Пояснения для первой:

1. **Импорт random** — нужен для генерации случайного числа.
    - random.sample берёт уникальные элементы, что гарантирует отсутствие повторяющихся цифр.
2. **Функция generate_secret_number**
    - Использую строку "0123456789".
    - random.sample(digits, length) → список уникальных символов.
    - Первую цифру меняем, если она "0", чтобы не было ведущего нуля.
    - Возвращаем строку (удобнее работать, чем со списком).
3. **Функция get_user_guess**
    - Запрашивает ввод у пользователя.
    - Проверяем:
        - isdigit() — все ли символы цифры.
        - len(guess) — совпадает ли длина.
        - len(set(guess)) — нет ли повторов.
    - Если что-то не так → цикл while True повторяет ввод.
4. **Функция count_cows_and_bulls**
    - cows - сколько совпало точно.
    - bulls - буквы есть, но на других позициях.
5. **Функция play_game**
    - Ведёт цикл попыток до победы.
    - Каждый ввод проверяется функцией get_user_guess.
    - Счётчик attempts фиксирует число попыток.
    - Если все цифры угаданы → возвращаем attempts.
6. **Функция main**
    - Отвечает за выбор сложности, статистику и повтор игры.
    - stats хранит количество попыток каждой игры.
    - После выхода показываем:
        - Кол-во игр (len(stats)).

    - Минимум, максимум, среднее.
7. **if __name__ == "__main__":**
8. Стандартный способ запуска.

Пояснения для второй:

Функция get_positive_integer

- Использую цикл while True, чтобы программа не завершалась на ошибке.
- int(input(...)) → пробуем преобразовать ввод в число.
- try/except ловит ошибку ValueError, если введены буквы или символы.
- Проверка if n > 0 — исключаем ноль и отрицательные числа.
- Если всё хорошо → возвращаем число.

Функция get_divisors

- Перебираем числа от 1 до n включительно.
- Условие n % i == 0 означает, что i делитель.
- Сохраняем i в список.

Функция is_prime

- Простое число должно делиться только на 1 и n.
- Если n < 2 → сразу не простое.
- Проверяю делители только до sqrt(n), потому что если есть делитель больше корня, то меньший уже встретился.
- Если нашли хотя бы один делитель → число не простое (false).

Функция is_perfect

- Совершенное число = сумма делителей, кроме самого себя.
- Перебираем все делители
- Суммируем → сравниваем с самим числом.

Главная функция main

- Запрашиваем число.
- Выводим делители.

- Проверяем простое/непростое.
- Проверяем совершенное/несовершенное.
- В случае совершенного числа вывожу уравнение вида 1+2+4+7+14=28.

if __name__ == "__main__":

- Стандартный способ запуска.

Пояснения для третьей:

Список WORDS

- Подготовленный список слов из 5 букв.
- Можно расширять, заменять на свой.

Функция choose_word

- random.choice(WORDS) выбирает одно слово случайно.

Функция get_user_word

- Запрашиваем у игрока слово.
- Проверяем:
  - len(guess) — длина должна совпадать.
  - isalpha() — только буквы (без цифр, пробелов).
- Приводим к нижнему регистру, чтобы не было проблем с заглавными.

Функция check_word − Здесь реализована логика проверки.

Первый цикл (ищем точные совпадения [X])

- Если guess[i] == secret[i] → значит буква на своём месте → [X].
- Эту букву в secret_letters помечаем None, чтобы её не пересчитать потом.
- Если не совпало → оставляем None в result.

Второй цикл (ищем буквы «не на месте» (X))

- Если в result ещё None → значит буква пока не определена.
- Проверяем: есть ли guess[i] в secret_letters.
  - Если есть → (X), и букву вычёркиваем (ставим None).

  (Про secret_letters[secret_letters.index(guess[i])] = None):
  - Берёт текущую букву из guess.
  - Находит **первое вхождение этой буквы** в списке secret_letters.

- Заменяет его на None, чтобы пометить «буква уже использована».
- Если нет → оставляем просто букву.

− Это нужно, чтобы при повторяющихся буквах мы не считали одну и ту же букву дважды.

− В конце объединяем результат через join.

Функция play_wordle

- У игрока 6 попыток.
- После каждой попытки выводим результат сравнения.
- Если угадал → поздравляем и завершаем.
- Если не угадал за 6 попыток → показываем слово.

if __name__ == "__main__":

- Стандартный способ запуска.

Пояснения для четвертой:

Список CHOICES

- Все возможные варианты, чтобы не писать их вручную в нескольких местах.
- Пользователь должен выбирать только из этого списка.

Словарь RULES

- Ключ = выбор игрока.
- Значение = список тех, кого он побеждает.
- Пример: "ножницы": ["бумага", "ящерица"].

Функция get_user_choice

- Запрашиваем ввод у игрока.
- Проверяем, что слово из списка CHOICES.
- Если ошибка — цикл while True повторяет ввод.

Функция get_winner

- Сначала проверяем случай ничьей.

- Если компьютерский выбор входит в список побеждаемых игроком (RULES[player]), то выигрывает игрок.
- Иначе выигрывает компьютер.

Функция play_game

- Спрашиваем, до скольки побед играем (max_wins).
- В цикле:
  - Игрок выбирает ход.
  - Компьютер делает случайный выбор (random.choice(CHOICES)).
  - Определяем победителя через get_winner.
  - Обновляем счёт.
- Цикл продолжается, пока кто-то не наберёт нужное количество побед.
- В конце объявляем общий результат.

if __name__ == "__main__":

- Стандартный способ запуска.

Пояснения для пятой:

Функция read_text_from_file

- Считывает текст из файла, путь к которому вводит пользователь.
- Проверяет, что файл существует и содержит минимум «min_length» символов

Функция clean_text

- lower() → всё в нижний регистр (чтобы "Python" и "python" считались одинаковыми).
- берём каждый символ из списка и заменяем его на пробел.
- split() → разбиваем строку по пробелам на слова.

Функция count_characters

- len(text) → количество символов с пробелами.
- len(text.replace(" ", "")) → убираем пробелы и считаем символы без них.

Функция get_statistics

- Counter(words) считает частоты слов.

- most_common(5) → берём 5 самых популярных слов.
- Для длинных слов:
    - сортируем sorted(words, key=len, reverse=True),
    - выбираем только уникальные.
- Средняя длина слова = сумма длин / количество слов.

Функция main

- Собирает все шаги:
    - ввод текста,
    - очистка,
    - подсчёт символов и слов,
    - статистика,
    - вывод.