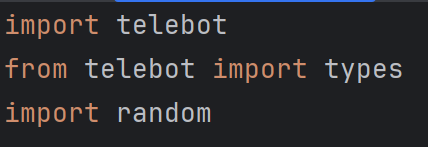
**Урок 10**

**Математический тренажёр**

На предыдущем уроке мы научили нашего бота отвечать на различные файлы, отправленные пользователем. Также сделали так, чтобы бот отправлял изображения, видео, аудио, текстовые документы, стикеры. Научились узнавать id стикера с помощью бота.

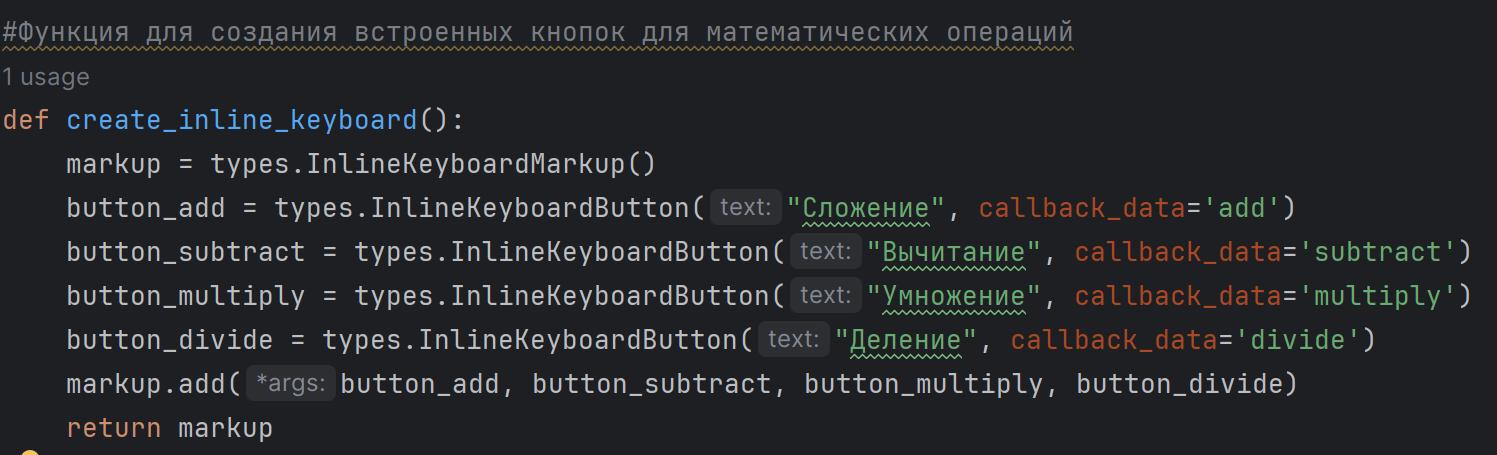
Сегодня мы сделаем бота «Математический тренажёр», с его помощью можно тренироваться в решении математических примеров. Бот будет предлагать пользователю разные задачи со случайными значениями (сложение, вычитание, умножение и деление) и предоставлять обратную связь по его ответам (правильный или неправильный ответ был дан)

Для начала необходимо импортировать библиотеку random и types



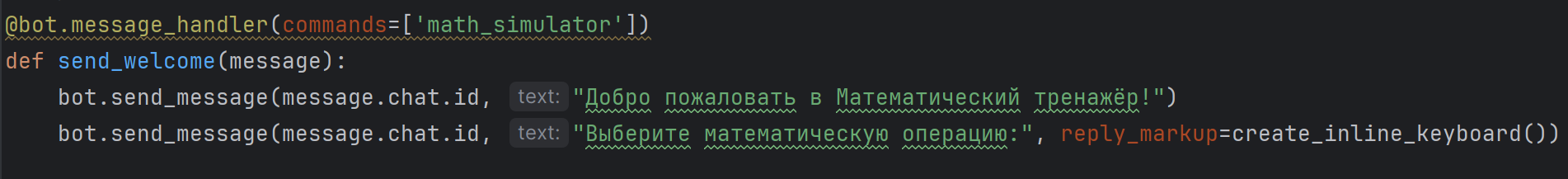
**Создаём кнопки с математическими операциями**

Мы сделаем inline-клавиатуру с кнопками для сложения, вычитания, умножения и деления. Пользователь может выбрать любое действие и решить задачу, сгенерированную ботом.

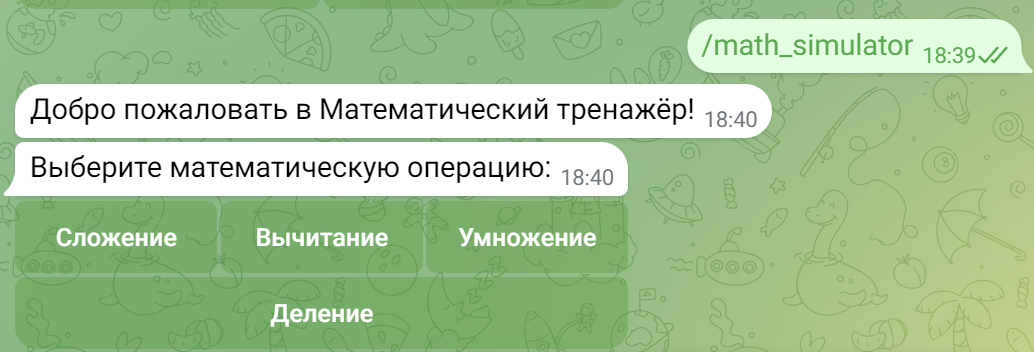


Обратите внимание — в параметрах каждой кнопки мы указываем callback\_data, по которому потом программа будет понимать на какую кнопку нажал пользователь.

Далее мы создаём обработчик для команды **/math\_simulator,** чтобы бот приветствовал пользователя и показывал встроенные кнопки.

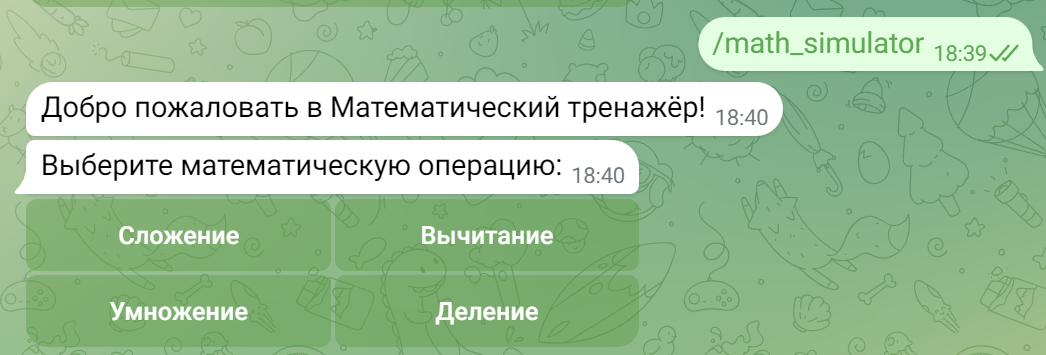


Протестируем программу на этом этапе:



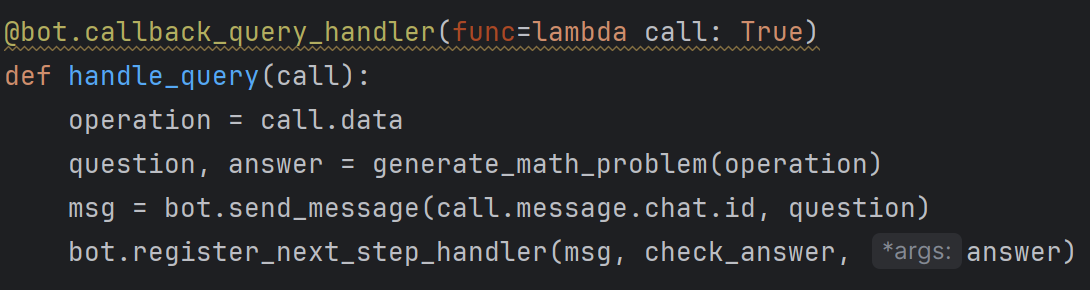
**Самостоятельная работа**

Сделайте так, чтобы в одном ряду располагалось две кнопки. Как это можно осуществить? С помощью параметра **row\_width**



**Принимаем ответ пользователя**

Сделаем обработчик, который будет принимать все ответы пользователя. Когда пользователь нажмёт на кнопку запуститься функция **generate\_math\_problem.** Аргументом функции будет выбор пользователя (сложение, деление и т.д.) — **operation.** Бот отправить пользователю сформированный вопрос, который хранится в переменной **question.** После этого мы создаём обработчик следующего шага (**bot.register\_next\_step\_handler**) для обработки ответа пользователя.

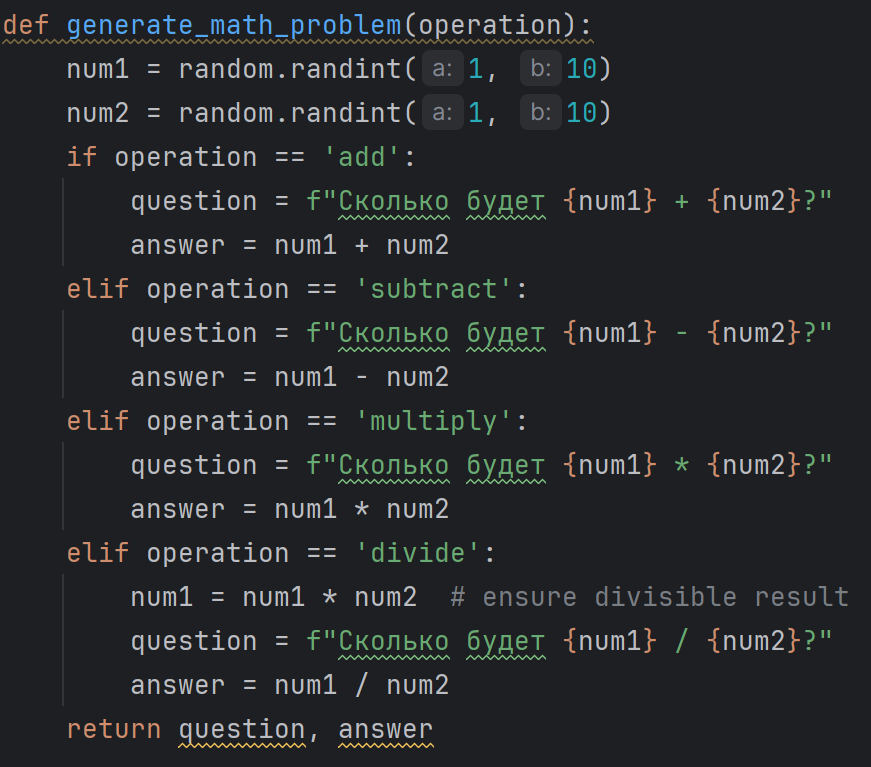


**Генерируем задачу**

Теперь создаём функцию, которую указывали ранее — **generate\_math\_problem.** В качестве аргумента мы указываем операцию, которую выбрал пользователь (**operation**).

Далее создаём две переменные (num1 и num2) и сохраняем в них два рандомных числа от 1 до 10. Можно указать диапазон рандомных чисел от 1 до 100, тогда задачи будут генерироваться более сложные.

В зависимости от операции мы генерируем соответствующую математическую задачу и ответ на неё сохраняем в переменную **answer.** После этого функция возвращает значения **question** и **answer.**

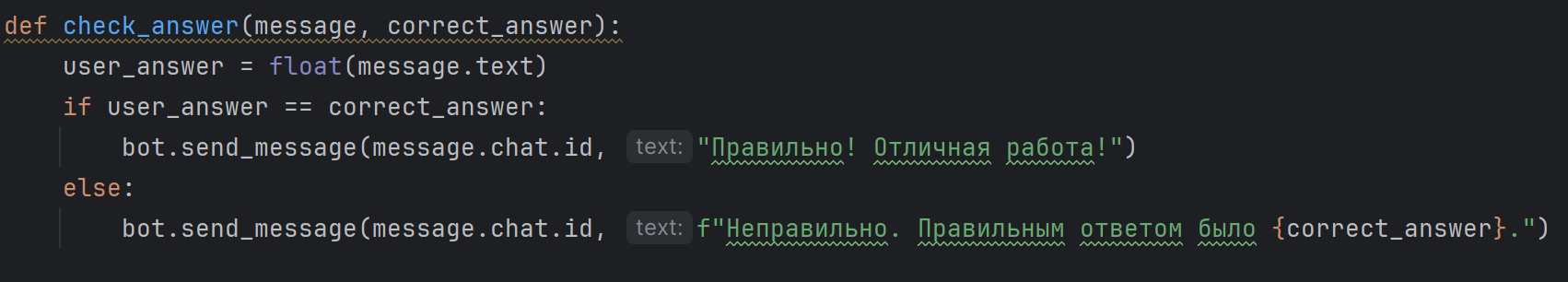


**Обрабатываем ответ пользователя**

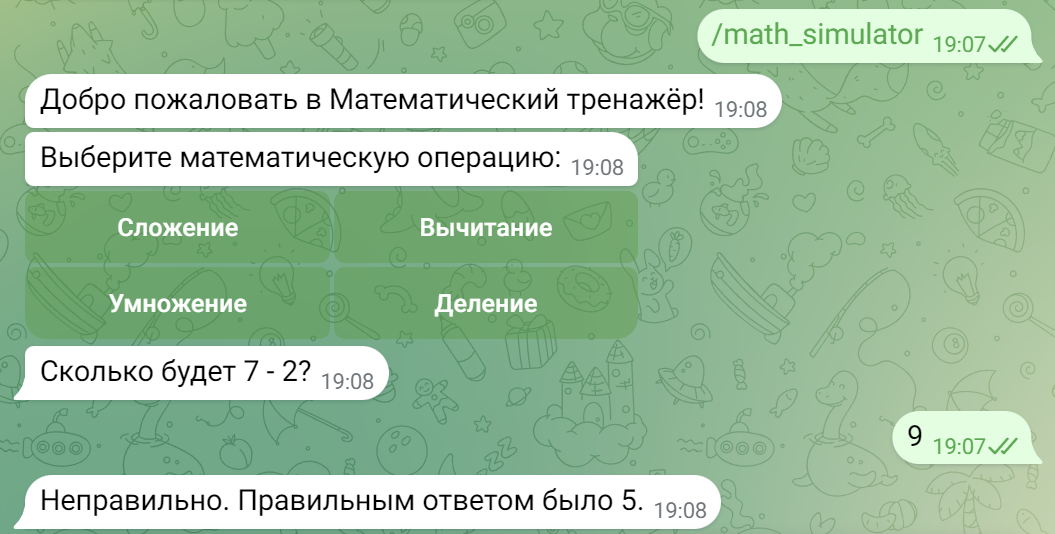
Создаём функцию **check\_answer**, которая проверит правильный ответ дал пользователь или нет. В качестве аргументов функция будет принимать само сообщение пользователя — **message**, и правильный ответ — **correct\_answer.**

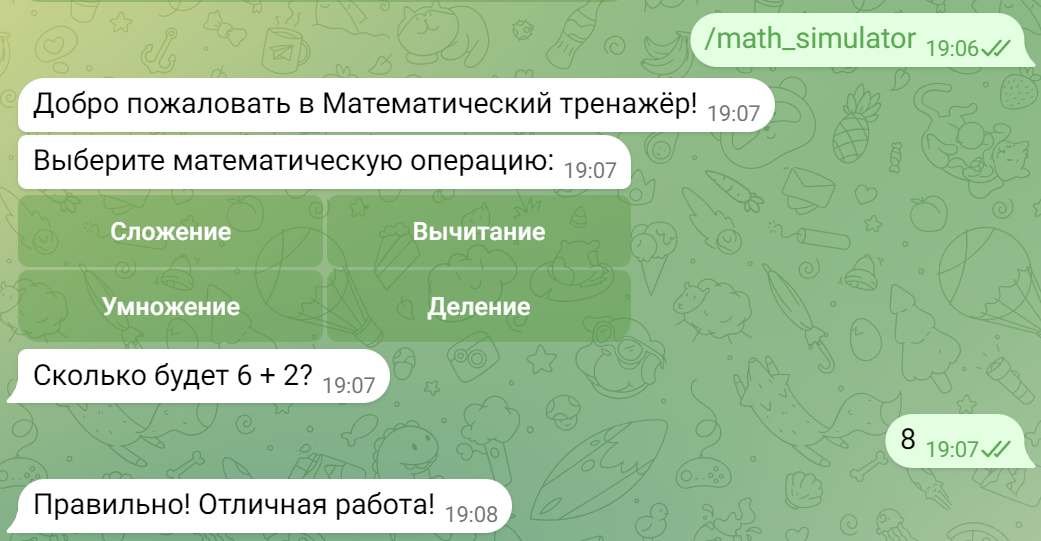
В переменную **user\_answer** мы сохраняем ответ пользователя. Однако текст сообщения, который отправляет пользователь, автоматически сохраняется как строка. Нам необходимо преобразовать эту информацию в число с плавающей запятой, для этого используем команду **float().**

Если ответ пользователя соответствует правильному ответу, бот отправляет сообщение «Правильно!». В противном случае он пишет «Неверно» и даёт правильный ответ.



Проверим как работает наш бот





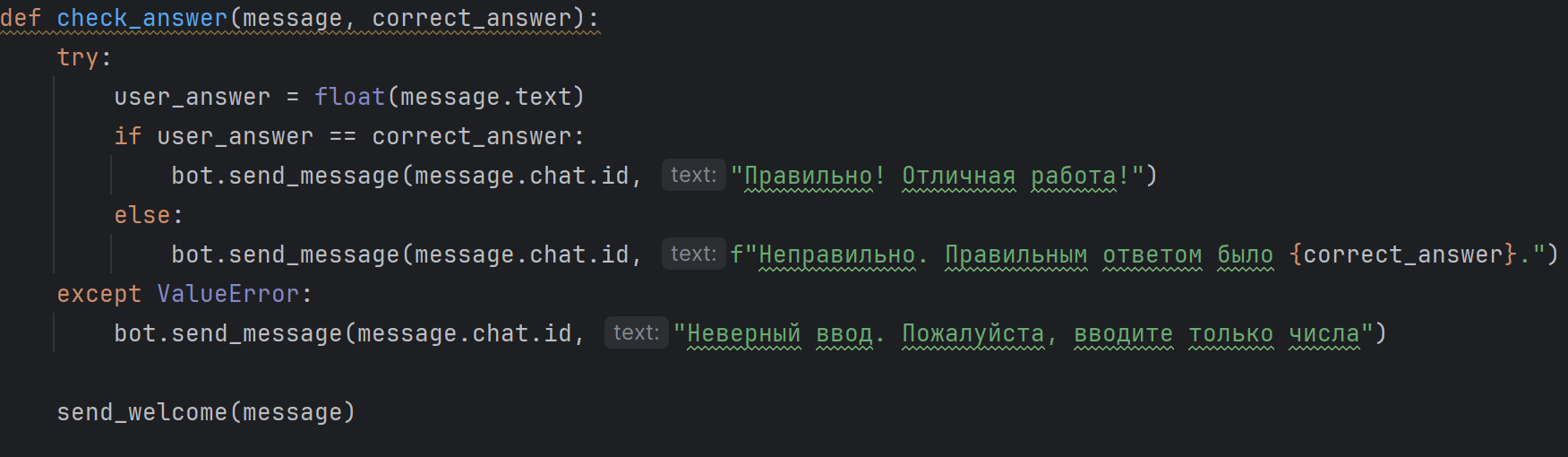
Всё работает! Однако есть одна проблема — если мы введёт не число, а например текст, будет ошибка и программа перестанет работать.

**Самостоятельная работа**

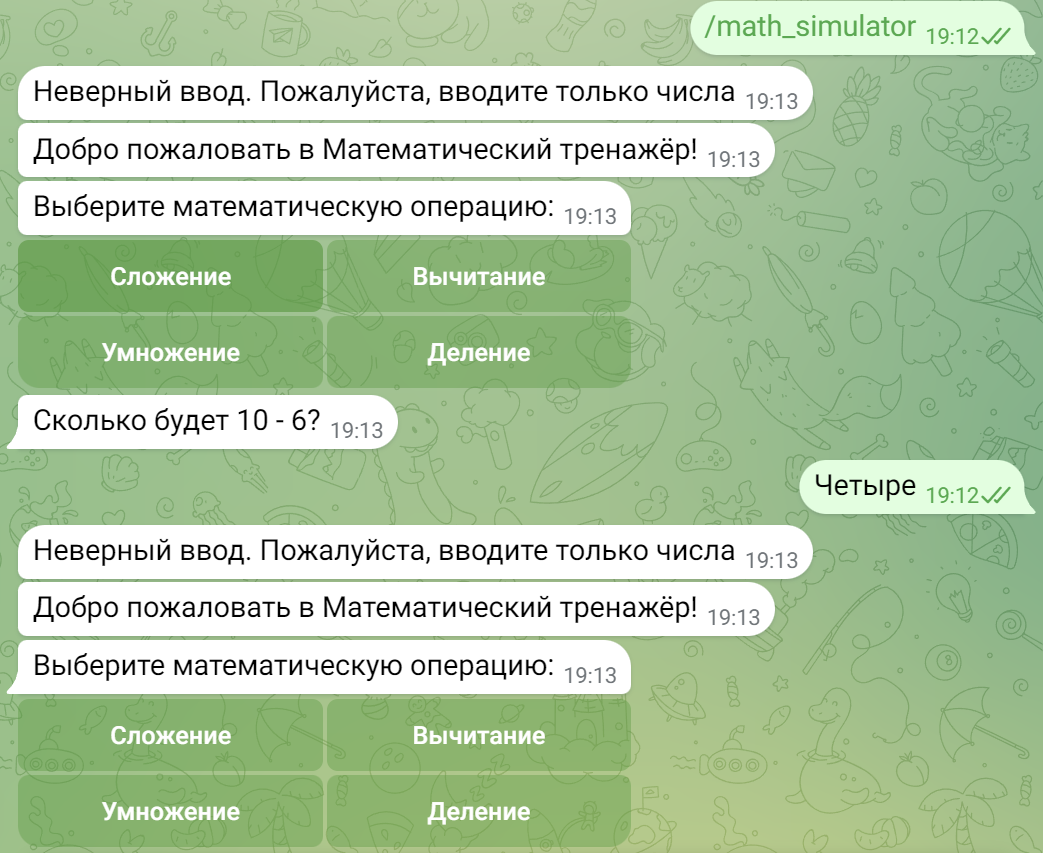
Как можно исправить функцию **check\_answer**, чтобы в случае, если пользователь ввёл недопустимый тип данных, программа не перестала работать, а предупредила пользователя что вводить можно только числа?

Для того чтобы избежать прерывания программы при возникновении подобных ошибок, применяется конструкция **try / except**. В блок **try** помещается код, который может вызвать ошибку, а в блок **except** — действия, которые будут выполнены в случае возникновения ошибки.

Исправим нашу функцию:



Протестируем бота



Теперь если пользователь ввёл неверные данные, бот предупреждает его об этом и далее перезапускается функция **send\_welcome()**, программа отправляет пользователя к приветственному сообщению и выбору кнопок.

**Рефлексия**

* Сегодня мы сделали бота «Математический тренажёр», который помогает тренироваться в решении математических примеров
* Сделали так, чтобы программа случайно генерировала математическую задачу с действием, которое выбрал пользователь
* Научились использовать конструкцию try / except

***Тайминг.***

|  |  |  |
| --- | --- | --- |
|  | Раздел урока | Время с начала урока, мин. |
|  | Повторение материала предыдущего урока. Постановка цели урока | 0 |
|  | Создаём кнопки с математическими операциями | 10 |
|  | Самостоятельная работа | 20 |
|  | Генерируем задачу | 30 |
|  | Обрабатываем ответ пользователя | 45 |
|  | Бескомпьютерная деятельность | 65 |
|  | Самостоятельная работа | 70 |
|  | Рефлексия | 85 |