

# 1 Техническое задание

## 1.1 Описание

Необходимо разработать короткий Python скрипт, для генерации математической задачи. Размер скрипта около **100-200** строк кода, при генерации без картинки. **200-400** при генерации с картинкой.

Даны векторы  $\vec{a}(-13; 4)$  и  $\vec{b}(-6; 1)$ . Найдите скалярное произведение  $\vec{a} \cdot \vec{b}$ .

Рис. 1: Исходная задача

Векторы  $\vec{a}(-8; 19)$  и  $\vec{b}(26; 12)$  заданы компонентами. Найдите их скалярное произведение.

Рис. 2: Сгенерированная задача

## 1.2 Основные требования

В процессе генерации задачи, необходимо удовлетворить некоторым основным требованиям, которые и вносят значительный вклад в сложность скрипта.

1. **Случайные шаблоны** - перед генерацией задачи, необходимо задать сет предсгенерированных шаблонов с условием. Для генерации лучше всего использовать ChatGPT. Важно, **ни один из шаблонов не должен в точности повторять исходную формулировку** задачи. В процессе генерации, для каждой задачи необходимо выбрать случайный паттерн и подставить туда правильные параметры.

```
patterns = [
    "Даны векторы  $a(\{a1\}; \{a2\})$  и  $b(\{b1\}; \{b2\})$ . Найдите их скалярное произведение.",
    "Векторы  $a(\{a1\}; \{a2\})$  и  $b(\{b1\}; \{b2\})$  заданы своими компонентами. Определите их скалярное произведение.",
    "Определите скалярное произведение векторов  $a(\{a1\}; \{a2\})$  и  $b(\{b1\}; \{b2\})$ .",
    "Найдите скалярное произведение векторов  $a(\{a1\}; \{a2\})$  и  $b(\{b1\}; \{b2\})$ .",
    "Векторы  $a(\{a1\}; \{a2\})$  и  $b(\{b1\}; \{b2\})$  заданы компонентами. Найдите их скалярное произведение."
]
```

Рис. 3: Пример паттернов

2. **Генерация параметров** - для генерации задачи необходимо выбрать некоторые параметры. Сделать это можно перебором в каком-то промежутке, или же абсолютно случайно. Главное требования - **все задачи должны быть уникальными**, а значит в случае случайного выбора необходимо вести set параметров, во избежание дубликатов.

Так же стоит отметить, что **все задачи должны иметь адекватные параметры**, сопоставимые по сложности с исходной задачей. Так в примере Рис. 6 самое сложное действие - это произведение  $13 \times 6$ . В этом плане произведение из Рис. 2 -  $26 \times 8$  или  $19 \times 12$  имеет повышенную сложность, но все еще сопоставимую. Но если бы необходимо было выполнять операцию на порядок сложнее: например трехзначное число умножить на двузначное, то такие параметры бы считались неадекватными.

3. **HTML оформление** - сгенерированная задача должна быть красиво оформлена в HTML виде, **используя теги  $\langle \text{math} \rangle$** , если внутри используются формулы. Помимо этого нужно учитывать другие визуальные нюансы. Например если имеется шаблон  $(a + b)$ , и генерируются параметры от -10 до 10. То в случае прямой вставки отрицательных чисел вместо  $b$ , может получиться например следующая строка  $(5 + -3)$ . В таком случае необходимо отдельно обрабатывать место шаблона, чтобы получить строку вида  $(5 - 3)$ . Аналогичные проблемы связаны с сокращением дробей и тд.

4. **Целый или конечный ответ** - так как задачи должны решаться онлайн без преподавателя, ответ должен быть **однозначно трактуемым для компьютера и человека**, а также его должно быть легко вписать в поле для ввода. Так например ответ корень из 2 не подойдет, тк может быть проблематично вписать его в поле для ответа.

Поэтому предполагается, что в большинстве задач ответом является целое число - что сильно уменьшает количество возможных вариантов при генерации. Так же в некоторых задачах, например про вероятность - ответ не может быть целым. Поэтому предполагается конечный десятичный вид. Например: 0.125 или 0.3 и так далее.

5. **Определение сложности** - для каждой задачи необходимо определять сложность, а также делать случайную выборку из сгенерированного пула в определенном соотношении по сложности. Например, всего возможно сгенерировать 1000 различных задач. В финальной выборке должно быть 50 задач. Из них 60 процентов легких, 30 процентов средних и 10 процентов сложных.

Сложность задачи оценивает ее **относительную сложность в рамках одного условия**, но с разными параметрами. Чтобы определить сложность задачи, предлагается полагаться на здравый смысл. Так если основным действием в задаче является взятие корня, то очевидно что произвести данную операцию над числами 4, 9, 16, 25 и тд. до 100 - очень просто, тк это квадраты первых 10-ти чисел. Таким задачам предлагается давать сложность легко. Если же квадрат берется для степени двойки или 5-ки, то задача будет средней, тк данные числа на слуху у школьников. Например корень 256 или 625. Если же это что-то большое, как 361 и тд. то данная задача точно будет сложной.

По такой же логике необходимо оценивать сложность и для других задач / операций.

**Важно**, иногда может оказаться, что нельзя сгенерировать больше 10 простых задач. В таком случае нужно взять все простые задачи и добрать какое-то количество средних и сложных, чтобы сохранить пропорцию.

### 1.3 Выход скрипта

Основным артефактом работы скрипта - является JSON файл, с массивом сгенерированных задач. Пример сгенерированного файла с массивом из одного элемента:

В данном JSON объекте присутствуют как обязательные, так и необязательные поля. Дальше будут описаны все обязательные и некоторые дополнительные поля.

1. **uid** - обязательное поле, произвольная строка из 10 символов. Допускаются только **заглавные** английские символы или цифры.
2. **task\_num** - обязательное поле, порядковый номер задачи в тесте. Выдается вместе с исходной задачей.
3. **task\_subject** - обязательное поле, предмет задачи. Всегда принимает значение **math**.
4. **task\_types** - обязательное поле, массив типов задачи. Выдается вместе с исходной задачей.
5. **fipi\_uid** - обязательное поле, уникальный номер задачи с сайта ФИПИ. Выдается вместе с исходной задачей.
6. **task\_text** - обязательное поле, текстовое описание задачи в читаемом для машины тексте. Необходимо для того, чтобы удобно читать задание из базы данных, пересылать ее в телеграмме, отправлять в wolfram для решения или ChatGPT. Например некоторые математические символы могут пропасть при копировании. Например корень или степень. Поэтому предлагается в данном поле записывать их буквенно, как `sqrt()` и т.д.
7. **task\_html** - обязательное поле, условие задачи оформленное в формате html. Корневым тегом является `<p></p>`. Для описание математических формул можно использовать тег `<math>` и остальные математические теги. Также могут пригодиться таблицы и тд.
8. **task\_answer** - обязательное поле, ответ на задачу. Тип поля - строка.
9. **task\_solution** - обязательное поле, оформленное в HTML решение задачи, с подставленными частными параметрами.

10. **task\_hints** - обязательное поле, массив оформленных в HTML подсказок для решения задачи. Иногда будет достаточно завернуть просто текст в `<p></p>` тег. Данный массив должен содержать все формулы и свойства, которые могут пригодиться при решении задачи.
11. **answer\_type** - обязательное поле, тип ответа. Выдается вместе с исходной задачей. Скорее всего просто short.
12. **task\_group\_key** - обязательное поле, ключ группы задачи. Каждый генератор должен иметь уникальный код, обозначающий тип генерируемой задачи, ее уникальное свойство - о чем задача.
13. **task\_group\_label** - обязательное поле, имя группы задачи. Каждый генератор должен иметь уникальное имя, расшифровку ключа.
14. **task\_images\_svg** - дополнительное поле, массив содержимого svg картинок.

```

1  [ {
2      "uid": "AY197D05G6",
3      "task_subject": "math",
4      "task_text": "Отрезки AC и BD - диаметры окружности с центром O. Угол ACB равен 68°. Найдите угол AOD. Ответ дайте в градусах.",
5      "task_html": "<p>Окружность с центром в точке O имеет диаметры AC и BD. Величина угла ACB равна 68°. Найдите величину угла AOD. Ответ дайте в градусах.</p>",
6      "task_solution": "<p>Угол <math>\angle AOD</math> является смежным для угла <math>\angle ACB</math>. Следовательно, <math>\angle AOD = 180^\circ - \angle ACB = 180^\circ - 68^\circ = 112^\circ</math>.</p>",
7      "task_answer": "112",
8      "task_images_svg": [
9          "<?xml version='1.0' encoding='UTF-8'?>\n<svg xmlns='\"http://www.w3.org/2000/svg\"'>\n  <circle center='O' radius='1' />\n  <point x='0' y='0' label='O' />\n  <point x='1' y='0' label='A' />\n  <point x='-1' y='0' label='C' />\n  <point x='0' y='1' label='B' />\n  <point x='0' y='-1' label='D' />\n  <angle x1='0' y1='0' x2='1' y2='0' label='ACB' />\n  <angle x1='0' y1='0' x2='-1' y2='0' label='AOD' />\n</svg>"
10 ],
11      "task_hints": [
12          "<p>Центральный угол равен удвоенному вписанному углу, опирающемуся на ту же дугу.</p>",
13          "<p>Используйте свойство, смежных углов. Сумма смежных углов - равняется 180°.</p>"
14 ],
15      "task_group_label": "Поиск центрального угла, через вписанный и смежный",
16      "task_types": [
17          "7.1 Фигуры на плоскости"
18 ],
19      "task_group_key": "circle_task_a",
20      "answer_type": "short",
21      "task_num": 1,
22      "fipi_uid": "80A34C",
23      "difficulty": "medium"
24 } ]

```

Listing 1: tasks.json

## 1.4 Промпт ChatGPT

Для простого начала, предлагается использовать следующий промпт для ChatGPT. Данный запрос позволит быстро сгенерировать основу Python скрипта.

### 1.4.1 Генерация задачи

**Первое сообщение:** Помоги мне написать Python скрипт для генерации следующей задачи:

<Условие задачи>

Мне необходимо получить 1) Условие задачи 2) Решение задачи 3) Ответ задачи 4) Сложность задачи [Добавь какое-то условие для автоматической проверки на сложность. Например если все операции проводятся с числами до 10, то это легкая задача. Сделай возможность редактировать функцию оценки сложности]

**Второе сообщение:** Давай хранить сгенерированную задачу в JSON формате с следующими полями:

1. uid - уникальный номер из 10 случайных, заглавных букв или цифр.
  2. task\_subject - всегда равняется math.
  3. task\_text - условие задачи без использования спец символов. Например вместо знака корень используй функцию `sqrt()`. Это нужно для того, чтобы было удобно читать условие из базы данных.
  4. task\_html - условие задачи оформленное с использованием HTML. Необходимо для красивой записи формул. Используй теги `<p>` `<math>` `<mi>` `<mo>` `<ms>` и тд.
  5. task\_answer - ответ на задачу в виде строки.
  6. task\_solution - решение задачи с подставленными параметрами, оформленное с помощью HTML.
  7. task\_hints - массив подсказок к задаче, которые являются строками, так же оформленными в формате HTML для красоты. Должен содержать различные формулы которые используются при решении или свойства. Пример подсказки: "Используйте формулу для вычисления длины гипотенузы ...".
  8. answer\_type - всегда равняется short.
  9. difficulty - одно из трех: easy, medium, hard.
- Сгенерируй 1000 задач, возьми 40 случайных и сохрани в файл tasks.json - массив JSON-ов задач.

**Третье сообщение:** Давай добавим вариативности за счет использования случайных пресетов для условия задачи для task\_html. Сгенерируй

5 равнозначных текстов для условий, при этом ни один текст не должен повторять исходный. И дальше во время генерации выбирай случайный и подставляй туда параметры. Условия должны быть на русском языке, а кодировка utf-8.

### 1.4.2 Генерация картинки

Используя библиотеку `from lxml import etree`, напиши скрипт на Python для генерации похожей картинки.

<Вставьте картинку из задачи>

<Опишите какие параметры к картинке должны подаваться на вход>

## 1.5 Полезные функции

### 1.5.1 Генерация JSON файла

```
def save_tasks_to_json(tasks, filename='tasks.json'):
    with open(filename, 'w', encoding='utf-8') as f:
        json.dump(tasks, f, ensure_ascii=False, indent=4)
```

### 1.5.2 Генерация HTML-ей из массива JSON-ов

```
import os

def generate_html_files(tasks, output_folder='html'):
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    else:
        for file_name in os.listdir(output_folder):
            file_path = os.path.join(output_folder, file_name)
            if os.path.isfile(file_path):
                os.unlink(file_path)

    for i, task in enumerate(tasks):
        task_html = task['task_html']
        task_images_svg = task['task_images_svg'][0]

        prev_link = ''
        next_link = ''
        if i > 0:
            prev_uid = tasks[i-1]['uid']
            prev_link = f'<a href="{prev_uid}.html">Previous Task</a>'
        if i < len(tasks) - 1:
            next_uid = tasks[i+1]['uid']
            next_link = f'<a href="{next_uid}.html">Next Task</a>'
```

```

html_content = f'''
<html>
<head>
    <meta charset="UTF-8">
    <title>Task {task["uid"]}</title>
</head>
<body>
    {task_html}
    <div>{task_images_svg}</div>
    <div>
        {prev_link} | {next_link}
    </div>
</body>
</html>
'''

file_name = os.path.join(output_folder, f'{task["uid"]}.html')
with open(file_name, 'w', encoding='utf-8') as html_file:
    html_file.write(html_content)

```

## 1.6 Тестирование

Для тестирования результатов, был разработан специальный JS скрипт, который можно запустить в браузере. Для этого необходимо скачать данную папку **JS тестирование** и открыть файл index.html или же просто воспользоваться ссылкой: <https://regeinput.madzhuga.com/>.

Должно появиться следующее окно:



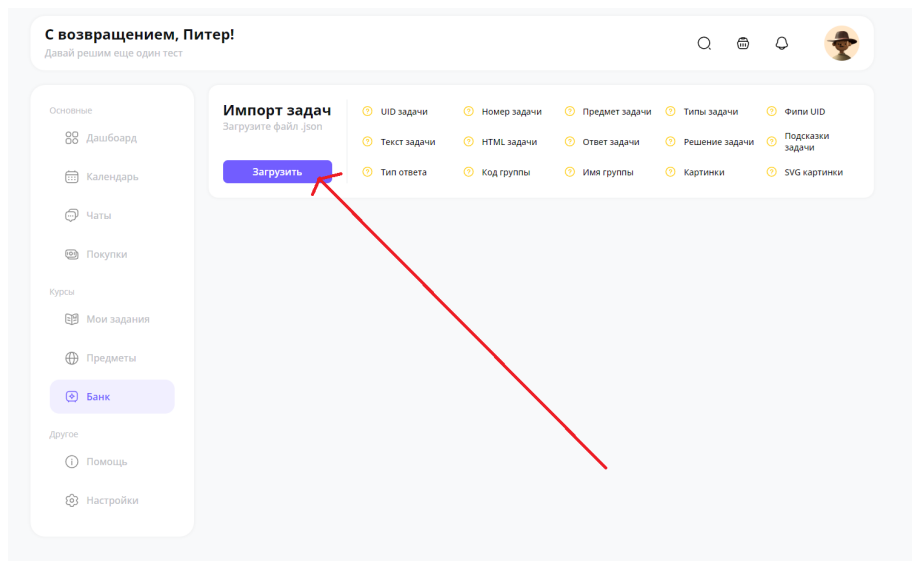


Рис. 4: Окно для тестирования

Необходимо нажать на кнопку Загрузить и выбрать сгенерированный .json файл. А именно tasks.json. Примеры файлов можно взять в репозитории с примером генераторов **Репозиторий с шаблонами**.

После загрузки должно повиться следующее окно:

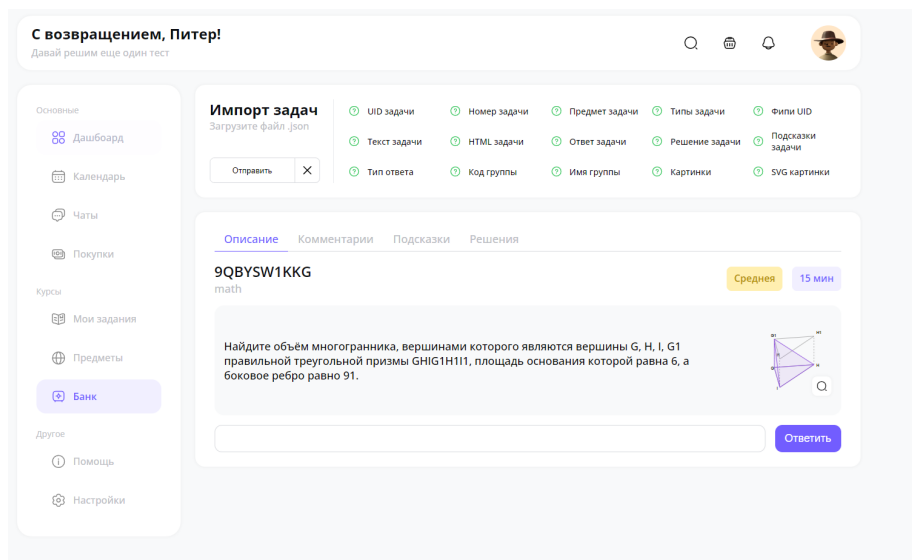


Рис. 5: Окно для тестирования

В данном меню необходимо проверить, что изображение сгенерировано правильно, все математические формулы верно отображаются. А также необходимо проверить меню Решения и Подсказки. В случае если какое-то поле не заполнено, можно открыть HTML код элемента, чтобы посмотреть название поля, которое неудачно запарсилось. Название поле зашифровано в id элемента.

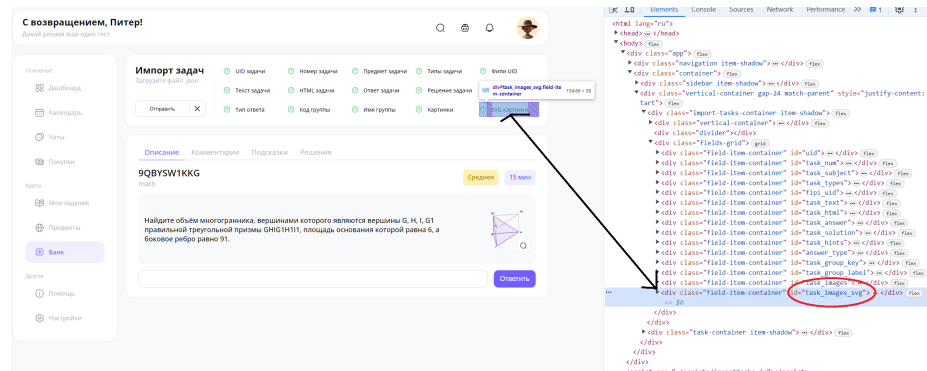


Рис. 6: Окно для тестирования

## 1.7 Вспомогательные файлы

Репозиторий с шаблонами  
HTML + JS для тестирования