Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Новгородский государственный университет имени Ярослава Мудрого»
Институт непрерывного педагогического образования

Кафедра технологического и художественного образования

Отчёт о выполнении практической работы №2
«Командный хакатон “Продвинутое обучение”»
по дисциплине «Основы прикладного программирования»

Направление (специальность): 44.03.05 Педагогическое образование
(с двумя профилями подготовки: «Технология» и «Информатика»)

Проверил:
ассистент кафедры
ТХО
А.С. Шустров
«30» марта 2023 года

Выполнили:
студенты гр. №9241
Д.Е. Плотников
Т.Н. Спиридонов
«30» марта 2023 года

Великий Новгород

Оглавление

Общая информация о проекте

Название проекта: «Игра “Тематическая виселица”»
Целевая аудитория проекта: учащиеся 7-9 классов
Язык программирования: Python
Среда программирования, в которой разработан проект: PyCharm Community Edition 2022.2.1
Распределение обязанностей по разработке проекта:
Д.Е. Плотников – отвечал за разработку, написание игры и оформление отчёта
Т.Н. Спиридонов – отвечал за консультирование в написании кода и логики игры
Сроки выполнения: 16.03.2023-30.03.2023

Задание на хакатон

Разработать проект «Продвинутое обучение» для тренировки обучающегося любому технологическому навыку.

Обоснование актуальности темы проекта

Мы живём в век цифровизации и развития передовых технологий. Ещё недавно мы не могли даже представить, что в будущем многие из видов работы или деятельности будут выполнятся с применением информационных технологий. Их развитие требует от человека человека знания различных языков программирования, однако многие люди начиная изучать какой-либо язык, теряются при виде неизвестного для них интерфейса программы или незнакомых слов и как следствие, не хотят погружаться в среду програмирования, считая данную деятельность недоступной для себя.

Создание вспомогательной программы для изучения различной терминологии в том числе и программирования, по нашему мнению, наиболее актуально, в будущем мы станем учителями технологии и информатики, и данная программа позволит нам, в нашей деятельности использовать данную программу для изучения и повторения основных понятий по любой из тем в интересной для детей игровой форме.

Для каждого человека любой новый материал воспринимается и усваивается с разной скоростью. Разработанный проект будет полезен для людей разных возрастных групп, основная целевая аудитория — это школьники 7-9 классов поскольку у данной группы потребителей, наиболее эффективный способ усвоения темы - это визуализация и возможность освоить знания на практике, в особенности в игровой форме. У учащихся существует потребность в получении новых, доступных знаний, но они не хотят читать длинные тексты инструкций, гораздо интереснее для них это простая и увлекательная форма подачи материала - игра. Плюс, судя по собственным наблюдениям, учащиеся гораздо активнее и с энтузиазмом подходят к восприятию материала, когда учитель использует в своей деятельности какие-либо инновационные технологии или предлагает работу на компьютере.

Благодаря использованию разработанного приложения повысится уровень усвоения учащимися знаний в терминологии различных пройденных тем, в том числе и программирования, обучающиеся более свободно будут ориентироваться в новой, неизвестной ранее информации, когда будут встречать уже знакомые изученные понятия. Знакомство с интерфейсом программы, позволит подойти к изучению тем по программированию с большим комфортом и без страха. Так же в процессе решения небольшой несложной задачи по разгадыванию букв загаданного слова с ограниченным количеством попыток, повысится уровень мыслительной активности учащихся, сформируется более ответственное отношение к принимаемым решениям. Простота программы, позволяет менять загадываемые слова в зависимости от потребностей занятия.

Данное приложение можно использовать при повторении терминологии разных тем или же для самопроверки знаний. Не менее важным преимуществом проекта является его постепенная дополняемость функций,

так к примеру, можно проработать более наглядный образ сопровождающих изображений, а также возможность давать досрочный ответ - вводить все слово сразу.

Данный проект важен и нужен современным школьникам, в качестве дополнительного тренажёра для изучения материала.

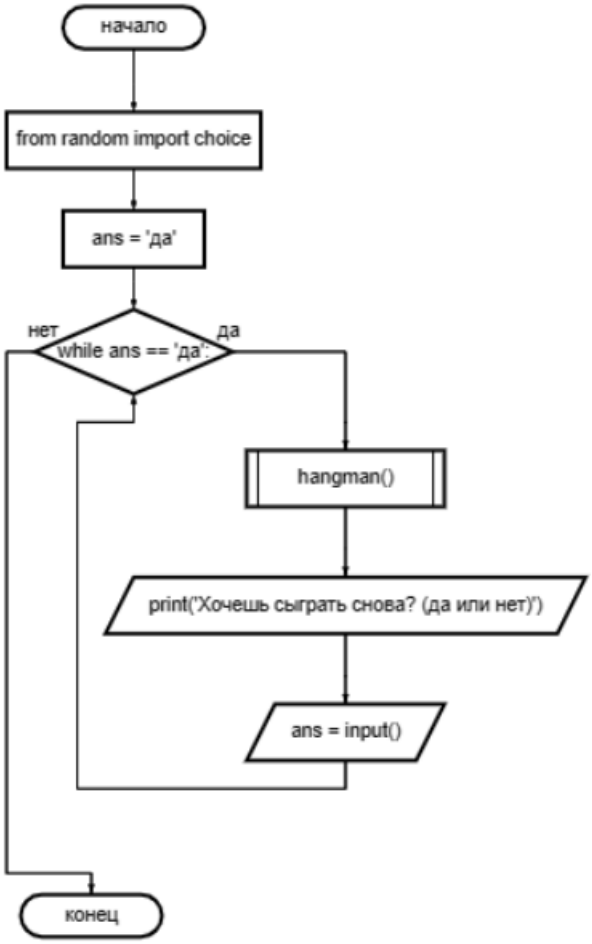
Ментальная карта проекта

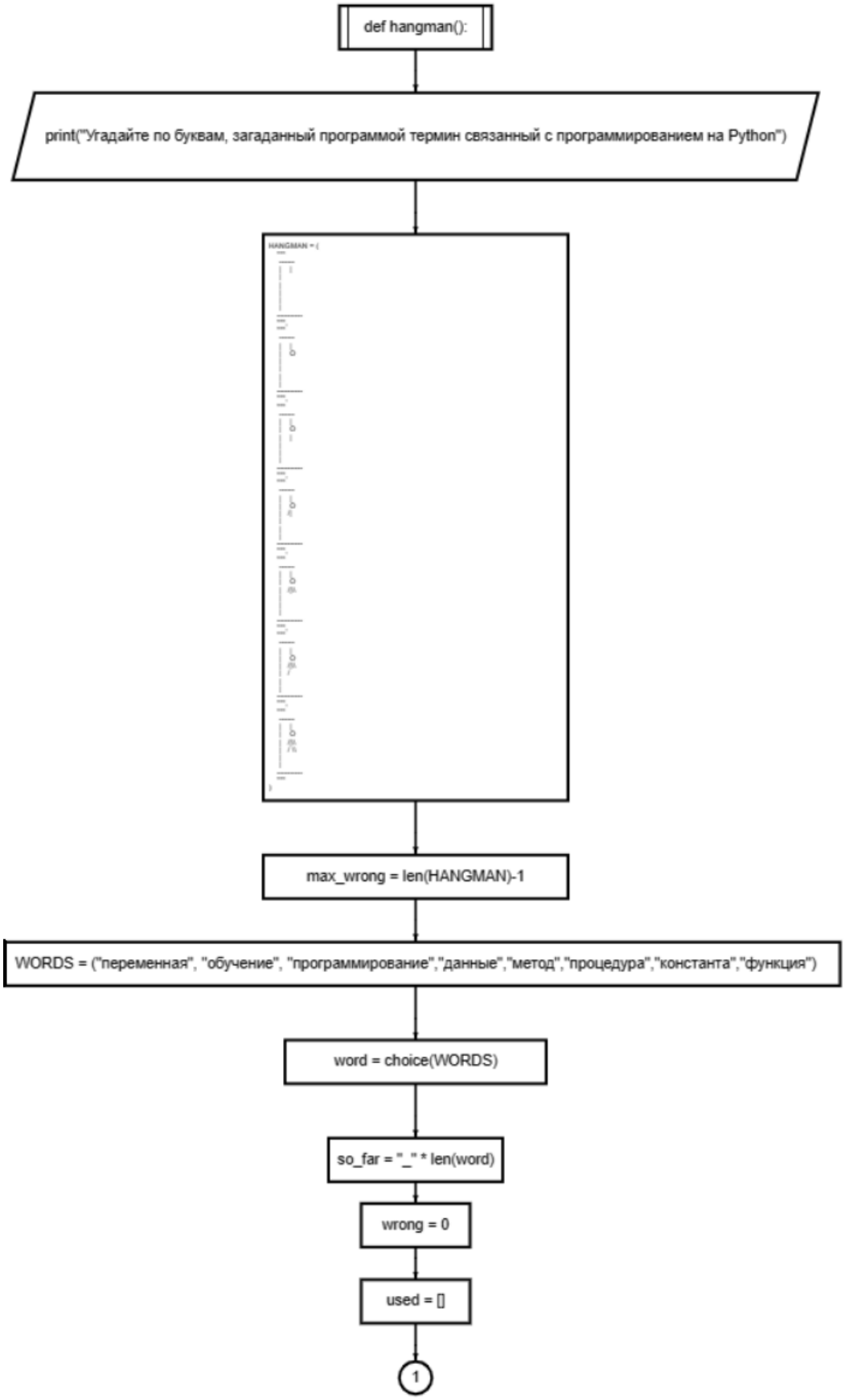
Описание переменных

Таблица №1 – Описание переменных

№	Название переменной	Описание переменной
1.	ans	Хранит в себе значение «да» для запуска игры
2.	hangman()	Название процедуры хранящей код логики игры
3.	max_wrong	количество неверных предположений игрока
4.	WORDS	Слова для угадывания
5.	word	Слово, которое нужно угадать
6.	so_far	Одна черточка в угадываемом слове
7.	wrong	Количество неверных предположений, сделанных игроком
8.	used	Буквы уже угаданы
9.	guess	Хранит предполагаемую пользователем букву
10.	new	Включает в себя введенную пользователем букву и запоминает её

Блок-схема проекта




def hangman():
print("Угадайте по буквам, загаданный программой термин связанный с программированием на Python")
HANGMAN = (
)
max_wrong = len(HANGMAN)-1
WORDS = ("переменная", "обучение", "программирование","данные","метод","процедура","константа","функция")
word = choice(WORDS)
so_far = "_" * len(word)
wrong = 0
used = []
1

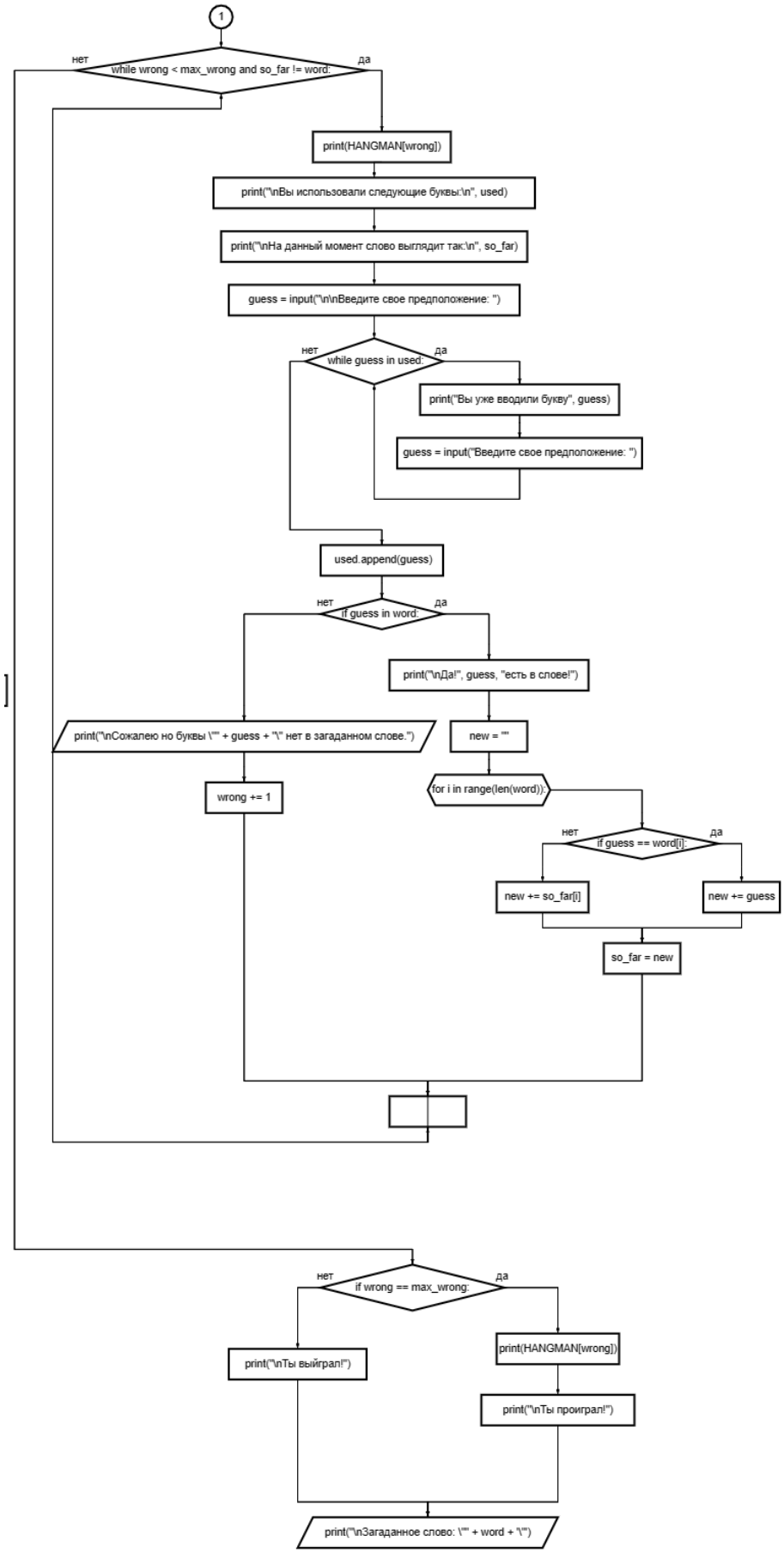


Рис. 1 – Блок-схема проекта

Листинг программы

Таблица №2 – Листинг программы

Номер шага	Программный код	Команды и алгоритмы, которые я использовал(-а)	Описание шага
1.	from random import choice	From Random Import choice	Осуществляем импорт функции «choice» из модуля «random» для выбора случайного слова из списка
2.	def hangman():	def - команда задания процедуры(функции)	Задаем процедуру с именем hangman, хранящую в себе код логики игры
3.	print("Угадайте по буквам, загаданный программой термин связанный с программированием на Python")	print	Вывод на экран пользователя "Угадайте по буквам, загаданный программой термин связанный с программированием на Python"
4.	HANGMAN = ()	присвоение	Присваиваем переменной HANGMAN значение массива из строк(граф изображение картинок) Сами картинки не добавлены, так как они громоздки и возможно в будущем будут реализованы через привлечение другой программы.
5.	max_wrong = len(HANGMAN) - 1	len	Присваиваем переменной max_wrong длину массива строк заключенных в переменной HANGMAN начиная не с нулевой, а с первой попытки которая использована поэтому -1
6.	WORDS = ("переменная", "обучение", "программирование","данные","метод","процедура", "константа","функция")	присваивание	Присваиваем переменной WORDS слова из которых программа буде выбирать одно случайное и загадывать его пользователю
7.	word = choice(WORDS)	Присвоение choice - функция выбора случайного элемента	Присваиваем переменной word случайное слово хранящееся в переменной WORDS
8.	so_far = "_" * len(word)	* len (происходит умножение строчного	Присваиваем переменной so_far значение взятого слова каждый символ которого заменён на тире

		элемента «_» на число символов в слове(длину))	
9.	wrong = 0	присвоение	Задаём переменную wrong хранящую количество ошибочно введённых букв
10.	used = []	Присвоение []- список	Присваиваем переменной used пустой список где будут храниться все введённые буквы пользователем
11.	while wrong < max_wrong and so_far != word:	While - пока условие выполняется... неравенство	Задаём цикл с условием, пока ошибок меньше чем максимально возможное количество элементов хранящихся в переменной max_wrong и пока переменная so_far не равна самому слову содержащемуся в переменной word
12.	print(HANGMAN[wrong]) print("\nВы использовали следующие буквы:\n", used) print("\nНа данный момент слово выглядит так:\n", so_far)	Вывод \n	Вывод на экран «элемента из списка содержащегося в переменной HANGMAN по индексу» и номеру неверной попытки(то-есть номер неверной попытки это и есть индекс), вывод букв которые использовались ранее (содержащиеся вписке used), и соответственно вывод зашифрованного слова в переменной so_far виде тире.
13.	guess = input("\n Введите свое предположение: ") # Пользователь вводит предполагаемую букву	Присвоение \n	Присваиваем переменной guess вводимую пользователем букву
14.	while guess in used:	while	Задаем цикл с условием. Пока введённая буква(guess) есть в списке used
15.	print("Вы уже вводили букву", guess)	Вывод input	Программа выводит сообщение, вы уже вводили данную букву guess, если буква уже вводилась ранее, то выводим соответствующее сообщение
16.	guess = input ("Введите своё предположение: ")		запрашивает ввести новую (идет новое присваивание переменной guess вводимого значения) Цикл повторяется пока не будет введена буква которой ещё нет в списке
17.	used.append(guess) # В список использованных	.append - добавление в конец списка	Добавляется в список used введённая пользователем новая буква

	букв добавляется введённая буква		
18.	if guess in word:	If - условие если	Условие, если введённая буква (хранящаяся в переменной guess) находится в загаданном слове
19.	print("\nДа!", guess, "есть в слове!")	Вывод \n	Вывод на экран Да!", буква(guess), "есть в слове!
20.	new = ""	Присвоение Пустое значение	Задаём переменную с пустым значением в которой будут хранится введённые буквы
21.	for i in range(len(word)):	For I in range len	Для элемента в рамках длинны слова в символах
22.	if guess == word[i]:	If [i]-обращение по индексу	Если введённая буква хранящаяся в переменной guess равна(есть где то в слове)
23.	new += guess	+= операция сложения(приклеивания)	В переменную new запишется эта буква
24.	else:	else - иначе	иначе
25.	new += so_far[i]	+= - приклеивание [i]-обращение по индексу	В переменную нью new запишется значение переменно so_far в тот индекс где эта буква стоит в слове
26.	so_far = new	присваивание	Переменной so_far присваивается полученное в результате значение переменной new
27.	else:	else	иначе
28.	print("\nИзвините, буквы \"" + guess + "\" нет в слове.")	Вывод на экран	Вывод на экран: Извините, введённой буквы guess нет в слове
29.	wrong += 1	+= - операция сложения(счетчик)	Счетчик (значение переменной wrong увеличится на 1)
30.	if wrong == max_wrong:	If - если == - равенство	Если значение переменной wrong равно максимально допустимым ошибкам(количеству элементов в переменной max_wrong)
31.	print(HANGMAN[wrong]) print("\nТы проиграл!")	Вывод на экран с индексом обращения	Выводит на экран последнюю картинку по индексу элементов в списке HANGMAN равное максимально допустимому количеству неверных попыток wrong и надпись ты проиграл
32.	else:	else	иначе

33.	print("\nВы угадали слово!")	else - иначе	Если кол-во ошибок не превышено, то на экран выводится: Ты выиграл
34.	print("\nЗагаданное слово было \"" + word + '\"')	Вывод на экран	На экран в консоль выводится загаданное программой слово хранящееся в переменной word
35.	ans = 'да'	присваивание	Создаем переменную ans, которой присваиваем значение «да»
36.	while ans == 'да':	While ==	Задаем цикл пока условие истинно, то есть значение переменной ans равно «да»
37.	hangman()	hangman() - вызов процедуры по имени	Вызываем процедуру hangman()
38.	print('Хочешь сыграть снова? (да или нет)')	print	Вывод на экран пользователя строки: Хочешь сыграть снова? (да или нет)
39.	ans = input()	input	Запрос на ввод с клавиатуры ответа на заданный вопрос, с последующим новым присваиванием значения переменной ans

Программный код проекта в Приложении А

Описание входных данных и результат вычислений

Таблица №3 - Описание входных данных и результат вычислений

Входные данные	Выходные данные
Угадайте по буквам, загаданный программой термин связанный с программированием на Python ------ | | | | | | | ---------- Вы использовали следующие буквы: [] На данный момент слово выглядит так: _____ Введите свое предположение:	Сожалею, но буквы "м" нет в загаданном слове. ------ | | | O | | | | ---------- Вы использовали следующие буквы: ['м'] На данный момент слово выглядит так: _____ Введите свое предположение:
Введите свое предположение:	Введите свое предположение: д Да! д есть в слове! ------ | | | O | | | | ---------- Вы использовали следующие буквы: ['м', 'д'] На данный момент слово выглядит так: д_____ Введите свое предположение:

------ | | | O | | | | ---------- Вы использовали следующие буквы: ['м', 'д', 'а', 'н', 'ы'] На данный момент слово выглядит так: данны_	Введите свое предположение: е Да! е есть в слове! Ты выиграл! Загаданное слово: "данные"
Введите свое предположение: к	Сожалею, но буквы "к" нет в загаданном слове. ------ | | | O | /|\ | / \ | | ---------- Ты проиграл! Загаданное слово: "процедура"
Хочешь сыграть снова? (да или нет) нет	Process finished with exit code 0
Хочешь сыграть снова? (да или нет) да	Угадайте по буквам, загаданный программой термин связанный с программированием на Python ------ | | | | | | | ---------- Вы использовали следующие буквы: [] На данный момент слово выглядит так: _____ Введите свое предположение:

Варианты продвижения проекта в Интернете

Возможности продвижения нашего проекта «Виселица» - это, прежде всего, создание группы или беседы в социальной сети «ВКонтакте», где учителя смогут обсуждать какие-то вопросы по работе в программе, пожелания по улучшению функционала. Также появится возможность обмена дидактическим и диагностическим материалом среди учителей информатики. Будет создан видео обзор функционала, по программированию данной игры «Виселица». Очень важно показать систематику данной игры, так как на её основе можно придумать и другие «текстовые» игры.

В качестве какого-либо удобного формата демонстрации проекта, можно создать презентацию об особенностях игры, плюсах и минусах, возможностях, открывающихся для учащихся и учителей при использовании данной программы, а также подробная информация по её созданию - программированию.

Также можно создать специализированный сайт для учителей и, возможно, учащихся, где будут размещены материалы по созданию похожих игр на языке Python, где при желании можно будет поделиться своей разработанной игрой.

В качестве улучшения проекта в будущем может быть реализация графического интерфейса, что бы игра была более наглядной и интересной.

Материально-техническая база

Таблица №4 – Материально-техническая база

Тип	Полное название
Программное обеспечение	PyCharm Community Edition 2022.2.1
Персональный компьютер	Имя устройства LAPTOP-4CR7NPUK Процессор Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz 2.70 GHz Оперативная память 6,00 ГБ (доступно: 5,87 ГБ) Код устройства 4910DBAD-6501-4EB4-A06F-B708EF685917 Код продукта 00327-30504-19560-AAOEM Тип системы 64-разрядная операционная система, процессор x64 Перо и сенсорный ввод Для этого монитора недоступен ввод с помощью пера и сенсорный ввод

Программный код проекта

```
from random import choice
def hangman():
    print("Угадайте по буквам, загаданный программой термин связанный с программированием на Python")
    HANGMAN = (
    """
     ------
     |   |
     |
     |
     |
     |
     |
    ----------
    """,
    """
     ------
     |   |
     |   O
     |
     |
     |
     |
    ----------
    """,
    """
     ------
     |   |
     |   O
     |   |
     |
     |
     |
    ----------
    """,
    """
     ------
     |   |
     |   O
     |  /|
     |
     |
     |
    ----------
    """,
    """
     ------
```

```
    |   |
    |   O
    |  /|\\
    |
    |
    |
   ----------
   """,
   """
    ------
    |   |
    |   O
    |  /|\\
    |  /
    |
    |
   ----------
   """,
   """
    ------
    |   |
    |   O
    |  /|\\
    |  / \\
    |
    |
   ----------
   """
   )
   max_wrong = len(HANGMAN) - 1
   WORDS = ("переменная", "обучение", "программирование", "данные", "метод",
"процедура", "константа",
            "функция")  # Слова для угадывания

   word = choice(WORDS)
   so_far = "_" * len(word)
   wrong = 0
   used = []
   while wrong < max_wrong and so_far != word:
      print(HANGMAN[wrong])
      print("\nВы использовали следующие буквы:\n", used)
      print("\nНа данный момент слово выглядит так:\n", so_far)

      guess = input("\nВведите свое предположение: ")
      while guess in used:
         print("Вы уже вводили букву", guess)
         guess = input("Введите свое предположение: ")
      used.append(guess)

      if guess in word:
         print("\nДа!", guess, "есть в слове!")
         new = ""
```

```
            for i in range(len(word)):
                if guess == word[i]:
                    new += guess
                else:
                    new += so_far[i]
            so_far = new
        else:
            print("\nСожалею но буквы \"" + guess + "\" нет в загаданном слове.")
            wrong += 1
    if wrong == max_wrong:
        print(HANGMAN[wrong])
        print("\nТы проиграл!")
    else:
        print("\nТы выйграл!")

    print("\nЗагаданное слово: \"" + word + '\"')



ans = 'да'
while ans == 'да':
    hangman()
    print('Хочешь сыграть снова? (да или нет)')
    ans = input()
```