1. Validating Postal Codes (<https://www.hackerrank.com/challenges/validating-postalcode/problem>)
2. regex\_integer\_in\_range = r"\_\_\_\_\_\_\_\_\_"   *# Do not delete 'r'.*
3. regex\_alternating\_repetitive\_digit\_pair = r"\_\_\_\_\_\_\_\_\_"  *# Do not delete 'r'.*
4. s = **input**()
5. **print**(s.isdigit() **and** 100000 <= **int**(s) <= 999999 **and**
6. **sum**([s[i] == s[i+2] **for** i **in** **range**(0, 4)]) < 2)
7. **import** re

**print** (**bool**(re.match(regex\_integer\_in\_range, P))

**and** **len**(re.findall(regex\_alternating\_repetitive\_digit\_pair, P)) < 2)

2. Matrix Script (<https://www.hackerrank.com/challenges/matrix-script/problem>)

**import** re

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

a = []

b = ""

**for** \_ **in** **range**(n):

    a.append(**input**())

**for** z **in** **zip**(\*a):

    b += "".join(z)

**print**(re.sub(r"(?<=\w)([^\w]+)(?=\w)", " ", b))

3. Triangle Quest 2 (<https://www.hackerrank.com/challenges/triangle-quest-2/problem>)

**for** x **in** **range**(1,**int**(**input**())+1):

**print**(((10\*\*x - 1)//9)\*\*2)

4. Reduce Function (<https://www.hackerrank.com/challenges/reduce-function/problem>)

**from** fractions **import** Fraction

**from** functools **import** **reduce**

**def** product(fracs):

    t = **reduce**(**lambda** x, y : x \* y, 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)

5. Regex Substitution (https://www.hackerrank.com/challenges/re-sub-regex-

substitution/problem)

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

**import** re

N = **int**(**input**())

**for** i **in** **range**(N):

**print**(re.sub(r'(?<= )(&&|\|\|)(?= )', **lambda** x: 'and' **if** x.group() == '&&' **else** 'or', **input**()))

6. Validating Credit Card Numbers (https://www.hackerrank.com/challenges/validating-

credit-card-number/problem)

**import** re

**for** \_ **in** **range**(**int**(**input**())):

    s = **input**()

**if** re.match(r"^[456]([\d]{15}|[\d]{3}(-[\d]{4}){3})$", s) **and** **not** re.search(r"([\d])\1\1\1", s.replace("-", "")):

**print**("Valid")

**else**:

**print**("Invalid")

7. Default Arguments (https://www.hackerrank.com/challenges/default-

arguments/problem)

**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=EvenStream()):

    stream.\_\_init\_\_()

**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())