(history.keys()))
    print(" ".join(map(str, history.values())))
```

8. Compress the string



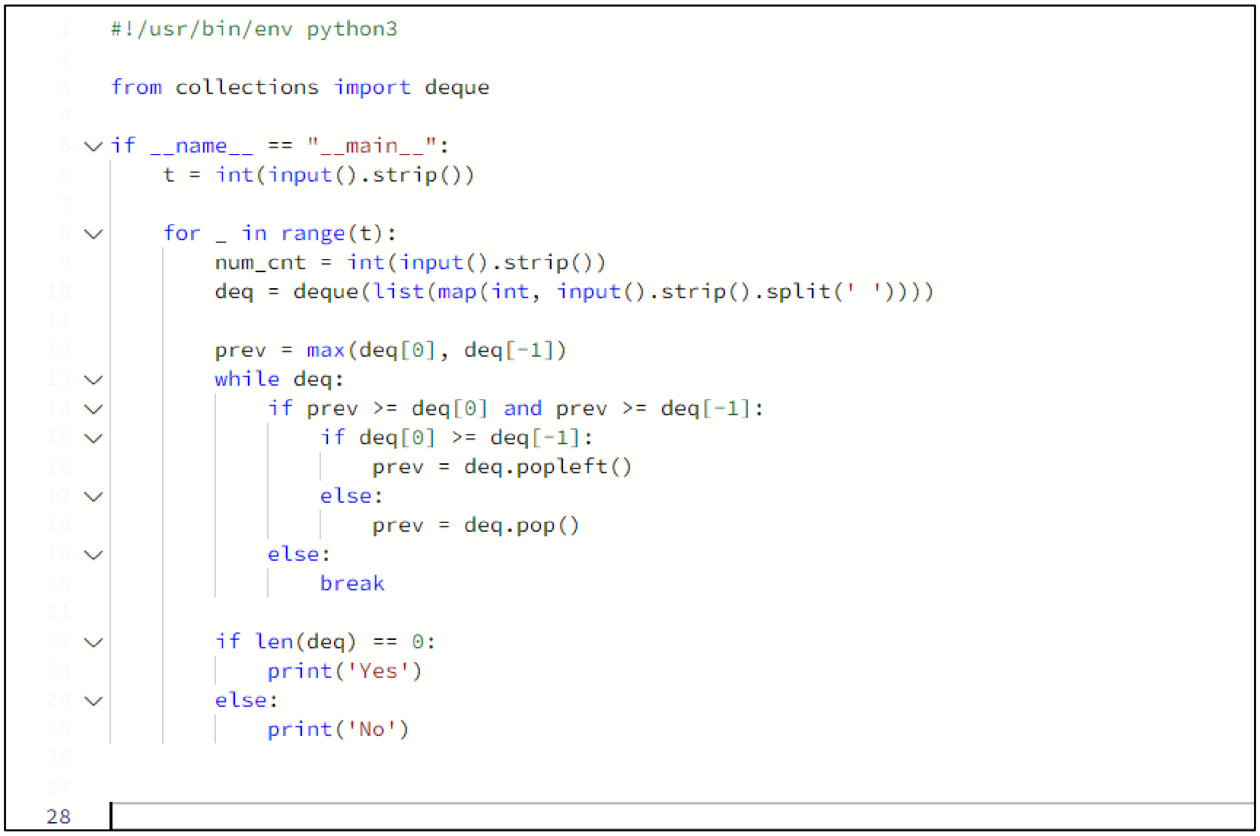
```python
#!/usr/bin/env python3

from itertools import groupby

if __name__ == "__main__":
    #in_data = input().strip().split(' ')

    for el, el_list in groupby(input()):
        print((len(list(el_list)), int(el)), end=' ')
```

9. Company logo



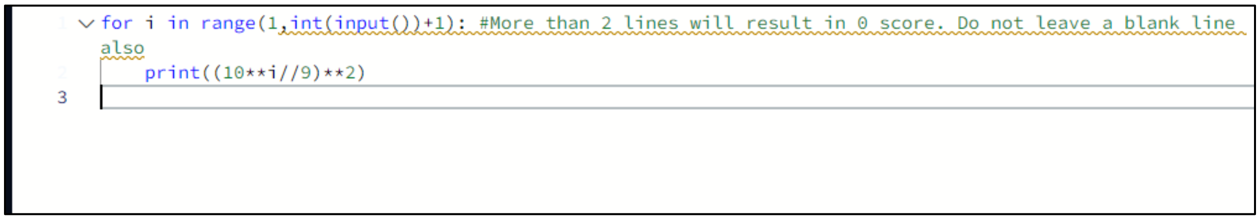
```python
if __name__ == '__main__':
    s = input()
    s = ''.join(sorted(s))
    my_dict = {}
    for char in s:
        if char not in my_dict:
            my_dict[char] = 1
        else:
            my_dict[char] += 1
    sorted_items_by_value = sorted(my_dict.items(), reverse = True, key=lambda x: x[1])
    sorted_items_by_value = sorted_items_by_value[:3]
    for key, value in sorted_items_by_value:
        print(f"{key} {value}")
```

10. Piling up



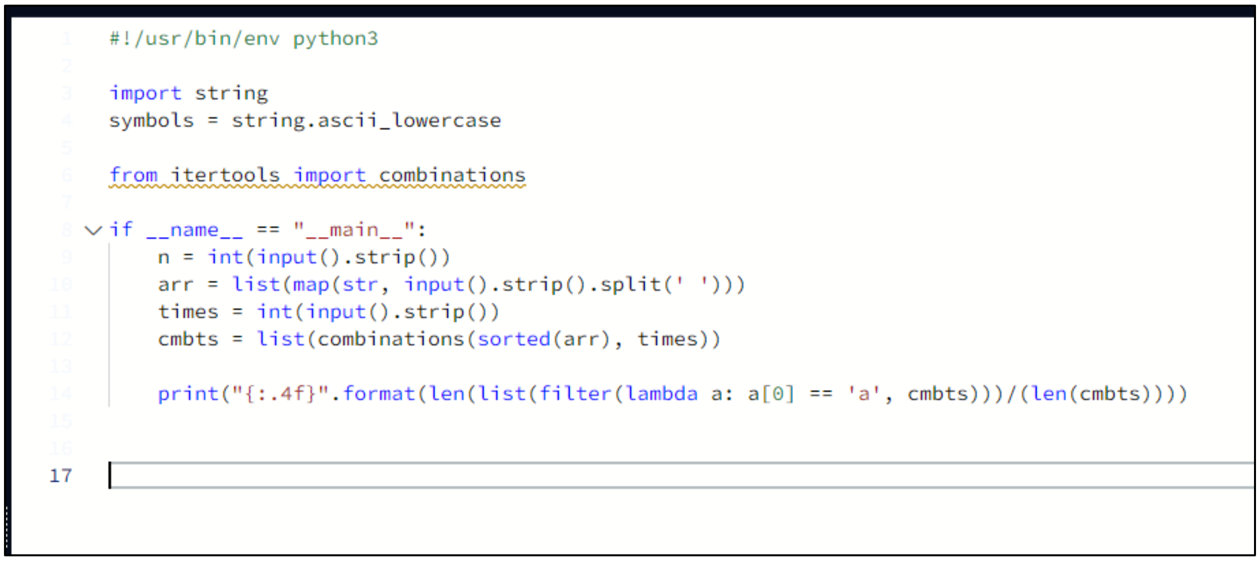
```
#!/usr/bin/env python3

from collections import deque

if __name__ == "__main__":
    t = int(input().strip())

    for _ in range(t):
        num_cnt = int(input().strip())
        deq = deque(list(map(int, input().strip().split(' '))))

        prev = max(deq[0], deq[-1])
        while deq:
            if prev >= deq[0] and prev >= deq[-1]:
                if deq[0] >= deq[-1]:
                    prev = deq.popleft()
                else:
                    prev = deq.pop()
            else:
                break

        if len(deq) == 0:
            print('Yes')
        else:
            print('No')
```

11. Triangular quest 2



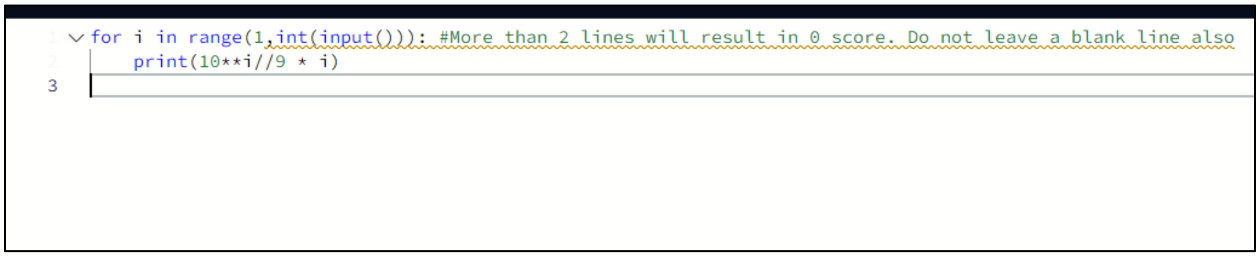
```
for i in range(1,int(input())+1): #More than 2 lines will result in 0 score. Do not leave a blank line also
    print((10**i//9)**2)
```

12. Iterables & Iterators



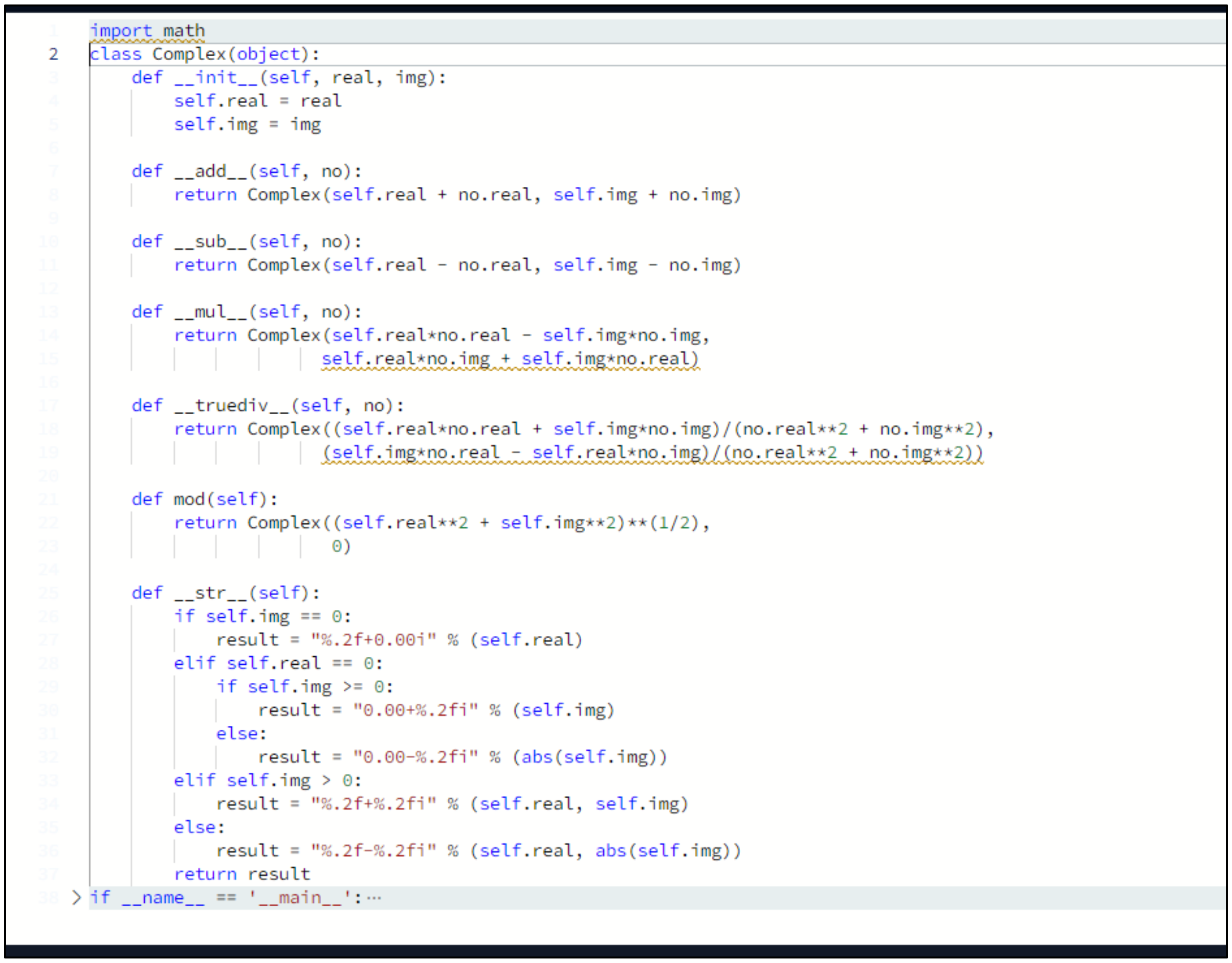
```
#!/usr/bin/env python3

import string
symbols = string.ascii_lowercase

from itertools import combinations

if __name__ == "__main__":
    n = int(input().strip())
    arr = list(map(str, input().strip().split(' ')))
    times = int(input().strip())
    cmbts = list(combinations(sorted(arr), times))

    print("{:.4f}".format(len(list(filter(lambda a: a[0] == 'a', cmbts)))/(len(cmbts))))
```

13. Triangular quest



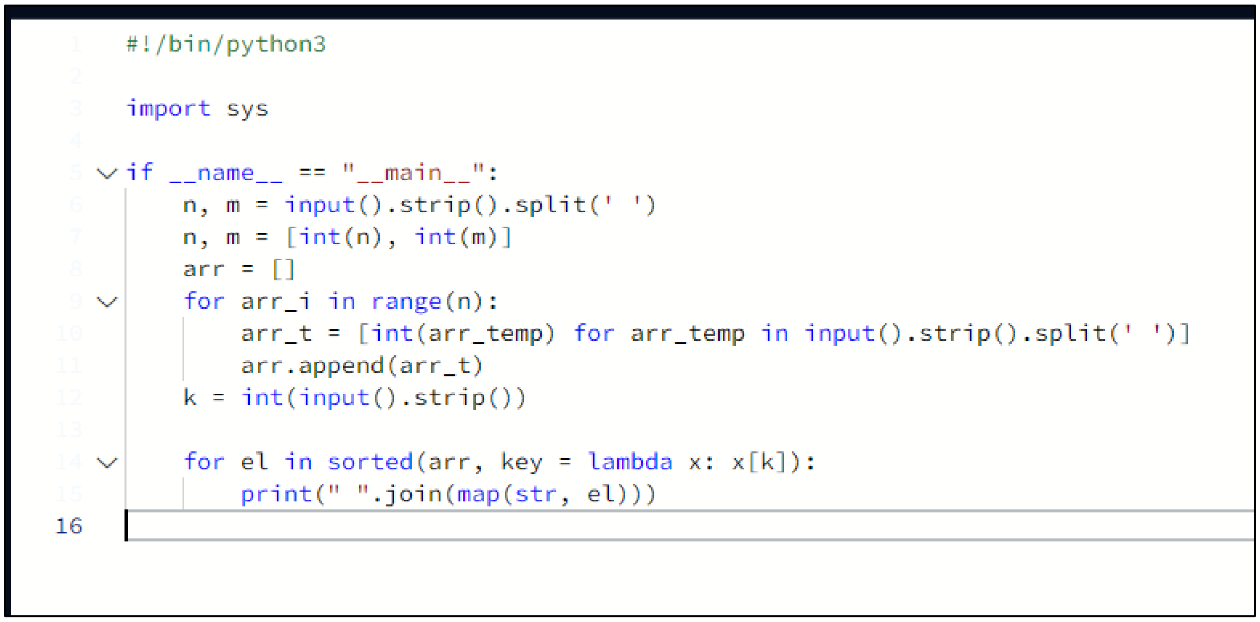
```python
for i in range(1,int(input())): #More than 2 lines will result in 0 score. Do not leave a blank line also
    print(10**i//9 * i)

```

14. Classes: dealing with complex number



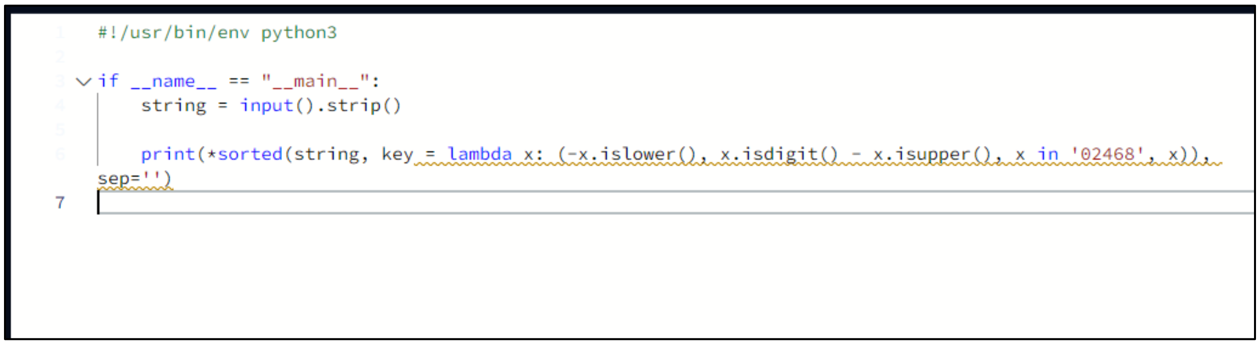
```python
import math
class Complex(object):
    def __init__(self, real, img):
        self.real = real
        self.img = img

    def __add__(self, no):
        return Complex(self.real + no.real, self.img + no.img)

    def __sub__(self, no):
        return Complex(self.real - no.real, self.img - no.img)

    def __mul__(self, no):
        return Complex(self.real*no.real - self.img*no.img,
                       self.real*no.img + self.img*no.real)

    def __truediv__(self, no):
        return Complex((self.real*no.real + self.img*no.img)/(no.real**2 + no.img**2),
                       (self.img*no.real - self.real*no.img)/(no.real**2 + no.img**2))

    def mod(self):
        return Complex((self.real**2 + self.img**2)**(1/2),
                        0)

    def __str__(self):
        if self.img == 0:
            result = "%.2f+0.00i" % (self.real)
        elif self.real == 0:
            if self.img >= 0:
                result = "0.00+%.2fi" % (self.img)
            else:
                result = "0.00-%.2fi" % (abs(self.img))
        elif self.img > 0:
            result = "%.2f+%.2fi" % (self.real, self.img)
        else:
            result = "%.2f-%.2fi" % (self.real, abs(self.img))
        return result
if __name__ == '__main__':···
```

15. Athelete sort



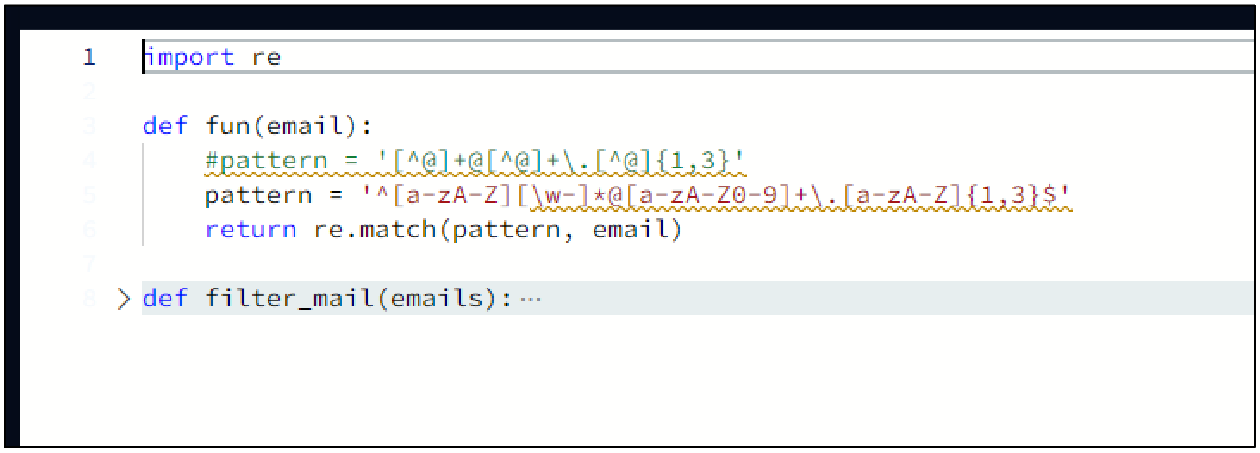
```
#!/bin/python3

import sys

if __name__ == "__main__":
    n, m = input().strip().split(' ')
    n, m = [int(n), int(m)]
    arr = []
    for arr_i in range(n):
        arr_t = [int(arr_temp) for arr_temp in input().strip().split(' ')]
        arr.append(arr_t)
    k = int(input().strip())

    for el in sorted(arr, key = lambda x: x[k]):
        print(" ".join(map(str, el)))

```

16. Ginortx



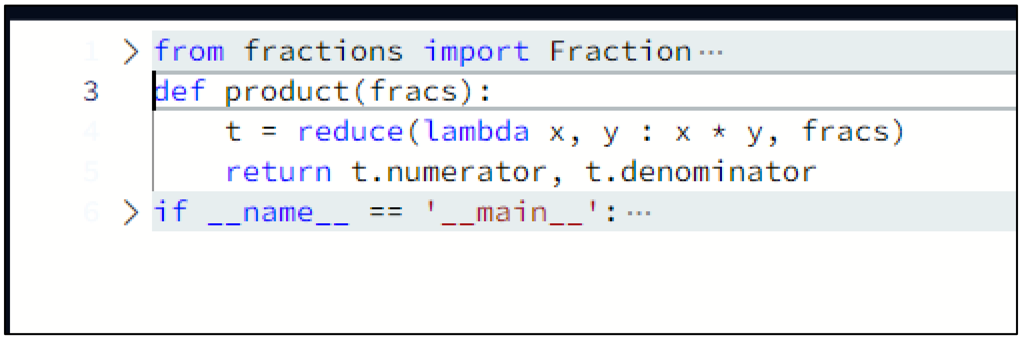
```
#!/usr/bin/env python3

if __name__ == "__main__":
    string = input().strip()

    print(*sorted(string, key = lambda x: (-x.islower(), x.isdigit() - x.isupper(), x in '02468', x)), sep='')

```

17. Validating Email address with a filter



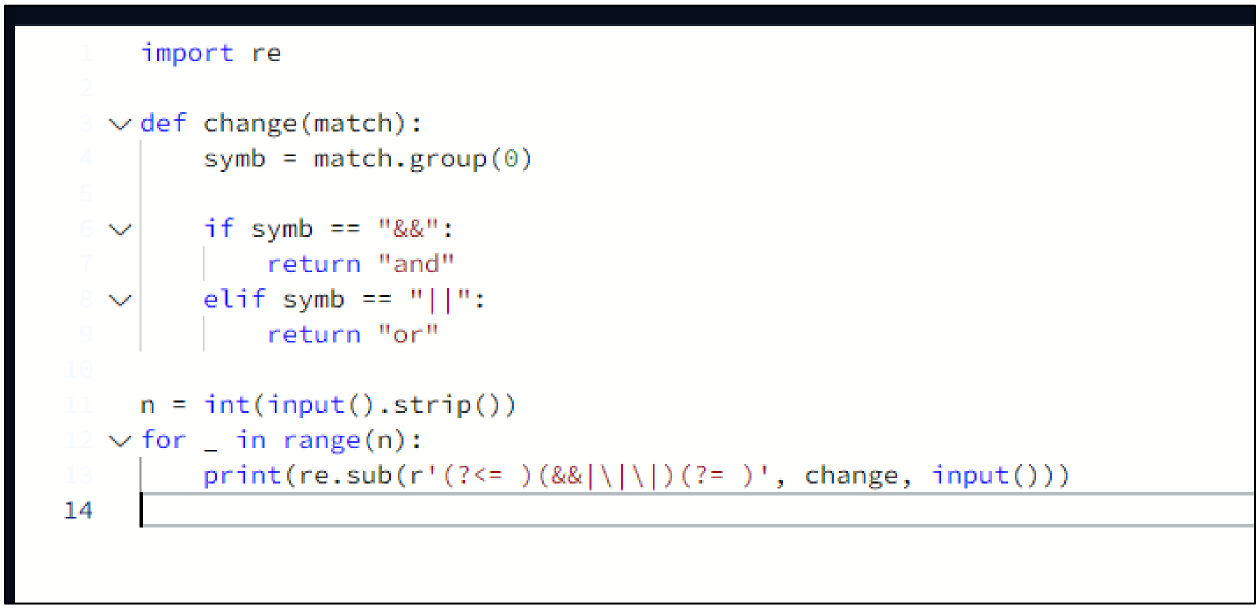
```
import re

def fun(email):
    #pattern = '[^@]+@[^@]+\.[^@]{1,3}'
    pattern = '^[a-zA-Z][\w-]*@[a-zA-Z0-9]+\.[a-zA-Z]{1,3}$'
    return re.match(pattern, email)

def filter_mail(emails):···
```

18. Reduce function



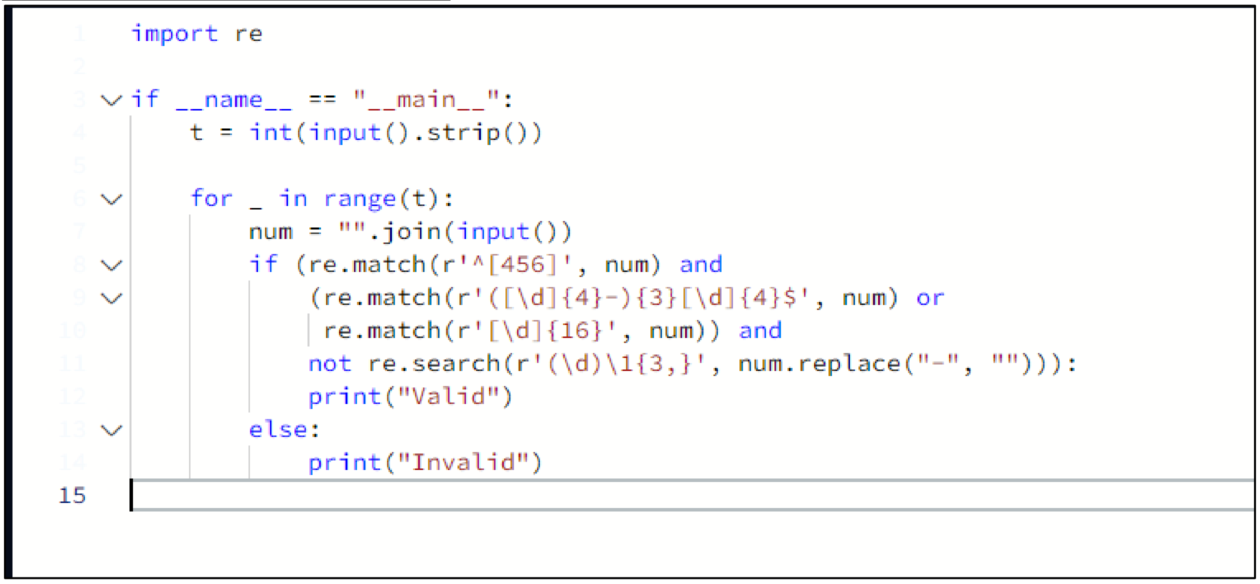
```python
from fractions import Fraction···
def product(fracs):
    t = reduce(lambda x, y : x * y, fracs)
    return t.numerator, t.denominator
if __name__ == '__main__':···
```

19. Regrex substitution



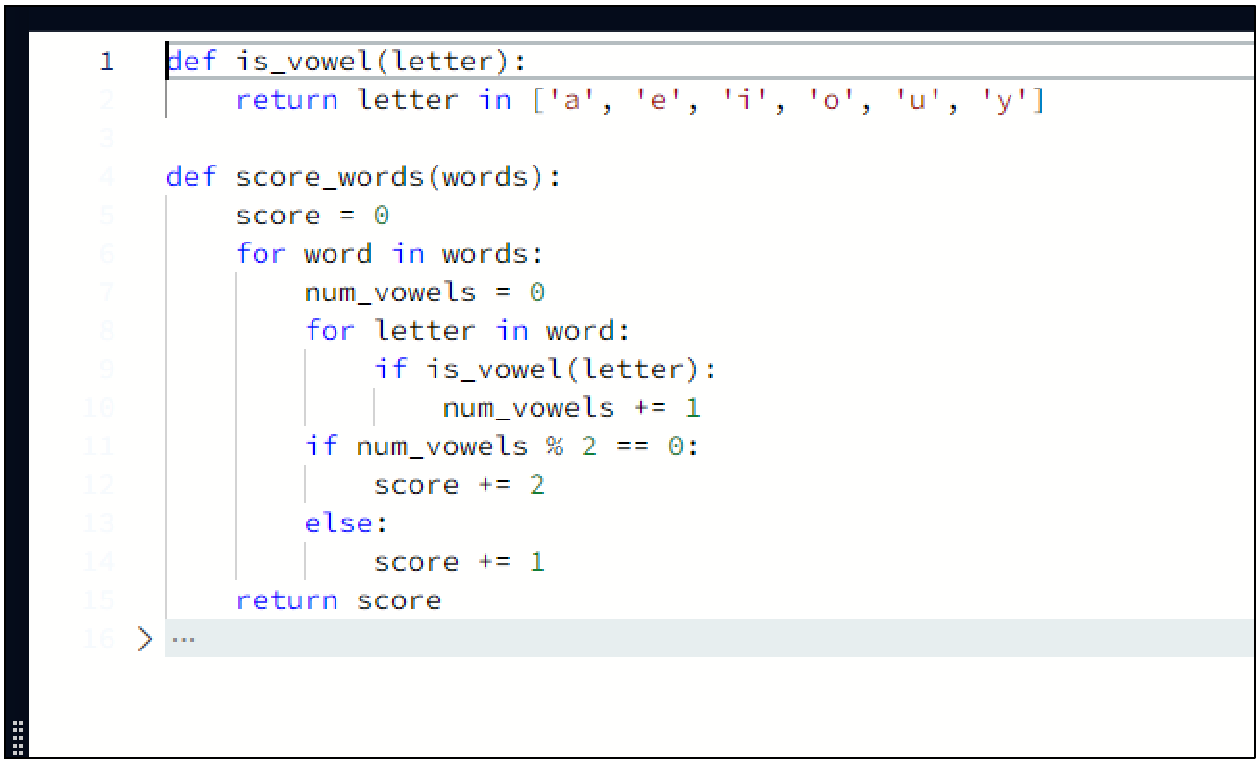
```python
import re

def change(match):
    symb = match.group(0)

    if symb == "&&":
        return "and"
    elif symb == "||":
        return "or"

n = int(input().strip())
for _ in range(n):
    print(re.sub(r'(?<= )(&&|\|\|)(?= )', change, input()))
```

20. Validating Credit card number



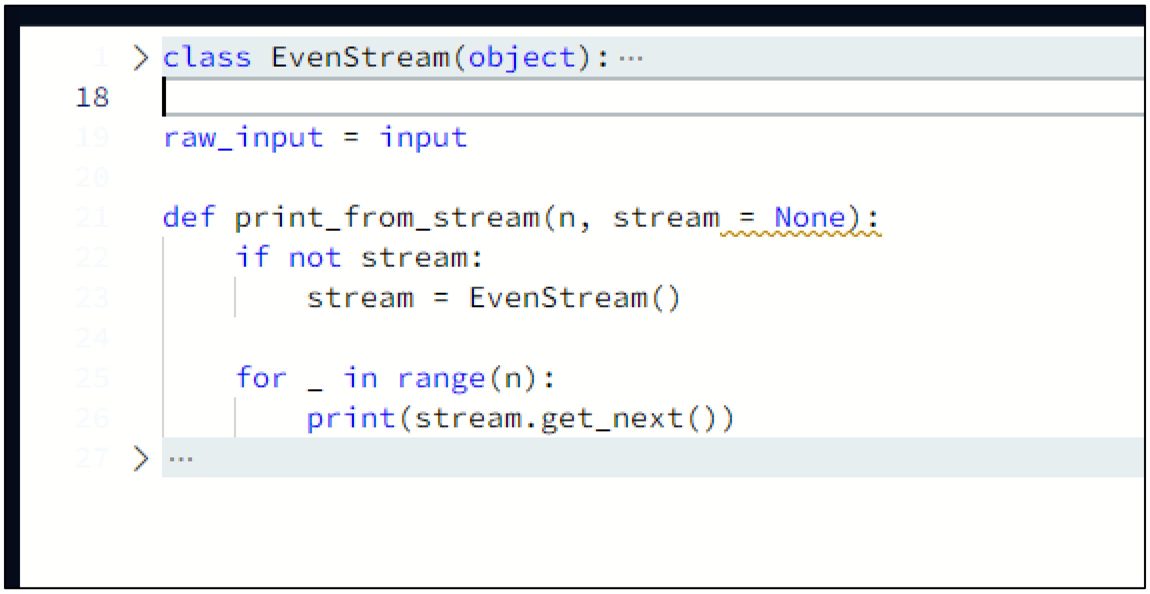
```python
import re

if __name__ == "__main__":
    t = int(input().strip())

    for _ in range(t):
        num = "".join(input())
        if (re.match(r'^[456]', num) and
            (re.match(r'([\d]{4}-){3}[\d]{4}$', num) or
             re.match(r'[\d]{16}', num)) and
            not re.search(r'(\d)\1{3,}', num.replace("-", ""))):
            print("Valid")
        else:
            print("Invalid")
```

21. Word score



```python
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
...
```

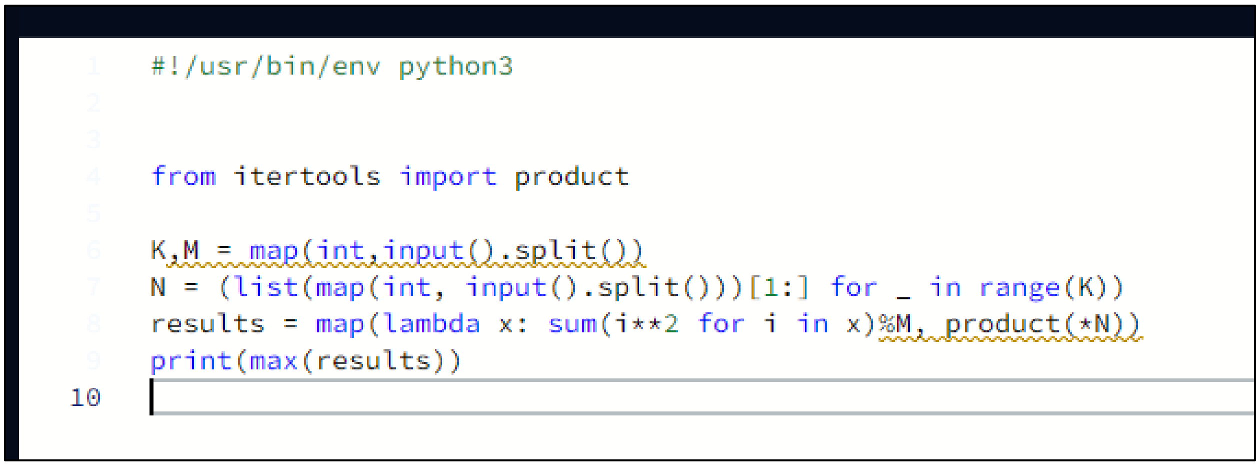
22. Default argument



```python
class EvenStream(object):...

raw_input = input

def print_from_stream(n, stream = None):
    if not stream:
        stream = EvenStream()

    for _ in range(n):
        print(stream.get_next())
...
```

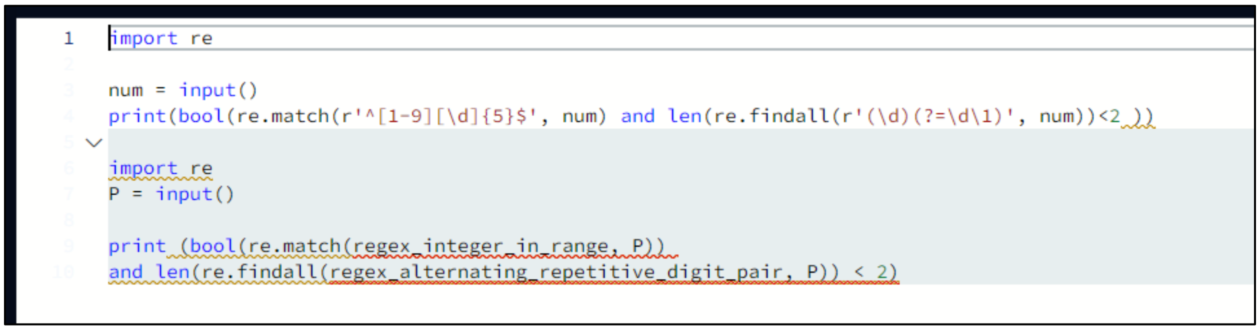
HARD LEVEL TASKS

1. Maximize it



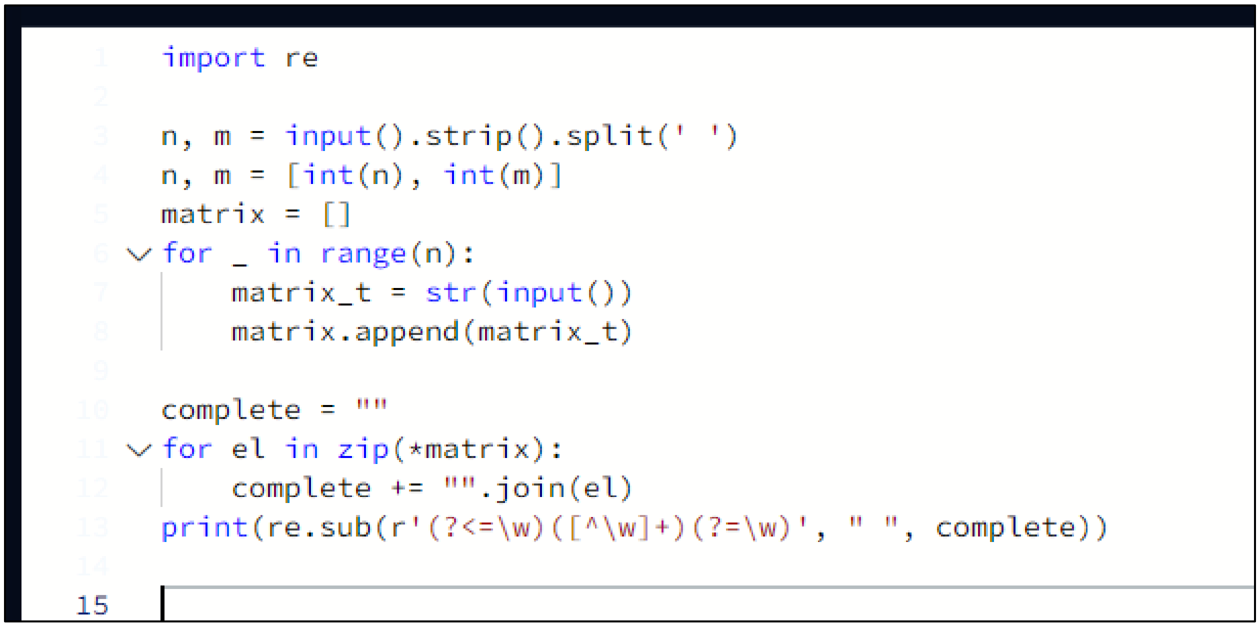
```
#!/usr/bin/env python3


from itertools import product

K,M = map(int,input().split())
N = (list(map(int, input().split()))[1:] for _ in range(K))
results = map(lambda x: sum(i**2 for i in x)%M, product(*N))
print(max(results))

```

2. Validating postal codes



```
import re

num = input()
print(bool(re.match(r'^[1-9][\d]{5}$', num) and len(re.findall(r'(\d)(?=\d\1)', num))<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)
```

3. Matrix script



```
import re

n, m = input().strip().split(' ')
n, m = [int(n), int(m)]
matrix = []
for _ in range(n):
    matrix_t = str(input())
    matrix.append(matrix_t)

complete = ""
for el in zip(*matrix):
    complete += "".join(el)
print(re.sub(r'(?<=\w)([^\w]+)(?=\w)', " ", complete))

```