**Пояснительная записка**

**Приложение «Challenge»**

**Краткое описание**

Приложение «Challenge» предназначено для создания учебных заданий, в которых формулировка задания генерируется по шаблону. В приложении две части

TeacherApp – приложение учителя

PupilApp – приложение ученика

Описание характеристик задания: формулировка, характеристики параметров и способ решения описываются в виде класса на языке программирования Python. Из этого следует, что для расширения базы заданий нужно владеть программированием на Python в части объектно-ориентированного программирования. Все задания группируются по темам. Задания одной темы описываются в одном python файле. Вот пример описания темы «Электронные таблицы», в этой теме два задания (файл excel\_table\_.py).

*"""Электронные таблицы"""*from .template import \*  
  
  
class CalcFromIndirectInformation(Task):  
 def statement(self):  
 return **"""В электронной таблице значение формулы =СРЗНАЧ(A1:C1) равно {{a}}. Чему равно значение ячейки D1, если значение формулы =СУММ(A1:D1) равно {{b}}?"""** def calculate(self, \*\*values\_params):  
 return str(self.values\_params[**'b'**] - self.values\_params[**'a'**] \* 3)  
  
 def set\_constraint\_params(self):  
 self.params[**'a'**] = SegmentConstraint(3, 20)  
 self.params[**'b'**] = SegmentConstraint(61, 80)  
  
  
class CalcFromWithPicture(Task):  
 def statement(self):  
 self.task\_image = PREFIX\_FOR\_PATH\_TO\_IMAGE + **'et1.JPG'** return **"""Чему равно значение ячейки D3, если в ячейку D3 введена формула =MAКС({{a}})."""** def calculate(self, \*\*values\_params):  
 data = [[1, 8, 5, 9], [9, 5, 3, 15], [7, 7, 9, None], [None, 24, 3, 6]]  
 the\_range = values\_params[**'a'**].split(**':'**)  
 x1, y1 = int(the\_range[0][1]) - 1, ord(the\_range[0][0]) - ord(**'A'**)  
 x2, y2 = int(the\_range[1][1]) - 1, ord(the\_range[1][0]) - ord(**'A'**)  
 maxi = data[x1][y1]  
 for r in range(x1, x2 + 1):  
 for c in range(y1, y2 + 1):  
 if data[r][c] > maxi:  
 maxi = data[r][c]  
 return str(maxi)  
  
 def set\_constraint\_params(self):  
 self.params[**'a'**] = ListConstraint(**'A1:C3'**, **'A1:C1'**, **'B1:C4'**, **'A1:B3'**, **'B1:D2'**, **'A2:C3'**)

При описании новой темы с заданиями нужно учитывать ряд факторов.

1. Файлы с заданиями должны быть размещены в подкаталоге topic.
2. Название темы на русском языке нужно сделать как doc-строку в начале файла с заданиями этой темы.
3. Все классы, необходимые для описания заданий, размещены в файле template подкаталога topic, их надо импортировать в начале файла описания командой from .template import \*
4. Любой класс задания должен быть унаследован от класса Task.
5. После создания нового файла с заданиями его import нужно добавить в файл all\_topics.py командой from .<имя файла> import \*, например

from .excel\_table\_ import \*

В классе задания обычно достаточно описать три метода:

statement – для описания формулировки задания, в тексте задания нужно выделить параметры. Параметры, это те части задания, которые мы хотим изменять при генерации. Параметры имеют имена и обрамляются двойными фигурными скобками, например **{{a}}**. Если в тексте есть рисунок, то его нужно поместить в подкаталог image, и в свойство класса task\_image записать полное имя файла картинки, например

self.task\_image = PREFIX\_FOR\_PATH\_TO\_IMAGE + **'et1.JPG'**

calculate – для описания процедур и формул определения правильного ответа. Здесь может использоваться весь арсенал языка программирования Python. Все значения параметров находятся в словаре self.values\_params, поэтому, для того, чтобы узнать текущее значение параметра c именем а, достаточно написать self.values\_params[**'a'**].

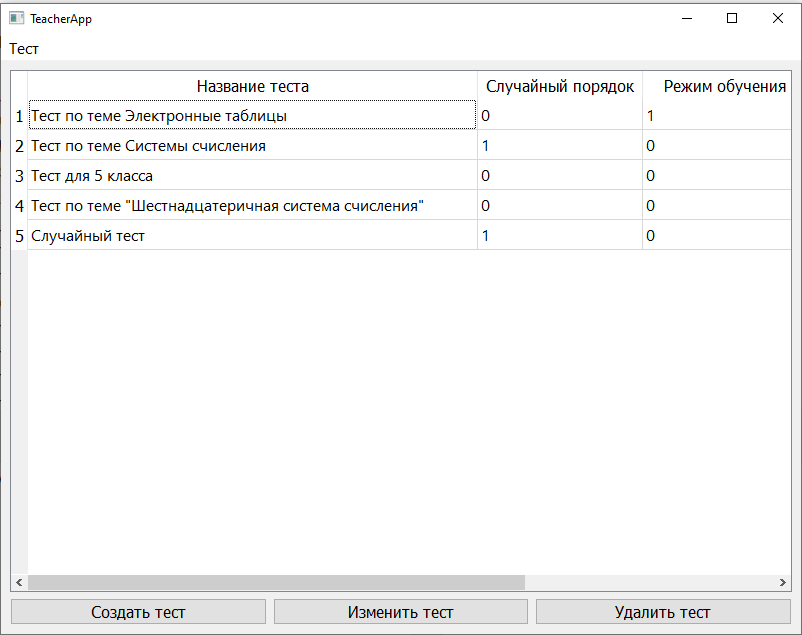
set\_constraint\_params – для описания типов и характеристик параметров задания. Бывают разные типы параметров, они описаны в файле template. Например, параметр, который генерируется как целое число из некоторого отрезка описывается так

self.params[**'a'**] = SegmentConstraint(3, 20)

А параметр, который может принимать значения из фиксированного списка возможных значений, так

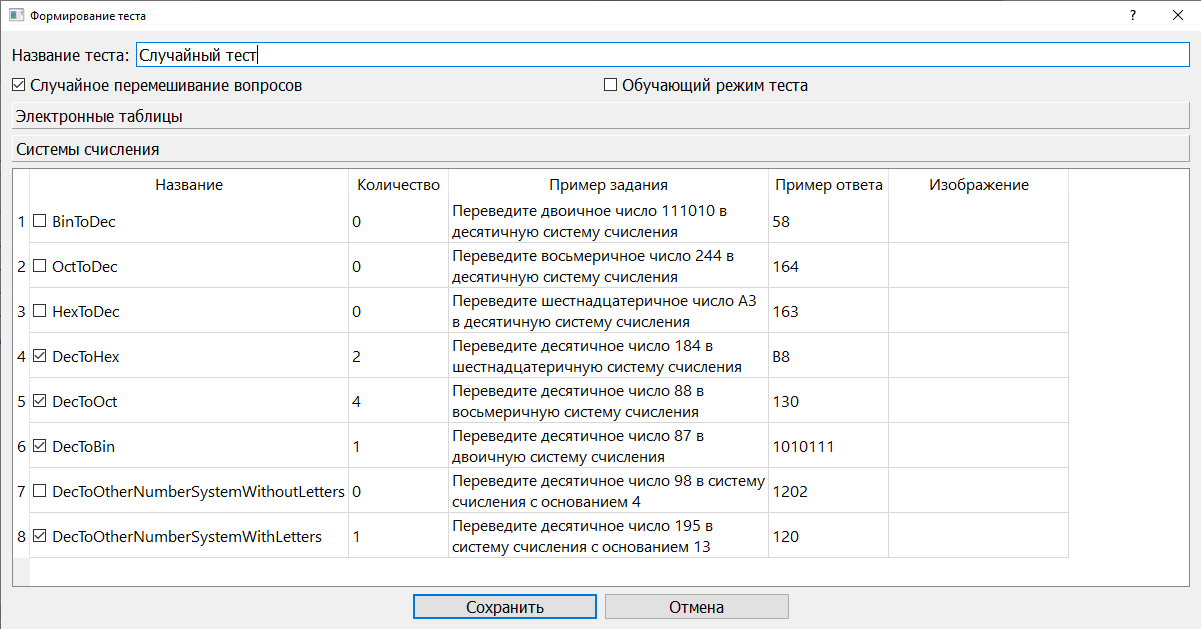
self.params[**'a'**] = ListConstraint(**'A1:C3'**, **'A1:C1'**, **'B1:C4'**, **'A1:B3'**)

Приложение учителя «TeacherApp» находится в подкаталоге TeacherApp и запускается файлом main\_teacher.py. Главное окно учительского приложения отображает список тестов из базы данных. Можно добавить тест в базу данных, изменить или удалить существующий тест. Пример окна главной формы приложения TeacherApp представлен на рисунке ниже.

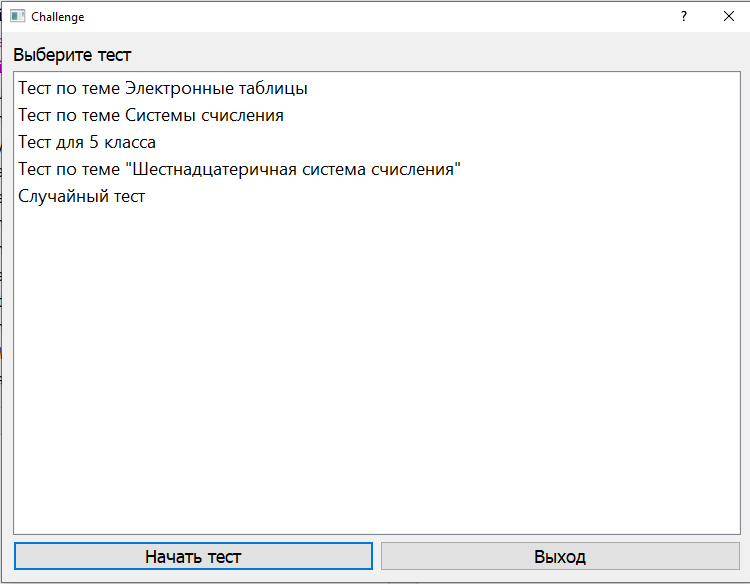


При создании нового теста или изменении параметров существующего теста открывается форма, а которой можно выбрать нужные задания по темам и указать их количество.

Для выбора тем используется контейнер типа «Аккордеон», щелчком на названии темы открываются задания данной темы. Для тех, заданий, которые надо включить в тест, надо поставить «галочки» в списке. При активизации «галочки» автоматически количество заданий данного типа устанавливается в 1. Это количество в дальнейшем можно изменить. Аналогично, изменение данных в столбце «Количество» влечет изменение состояния соответствующего CheckBox.

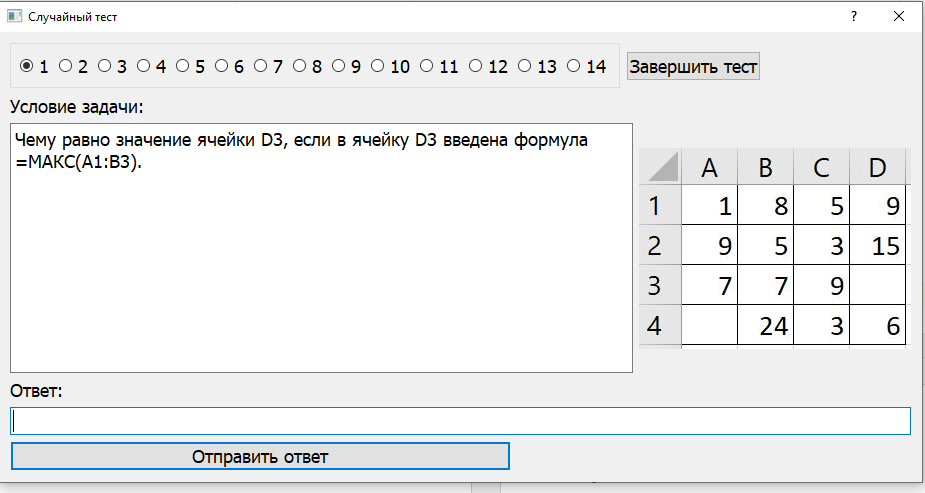


Приложение учителя «PupilApp» находится в подкаталоге PupilApp и запускается файлом main.py. Главное окно приложения ученика отображает список тестов из базы данных, позволяет выбрать тест и начать его выполнение.



При выполнении теста ученик может выполнять задания в произвольном порядке. После нажатия на кнопку «Отправить ответ» дальнейшее изменение ответа становится невозможным. Кнопка «Отправить ответ» сделана кнопкой по умолчанию, поэтому активна при нажатии клавиши Enter. Интерфейс формы генерируется автоматически в зависимости от количества заданий в тесте.

После завершения теста, на форме сохранятся ответы ученика и он сможет проанализировать свои ошибки. Если для теста установлен «Режим обучения», то после завершения теста ученику будут показаны правильные ответы к заданиям.



Важно, что формулировки заданий генерируются «на лету», поэтому можно запускать один и тот же тест многократно и отрабатывать навыки, столько, сколько потребуется. Такой подход к формированию тестов минимизирует возможные списывания и запоминание правильных ответов учащимися, что позволяет повысить качество обучения.

Приложение «Challenge» можно доработать в следующих направлениях:

- расширение базы данных, хранение тем отдельной таблицей и возможность выбора темы перед выбором теста в учительском приложении и приложении учащегося.

- наполнение базы учебных заданий

- дополнение интерфейса приложения TeacherApp таким образом, чтобы автоматизировать создание python-кода при описании заданий.

- создание Web-приложения, идейно совпадающего с данным настольным приложением.