Лабораторная работа 0. Основы программирования на языке Python

Цель работы:

- овладеть фундаментальными концепциями, навыками и инструментами, необходимыми для эффективного создания программ на языке Python.

Задание на лабораторную работу:

Из заданий **1-4** необходимо выполнить одно на выбор (но так, чтобы в группе они были распределены равномерно) Задания **5-10** обязательны для всех.

Задание 1. Виселица

Создайте игру "Виселица", в которой программа выбирает случайное слово, а игрок должен угадать его, вводя буквы по одной. Программа должна выводить текущее состояние слова с угаданными буквами и количество оставшихся попыток.

1. Шаг 1: Загрузка списка слов
    - Создайте список слов для игры. Например: words = ['яблоко', 'груша', 'шоколад', 'пони'].
    - Выберите случайное слово из списка с помощью функции random.choice(). Например: word = random.choice(words).
    - Для работы с функцией необходимо предварительно импортировать в программу нужный модуль командой import random
2. Шаг 2: Инициализация переменных
    - Создайте переменную guesses и установите ее значение равным пустой строке. В этой переменной будет храниться текущий прогресс угадывания слова.

- Создайте переменную max_attempts и установите ее значение равным количеству попыток, которые игрок имеет для угадывания слова. Например: max_attempts = 6.
- Создайте переменную attempt и установите ее значение равным 0. В этой переменной будет храниться текущее количество использованных попыток.

3. Шаг 3: Основной цикл игры
    - Создайте цикл while, который будет выполняться до тех пор, пока не будет достигнуто условие окончания игры (победа или поражение).
    - Выведите сообщение с инструкцией для игрока и текущим состоянием слова, в котором угаданные буквы отображаются, а неправильные попытки обозначаются символом "_". Например: print("Угадайте слово:", guesses).
    - Запросите у игрока ввод буквы и сохраните ее в переменной letter. Например: letter = input("Введите букву: ").
4. Шаг 4: Проверка введенной буквы
    - Проверьте, является ли введенная буква одной из букв загаданного слова. Если да, добавьте ее к переменной guesses. Если нет, увеличьте значение переменной attempt на 1.
    - Обновите переменную guesses, чтобы отображать угаданные буквы и символы "_" для неправильных попыток. Например: guesses += letter if letter in word else "_".
5. Шаг 5: Проверка условия окончания игры
    - Проверьте, если все буквы в слове были угаданы (победа), выведите сообщение о победе и завершите цикл.

- Проверьте, если количество использованных попыток (attempt) достигло максимального значения (max_attempts) (поражение), выведите сообщение о поражении и завершите цикл.

6. Шаг 6: Завершение игры
    - Выведите загаданное слово, чтобы игрок увидел его даже в случае поражения. Например: print("Загаданное слово:", word).
    - Выведите сообщение о том, что игра завершена.

Это общая структура решения. Вы можете добавить дополнительные проверки, улучшения и декорации для игры "Виселица", чтобы сделать ее более интерактивной и увлекательной.

Задание 2. Крестики-нолики

Реализуйте классическую игру "Крестики-нолики" для двух игроков. Создайте игровое поле, где игроки будут ходить по очереди и ставить свои символы (крестик или нолик) в пустые ячейки. Проверяйте наличие выигрышной комбинации после каждого хода.

Шаг 1: Создание игрового поля

- Создайте функцию create_board(), которая будет инициализировать пустое игровое поле. В данном случае, мы можем использовать двумерный список размером 3x3. Например:

```
def create_board():
    board = [['_', '_', '_'],
             ['_', '_', '_'],
             ['_', '_', '_']]
    return board
```

- Напишите функцию display_board(board), которая будет выводить текущее состояние игрового поля на экран. Используйте символы "X", "O" и пустую ячейку "_" для представления крестиков, ноликов и пустых ячеек соответственно. Например:

```
def display_board(board):
    for row in board:
        print(' '.join(row))
```

Шаг 2: Определение игроков

- Запросите у игроков их имена или установите фиксированные имена для игроков "Игрок 1" и "Игрок 2".

Шаг 3: Определение хода игроков

- Создайте функцию get_move(player), которая будет запрашивать у игрока координаты хода (строка и столбец) и возвращать их в виде кортежа. Например:

```
def get_move(player):
    print(player + ", введите координаты хода (строка и столбец): ")
    row = int(input("Строка: "))-1
```

```
    col = int(input("Столбец: "))-1
    return (row, col)
```

Шаг 4: Проверка возможности хода

- Напишите функцию is_valid_move(move, board), которая будет проверять, является ли ход игрока допустимым (корректным) и ячейка на игровом поле пуста. Например:

```
def is_valid_move(move, board):
    row, col = move
    if 0 <= row < 3 and 0 <= col < 3 and board[row][col] == '_':
        return True
    else:
        return False
```

Шаг 5: Обновление игрового поля

- Создайте функцию make_move(move, player, board), которая будет обновлять игровое поле, помещая символ игрока в соответствующую ячейку. Например:

```
def make_move(move, player, board):
    row, col = move
    board[row][col] = player
```

Шаг 6: Проверка условий победы

- Напишите функцию check_win(player, board), которая будет проверять, выполнились ли условия победы для заданного игрока. Проверьте все возможные комбинации трех символов в ряд по горизонтали, вертикали и диагонали.

Шаг 7: Основной цикл игры

- Создайте основной цикл игры, который будет выполняться до тех пор, пока не будет достигнуто условие победы или ничья.

- Поочередно вызывайте функции get_move(), is_valid_move(), make_move() и check_win() для каждого игрока.

Шаг 8: Завершение игры

- Выведите сообщение о завершении игры и спросите у игроков, хотят ли они сыграть еще раз.

Задание 3. Камень-ножницы-бумага

Реализуйте игру "Камень, ножницы, бумага". Программа должна предложить игроку выбрать один из трех вариантов, затем случайным образом выбрать ход компьютера и определить победителя.

Шаг 1: Определение игровых элементов

- Создайте список choices, содержащий доступные варианты игры: "камень", "ножницы" и "бумага". Например: choices = ["камень", "ножницы", "бумага"]

Шаг 2: Определение игроков

- Запросите у игрока имя или установите фиксированное имя.

Шаг 3: Получение ходов игроков

- Создайте функцию get_player_choice(player), которая будет запрашивать у игрока его выбор (камень, ножницы или бумагу) и возвращать его выбор в виде строки. Например:

```
def get_player_choice(player):
    print(player + ", сделайте свой выбор (камень, ножницы, бумага): ")
    choice = input().lower()
    while choice not in choices:
        print("Некорректный выбор. Попробуйте снова.")
        choice = input().lower()
    return choice
```

Шаг 4: Определение победителя

- Создайте функцию get_winner(player1_choice, player2_choice), которая будет определять победителя на основе выбора игроков. Используйте условия для проверки всех возможных комбинаций.

Шаг 5: Основной цикл игры

- Создайте основной цикл игры, который будет выполняться до тех пор, пока игроки хотят продолжать игру.
- В цикле вызывайте функции get_player_choice() для первого игрока и random.choice() для второго. Обратите внимание, для использования функции необходимо сначала импортировать модуль командой import random. Затем используйте функцию get_winner() для определения победителя и вывода результатов.

Шаг 6: Завершение игры

- Выведите сообщение о завершении игры.

Задание 4. Викторина

Игрокам предлагаются вопросы на различные темы, и они должны выбрать правильные ответы из предложенных вариантов. За каждый правильный ответ игрок получает очки, а за неправильный ответ теряет очки. В конце игры выводится итоговый счет.

1. Шаг 1: Подготовка вопросов и ответов
    - Создайте список questions, содержащий вопросы для викторины и список answers, содержащий соответствующие ответы на вопросы. Индекс каждого ответа должен соответствовать индексу соответствующего вопроса.
2. Шаг 2: Счетчик очков
    - Установите счетчик score в начальное значение 0. Он будет отслеживать количество правильных ответов.
3. Шаг 3: Основной цикл игры
    - Создайте цикл, который будет выполняться до тех пор, пока есть доступные вопросы в викторине. В каждой итерации цикла:
        - Выберите случайный вопрос из списка questions.
        - Выведите вопрос на экран и запросите ответ у игрока.
        - Сравните ответ игрока с правильным ответом из списка answers. Если ответ игрока верный, увеличьте счетчик score на 1.
        - Удалите использованный вопрос и ответ из списков questions и answers.
4. Шаг 4: Вывод результата
        - Выведите сообщение с итоговым счетом игрока.

Задание 5. Задача о последовательности Фибоначчи

Напишите программу, которая выводит n-ое число в последовательности Фибоначчи, где n - это целое положительное число, введенное пользователем. При этом, вычисления должны выполняться без использования рекурсии.

Задание 6. Палиндром

Напишите функцию, которая проверяет, является ли заданная строка палиндромом (читается одинаково слева направо и справа налево).

Задание 7. Поиск простых чисел

Напишите программу, которая находит все простые числа в заданном диапазоне.

Задание 8. Сумма чисел

Напишите программу, которая запрашивает у пользователя несколько чисел и выводит их сумму.

Задание 9. Элементы списка

Дан список list = [11, 5, 8, 32, 15, 3, 20, 132, 21, 4, 555, 9, 20]. Необходимо вывести элементы, которые одновременно 1) меньше 30 2) делятся на 3 без остатка. Все остальные элементы списка необходимо просуммировать и вывести конечный результат

Задание 10. Обратное число

Вводится целое число. Вывести число, обратное введенному по порядку составляющих его цифр. Например, введено 3425, надо вывести 5243.