**Работа с матрицами**

# coding=windows-1251

def format\_matrix (matrix\_string: str)->str:

lst = matrix\_string.split(',')

rows = int(lst[0])

cols = int(lst[1])

\_str = ""

for r in range(0,rows):

for c in range(0, cols):

#print(round(float(lst[2 + r\*cols + c]),3))

\_str += str(round(float(lst[2 + r\*cols + c]),3))

\_str += " "

\_str += "\n"

return \_str

#matrix\_string = input()

matrix\_string = "3,4,1.1349,2.6879,3.99999,4.5678,5.8712,6.00001,7.19231,8.123012,5.8712,6.00001,7.19231,1.66666"

formatted\_matrix = format\_matrix(matrix\_string)

print(formatted\_matrix)

**Решение уравнений**

# coding=windows-1251

from os import replace

def solve\_linear\_equation (equation: str)->str:

variable = ''

for i in equation:

if(i.isalpha() == True):

variable = i

equation = equation.replace('- ', '-')

lst = equation.split(' ')

variables = []

numbers = []

beforeEqual = True

for k in lst:

lastSymbol = k[-1] #забираем последний символ

if(lastSymbol == '='): beforeEqual = False

#если перед знаком равенства

if(beforeEqual):

#если в конце x

if(lastSymbol == variable):

if(len(k) > 1):

variables.append(float(k.rstrip(lastSymbol)))

else: variables.append(float(1))

#если в конце цифра

elif(lastSymbol.isdigit() == True):

numbers.append(-float(k))

#если после знака равенства

else:

#если в конце x

if(lastSymbol == variable):

variables.append(-float(k.rstrip(lastSymbol)))

#если в конце цифра

elif(lastSymbol.isdigit() == True):

numbers.append(float(k))

leftSum = sum(variables)

rightSum = sum(numbers)

result = variable + " = " + str(round(rightSum / leftSum, 3))

return result

#equation = input()

equation = "g + 2 = 5g - 10"

solution = solve\_linear\_equation(equation)

print(solution)

**Столярная мастерская**

# coding=windows-1251

import math

def schedule(timetable: str, orders: str)->str:

ordersList = []

daysList = []

dates = []

count = 0

for i in orders.split(';'):

ordersList.append(math.prod(map(int,i.split(','))))

for i in timetable.split(';'):

hours = list(map(int,i.split(',')))

minutes = (hours[1] - hours[0])\*60

daysList.append(minutes)

leftMinutes = daysList[0]

print(f"Начальные минуты из первого дня = {leftMinutes}")

i = 0

while(i < len(ordersList)):

print(f"i={i}")

leftMinutes = leftMinutes - ordersList[i]

print(f"leftMinutes = {leftMinutes}")

if(leftMinutes > 0):

dates.append(count)

else:

daysList.pop(0)

count += 1

i -= 1

leftMinutes = daysList[0]

print(f"Выкинули день, теперь leftMinutes = {leftMinutes}")

i += 1

return ', '.join(map(str, dates))

#timetable = input()

#orders = input()

timetable = "9,11;10,10;7,14"

orders = "1,60;5,30;10,10;2,32"

dates = schedule(timetable, orders)

print(dates)

**Планировщик**

# coding=windows-1251

import math

import re

import datetime

from datetime import datetime

class ScheduleManager:

def \_\_init\_\_(self, schedule: str):

self.schedilesList = schedule.split(", ")

def add\_task(self, task: str):

tasksForDelay = []

taskTimes = re.findall(r"\d{2}:\d{2}", task)

taskStartTime = datetime.strptime(taskTimes[0], "%H:%M").time()

taskEndTime = datetime.strptime(taskTimes[1], "%H:%M").time()

for scheduleString in self.schedilesList:

name = re.findall(r"^.+?\d{2}:\d{2}", scheduleString)[0]

name = name[0:len(name) - 6]

priority = re.findall(r"\b\w+$", scheduleString)[0]

times = re.findall(r"\d{2}:\d{2}", scheduleString)

startTime = datetime.strptime(times[0], "%H:%M").time()

endTime = datetime.strptime(times[1], "%H:%M").time()

if (taskStartTime >= startTime and taskStartTime < endTime or

taskEndTime > startTime and taskEndTime <= endTime or

taskStartTime < startTime and taskEndTime > endTime):

if(priority == "высокий"):

print(f"Время занято")

return

elif(priority == "обычный"): tasksForDelay.append(name)

if(len(tasksForDelay) > 0):

taskNames = ""

for s in tasksForDelay: taskNames += s + ', '

taskNames = taskNames[0:len(taskNames) - 2]

print(f"Придётся отложить задачи: {taskNames}")

else: print("Запись сделана")

# schedule = input()

# new\_task = input()

schedule = "Задача 12 13:00 до 14:00 высокий, Задача 13 14:30 до 15:30 обычный"

new\_task = "15:30 до 16:30"

manager = ScheduleManager(schedule)

manager.add\_task(new\_task)