Реализовать цикл

class Task(namedtuple):

########

# в объекте task описана задача, которую нужно решить и

# все необходимые параметры

class TaskQueue(Queue):

########

# При добавлении проверять принадлежит ли объект к классу Task

class Worker():

def \_\_init\_\_(self, \*\*init\_params):

self.task\_queue = TaskQueue()

self.driver = $SELENIUM.WebDriver()$

class Mainer(Worker):

def \_\_init\_\_(self, \*\*init\_params):

super(Worker, self).\_\_init\_\_(init\_params)

self.skills = {‘get\_tournaments’: self.get\_tournaments,

‘get\_games’: self.get\_games}

#########

def \_work(self):

if len(self.task\_queue) == 0:

return self.get\_new\_tasks() # Запрос с сервера заданий для

# текущего клиента

task = self.task\_queue.pop()

result = self.do(task) # Выполнение задания

self.report(result) # Отправка результата на сервер (Если произошла

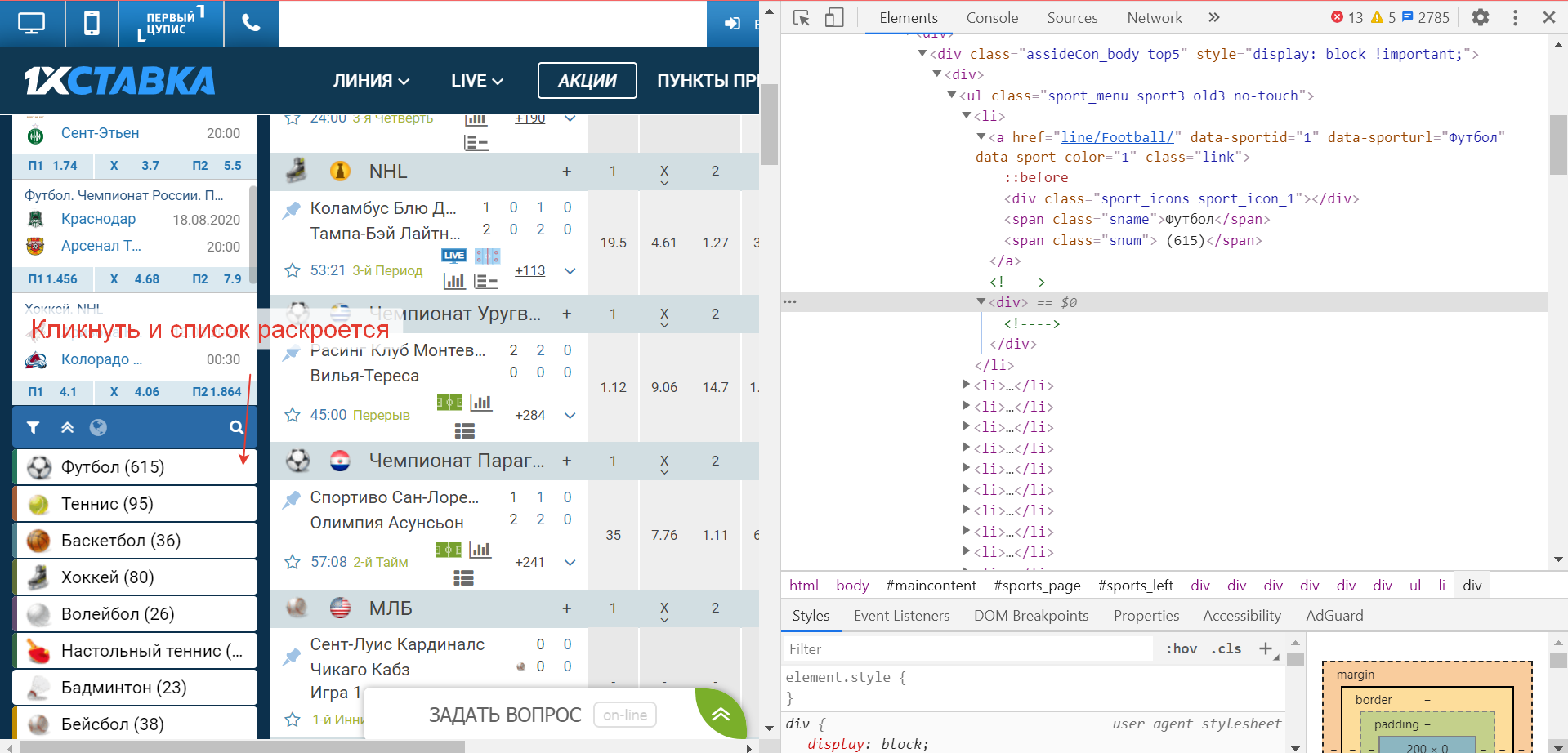
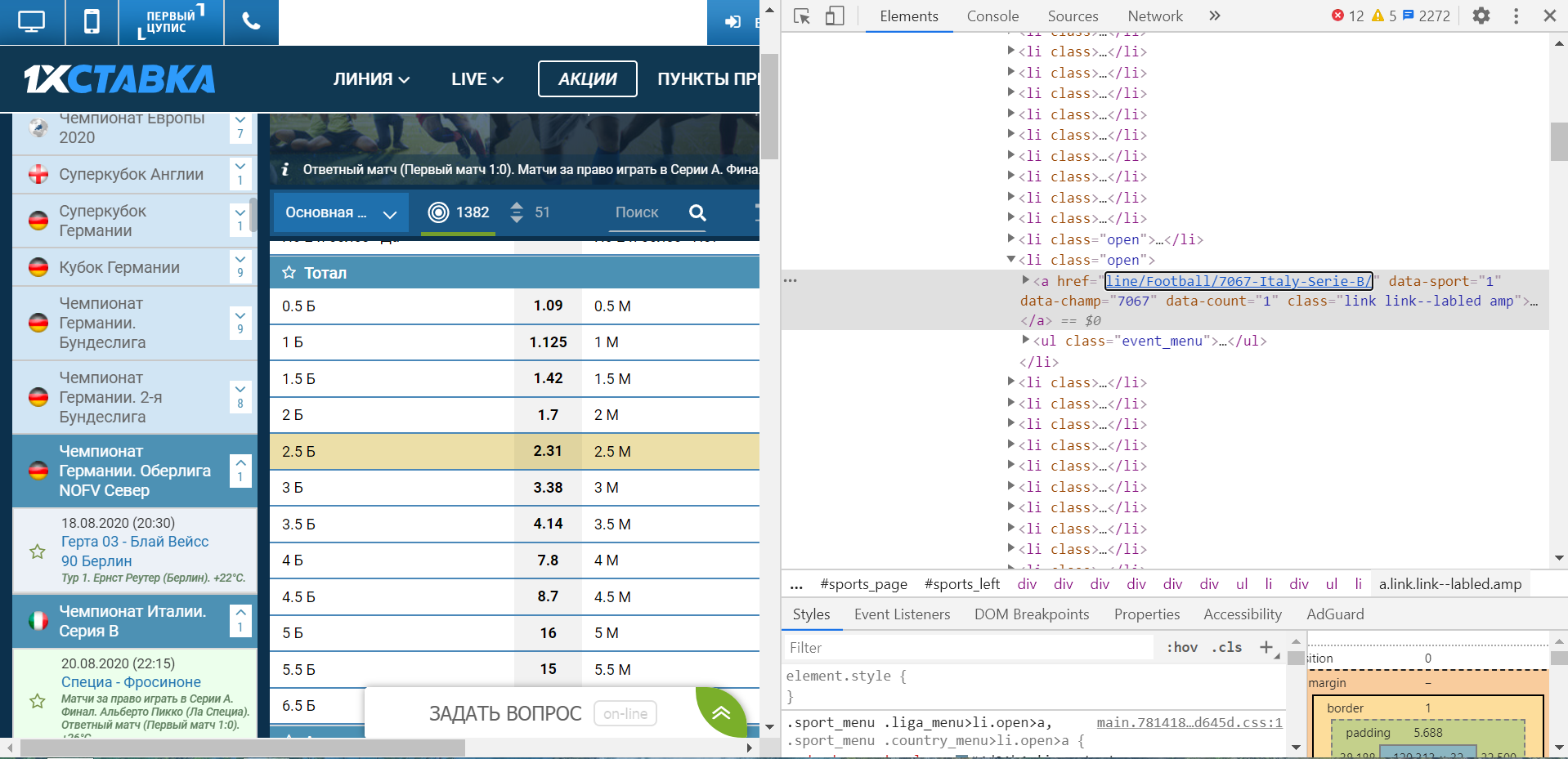
# ошибка, то сообщить об этом

def work(self):

while True:

self.\_work()

def get\_tournaments(self, is\_line: bool):

# Вернуть кортеж (название турнира, абсолютный URL на турнир)

def get\_games(self, is\_line: bool, tournament\_url: str):

# Вернуть (название левой команды, название правой команды, ссылка на матч, все коэффициенты, дата и время начала матча) - для некоторых коэффициентов нужно будет кликнуть на элемент

