ВСЕРОССИЙСКИЙ КОНКУРС
НАУЧНО-ИССЛЕДОВАТЕЛЬСКИХ И ТВОРЧЕСКИХ РАБОТ МОЛОДЁЖИ
«МЕНЯ ОЦЕНЯТ В XXI ВЕКЕ»

Направление: Педагогика

КОНКУРСНАЯ РАБОТА

РАЗРАБОТКА ПРАКТИЧЕСКИХ ЗАДАНИЙ ПО ПРОГРАММИРОВАНИЮ НА ЯЗЫКЕ PYTHON С ИСПОЛЬЗОВАНИЕМ СИМУЛЯТОРА ПОЛЕТА БПЛА

Соискатель: Копанева Алина Алексеевна,
обучающаяся 5 курса очной формы обучения

Научный руководитель: Бунаков П. Ю.,
доктор технических наук, профессор

Место выполнения работы:
Государственное образовательное учреждение высшего образования Московской области
«Государственный социально-гуманитарный университет»
Факультет математики, физики, химии, информатики

Коломна
2026

Аннотация

В работе представлен комплекс практических заданий по программированию на языке Python с использованием учебного браузерного симулятора PyDrone Lab. Разработка направлена на организацию последовательной работы с переменными, списками, циклами, условиями и функциями через управление виртуальным беспилотным летательным аппаратом.

Основной результат состоит в объединении шести учебных миссий, заданий разной сложности и критериев оценивания. Для каждой миссии предусмотрен переход от запуска готового примера к изменению параметров и самостоятельному составлению программы. Виртуальный полигон позволяет сопоставлять код с координатами, высотой и траекторией движения. В работе описана проверка базового сценария взлета, ожидания и посадки.

Практическая значимость разработки заключается в возможности использования комплекса на занятиях по информатике и во внеурочной деятельности. Симулятор предназначен для освоения алгоритмов и не воспроизводит точную физику полета. Выводы ограничены результатами разработки учебных материалов и проверкой отдельного сценария; оценка влияния комплекса на учебные достижения требует дальнейшей апробации.

Ключевые слова: Python, практические задания, обучение программированию, виртуальный БПЛА, PyDrone Lab.

Содержание

Введение 3
1 Основы разработки практических заданий 4
1.1 Практические задания в обучении Python 4
1.2 Возможности учебного симулятора PyDrone Lab 5
1.3 Требования к содержанию и последовательности заданий 7
2 Комплекс практических заданий 8
2.1 Система заданий и уровни сложности 8
2.2 Задания на основные конструкции Python 9
2.3 Маршрутные и проектные задания 13
2.4 Результаты разработки и границы их применения 15
Заключение 16
Список использованной литературы 17
Приложение 1 Инструкция по работе с PyDrone Lab 18
Приложение 2 Карточки практических заданий 19
Приложение 3 Пример итоговой программы 23
Приложение 4 Устройство учебного симулятора 24

Введение

Изучение программирования предполагает переход от понимания отдельных команд к самостоятельному составлению, запуску и исправлению программ. При выполнении однотипных вычислительных упражнений связь между текстом программы и поведением управляемого объекта может оставаться недостаточно наглядной. Поэтому представляет интерес использование виртуального исполнителя, движение которого определяется кодом на языке Python.

В федеральной рабочей программе по информатике для 7–9 классов предусмотрена работа с алгоритмами, программами и исполнителями [3]. Разработанный комплекс связан с этими направлениями содержания: изменение переменной задает новую координату, цикл обеспечивает прохождение нескольких точек, а условный оператор позволяет выбрать маршрут. Визуальное представление результата дополняет анализ исходного кода.

Проблема работы заключается в необходимости создания понятной последовательности практических заданий, связывающих основные конструкции Python с управлением виртуальным БПЛА. Отдельного демонстрационного примера недостаточно: требуется система упражнений, в которой постепенно увеличиваются сложность алгоритма и самостоятельность обучающегося.

Цель работы: разработать комплекс практических заданий по программированию на языке Python с использованием симулятора полета БПЛА.

Объект работы: процесс обучения программированию на языке Python с использованием виртуального исполнителя.

Предмет работы: содержание, структура и последовательность практических заданий по управлению виртуальным БПЛА.

Для достижения цели были поставлены следующие задачи:

1. Рассмотреть роль практических заданий в обучении программированию и возможности учебного симулятора.
2. Определить требования к содержанию упражнений, их последовательности и проверке результата.
3. Разработать систему миссий на применение переменных, списков, циклов, условий и функций.
4. Подготовить маршрутные и проектные задания, критерии оценивания и инструкцию по работе.
5. Описать результаты разработки и проверки базового сценария.

В работе использовались анализ учебно-методической и технической литературы, сравнение, классификация, педагогическое проектирование, моделирование учебных ситуаций, разработка программного продукта и практическая проверка отдельного сценария.

Практическим результатом является комплекс упражнений для PyDrone Lab с примерами кода, самостоятельными вариантами и критериями оценивания. Его можно использовать на уроках информатики, факультативных занятиях и во внеурочной деятельности. Авторский вклад представлен содержанием миссий, их последовательностью и объединением заданий с учебной средой.

1 Основы разработки практических заданий

1.1 Практические задания в обучении Python

Практическое задание по программированию представляет собой учебную задачу, для решения которой необходимо составить или изменить программу и получить проверяемый результат. Его назначение связано с применением изученного материала, выбором алгоритма и анализом ошибок. Общие вопросы организации обучения информатике раскрываются в учебном пособии А. А. Кузнецова, Т. Б. Захаровой и А. С. Захарова [1].

В разработанном комплексе использован единый исполнитель – виртуальный БПЛА. Такой выбор позволяет сохранять знакомый контекст при переходе к новой конструкции языка. Сначала аппарат выполняет последовательность команд. Затем координаты задаются через переменные, маршрут сохраняется в списке, а повторяющиеся действия оформляются циклом.

При изучении переменной существенна связь между ее значением и наблюдаемым результатом. Изменение координаты приводит к изменению конечной точки. При изучении цикла становится заметно, что одинаковые команды не требуется многократно записывать вручную. Функция позволяет выделить повторно используемый фрагмент маршрута. Для школьного уровня такой переход согласуется с постепенным освоением программирования [2].

Таблица 1 - Формы практических заданий

Форма задания	**Действия обучающегося**	**Назначение**
По образцу	Запуск готового кода и изменение параметров	Знакомство с командами
С пропусками	Дополнение отдельных строк	Закрепление конструкции
По условию	Самостоятельное составление алгоритма	Выбор способа решения
Маршрутная задача	Объединение нескольких команд и конструкций	Применение изученного
Проектная задача	Создание собственной полетной миссии	Комплексная проверка умений

Визуальный результат не должен заменять объяснение программы. После запуска важно установить, какая строка задала высоту, почему сработала определенная ветвь и как изменился маршрут после редактирования списка. Поэтому условия заданий включают самостоятельные изменения и критерии проверки, а не ограничиваются воспроизведением образца.

Работа с ошибками включается в само выполнение упражнения. Обучающийся сопоставляет ожидаемое и фактическое движение, находит причину расхождения, изменяет программу и повторяет запуск. В результате программирование рассматривается как последовательность взаимосвязанных действий.

1.2 Возможности учебного симулятора PyDrone Lab

Для выполнения комплекса был разработан PyDrone Lab - браузерная учебная среда, в которой Python-код управляет виртуальным БПЛА. В одном интерфейсе размещены описание миссии, координатный полигон, редактор программы и консоль. Общий вид среды представлен на рисунке 1.

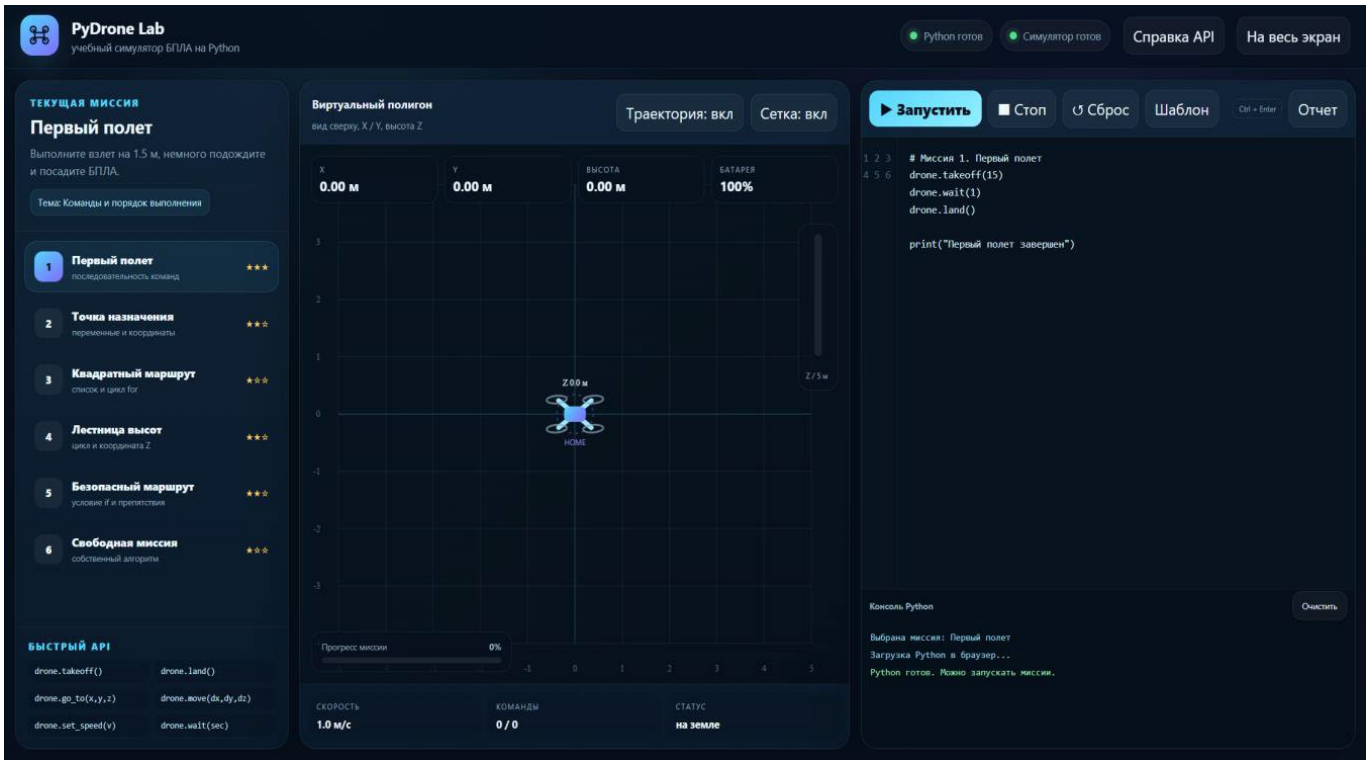


Рисунок 1 - Интерфейс учебного симулятора PyDrone Lab

В левой части окна выбирается миссия и отображается ее условие. В центре расположен полигон с координатной сеткой и моделью аппарата. Справа находятся редактор Python-кода и консоль. Панель управления позволяет запустить выполнение, остановить его и сбросить состояние. Показатели координат и высоты используются для сопоставления задания с движением аппарата.

Положение на плоскости определяется координатами x и y, высота - координатой z. Отображение траектории помогает проследить порядок прохождения точек. При изменении значений в программе можно сравнивать несколько вариантов маршрута в одной и той же учебной ситуации.

Графическая часть среды реализована с использованием Canvas API, предназначенного для рисования средствами JavaScript [5]. Выполнение Python в браузере обеспечивается Pyodide [6]. Это позволяет объединить ввод программы и просмотр результата в окне браузера. Для загрузки используемой базовой версии среды требуется подключение к сети.

PyDrone Lab представляет собой упрощенную учебную модель. В ней не воспроизводятся точная аэродинамика, воздействие ветра и характеристики двигателей. Такое ограничение соответствует назначению разработки: основное внимание уделяется алгоритму, структуре программы и проверке результата.

Команды управления и обратная связь

Для управления виртуальным аппаратом используется объект drone. Команды этого объекта относятся к интерфейсу PyDrone Lab и не являются встроенными командами языка Python. Они доступны в подготовленной учебной среде. Основные команды представлены в таблице 2.

Таблица 2 - Команды учебного исполнителя

Команда	**Назначение**
drone.takeoff(height)	Взлет на указанную высоту
drone.land()	Посадка
drone.go_to(x, y, z)	Перемещение в заданную точку
drone.move(dx, dy, dz)	Перемещение относительно текущего положения
drone.set_speed(value)	Установка скорости
drone.wait(seconds)	Ожидание в течение заданного времени
drone.home()	Возвращение в исходную точку
drone.say(text)	Вывод сообщения

Команды объединяются с обычными конструкциями Python. Например, координаты могут храниться в переменных, а последовательность точек - в списке. Внутри цикла происходит обращение к очередной точке, после чего вызывается команда перемещения. Синтаксис циклов, условий и функций соответствует описанию языка Python [4].

После завершения миссии предусмотрена автоматическая проверка отдельных условий. В задании «Первый полет» контролируются признаки взлета и посадки, а также завершение выполнения на земле. Результат отображается в виде звезд. Такая обратная связь помогает заметить невыполненное условие, но не заменяет оценку читаемости кода и объяснение алгоритма.

В ходе подготовки работы был проверен базовый сценарий «Первый полет»: виртуальный аппарат выполнил последовательность взлета, ожидания и посадки; системой были отмечены три критерия выполнения. Этот результат относится к одному сценарию и не используется как подтверждение проверки всех возможностей среды.

При дальнейшей работе необходимо учитывать особенности браузерной среды Python и ограничения доступных библиотек [7]. Для представленных упражнений используются основные конструкции языка и команды учебного исполнителя. Подключение реального БПЛА для выполнения заданий не требуется.

1.3 Требования к содержанию и последовательности заданий

При разработке комплекса были определены требования, позволяющие связать учебную цель с конкретным действием в симуляторе. Первое требование – это соответствие изучаемой теме. В задании на переменные основной трудностью должно быть использование значений, а не сложный выбор траектории. В упражнении на цикл существенным становится повторение действий.

Второе требование – понятность и проверяемость условия. Вместо общей формулировки «запрограммировать полет» задаются высота, координаты или число повторений. Например, требуется взлететь на 1,5 м, пройти заданную точку и выполнить посадку. Эти условия позволяют сравнить программу с ожидаемым результатом.

Третье требование – постепенное усложнение. Знакомство начинается с последовательных команд. Затем добавляются переменные, списки, циклы, условия и функции. В базовом упражнении вводится ограниченное число новых элементов, чтобы причину ошибки можно было установить без одновременного разбора нескольких незнакомых конструкций.

Таблица 3 - Уровни сложности заданий

Уровень	**Характер задания**	**Самостоятельность**
Базовый	Запуск образца и изменение параметров	Работа с опорой на готовый код
Средний	Дополнение или изменение алгоритма	Самостоятельное редактирование
Повышенный	Выбор конструкции и варианта маршрута	Обоснование способа решения
Комплексный	Разработка собственного сценария	Самостоятельная программа

Четвертое требование – возможность повторной проверки. После неудачного запуска программа должна быть доступна для редактирования. Пятое - постепенное уменьшение объема подсказок: от полного образца к условию, для которого алгоритм выбирается самостоятельно.

Шестое требование связано с возможностями полигона. Координаты, высота и расположение препятствий должны соответствовать выбранной миссии. Изменение маршрута оценивается по фактической траектории. Наличие оператора if само по себе не доказывает безопасность движения.

Принятые требования определили структуру комплекса: каждое упражнение содержит тему, условие, исходный пример или ориентир, самостоятельную часть и критерии. Завершающее задание объединяет ранее изученные элементы.

2 Комплекс практических заданий

2.1 Система заданий и уровни сложности

Основной комплекс включает шесть миссий. Последовательность построена так, чтобы результат предыдущего задания становился опорой для следующего. После взлета и посадки вводится перемещение в точку, затем маршрут из нескольких точек, изменение высоты, выбор пути и самостоятельная полетная программа.

Таблица 4 - Содержание основного комплекса

Миссия	Тема Python	Основное действие	Уровень
Первый полет	Последовательность команд	Взлет, ожидание, посадка	Базовый
Точка назначения	Переменные	Перемещение по координатам	Базовый
Квадратный маршрут	Список и цикл for	Последовательный обход точек	Средний
Лестница высот	Цикл и координата z	Изменение высоты на маршруте	Средний
Безопасный маршрут	Условный оператор if	Выбор варианта движения	Повышенный
Свободная миссия	Функции, списки и циклы	Создание собственного сценария	Комплексный

Для каждой миссии предусмотрены три этапа. Сначала выполняется исходный пример и объясняется его результат. Затем изменяется один или несколько параметров. После этого выполняется самостоятельное задание. Такая организация позволяет использовать один материал с обучающимися разного уровня подготовки.

Например, в миссии «Первый полет» сначала запускается программа с высотой 1,5 м. Затем изменяются высота и длительность ожидания. В самостоятельной части требуется объяснить порядок команд и проверить завершение полета. В маршрутных заданиях изменяются не только числа, но и структура списка или способ повторения действий.

Переход к следующему уровню связывается с пониманием решения. Успешный запуск готового кода рассматривается как начальный шаг. Для проверки самостоятельности используются изменение условия, объяснение выбранной конструкции и поиск ошибки в программе.

В следующих разделах приведены основные миссии, маршрутные задания и итоговые критерии. Карточки для самостоятельной работы размещены в приложении 2. Команды приведены для PyDrone Lab. Перед повторным запуском симулятор возвращается в исходное состояние.

2.2 Задания на основные конструкции Python

Задание 1. Первый полет

Тема: последовательное выполнение команд. Цель: установить связь между порядком строк и действиями исполнителя. Требуется выполнить взлет на высоту 1,5 м, сделать паузу в одну секунду и посадить аппарат.

```
drone.takeoff(1.5)
drone.wait(1)
drone.land()
print("Первый полет завершен")
```

Ожидаемый результат: аппарат поднимается на заданную высоту, некоторое время остается в воздухе и завершает полет на земле. После запуска сопоставляются команды взлета, ожидания и посадки с наблюдаемыми действиями.

Самостоятельная часть: изменить высоту на 2,5 м, а длительность ожидания — на две секунды. Критерии выполнения: сохранен правильный порядок команд, использованы новые параметры, выполнена посадка.

Задание 2. Точка назначения

Тема: переменные. Цель: сохранить координаты в переменных и использовать их при вызове команды перемещения. Требуется пройти точку x = 3, y = 2, z = 1,5 и выполнить посадку.

```
x = 3
y = 2
z = 1.5
drone.takeoff(z)
drone.go_to(x, y, z)
drone.land()
```

Самостоятельная часть: изменить значения так, чтобы аппарат достиг точки x = −2, y = 3, z = 2. Перед запуском необходимо предсказать расположение точки на полигоне, а после выполнения сопоставить прогноз с траекторией.

Критерии выполнения: координаты заданы через переменные, достигнута нужная точка, выполнена посадка. Координата z проверяется во время движения, поскольку после посадки высота становится равной нулю. Практический смысл переменной раскрывается через изменение поведения исполнителя без переписывания всей программы.

Задания 3 и 4 Списки и циклы

Задание 3. Квадратный маршрут

Тема: список и цикл for. Требуется построить квадрат со стороной 3 м на высоте 1,5 м. При исходном положении x = 0, y = 0 последовательно проходятся четыре точки.

```
points = [(3, 0, 1.5), (3, 3, 1.5),
          (0, 3, 1.5), (0, 0, 1.5)]
drone.takeoff(1.5)
for x, y, z in points:
   drone.go_to(x, y, z)
drone.land()
```

Самостоятельная часть: уменьшить сторону до 2 м, затем составить список для прямоугольника размером 4 × 2 м. Критерии: используется список и цикл for, точки пройдены в заданном порядке, получена замкнутая траектория. Перестановка элементов списка позволяет показать зависимость формы маршрута от порядка данных.

Задание 4 Лестница высот

Тема: цикл и изменение координаты z. Требуется последовательно пройти три точки с увеличением высоты от 1 до 3 м.

```
points = [(1, 0, 1.0), (2, 1, 2.0), (3, 2, 3.0)]
drone.takeoff(1.0)
for point in points:
   drone.go_to(*point)
drone.land()
```

Запись *point передает три элемента очередной точки в качестве отдельных аргументов команды. На начальном этапе допускается более явная форма: for x, y, z in points с вызовом drone.go_to(x, y, z). Это позволяет сосредоточиться на цикле, если распаковка аргументов еще не изучалась [4].

Самостоятельная часть: добавить две точки так, чтобы после достижения 3 м аппарат постепенно снизился до 1 м. Критерии: пройдены все контрольные точки, достигнута максимальная высота 3 м, изменение высоты выполнено до посадки, используется цикл. Проекция маршрута на плоскость дополняется наблюдением за показателем высоты.

Задание 5. Безопасный маршрут

Тема: условный оператор if. Цель: связать логическое условие с выбором пути. На учебном полигоне рассматривается препятствие высотой 2,2 м. Требуется достичь точки B с координатами x = 4, y = 3. Если выбранная высота превышает высоту препятствия, используется прямое перемещение; иначе выполняется обход по указанным промежуточным точкам.

```
height = 1.5
drone.takeoff(height)
if height > 2.2:
    drone.go_to(4, 3, height)
else:
    drone.go_to(0, 3.5, height)
    drone.go_to(4, 3.5, height)
    drone.go_to(4, 3, height)
drone.land()
```

В примере явно выделены два варианта решения. При значении height = 1.5 выполняется ветвь else, при height = 2.5 - первая ветвь. Равенство высоты значению 2,2 м не считается достаточным условием для пролета над препятствием. Это позволяет обсудить выбор знака сравнения и пограничное значение.

Самостоятельная часть: сравнить выполнение программы при высотах 1,5; 2,2 и 2,5 м. Перед каждым запуском требуется определить, какая ветвь будет выполнена. Затем необходимо проверить траекторию и объяснить полученный результат.

Таблица 5 - Ожидаемый выбор ветви программы

Высота	**Результат проверки height > 2.2**	**Выбранный вариант**
1,5 м	Ложь	Обход по промежуточным точкам
2,2 м	Ложь	Обход по промежуточным точкам
2,5 м	Истина	Прямое перемещение

Заданные точки обхода применяются для соответствующей конфигурации учебного полигона. При другом расположении препятствий их необходимо изменить. Таблица отражает ожидаемую логику программы, а не протокол дополнительных выполненных испытаний.

Критерии выполнения: условие связано с высотой, правильно определена выполняемая ветвь, достигнута конечная точка, по фактическому движению не зафиксировано столкновение, выполнена посадка. Проверка включает чтение кода и наблюдение за маршрутом.

Задание 6 Свободная миссия

Тема: пользовательские функции и комплексное применение конструкций Python. Цель: выделить алгоритм движения в функцию и использовать параметры для изменения маршрута. Исходный пример задает треугольный маршрут с возвращением в точку x = 0, y = 0.

```
def fly_triangle(size, z):
    points = [(size, 0, z), (0, size, z), (0, 0, z)]
    for point in points:
        drone.go_to(*point)

drone.set_speed(1.4)
drone.takeoff(1.5)
fly_triangle(3, 1.5)
drone.say("Маршрут завершен")
drone.land()
```

Параметр size определяет размеры маршрута, а z — высоту. Определение функции отделено от ее вызова. Благодаря этому маршрут можно изменять через аргументы, сохраняя основной алгоритм. При разборе примера уточняется, что координаты точек абсолютные и отсчитываются в системе координат полигона.

Самостоятельная часть: составить функцию fly_rectangle(length, width, z), принимающую длину, ширину и высоту. Внутри функции должен формироваться список точек прямоугольника и выполняться их последовательный обход. Вызов функции располагается между командами взлета и посадки.

Дополнительный вариант: выполнить два прямоугольных маршрута разных размеров с помощью одной функции. Для сравнения используются одинаковая исходная точка и разные значения параметров. Усложнение связано с повторным применением алгоритма, а не с увеличением количества отдельных команд.

Таблица 6 - Критерии выполнения свободной миссии

Критерий	**Признак выполнения**
Пользовательская функция	Функция определена и вызывается
Использование параметров	Размеры и высота передаются через аргументы
Список и цикл	Точки хранятся в списке и обходятся циклом
Полнота маршрута	Пройдены не менее четырех заданных точек
Завершение	Программа заканчивается посадкой

При оценивании важно установить, может ли обучающийся объяснить назначение параметров и предсказать результат их изменения. Наличие функции без ее вызова не считается выполнением условия.

2.3 Маршрутные и проектные задания

После освоения основных миссий предлагаются задания с меньшим объемом готового кода. В них требуется выбрать структуру данных, определить порядок действий и объединить несколько конструкций Python. Для выполнения используются те же команды учебного исполнителя.

Патрулирование территории

Условие: трижды пройти квадратный маршрут со стороной 2 м на высоте 1,5 м. В программе используется внешний цикл для повторения маршрута и внутренний - для обхода точек.

```
route = [(2, 0, 1.5), (2, 2, 1.5),
        (0, 2, 1.5), (0, 0, 1.5)]
drone.takeoff(1.5)
for repeat in range(3):
    for point in route:
        drone.go_to(*point)
drone.land()
```

Критерии: маршрут повторяется три раза, внутри каждого повторения сохраняется порядок четырех точек, аппарат возвращается к началу квадрата и после завершения выполняет посадку. В этой программе команда перемещения вызывается 12 раз. Проверка числа вызовов помогает связать вложенные циклы с объемом повторяющихся действий.

Разведывательный маршрут

Условие: пройти пять точек. Третья точка должна находиться на высоте 3 м, остальные - на высоте 1,5 м. Список координат составляется самостоятельно. Один из вариантов исходных данных приведен ниже.

```
route = [(1, 0, 1.5), (2, 1, 1.5), (3, 2, 3.0),
        (2, 3, 1.5), (0, 2, 1.5)]
```

Необходимо дополнить данные командами взлета, циклом обхода и посадкой. Критерии: сохранены пять точек, изменена высота именно третьей точки, после нее выполнено снижение. В объяснении решения указывается, какой элемент кортежа отвечает за высоту.

Возвращение домой

Условие: самостоятельно задать несколько точек, выполнить маршрут и перед посадкой вызвать drone.home(). Критерии: пройдены заданные точки, выполнен возврат в исходную точку, затем произведена посадка. В процессе разбора различаются перемещение домой и завершение полета.

Итоговое задание и оценивание

Итоговое проектное задание «Собственная полетная миссия» выполняется без полного готового решения. Требуется разработать программу автономного движения виртуального БПЛА и объяснить выбранный алгоритм.

Обязательные условия: взлет, прохождение не менее пяти точек, изменение высоты, использование списка координат, цикла и пользовательской функции, возвращение в исходную область и посадка. Дополнительное условие повышенного уровня - осмысленное применение if, влияющего на действия исполнителя.

При подготовке решения сначала определяется маршрут, затем выбирается способ хранения точек. После этого составляется функция, выполняющая повторяющиеся действия. Проверка проводится по траектории, координатам, высоте и структуре кода. Пример возможной программы приведен в приложении 3.

Таблица 7 - Критерии оценивания итогового задания

Критерии	**Баллы**
Программа запускается без синтаксических ошибок	1
Выполнены взлет и посадка	1
Пройдены не менее пяти точек маршрута	1
Используется список координат	1
Используется цикл	1
Определена и вызвана пользовательская функция	1
На маршруте изменяется высота	1
Выполнено возвращение в исходную область	1
Код имеет понятную структуру	1
Дополнительно применено осмысленное условие if	1

Максимальный результат составляет 10 баллов. Для разработанного комплекса принята следующая шкала: 9–10 баллов — высокий уровень выполнения, 6–8 — средний, 3–5 — базовый, менее 3 — необходимость доработки. Это рабочая шкала оценивания задания, а не стандартизированная диагностика учебных достижений.

Автоматическая проверка миссии и оценивание по таблице выполняют разные функции. Система отмечает отдельные признаки движения, а проверка кода позволяет установить наличие списка, функции, цикла и условия. При повторном выполнении фиксируется, какие ошибки были исправлены и почему новое решение соответствует заданию.

2.4 Результаты разработки и границы их применения

В результате работы подготовлен комплекс из шести основных миссий, маршрутных упражнений и итогового проектного задания. Для основных миссий определены темы Python, ожидаемые действия, самостоятельные варианты и критерии. Содержание связано единым исполнителем и постепенным переходом от работы по образцу к собственному алгоритму.

Средством выполнения заданий является PyDrone Lab. В нем объединены редактор Python-кода, виртуальный полигон, отображение координат, высоты и траектории, консоль и проверка отдельных условий. Инструкция приведена в приложении 1, схема интерфейса - в приложении 4.

Таблица 8 - Полученные результаты и основания выводов

Результат	**Представление в работе**	**Граница вывода**
Комплекс основных миссий	Шесть заданий с кодом и самостоятельной частью	Подтверждает подготовку учебных материалов
Маршрутные и проектные задачи	Условия, примеры и шкала оценивания	Описывает способы применения комплекса
Учебная среда	Интерфейс и команды PyDrone Lab	Показывает организацию работы с исполнителем
Проверка базового сценария	Взлет, ожидание и посадка в первой миссии	Относится к отдельному сценарию

Практическая проверка, описанная в работе, выполнена для миссии «Первый полет». Зафиксированы выполнение базовой последовательности команд и отметка трех критериев системой. Этот результат позволяет описать работу начального сценария, но не подтверждает полноту тестирования всех маршрутов и пограничных условий.

Педагогическая апробация с измерением результатов обучающихся в данной работе не представлена. Поэтому не формулируется вывод о доказанном повышении успеваемости или мотивации. Значимость результата связана с подготовкой конкретных заданий, доступных для последующего использования и проверки на занятиях.

Для дальнейшей апробации целесообразно сопоставить самостоятельное выполнение однотипных заданий до и после работы с комплексом, фиксировать типичные ошибки и степень необходимой помощи. Техническое развитие может включать проверку разных браузеров, новых конфигураций препятствий и всех основных миссий. Эти действия рассматриваются как дальнейшая работа.

Заключение

В ходе работы был разработан комплекс практических заданий по программированию на языке Python с использованием учебного симулятора полета БПЛА. Поставленная цель реализована через подготовку последовательности миссий, примеров кода, самостоятельных вариантов и критериев оценивания.

Было рассмотрено значение практических заданий в освоении программирования. Основное внимание уделено переходу от запуска готового примера к изменению программы и самостоятельному выбору алгоритма. Использование виртуального исполнителя позволяет связать текст кода с наблюдаемыми координатами, высотой и траекторией.

Были определены требования к содержанию комплекса: соответствие изучаемой конструкции, понятность условия, постепенное усложнение, наличие проверяемого результата, возможность повторного запуска и повышение самостоятельности. На их основе выделены базовый, средний, повышенный и комплексный уровни заданий.

Подготовлены шесть основных миссий. В них последовательно используются команды управления, переменные, списки, циклы, условия и функции. Задания дополнены упражнениями на патрулирование, прохождение точек с разной высотой и возвращение в исходную точку. Итоговая миссия объединяет изученные конструкции в самостоятельной программе.

Для выполнения упражнений разработан PyDrone Lab. Браузерная среда объединяет редактор программы и виртуальный полигон. Отображение траектории и автоматическая проверка отдельных условий дополняются анализом кода. Для итогового задания предложена шкала из десяти критериев, позволяющая отдельно учитывать движение исполнителя и структуру решения.

Практическая проверка базового сценария показала выполнение последовательности взлета, ожидания и посадки. Этот результат ограничен первой миссией. Выводы о влиянии комплекса на учебные достижения требуют отдельной апробации с участием обучающихся.

Практическая ценность работы состоит в подготовке согласованных учебных материалов и среды для их выполнения. Комплекс может использоваться на занятиях по информатике и во внеурочной деятельности с учетом уровня подготовки класса. Перспективы развития связаны с расширением заданий, технической проверкой сценариев и оценкой результатов применения комплекса в учебном процессе.

Список использованной литературы

1. Кузнецов, А. А. Общая методика обучения информатике. Часть I : учебное пособие для студентов педагогических вузов / А. А. Кузнецов, Т. Б. Захарова, А. С. Захаров. — Москва : Прометей, 2016. — 300 с. — ISBN 978-5-9907452-1-6.

2. Босова, Л. Л. Информатика. 8 класс. Базовый уровень : электронная форма учебника / Л. Л. Босова, А. Ю. Босова. — Москва : Просвещение, 2023. — URL: https://media.prosv.ru/content/item/12824/ (дата обращения: 04.10.2026). — Текст : электронный.

3. Федеральная рабочая программа основного общего образования. Информатика. Базовый уровень. Для 7–9 классов образовательных организаций. — Москва, 2025. — 66 с. — URL: https://edsoo.ru/wp-content/uploads/2025/07/2025_ooo_frp_informatika-7-9_baza.pdf (дата обращения: 05.10.2026). — Текст : электронный.

4. Больше инструментов управления потоком выполнения программы // Документация Python / Python Software Foundation. — URL: https://docs.python.org/ru/3/tutorial/controlflow.html (дата обращения: 05.10.2026). — Текст : электронный.

5. Canvas // MDN Web Docs. — URL: https://developer.mozilla.org/ru/docs/Web/API/Canvas_API (дата обращения: 05.10.2026). — Текст : электронный.

6. Getting started // Pyodide : официальная документация. — URL: https://pyodide.org/en/stable/usage/quickstart.html (дата обращения: 05.10.2026). — Текст : электронный.

7. Pyodide Python compatibility // Pyodide : официальная документация. — URL: https://pyodide.org/en/stable/usage/wasm-constraints.html (дата обращения: 05.10.2026). — Текст : электронный.

Приложение 1 Инструкция по работе с PyDrone Lab

Порядок запуска и выполнения миссии представлен на рисунке 2. До начала занятия необходимо убедиться, что среда Python загружена и доступна выбранная миссия. Повторный запуск выполняется после анализа результата и необходимых изменений программы.



Рисунок 2 - Инструкция по выполнению задания в PyDrone Lab

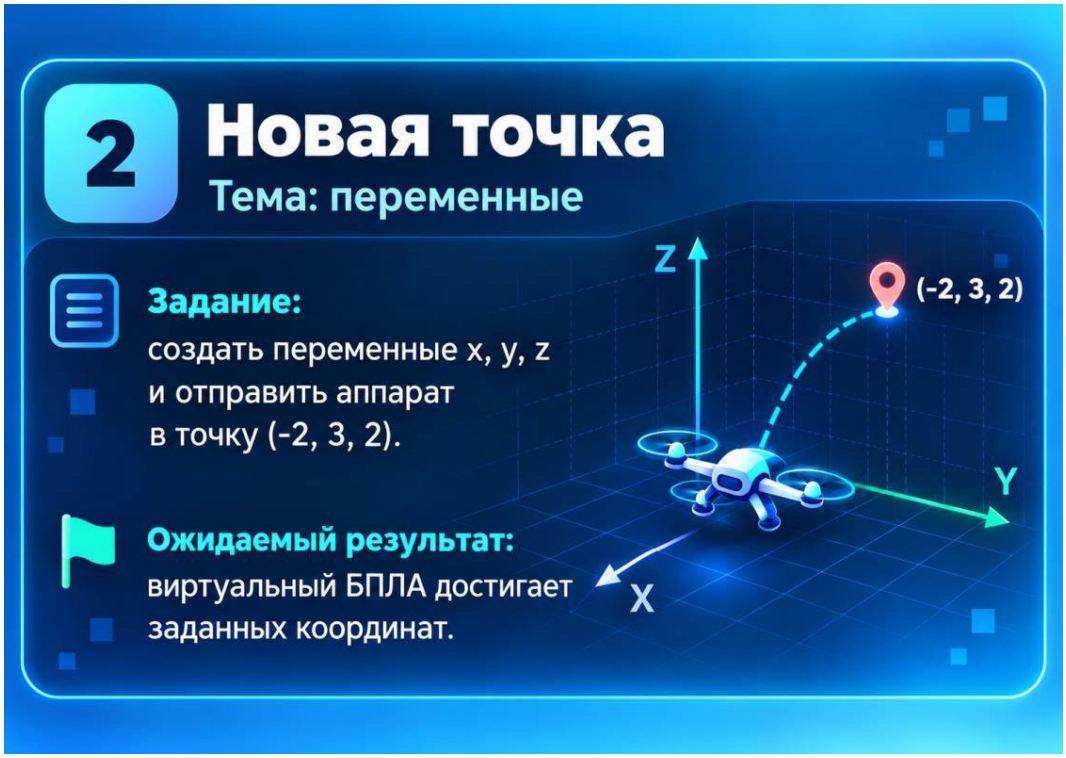
Приложение 2

Карточки практических заданий

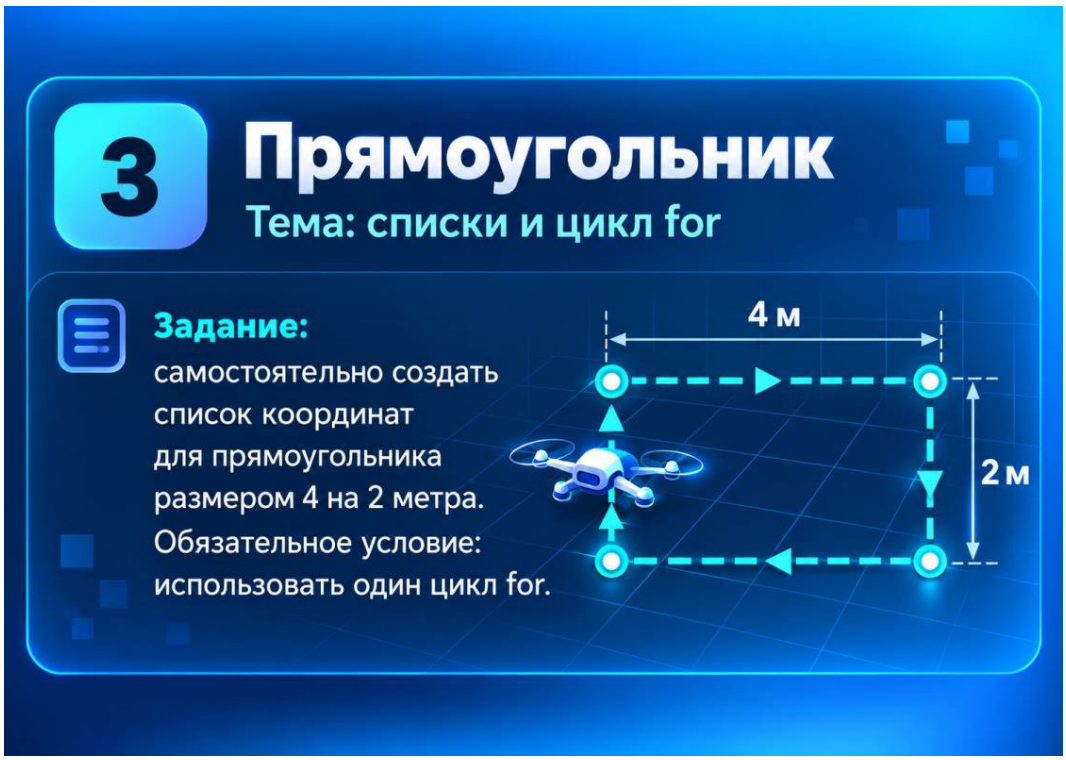
Карточки содержат варианты самостоятельной работы к основным миссиям и маршрутным заданиям.



Карточка 1 - Первый полет



Карточка 2 - Новая точка



Карточка 3 - Прямоугольник



Карточка 4 - Подъем и снижение



Карточка 5 - Обход препятствия

Уточнение: при высоте ровно 2,2 м также выбирается обход; пролет над препятствием допускается только при height > 2.2.



Карточка 6 - Собственный маршрут

Уточнение к коду карточки 6: вместо drone.move_to используется команда PyDrone Lab — drone.go_to(point[0], point[1], point[2]).



Карточка 7 - Патруль

Порядок работы с карточками

Перед выполнением определяется тема задания и ожидаемое движение аппарата. Затем составляется программа, выполняется запуск и проверяются координаты, высота и траектория. После исправления ошибок задание выполняется повторно.

Карточки 1–6 используются после разбора соответствующей основной миссии. Карточка 7 предназначена для закрепления вложенных циклов. Числовые параметры самостоятельных вариантов могут отличаться от исходных примеров основной части.

При оценивании учитываются выполнение условия, применение указанной конструкции Python и объяснение алгоритма. Для карточек 5 и 6 применяются приведенные выше уточнения. Порядок команд и отступы в коде проверяются до запуска.

Приложение 3

Пример итоговой программы

Ниже приведен вариант структуры итоговой миссии. В нем предусмотрены пять точек, изменение высоты, список, функция и цикл. Возвращение и посадка выполняются после прохождения маршрута. Пример иллюстрирует выполнение обязательной части задания; условие if добавляется при дальнейшем усложнении.

```
def fly_route(points):
    for x, y, z in points:
        drone.go_to(x, y, z)

route = [
    (1, 0, 1.5),
    (2, 1, 2.0),
    (3, 2, 2.5),
    (1, 3, 1.5),
    (0, 1, 1.5)
]

drone.takeoff(1.5)
fly_route(route)
drone.home()
drone.land()
```

Маршрут предназначен для полигона, на котором выбранные точки и соединяющие их участки доступны для движения. Перед запуском проверяются координаты и расположение препятствий. Отсутствие синтаксической ошибки само по себе не подтверждает правильность траектории.

Вопросы для разбора решения

1. Какие строки программы задают порядок прохождения точек?

2. В какой точке достигается наибольшая высота и чему она равна?

3. Как изменить маршрут, не изменяя функцию fly_route?

4. Чем вызов drone.home() отличается от drone.land()?

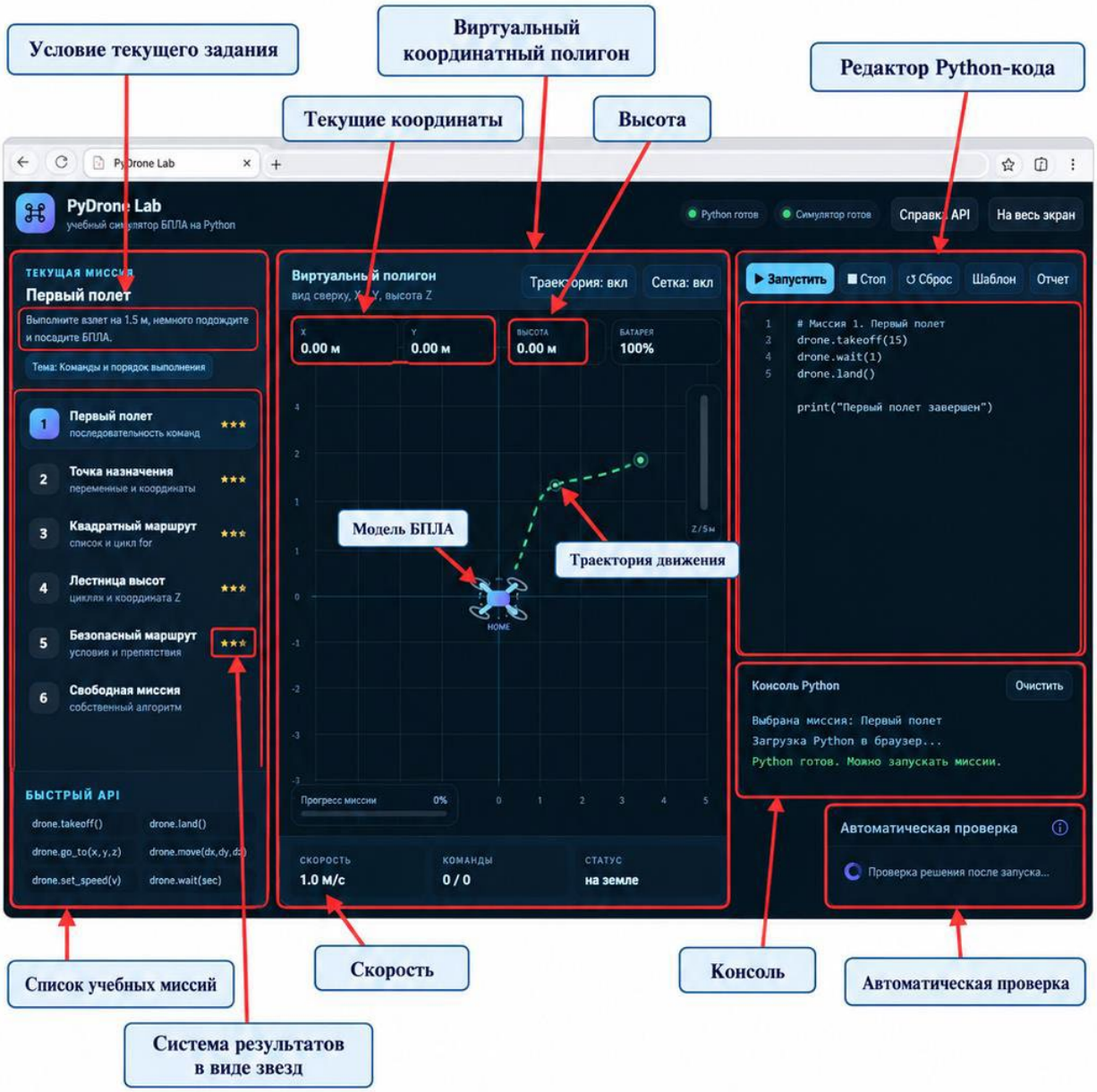
5. Как добавить условие, которое действительно изменит действия исполнителя?

В качестве самостоятельного продолжения предлагается изменить не менее двух точек, добавить условие выбора высоты и объяснить различия между исходной и новой программой. Оценивание выполняется по критериям таблицы 7.

Приложение 4

Устройство учебного симулятора

Схема показывает расположение элементов PyDrone Lab, используемых при выполнении практических заданий.



PyDrone Lab представляет собой учебный браузерный симулятор полета виртуального БПЛА. Он предназначен для выполнения практических заданий по программированию на языке Python. В качестве Python-среды используется Pyodide. Графическая визуализация реализована средствами HTML Canvas. PyDrone Lab является учебной имитационной моделью и предназначен для отработки алгоритмов программирования.

Рисунок 3 - Основные элементы интерфейса PyDrone Lab