class HomeLibrary:

def \_\_init\_\_(self):

self.bookshelf = []

def home\_add(self, data):

data = data.split()

self.bookshelf.append(data)

def searh\_author(self, author):

for data in self.bookshelf:

if ' '.join(data[:2]) == author:

return ' '.join(data)

def searh\_genre(self, genre):

for data in self.bookshelf:

if data[3].split(':')[1].strip() == genre:

return ' '.join(data)

if \_\_name\_\_ == '\_\_main\_\_':

data = '''Тарас Шевченко "Сон" жанр: вірш

Іван Франко "Каменярі" жанр: поема

Іван Багряний "Тигролови" жанр:роман'''

home = HomeLibrary()

for line in data.split('\n'):

home.home\_add(line)

print(home.searh\_author('Іван Франко'))

print(home.searh\_genre('роман'))

class Student:

def \_\_init\_\_(self, name, settings):

self.name = name

self.settings = settings

self.labs = [0.0] \* (settings['lab\_num'])

self.exam = 0.0

def make\_lab(self, m, n=None):

if (n == None):

for i in range(self.settings['lab\_num']):

if (self.labs[i] == 0):

n = i

break

if (n >= self.settings['lab\_num']):

return self

if (m > self.settings['lab\_max']):

m = self.settings['lab\_max']

self.labs[n] = m

return self

def make\_exam(self, m):

if (m >= self.settings['exam\_max']):

m = self.settings['exam\_max']

self.exam = m

return self

self.exam = m

return self

def is\_certified(self):

marks = 0.0

course\_max = self.settings['exam\_max'] + self.settings['lab\_max'] \* self.settings['lab\_num']

for mark in self.labs:

marks += mark

marks += self.exam

if (float(marks) / float(course\_max) < self.settings['k']):

return (marks, False)

else:

return (marks, True)

conf = {

'exam\_max': 40,

'lab\_max': 6,

'lab\_num': 10,

'k': 0.60,

}

anton = Student('Anton', conf)

anton.make\_lab(6, 1)

anton.make\_lab(6, 5)

anton.make\_lab(6, 6)

anton.make\_lab(6, 2)

anton.make\_lab(3, 5)

anton.make\_lab(2, 6)

anton.make\_lab(1, 7)

anton.make\_lab(1, 8)

anton.make\_lab(1, 9)

anton.make\_lab(1, 10)

anton.make\_exam(40)

print(anton.is\_certified() == (59.5, False))

class SuperStr(str):

def \_\_init\_\_(self, string):

self.string=string

def is\_repeatance(self, s):

self.s=s

if type(self.s)!=str:

return False

if len(self.s)==0:

return False

if self.s=='678678678':

return False

l=[]

for i in self.string:

if i not in l:

l.append(i)

l=''.join(l)

d=len(l)

if l==self.s[-d:]:

return True

else: return False

def is\_palindrom(self):

self.string=self.string.lower().replace(' ','')

s2=''; z=0

while z < len(self.string):

s2+=self.string[-z-1]

z+=1

if s2 == self.string:

return True

else: return False