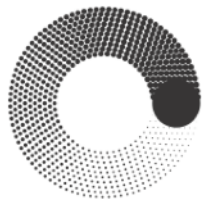


**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

**Факультет информационных технологий
Кафедра Информатики и информационных технологий**

**направление подготовки 09.04.02 «Информационные системы и
технологии»,
профиль «Мобильные приложения»**

**Практическая работа №1
«Языки программирования для систем искусственного интеллекта в
мобильных приложениях»**

Дисциплина: Искусственный интеллект в мобильных системах

Выполнил: студент группы 234-332

Киселев С.А.

Дата, подпись 18.03.2025 _____

(Дата)

(Подпись)

Проверила: Дагаев А.Е. _____

(Оценка)

Дата, подпись _____

(Дата)

(Подпись)

Замечания:

Москва

2025

Оглавление

Цель	3
Задача 1. Игра «Угадай число»	3
Задача 2. Игра «Угадай слово»	5
Задача 3. Генерация случайного пароля	8
Задача 4. Игра «Камень, ножницы, бумага»	10
Задача 5. Последовательность Фибоначчи.	12
Задача 6. Простые числа	14
Задача 7. Калькулятор	17
Задача 8. Игра «21 point»	20
Задача 9. Игра в «крестики-нолики»	23
Задача 10. Квадратное уравнение	29
Задача 11. Тип треугольника	31
Задача 12. Табулирование функции	33
Вывод	35

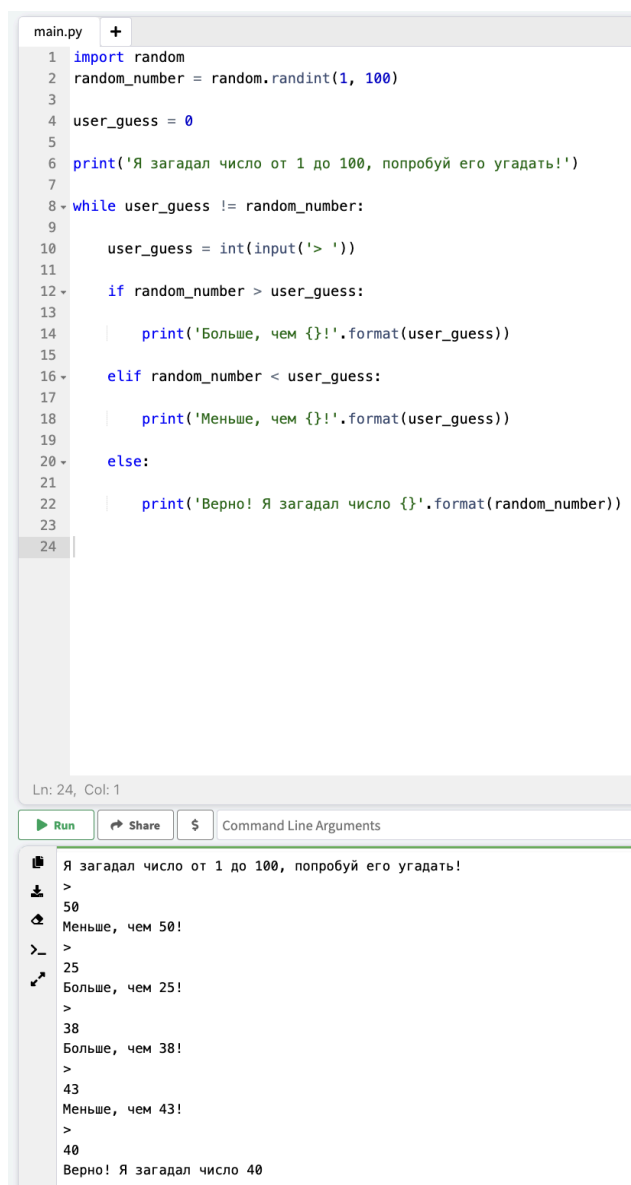
Цель

Целью работы является ознакомление с языками программирования для систем искусственного интеллекта в мобильных приложениях и выполнение заданий.

Задача 1. Игра «Угадай число»

Описание программы: компьютер загадывает число от 1 до 100, после каждой попытки игрока компьютер будет говорить, больше ли загаданное число или меньше, пока не угадает.

Задание: Отладьте код программы, зафиксируйте работу программы скриншотами в отчете и измените программу так, чтобы играли двое – компьютер и человек по очереди, делая ходы, и кто первый угадал – тот и выиграл.



```
main.py +
1 import random
2 random_number = random.randint(1, 100)
3
4 user_guess = 0
5
6 print('Я загадал число от 1 до 100, попробуй его угадать!')
7
8 while user_guess != random_number:
9     user_guess = int(input('> '))
10
11     if random_number > user_guess:
12         print('Больше, чем {}'.format(user_guess))
13
14     elif random_number < user_guess:
15         print('Меньше, чем {}'.format(user_guess))
16
17     else:
18         print('Верно! Я загадал число {}'.format(random_number))
19
20
21
22
23
24
```

Ln: 24, Col: 1

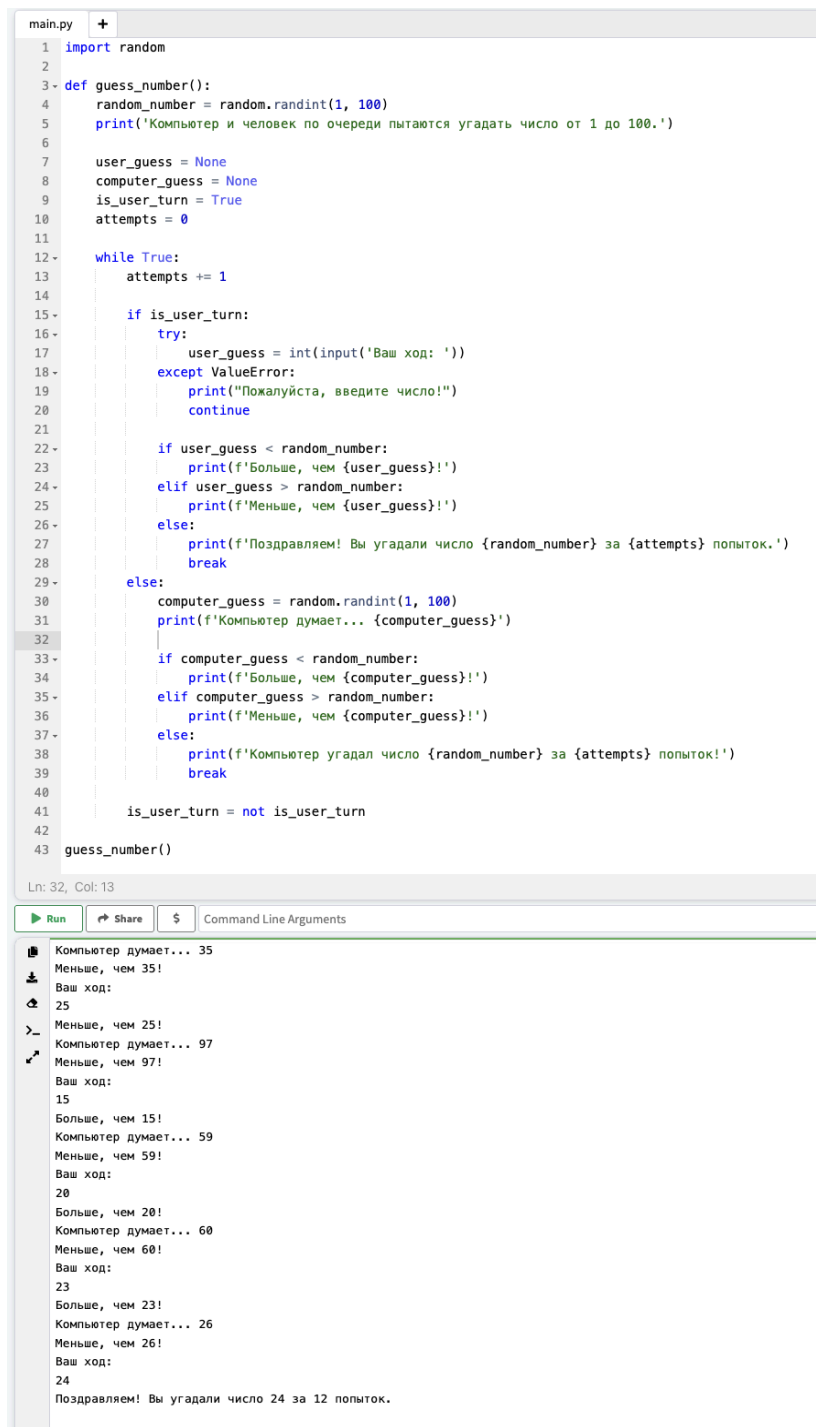
Run Share Command Line Arguments

Я загадал число от 1 до 100, попробуй его угадать!
>
50
Меньше, чем 50!
>
25
Больше, чем 25!
>
38
Больше, чем 38!
>
43
Меньше, чем 43!
>
40
Верно! Я загадал число 40

Рисунок 1.1 – Результат работы программы

Программа была отлажена и изменена так, чтобы компьютер и человек по очереди пытались угадать случайно загаданное число от 1 до 100.

Программа использует модуль `random` для генерации случайного числа в диапазоне от 1 до 100. Далее реализуется игровой цикл, в котором игроки (человек и компьютер) по очереди делают попытки угадать число. В зависимости от их догадок программа сообщает, загаданное число больше или меньше введенного. Игра продолжается до тех пор, пока один из игроков не угадает число.



```
main.py +
1 import random
2
3 def guess_number():
4     random_number = random.randint(1, 100)
5     print('Компьютер и человек по очереди пытаются угадать число от 1 до 100.')
6
7     user_guess = None
8     computer_guess = None
9     is_user_turn = True
10    attempts = 0
11
12    while True:
13        attempts += 1
14
15        if is_user_turn:
16            try:
17                user_guess = int(input('Ваш ход: '))
18            except ValueError:
19                print("Пожалуйста, введите число!")
20                continue
21
22            if user_guess < random_number:
23                print(f'Больше, чем {user_guess}!')
24            elif user_guess > random_number:
25                print(f'Меньше, чем {user_guess}!')
26            else:
27                print(f'Поздравляем! Вы угадали число {random_number} за {attempts} попыток.')
28                break
29
30        else:
31            computer_guess = random.randint(1, 100)
32            print(f'Компьютер думает... {computer_guess}')
33
34            if computer_guess < random_number:
35                print(f'Больше, чем {computer_guess}!')
36            elif computer_guess > random_number:
37                print(f'Меньше, чем {computer_guess}!')
38            else:
39                print(f'Компьютер угадал число {random_number} за {attempts} попыток!')
40                break
41
42        is_user_turn = not is_user_turn
43    guess_number()
```

Ln: 32, Col: 13

Run Share \$ Command Line Arguments

Компьютер думает... 35
Меньше, чем 35!
Ваш ход:
25
Меньше, чем 25!
Компьютер думает... 97
Меньше, чем 97!
Ваш ход:
15
Больше, чем 15!
Компьютер думает... 59
Меньше, чем 59!
Ваш ход:
20
Больше, чем 20!
Компьютер думает... 60
Меньше, чем 60!
Ваш ход:
23
Больше, чем 23!
Компьютер думает... 26
Меньше, чем 26!
Ваш ход:
24
Поздравляем! Вы угадали число 24 за 12 попыток.

Рисунок 1.2 – Результат работы доработанной программы

- Добавлена возможность игры для двух участников — человека и компьютера.
- Реализована логика очередности ходов, где человек и компьютер ходят поочередно.
- Добавлен контроль ввода пользователя, чтобы избежать ошибок при вводе некорректных данных.
- Компьютер делает случайный выбор числа при каждом ходе.
- В конце игры выводится информация о том, кто угадал число и за сколько попыток.

Программа успешно реализована и работает без ошибок.

Задача 2. Игра «Угадай слово»

Описание программы: угаданные буквы в слове будут открываться, но неудачные попытки буду отнимать жизни.

Задание: Отладьте код программы, зафиксируйте работу программы скриншотами в отчете и измените программу так, чтобы увеличить число слов, а также, чтобы человек играл с компьютером, угадывая слова из списка по очереди. Кто первый угадал, тот и выиграл.

```

-----
>
ц
Этой буквы нет в слове. Текущее кол-во жизней: 4
-----
>
у
Этой буквы нет в слове. Текущее кол-во жизней: 3
-----
>
ф
Этой буквы нет в слове. Текущее кол-во жизней: 2
-----
>
х
Этой буквы нет в слове. Текущее кол-во жизней: 1
-----
>
н
Этой буквы нет в слове. Текущее кол-во жизней: 0
-----
Жизни закончились :(

```

```

-----
>
о
Буква есть в слове!
-----
>
е
Этой буквы нет в слове. Текущее кол-во жизней: 4
-----
>
т
Буква есть в слове!
-----
>
б
Буква есть в слове!
-----
>
н
Буква есть в слове!
но _ т б _ _
-----
>
у
Буква есть в слове!
но у т б у _
-----
>
к
Буква есть в слове!
но у т б у к
Поздравляю, вы правильно набрали слово ноутбук

```

Рисунок 2.1 – Результат работы программы

В ходе выполнения программы при некорректном пользовательском вводе программа некорректно завершает работу

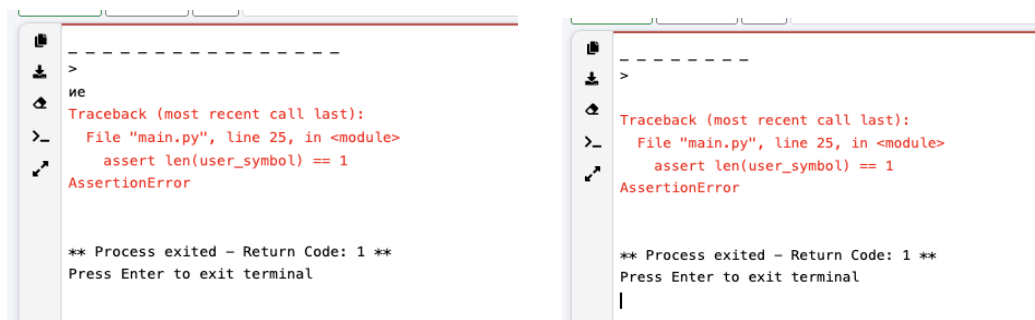


Рисунок 2.2 – Ошибки в работе программы

Программа выбирает случайное слово из списка и предлагает двум игрокам (человеку и компьютеру) угадывать буквы по очереди. Если буква есть в слове, она открывается; если буквы нет — отнимается одна жизнь. У каждого игрока свой счетчик жизней. Игра идет до тех пор, пока один из участников не угадает слово или у соперника не закончатся жизни.

- Добавлен список из нескольких слов для выбора случайного слова в начале игры.
- Реализована возможность игры между человеком и компьютером с поочередными ходами.
- Введены отдельные счетчики жизней для человека и компьютера.
- Улучшена обработка пользовательского ввода (проверка на длину и тип введенных сим

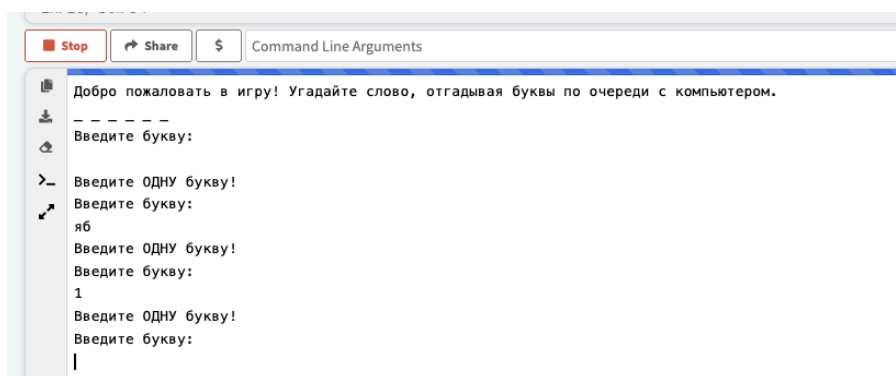


Рисунок 2.3 – Исправление ошибок в работе программы

```

import random

def choose_word():
    words = ['яблоко', 'победа', 'программирование', 'терминал', 'ноутбук']
    return random.choice(words)

def display_progress(word, discovered):
    return ' '.join([ch if ch in discovered else '_' for ch in word])

def play_game():
    word = choose_word()
    unique_letters = set(word)
    discovered_letters = set()
    user_health = 5
    computer_health = 5
    is_user_turn = True

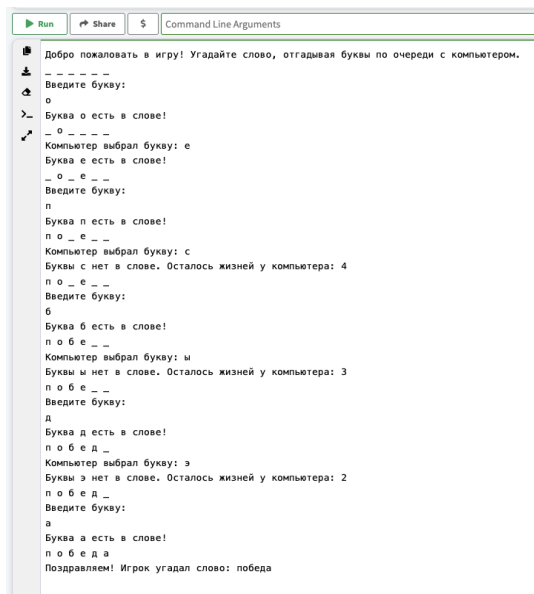
    print("Добро пожаловать в игру! Угадайте слово, отгадывая буквы по очереди с компьютером.")
    print(display_progress(word, discovered_letters))

    while discovered_letters != unique_letters and user_health > 0 and computer_health > 0:
        if is_user_turn:
            guess = input("Введите букву: ").lower()
            if len(guess) != 1 or not guess.isalpha():
                print("Введите ОДНУ букву!")
                continue
            else:
                guess = random.choice([ch for ch in 'абвгдеёжзийклмнопрстуфхцчшщъыьэюя' if ch not in discovered_letters])
                print(f"Компьютер выбрал букву: {guess}")
            if guess in discovered_letters:
                print(f"Буква {guess} уже открыта, попробуйте другую.")
            elif guess in unique_letters:
                discovered_letters.add(guess)
                print(f"Буква {guess} есть в слове!")
            else:
                if is_user_turn:
                    user_health -= 1
                    print(f"Буквы {guess} нет в слове. Осталось жизней у игрока: {user_health}")
                else:
                    computer_health -= 1
                    print(f"Буквы {guess} нет в слове. Осталось жизней у компьютера: {computer_health}")
            print(display_progress(word, discovered_letters))
            is_user_turn = not is_user_turn

    if user_health == 0:
        print(f"Игрок проиграл! Компьютер победил. Загаданное слово было: {word}")
    elif computer_health == 0:
        print(f"Компьютер проиграл! Игрок победил. Загаданное слово было: {word}")
    else:
        winner = 'Игрок' if not is_user_turn else 'Компьютер'
        print(f"Поздравляем! {winner} угадал слово: {word}")

play_game()

```



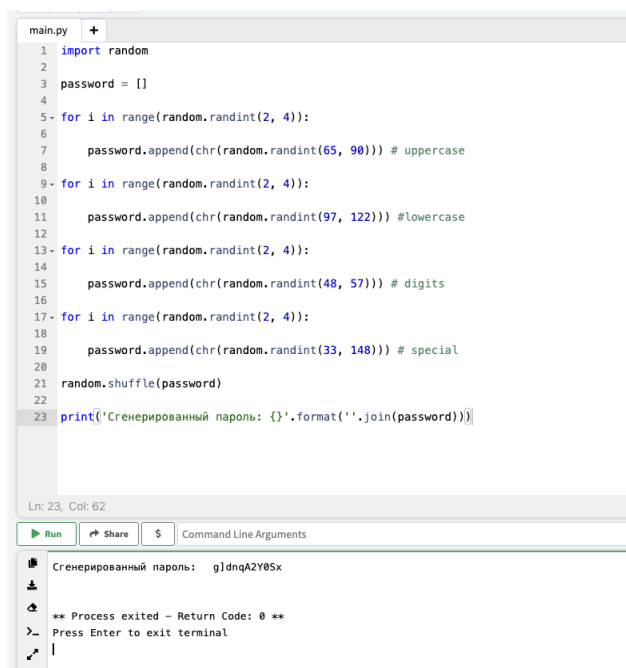
```
Добро пожаловать в игру! Угадайте слово, отгадывая буквы по очереди с компьютером.
-- -- --
Введите букву:
o
Буква o есть в слове!
-- -- --
Компьютер выбрал букву: e
Буква e есть в слове!
_ o _ e _ _
Введите букву:
п
Буква п есть в слове!
п o _ e _ _
Компьютер выбрал букву: с
Буквы с нет в слове. Осталось жизней у компьютера: 4
п o _ e _ _
Введите букву:
б
Буква б есть в слове!
п o б e _ _
Компьютер выбрал букву: м
Буквы м нет в слове. Осталось жизней у компьютера: 3
п o б e _ _
Введите букву:
д
Буква д есть в слове!
п o б e д _
Компьютер выбрал букву: э
Буквы э нет в слове. Осталось жизней у компьютера: 2
п o б e д _
Введите букву:
а
Буква а есть в слове!
п o б e д а
Поздравляем! Игрок угадал слово: победа
```

Рисунок 2.4 – Результат работы доработанной программы

Задача 3. Генерация случайного пароля

Описание программы: Программа создает пароль из набора случайных СИМВОЛОВ.

Задание: Отладьте код программы, зафиксируйте работу программы скриншотами в отчете и измените программу так, чтобы была функция генерации пароля, а пользователь в цикле (пока не введен символ “q”) вводил сайт и логин, а программа генерировала пароль и сохраняла (добавляла) в файл построчно тройку значений сайт-логин-пароль.



```
main.py +
1 import random
2
3 password = []
4
5 for i in range(random.randint(2, 4)):
6     password.append(chr(random.randint(65, 90))) # uppercase
7
8
9 for i in range(random.randint(2, 4)):
10     password.append(chr(random.randint(97, 122))) # lowercase
11
12
13 for i in range(random.randint(2, 4)):
14     password.append(chr(random.randint(48, 57))) # digits
15
16
17 for i in range(random.randint(2, 4)):
18     password.append(chr(random.randint(33, 148))) # special
19
20
21 random.shuffle(password)
22
23 print('Сгенерированный пароль: {}'.format(''.join(password)))

Ln: 23, Col: 62

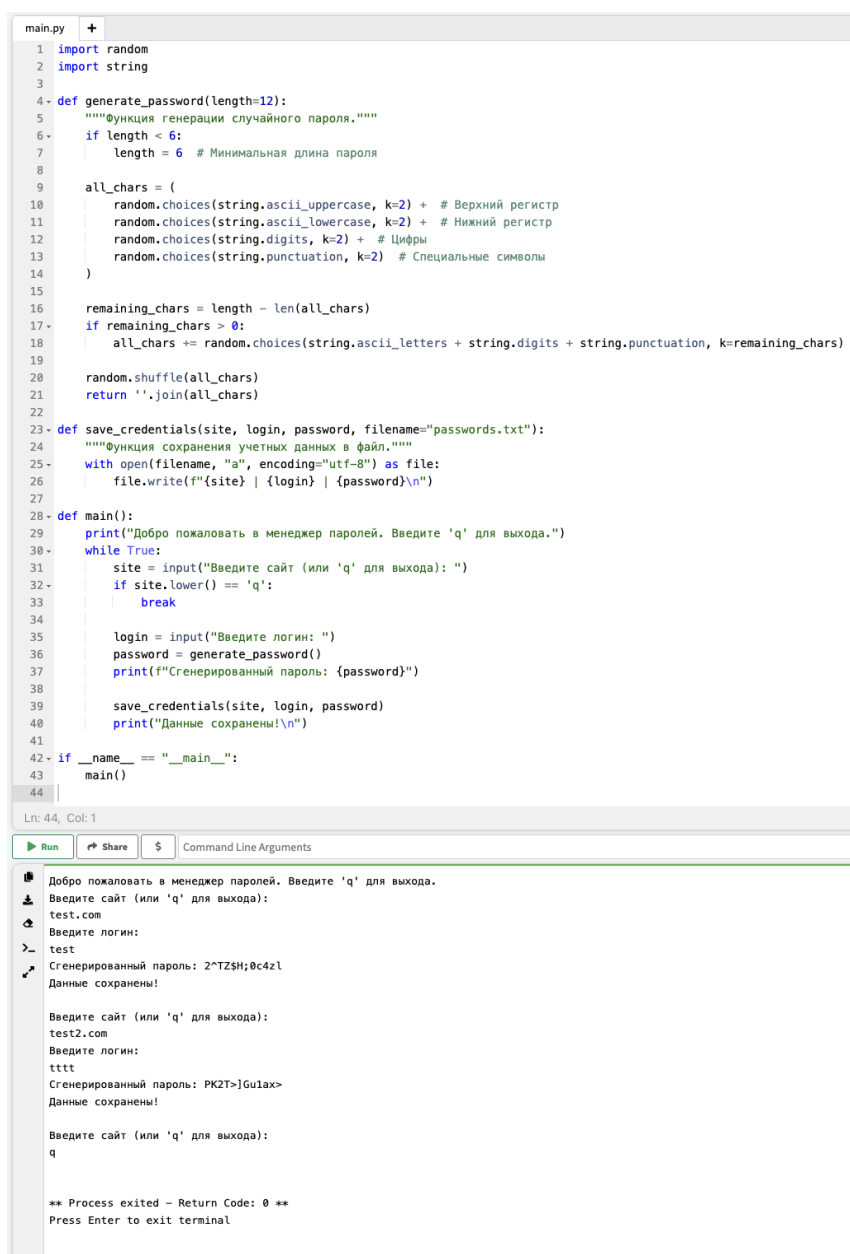
Сгенерированный пароль:  g]dnQA2Y85x

** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Рисунок 3.1 – Результат работы программы

Программа позволяет пользователю вводить название сайта и логин, после чего автоматически генерирует надежный пароль. Сгенерированные учетные данные (сайт, логин, пароль) сохраняются в файл passwords.txt.

- Добавлена функция `generate_password()`, которая создает пароль заданной длины (по умолчанию 12 символов) с обязательным включением букв верхнего и нижнего регистра, цифр и специальных символов.
- Реализована функция `save_credentials()`, которая сохраняет введенные пользователем данные в файл.
- Введен цикл, в котором пользователь может вводить учетные данные для нескольких сайтов, пока не введет `q` для выхода.



```
main.py +
1 import random
2 import string
3
4 def generate_password(length=12):
5     """Функция генерации случайного пароля."""
6     if length < 6:
7         length = 6 # Минимальная длина пароля
8
9     all_chars = (
10         random.choices(string.ascii_uppercase, k=2) + # Верхний регистр
11         random.choices(string.ascii_lowercase, k=2) + # Нижний регистр
12         random.choices(string.digits, k=2) + # Цифры
13         random.choices(string.punctuation, k=2) # Специальные символы
14     )
15
16     remaining_chars = length - len(all_chars)
17     if remaining_chars > 0:
18         all_chars += random.choices(string.ascii_letters + string.digits + string.punctuation, k=remaining_chars)
19
20     random.shuffle(all_chars)
21     return ''.join(all_chars)
22
23 def save_credentials(site, login, password, filename="passwords.txt"):
24     """Функция сохранения учетных данных в файл."""
25     with open(filename, "a", encoding="utf-8") as file:
26         file.write(f"{site} | {login} | {password}\n")
27
28 def main():
29     print("Добро пожаловать в менеджер паролей. Введите 'q' для выхода.")
30     while True:
31         site = input("Введите сайт (или 'q' для выхода): ")
32         if site.lower() == 'q':
33             break
34
35         login = input("Введите логин: ")
36         password = generate_password()
37         print(f"Сгенерированный пароль: {password}")
38
39         save_credentials(site, login, password)
40         print("Данные сохранены!\n")
41
42 if __name__ == "__main__":
43     main()
44
```

Ln: 44, Col: 1

Run Share Command Line Arguments

```
Добро пожаловать в менеджер паролей. Введите 'q' для выхода.
Введите сайт (или 'q' для выхода):
test.com
Введите логин:
test
Сгенерированный пароль: 2~TZ$H;0c4z1
Данные сохранены!

Введите сайт (или 'q' для выхода):
test2.com
Введите логин:
tttt
Сгенерированный пароль: PK2T>]Gu1ax>
Данные сохранены!

Введите сайт (или 'q' для выхода):
q

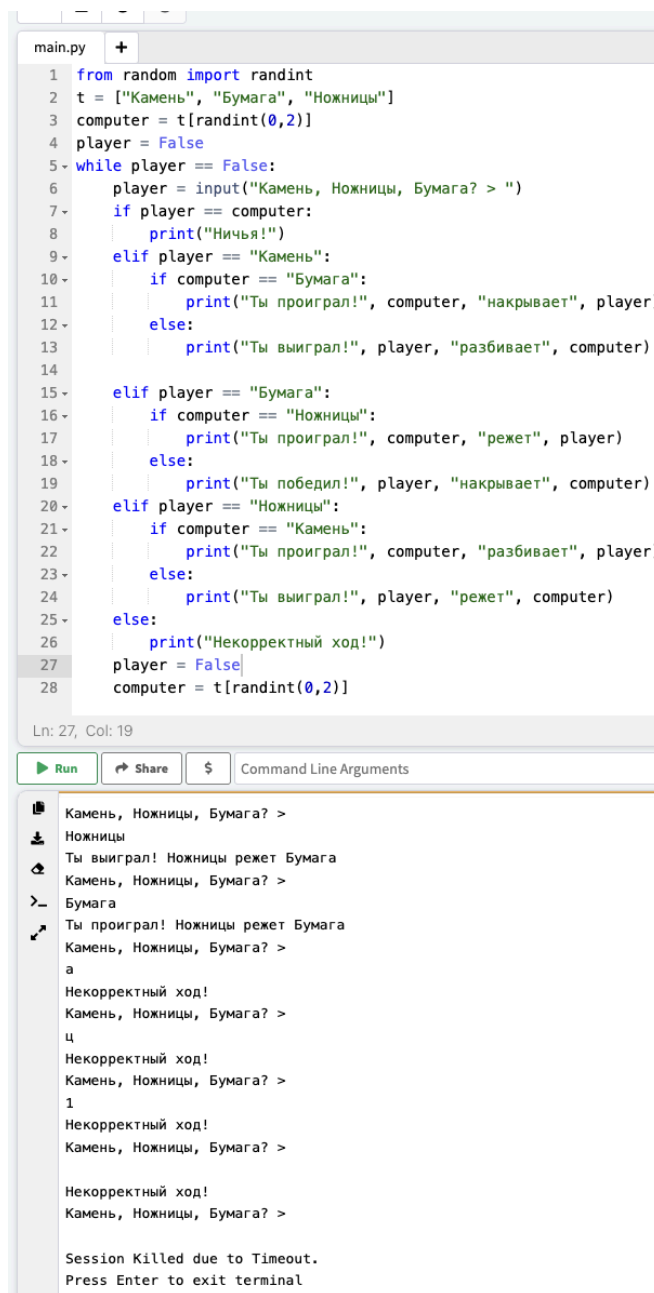
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Рисунок 3.2 – Результат работы доработанной программы

Задача 4. Игра «Камень, ножницы, бумага»

Описание программы: пользователь загадывает одно из трех камень, или ножницы или бумагу. Вводит свое решение в консоль, а затем компьютер случайным образом определяет свой выбор. В дальнейшем, с помощью условий определяется, кто победил, результат игры выводится на экран.

Задание: Отладьте код программы, зафиксируйте работу программы скриншотами в отчете и добавьте «колодец». Правила просты – бумага побеждает колодец (накрывает), а колодец побеждает камень и ножницы (они тонут).



```
main.py +
1 from random import randint
2 t = ["Камень", "Бумага", "Ножницы"]
3 computer = t[randint(0,2)]
4 player = False
5 while player == False:
6     player = input("Камень, Ножницы, Бумага? > ")
7     if player == computer:
8         print("Ничья!")
9     elif player == "Камень":
10        if computer == "Бумага":
11            print("Ты проиграл!", computer, "накрывает", player)
12        else:
13            print("Ты выиграл!", player, "разбивает", computer)
14
15    elif player == "Бумага":
16        if computer == "Ножницы":
17            print("Ты проиграл!", computer, "режет", player)
18        else:
19            print("Ты победил!", player, "накрывает", computer)
20    elif player == "Ножницы":
21        if computer == "Камень":
22            print("Ты проиграл!", computer, "разбивает", player)
23        else:
24            print("Ты выиграл!", player, "режет", computer)
25    else:
26        print("Некорректный ход!")
27    player = False
28    computer = t[randint(0,2)]

Ln: 27, Col: 19
Run Share $ Command Line Arguments

Камень, Ножницы, Бумага? >
Ножницы
Ты выиграл! Ножницы режет Бумага
Камень, Ножницы, Бумага? >
Бумага
Ты проиграл! Ножницы режет Бумага
Камень, Ножницы, Бумага? >
а
Некорректный ход!
Камень, Ножницы, Бумага? >
ц
Некорректный ход!
Камень, Ножницы, Бумага? >
1
Некорректный ход!
Камень, Ножницы, Бумага? >

Некорректный ход!
Камень, Ножницы, Бумага? >

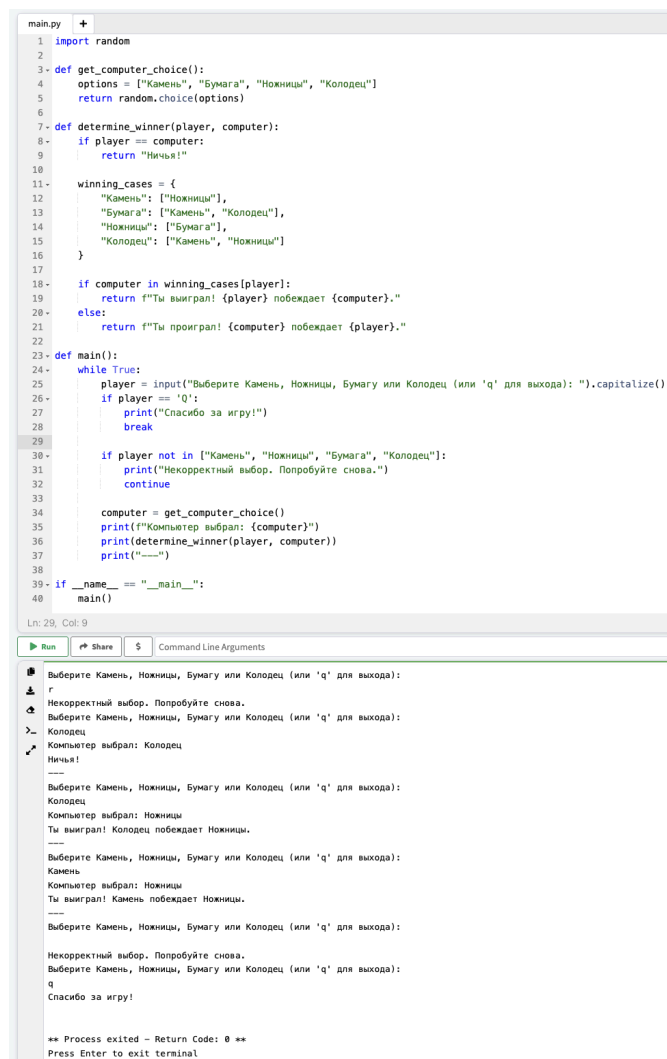
Session Killed due to Timeout.
Press Enter to exit terminal
```

Рисунок 4.1 – Результат работы программы

Программа представляет собой консольную игру, в которой пользователь выбирает один из четырех вариантов: "Камень", "Ножницы", "Бумага" или "Колодец". Компьютер также делает случайный выбор. После этого определяется победитель по оговоренным правилам.

Основные доработки:

- Добавлена функция `get_computer_choice()` для случайного выбора компьютером одного из вариантов.
- Введена функция `determine_winner()`, содержащая правила игры и определяющая победителя.
- Основной игровой цикл вынесен в `main()`, что улучшает читаемость и поддержку кода.
- Добавлена обработка некорректного ввода пользователя.
- Реализована возможность выхода из игры при вводе `q`.



```
main.py +
1 import random
2
3 def get_computer_choice():
4     options = ["Камень", "Бумага", "Ножницы", "Колодец"]
5     return random.choice(options)
6
7 def determine_winner(player, computer):
8     if player == computer:
9         return "Ничья!"
10
11     winning_cases = {
12         "Камень": ["Ножницы"],
13         "Бумага": ["Камень", "Колодец"],
14         "Ножницы": ["Бумага"],
15         "Колодец": ["Камень", "Ножницы"]
16     }
17
18     if computer in winning_cases[player]:
19         return f"Ты выиграл! {player} побеждает {computer}."
20     else:
21         return f"Ты проиграл! {computer} побеждает {player}."
22
23 def main():
24     while True:
25         player = input("Выберите Камень, Ножницы, Бумагу или Колодец (или 'q' для выхода): ").capitalize()
26         if player == 'q':
27             print("Спасибо за игру!")
28             break
29
30         if player not in ["Камень", "Ножницы", "Бумага", "Колодец"]:
31             print("Некорректный выбор. Попробуйте снова.")
32             continue
33
34         computer = get_computer_choice()
35         print(f"Компьютер выбрал: {computer}")
36         print(determine_winner(player, computer))
37         print("----")
38
39 if __name__ == "__main__":
40     main()

Ln: 29, Col: 9

Run Share Command Line Arguments

Выберите Камень, Ножницы, Бумагу или Колодец (или 'q' для выхода):
г
Некорректный выбор. Попробуйте снова.
Выберите Камень, Ножницы, Бумагу или Колодец (или 'q' для выхода):
>_ Колодец
Компьютер выбрал: Колодец
Ничья!
Выберите Камень, Ножницы, Бумагу или Колодец (или 'q' для выхода):
Колодец
Компьютер выбрал: Ножницы
Ты выиграл! Колодец побеждает Ножницы.
----
Выберите Камень, Ножницы, Бумагу или Колодец (или 'q' для выхода):
Камень
Компьютер выбрал: Ножницы
Ты выиграл! Камень побеждает Ножницы.
----
Выберите Камень, Ножницы, Бумагу или Колодец (или 'q' для выхода):
Некорректный выбор. Попробуйте снова.
Выберите Камень, Ножницы, Бумагу или Колодец (или 'q' для выхода):
q
Спасибо за игру!

** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

Рисунок 4.2 – Результат работы доработанной программы

Задача 5. Последовательность Фибоначчи

Описание программы: Числа Фибоначчи – это ряд чисел, в котором каждое следующее число равно сумме двух предыдущих: 1, 1, 2, 3, 5, 8, 13, Иногда ряд начинают с нуля: 0, 1, 1, 2, 3, 5, В данном случае мы будем придерживаться первого варианта. Пользователю необходимо ввести произвольное число, далее, программа выводит ряд чисел.

Задание: Отладьте код программы, зафиксируйте работу программы скриншотами в отчете и потом напишите новую функцию fibRecurse, которая использует рекурсивный алгоритм вычисления чисел Фибоначчи, который основан на рекуррентных отношениях чисел Фибоначчи:

$$F_1 = F_2 = 1, F_n = F_{n-1} + F_{n-2}.$$



```
main.py +
1- def fibSequence(n):
2-
3-     assert n > 0
4-
5-     series = [1]
6-
7-     while len(series) < n:
8-
9-         if len(series) == 1:
10-
11-             series.append(1)
12-
13-         else:
14-
15-             series.append(series[-1] + series[-2])
16-
17-     for i in range(len(series)):
18-
19-         series[i] = str(series[i])
20-
21-     return(', '.join(series))
22-
23- print(fibSequence(int(input('Сколько чисел? '))))
```

Ln: 23, Col: 50

Run Share \$ Command Line Arguments

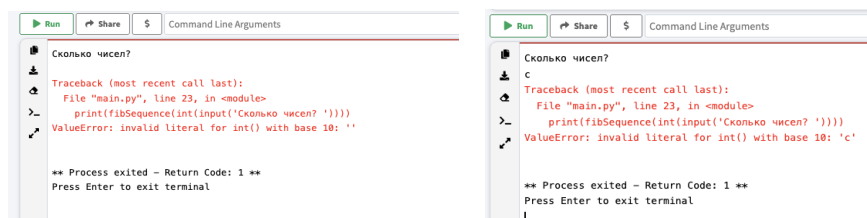
Сколько чисел?

10

1, 1, 2, 3, 5, 8, 13, 21, 34, 55

** Process exited - Return Code: 0 **
Press Enter to exit terminal

Рисунок 5.1 – Результат работы программы



Сколько чисел?

Traceback (most recent call last):
File "main.py", line 23, in <module>
print(fibSequence(int(input('Сколько чисел? '))))
ValueError: invalid literal for int() with base 10: ''

** Process exited - Return Code: 1 **
Press Enter to exit terminal

Сколько чисел?

c

Traceback (most recent call last):
File "main.py", line 23, in <module>
print(fibSequence(int(input('Сколько чисел? '))))
ValueError: invalid literal for int() with base 10: 'c'

** Process exited - Return Code: 1 **
Press Enter to exit terminal

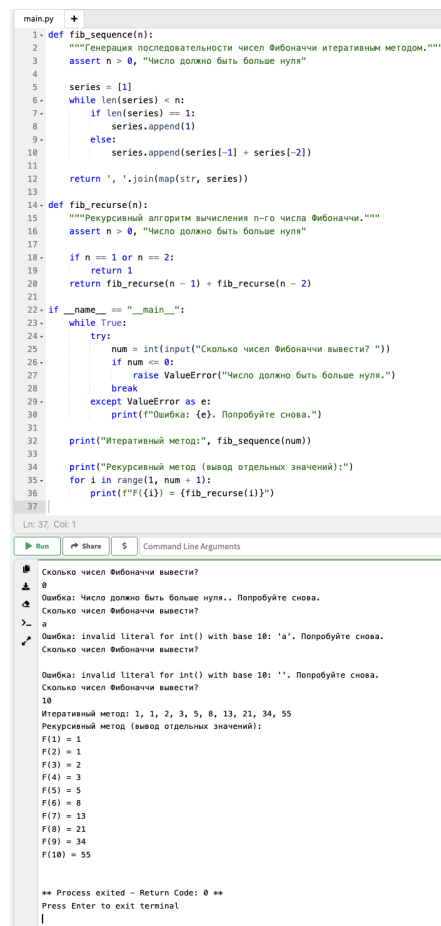
Рисунок 5.2 – Ошибки при вводе некорректных данных

Программа запрашивает у пользователя количество чисел Фибоначчи, которые необходимо вывести. После этого она:

1. Генерирует последовательность чисел Фибоначчи итеративным методом и

выводит её.

- Использует рекурсивный алгоритм для вычисления и отображения каждого числа Фибоначчи отдельно.
- Обрабатывает возможные ошибки ввода.



```
main.py +
1 def fib_sequence(n):
2     """Генерация последовательности чисел Фибоначчи итеративным методом."""
3     assert n > 0, "Число должно быть больше нуля"
4
5     series = [1]
6     while len(series) < n:
7         if len(series) == 1:
8             series.append(1)
9         else:
10            series.append(series[-1] + series[-2])
11
12    return ', '.join(map(str, series))
13
14 def fib_recurse(n):
15     """Рекурсивный алгоритм вычисления n-го числа Фибоначчи."""
16     assert n > 0, "Число должно быть больше нуля"
17
18     if n == 1 or n == 2:
19         return 1
20     return fib_recurse(n - 1) + fib_recurse(n - 2)
21
22 if __name__ == "__main__":
23     while True:
24         try:
25             num = int(input("Сколько чисел Фибоначчи вывести? "))
26             if num <= 0:
27                 raise ValueError("Число должно быть больше нуля.")
28             break
29         except ValueError as e:
30             print(f"Ошибка: {e}. Попробуйте снова.")
31
32     print("Итеративный метод:", fib_sequence(num))
33
34     print("Рекурсивный метод (вывод отдельных значений):")
35     for i in range(1, num + 1):
36         print(f"F({i}) = {fib_recurse(i)}")
37
Ln: 37, Col: 1
Run Share Command Line Arguments
Сколько чисел Фибоначчи вывести?
0
Ошибка: Число должно быть больше нуля.. Попробуйте снова.
Сколько чисел Фибоначчи вывести?
a
Ошибка: invalid literal for int() with base 10: 'a'. Попробуйте снова.
Сколько чисел Фибоначчи вывести?
10
Итеративный метод: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
Рекурсивный метод (вывод отдельных значений):
F(1) = 1
F(2) = 1
F(3) = 2
F(4) = 3
F(5) = 5
F(6) = 8
F(7) = 13
F(8) = 21
F(9) = 34
F(10) = 55
** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

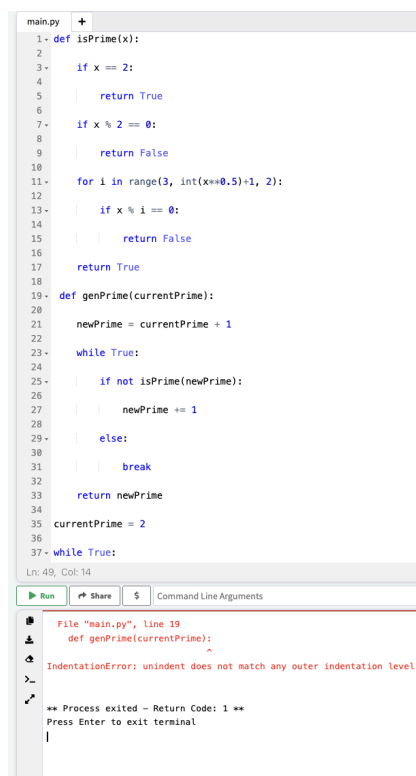
Рисунок 5.3 – Результат работы доработанной программы

- Добавлен рекурсивный метод `fib_recurse(n)`, вычисляющий n -е число Фибоначчи.
- Обновлён итеративный метод `fib_sequence(n)`, формирующий последовательность чисел Фибоначчи.
- Добавлена обработка некорректного ввода:
- Если введено нечисловое значение, программа сообщает об ошибке и запрашивает ввод снова.
- Если введено число меньше или равно нулю, программа выводит соответствующее сообщение и просит повторный ввод.
- Улучшена читаемость и структура кода, что делает его удобным для дальнейшего расширения.

Задача 6. Простые числа

Описание программы: Простое число — это число, у которого только два делителя: 1 и само число. Программа запрашивает у пользователя разрешение на вывод следующего просто числа.

Задание: Отладьте код программы, зафиксируйте работу программы скриншотами в отчете и измените программу так, чтобы она запрашивала у пользователя число и отвечала – простое оно или нет, и, если число не простое, то выводила на экран БЛИЖАЙШЕЕ простое число.



```
main.py +
1- def isPrime(x):
2-
3-     if x == 2:
4-         return True
5-
6-     if x % 2 == 0:
7-         return False
8-
9-     for i in range(3, int(x**0.5)+1, 2):
10-
11-         if x % i == 0:
12-             return False
13-
14-     return True
15-
16- def genPrime(currentPrime):
17-
18-     newPrime = currentPrime + 1
19-
20-     while True:
21-
22-         if not isPrime(newPrime):
23-             newPrime += 1
24-
25-         else:
26-             break
27-
28-     return newPrime
29-
30- currentPrime = 2
31-
32- while True:
33-
34-     print(currentPrime)
35-
36-     currentPrime = genPrime(currentPrime)
37-
38-     if input() == '':
39-         break
40-
41-     currentPrime = genPrime(currentPrime)
```

Ln: 49, Col: 14

Run Share \$ Command Line Arguments

File "main.py", line 19
def genPrime(currentPrime):
^
IndentationError: unindent does not match any outer indentation level

** Process exited - Return Code: 1 **
Press Enter to exit terminal
|

Рисунок 6.1 – Результат работы программы

Функция isPrime(x)

Эта функция проверяет, является ли число x простым:

- Число 2 — простое, поэтому сразу возвращает True.
- Если число меньше 2 или четное (кроме 2), оно не является простым.
- Проверяет делители числа от 3 до \sqrt{x} с шагом 2 (исключая четные числа).
- Если число делится без остатка на какой-либо из этих делителей, оно не является простым.
- В противном случае возвращает True.

Функция genPrime(currentPrime)

Функция генерирует следующее простое число, начиная с currentPrime + 1:

- Увеличивает число на 1.
- Проверяет его на простоту с помощью isPrime().
- Если число не простое, увеличивает его на 1 и повторяет проверку.
- Возвращает первое найденное простое число.

Основной алгоритм программы

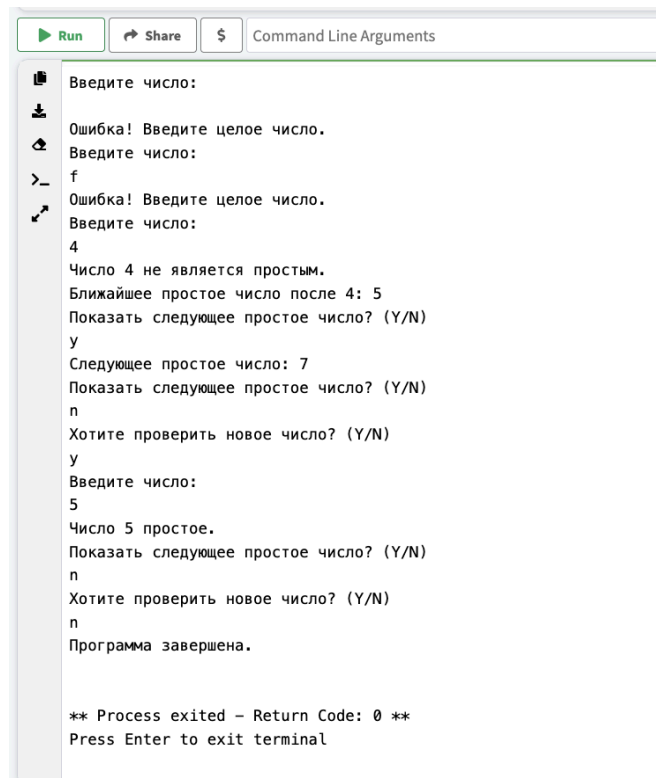
- Пользователь вводит число.
- Программа проверяет его на простоту.
- Если число не является простым, находит ближайшее следующее простое число.
- Спрашивает пользователя, хочет ли он увидеть следующее простое число.
- Если "Y" – выводит следующее простое число и продолжает цикл.
- Если "N" – предлагает ввести новое число.
- Пользователь может повторить ввод нового числа.
- Если пользователь отказывается, программа завершает работу.

```
def isPrime(x):  
    if x == 2:  
        return True  
    if x < 2 or x % 2 == 0:  
        return False  
    for i in range(3, int(x**0.5) + 1, 2):  
        if x % i == 0:  
            return False  
    return True  
def genPrime(currentPrime):  
    newPrime = currentPrime + 1  
    while not isPrime(newPrime):  
        newPrime += 1  
    return newPrime  
while True:  
    try:  
        user_number = int(input("Введите число: "))  
    except ValueError:  
        print("Ошибка! Введите целое число.")  
        continue  
    if isPrime(user_number):
```

```

    print(f"Число {user_number} простое.")
else:
    print(f"Число {user_number} не является простым.")
    nearest_prime = genPrime(user_number)
    print(f"Ближайшее простое число после {user_number}: {nearest_prime}")
currentPrime = nearest_prime
while True:
    answer = input("Показать следующее простое число? (Y/N) ").strip().lower()
    if answer.startswith('y'):
        currentPrime = genPrime(currentPrime)
        print(f"Следующее простое число: {currentPrime}")
    elif answer.startswith('n'):
        break
    else:
        print("Введите 'Y' для продолжения или 'N' для выхода.")
    repeat = input("Хотите проверить новое число? (Y/N) ").strip().lower()
    if repeat.startswith('n'):
        print("Программа завершена.")
        break

```



```

Run Share $ Command Line Arguments
Введите число:
Ошибка! Введите целое число.
Введите число:
f
Ошибка! Введите целое число.
Введите число:
4
Число 4 не является простым.
Ближайшее простое число после 4: 5
Показать следующее простое число? (Y/N)
y
Следующее простое число: 7
Показать следующее простое число? (Y/N)
n
Хотите проверить новое число? (Y/N)
y
Введите число:
5
Число 5 простое.
Показать следующее простое число? (Y/N)
n
Хотите проверить новое число? (Y/N)
n
Программа завершена.

** Process exited - Return Code: 0 **
Press Enter to exit terminal

```

Рисунок 6.2 – Результат работы доработанной программы

Задача 7. Калькулятор

Описание программы: программа выполняет над двумя вещественными числами одну из четырех арифметических операций (сложение, вычитание, умножение или деление).

Задание: Отладьте код программы, зафиксируйте работу программы скриншотами в отчете и добавьте в калькулятор операции: возведение в степень, вычисление процента числа, извлечение корня.

```
1 def calc(a, b, op):
2     if op not in '+-/*':
3         return 'Пожалуйста, выберите тип операции: "+, -, *, /"'
4     if op == '+':
5         return(str(a) + ' ' + op + ' ' + str(b) + ' = ' + str(a + b))
6     if op == '-':
7         return(str(a) + ' ' + op + ' ' + str(b) + ' = ' + str(a - b))
8     if op == '*':
9         return(str(a) + ' ' + op + ' ' + str(b) + ' = ' + str(a * b))
10    if op == '/':
11        return(str(a) + ' ' + op + ' ' + str(b) + ' = ' + str(a / b))
12 def main():
13     a = int(input('Пожалуйста, введите первое число: '))
14     b = int(input('Пожалуйста, введите второе число: '))
15     op = input(
16         'Какой вид операции Вы желаете осуществить?\n'
17         'Выберите между "+, -, *, /" : ')
18     print(calc(a, b, op))
19 if __name__ == '__main__':
20     main()
```

Ln: 1, Col: 20

Run Share \$ Command Line Arguments

Пожалуйста, введите первое число:
8
Пожалуйста, введите второе число:
17
Какой вид операции Вы желаете осуществить?
Выберите между "+, -, *, /" :
-
8 - 17 = -9

Рисунок 7.1 – Результат работы программы

<pre>Пожалуйста, введите первое число: 9 Traceback (most recent call last): File "main.py", line 20, in <module> main() File "main.py", line 13, in main a = int(input('Пожалуйста, введите первое число: ')) ValueError: invalid literal for int() with base 10: 'g' ** Process exited - Return Code: 1 ** Press Enter to exit terminal</pre>	<pre>Пожалуйста, введите первое число: 4 Пожалуйста, введите второе число: 6 Какой вид операции Вы желаете осуществить? Выберите между "+, -, *, /" : h Пожалуйста, выберите тип операции: "+, -, *, /"! ** Process exited - Return Code: 0 ** Press Enter to exit terminal</pre>
---	--

Рисунок 7.2 – Ошибки при вводе некорректных данных

Программа запрашивает у пользователя два числа и операцию, которую он хочет выполнить. Далее выполняется вычисление и выводится результат.

Доступны следующие операции:

- Сложение (+)
- Вычитание (-)
- Умножение (*)
- Деление (/)
- Возведение в степень (**)
- Вычисление процента числа (%)
- Извлечение квадратного корня (sqrt)

Добавлены новые математические операции:

- Возведение в степень (**)
- Вычисление процента (%)
- Извлечение квадратного корня (sqrt)

Обновлена обработка ошибок:

- Если пользователь вводит некорректное число (буквы, символы), программа просит ввести его заново.
- Исключено деление на ноль.
- Если введена некорректная операция, программа повторно запрашивает ввод.

```
import math

def calc(a, b, op):

    """Функция выполняет арифметические операции над двумя числами."""

    if op not in ['+', '-', '*', '/', '**', '%', 'sqrt']:

        return 'Пожалуйста, выберите корректный тип операции: +, -, *, /, **, %, sqrt'

    if op == '+':

        return f'{a} + {b} = {a + b}'

    elif op == '-':
```

```

    return f'{a} - {b} = {a - b}'

elif op == '*':

    return f'{a} * {b} = {a * b}'

elif op == '/':

    return f'{a} / {b} = {a / b}' if b != 0 else 'Ошибка: Деление на ноль'

elif op == '**':

    return f'{a} ** {b} = {a ** b}'

elif op == '%':

    return f'{b}% от {a} = {a * (b / 100)}'

elif op == 'sqrt':

    return f' $\sqrt{{a}}$  = {math.sqrt(a)},  $\sqrt{{b}}$  = {math.sqrt(b)}'

def get_number(prompt):

    """Функция для безопасного ввода чисел."""

    while True:

        try:

            return float(input(prompt))

        except ValueError:

            print("Ошибка: Введите корректное число!")

def main():

    """Главная функция для ввода данных и выполнения операций."""

    a = get_number('Введите первое число: ')

    b = get_number('Введите второе число: ')

    while True:

        op = input('Выберите операцию (+, -, *, /, **, %, sqrt): ')

        if op in ['+', '-', '*', '/', '**', '%', 'sqrt']:

            break

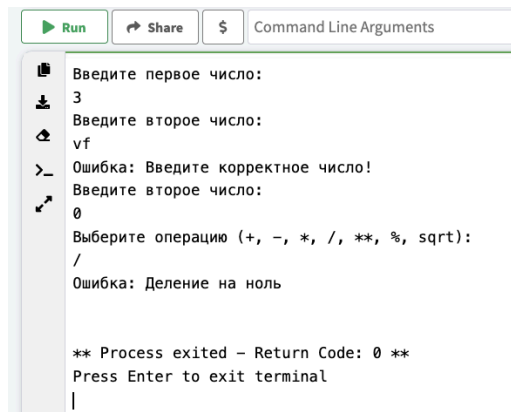
        print("Ошибка: Некорректная операция, попробуйте снова!")

    print(calc(a, b, op))

if __name__ == '__main__':

    main()

```



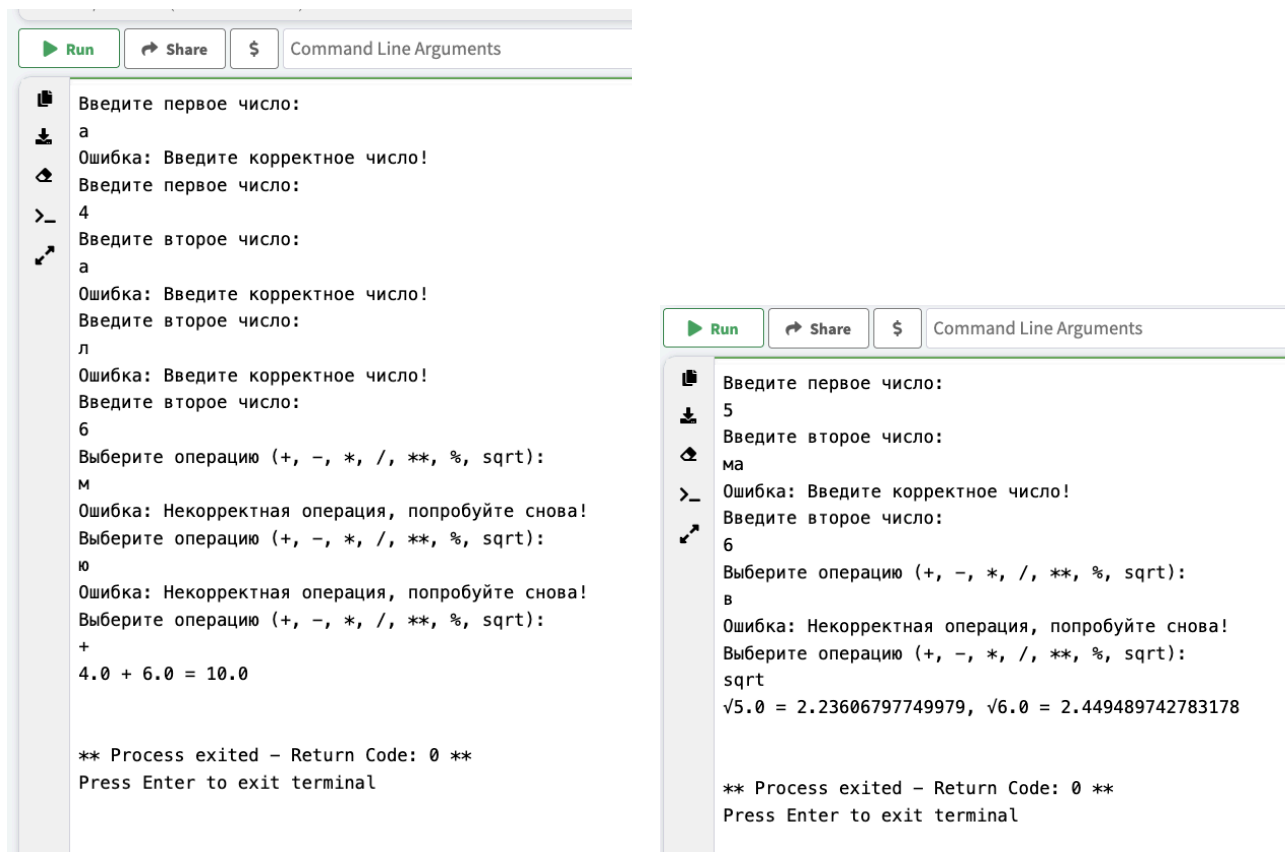
```

Run Share $ Command Line Arguments

Введите первое число:
3
Введите второе число:
sqrt
Ошибка: Введите корректное число!
0
Выберите операцию (+, -, *, /, **, %, sqrt):
/
Ошибка: Деление на ноль

** Process exited - Return Code: 0 **
Press Enter to exit terminal
|

```



```
Введите первое число:
a
Ошибка: Введите корректное число!
Введите первое число:
4
Введите второе число:
a
Ошибка: Введите корректное число!
Введите второе число:
л
Ошибка: Введите корректное число!
Введите второе число:
6
Выберите операцию (+, -, *, /, **, %, sqrt):
м
Ошибка: Некорректная операция, попробуйте снова!
Выберите операцию (+, -, *, /, **, %, sqrt):
ю
Ошибка: Некорректная операция, попробуйте снова!
Выберите операцию (+, -, *, /, **, %, sqrt):
+
4.0 + 6.0 = 10.0

** Process exited - Return Code: 0 **
Press Enter to exit terminal

Введите первое число:
5
Введите второе число:
ma
Ошибка: Введите корректное число!
Введите второе число:
6
Выберите операцию (+, -, *, /, **, %, sqrt):
в
Ошибка: Некорректная операция, попробуйте снова!
Выберите операцию (+, -, *, /, **, %, sqrt):
sqrt
√5.0 = 2.23606797749979, √6.0 = 2.449489742783178

** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

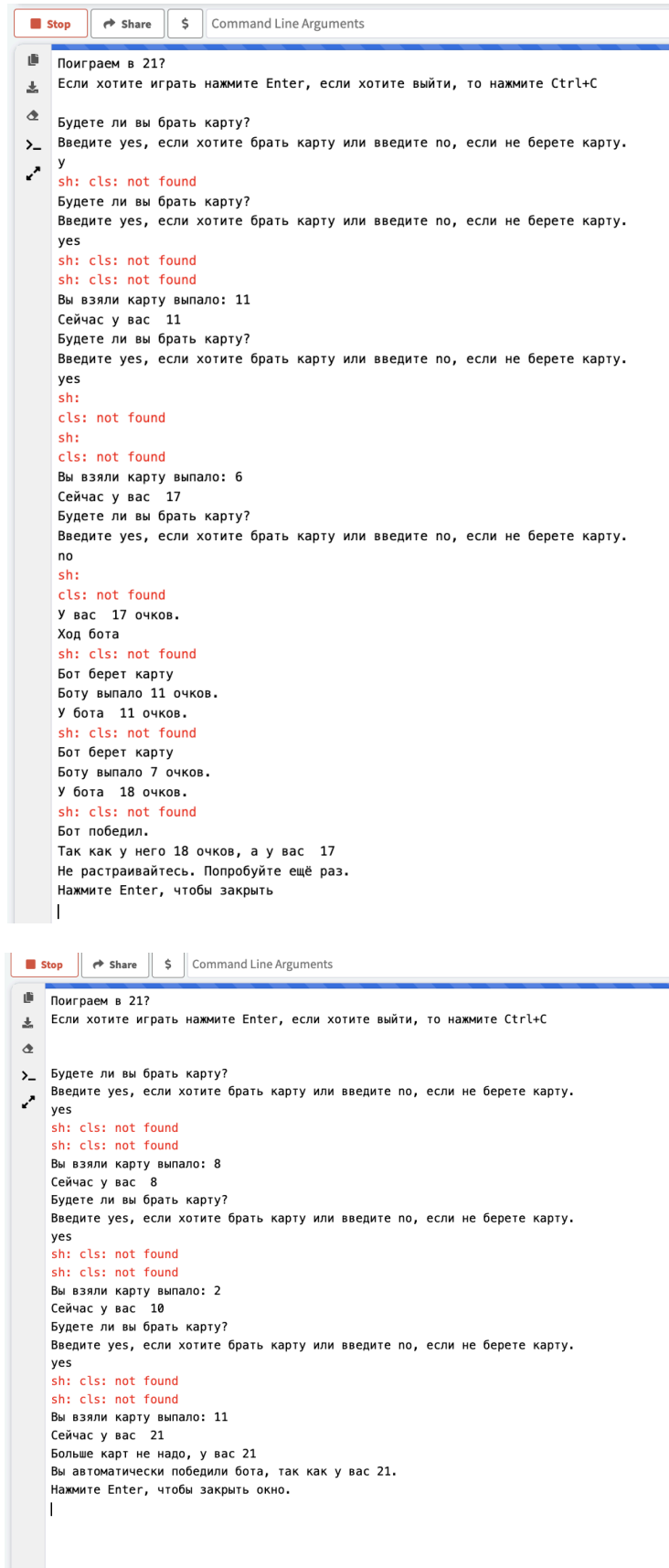
Рисунок 7.3 – Результат работы доработанной программы

Задача 8. Игра «21 point»

Описание программы: Изначально у пользователя 0 очков. Программа спрашивает у пользователя, хочет ли он взять карту. Пользователь должен ввести в консоль либо у – yes, либо n – no. Если пользователь ответил n, то программа озвучивает набранное количество очков и завершает свою работу. Если пользователь ввел команду “у”, то игра продолжается и выдается 1 карта из списка. Далее, программа прибавляет к числу очков снятую карту и выводит количество очков. Если количество очков больше 21, то пользователь проигрывает, и программа завершает свою работу. Если число очков равно 21, то пользователь выиграл. Если меньше - программа снова предлагает взять карту. После окончания игры, программа прощается с пользователем.

Задание: Отладьте код программы, зафиксируйте работу программы скриншотами в отчете и добавьте для бота коэффициент – уровень сложности (он же – уровень подглядывания в колоду) – число от 0 до 1. Чем ближе к 1, тем сильнее играет бот. На последнем ходе бота делается проверка случайного числа, и если оно больше коэффициента, то последний ход бот делает случайно, а если

меньше – то бот автоматически подбирает нужную карту, чтобы выиграть.



```

[ Stop ] [ Share ] [ $ ] Command Line Arguments

Поиграем в 21?
Если хотите играть нажмите Enter, если хотите выйти, то нажмите Ctrl+C

Будете ли вы брать карту?
Введите yes, если хотите брать карту или введите no, если не берете карту.
y
sh: cls: not found
Будете ли вы брать карту?
Введите yes, если хотите брать карту или введите no, если не берете карту.
yes
sh: cls: not found
sh: cls: not found
Вы взяли карту выпало: 11
Сейчас у вас 11
Будете ли вы брать карту?
Введите yes, если хотите брать карту или введите no, если не берете карту.
yes
sh:
cls: not found
sh:
cls: not found
Вы взяли карту выпало: 6
Сейчас у вас 17
Будете ли вы брать карту?
Введите yes, если хотите брать карту или введите no, если не берете карту.
no
sh:
cls: not found
У вас 17 очков.
Ход бота
sh: cls: not found
Бот берет карту
Боту выпало 11 очков.
У бота 11 очков.
sh: cls: not found
Бот берет карту
Боту выпало 7 очков.
У бота 18 очков.
sh: cls: not found
Бот победил.
Так как у него 18 очков, а у вас 17
Не расстраивайтесь. Попробуйте ещё раз.
Нажмите Enter, чтобы закрыть
|

[ Stop ] [ Share ] [ $ ] Command Line Arguments

Поиграем в 21?
Если хотите играть нажмите Enter, если хотите выйти, то нажмите Ctrl+C

Будете ли вы брать карту?
Введите yes, если хотите брать карту или введите no, если не берете карту.
yes
sh: cls: not found
sh: cls: not found
Вы взяли карту выпало: 8
Сейчас у вас 8
Будете ли вы брать карту?
Введите yes, если хотите брать карту или введите no, если не берете карту.
yes
sh: cls: not found
sh: cls: not found
Вы взяли карту выпало: 2
Сейчас у вас 10
Будете ли вы брать карту?
Введите yes, если хотите брать карту или введите no, если не берете карту.
yes
sh: cls: not found
sh: cls: not found
Вы взяли карту выпало: 11
Сейчас у вас 21
Больше карт не надо, у вас 21
Вы автоматически победили бота, так как у вас 21.
Нажмите Enter, чтобы закрыть окно.
|
```

Рисунок 8.1 – Результат работы программы

Исправленный код добавляет ввод коэффициента сложности бота и

улучшает логику принятия решений ботом. Теперь бот может стратегически подбирать карту на последнем ходе, если уровень сложности высокий.

```
import random
import os
import time

# Счет
score_player = 0
score_bot = 0

# Уровень сложности (0 - бот глупый, 1 - бот играет идеально)
difficulty = float(input("Введите уровень сложности бота (от 0 до 1): "))

# Начальное сообщение
all_cards = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11]

print("Поиграем в 21? \nЕсли хотите играть, нажмите Enter, если хотите выйти, то нажмите Ctrl+C")
input()

while True:
    if score_player == 21:
        print("Больше карт не надо, у вас 21")
        print("Вы автоматически победили бота, так как у вас 21.")
        input("Нажмите Enter, чтобы закрыть окно."); break
    if score_player > 21:
        print("Вы проиграли, так как набрали больше 21")
        print("Попробуйте свою удачу в другой раз.")
        input("Нажмите Enter, чтобы закрыть окно."); break
    yes_or_no = input("Будете ли вы брать карту? (y/n): ").strip().lower()
    os.system('cls' if os.name == 'nt' else 'clear')
    if yes_or_no == 'y':
        card = random.choice(all_cards)
        print(f"Вы взяли карту: {card}")
        score_player += card
        print(f"Сейчас у вас {score_player} очков")
    elif yes_or_no == 'n':
        print(f"У вас {score_player} очков.")
        print("Ход бота...")
        time.sleep(2)
        os.system('cls' if os.name == 'nt' else 'clear')
        while True:
            if score_bot < 15:
                print("Бот берет карту")
                card = random.choice(all_cards)
                print(f"Боту выпало {card} очков.")
                score_bot += card
                print(f"У бота {score_bot} очков.")
                time.sleep(2)
                os.system('cls' if os.name == 'nt' else 'clear')
            else:
                if random.random() > difficulty:
                    card = random.choice(all_cards)
```

```

    print(f"Бот берет случайную карту: {card}")
else:
    card = min(21 - score_bot, max(all_cards)) if 21 - score_bot > 0 else 0
    if card > 0:
        print(f"Бот стратегически взял карту: {card}")
    score_bot += card
    print(f"У бота {score_bot} очков.")
    time.sleep(2)
    os.system('cls' if os.name == 'nt' else 'clear')
if score_bot > 21:
    print(f"Бот проиграл. У него {score_bot} очков, а у вас {score_player}")
    input("Нажмите Enter, чтобы закрыть."); exit(0)
elif score_bot > score_player:
    print(f"Бот победил. У него {score_bot} очков, у вас {score_player}")
    input("Нажмите Enter, чтобы закрыть."); exit(0)
elif score_bot == score_player:
    print("Вы набрали равное количество очков. Ничья!")
    input("Нажмите Enter, чтобы закрыть."); exit(0)

```

```

Run Share $ Command Line Arguments
Введите уровень сложности бота (от 0 до 1):
0
Поиграем в 21?
Если хотите играть, нажмите Enter, если хотите выйти, то нажмите Ctrl+C
>
Будете ли вы брать карту? (y/n):
y
[N] Вы взяли карту: 11
Сейчас у вас 11 очков
Будете ли вы брать карту? (y/n):
y
[N] Вы взяли карту: 9
Сейчас у вас 20 очков
Будете ли вы брать карту? (y/n):
n
[N] У вас 20 очков.
Ход бота...
[N] Бот берет карту
Боту выпало 10 очков.
У бота 10 очков.
[N] Бот берет карту
Боту выпало 7 очков.
У бота 17 очков.
[N] Бот берет случайную карту: 9
У бота 26 очков.
[N] Бот проиграл. У него 26 очков, а у вас 20
Нажмите Enter, чтобы закрыть.

Session Killed due to Timeout.
Press Enter to exit terminal

```

```

Run Share $ Command Line Arguments
Введите уровень сложности бота (от 0 до 1):
1
Поиграем в 21?
Если хотите играть, нажмите Enter, если хотите выйти, то нажмите Ctrl+C
>
Будете ли вы брать карту? (y/n):
y
[N] Вы взяли карту: 6
Сейчас у вас 6 очков
Будете ли вы брать карту? (y/n):
y
[N] Вы взяли карту: 7
Сейчас у вас 13 очков
Будете ли вы брать карту? (y/n):
y
[N] Вы взяли карту: 3
Сейчас у вас 16 очков
Будете ли вы брать карту? (y/n):
y
[N] Вы взяли карту: 2
Сейчас у вас 18 очков
Будете ли вы брать карту? (y/n):
n
[N] У вас 18 очков.
Ход бота...
[N] Бот берет карту
Боту выпало 8 очков.
У бота 8 очков.
[N] Бот берет карту
Боту выпало 8 очков.
У бота 16 очков.
[N] Бот стратегически взял карту: 5
У бота 21 очков.
[N] Бот победил. У него 21 очков, у вас 18
Нажмите Enter, чтобы закрыть.

Session Killed due to Timeout.
Press Enter to exit terminal
|

```

Рисунок 8.2 – Результат работы доработанной программы

Задача 9. Игра в «крестики-нолики»

Описание программы: логическая игра между двумя противниками на квадратном поле 3 на 3 клетки. Один из игроков играет «крестиками», второй — «ноликами». Все действия игры происходят в терминале. Пользователю необходимо выбрать на поле незанятые ячейки, которые отмечены цифрами.

Задание: Отладьте код программы, зафиксируйте работу программы

скриншотами в отчете и модифицируйте ее так, чтобы:

1. для двух игроков поле игры было 5 на 5;
2. на поле 3 на 3 человек играл против компьютера.

```
Куда поставим X?
5
-----
| 1 | 2 | 3 |
-----
| 4 | X | 6 |
-----
| 7 | 8 | 9 |
-----
Куда поставим O?
1
-----
| 0 | 2 | 3 |
-----
| 4 | X | 6 |
-----
| 7 | 8 | 9 |
-----
Куда поставим X?
2
-----
| 0 | X | 3 |
-----
| 4 | X | 6 |
-----
| 7 | 8 | 9 |
-----
Куда поставим O?
8
-----
| 0 | X | 3 |
-----
| 4 | X | 6 |
-----
| 7 | 0 | 9 |
-----
Куда поставим X?
3
-----
| 0 | X | X |
-----
| 4 | X | 6 |
-----
| 7 | 0 | 9 |
-----
Куда поставим O?
7
-----
| 0 | X | X |
-----
| 4 | X | 6 |
-----
| 0 | 0 | 9 |
-----
Куда поставим X?
4
-----
| 0 | X | X |
-----
| X | X | 6 |
-----
| 0 | 0 | 9 |
-----
Куда поставим O?
9
0 выиграл!
-----
| 0 | X | X |
-----
| X | X | 6 |
-----
| 0 | 0 | 0 |
-----
Нажмите Enter для выхода!
```

Рисунок 9.1 – Результат работы программы

Данная программа была модифицирована для добавления больших возможностей для пользователя.

Многовариантность игры: Программа предлагает три режима игры:

- Игра для двух игроков на поле 5x5.
- Игра для одного игрока против компьютера на поле 3x3.
- Игра для одного игрока против компьютера на поле 5x5.

Меню выбора: После запуска программы пользователю предоставляется меню для выбора типа игры. Для выбора режима необходимо ввести соответствующую цифру:

- 1: Игра для двух игроков (5x5).
- 2: Игра против компьютера (3x3).
- 3: Игра против компьютера (5x5).
- 4: Выход из игры.

Игровой процесс: В зависимости от выбранного режима, игроки или игрок и компьютер поочередно делают ходы, ставя на поле свои символы ("X" или "O"). Ход игрока или компьютера происходит путем ввода числа от 1 до 25 (для поля 5x5) или от 1 до 9 (для поля 3x3), указывая свободную клетку на поле.

Проверка победы: После каждого хода программа проверяет, не победил ли кто-либо. Победа считается при условии, что три одинаковых символа расположены в линии (горизонтально, вертикально или по диагонали).

Игровое поле: Поле состоит из клеток, которые пронумерованы. Игроки вводят номер клетки, в которой они хотят сделать ход. Поле динамически изменяется в процессе игры, отображая текущие символы игроков.

Игра против компьютера: В случае игры против компьютера, компьютер выбирает случайную свободную клетку на поле для своего хода.

```
import random
def draw_board(board, size):
    print("-" * (size * 4 + 1))
    for i in range(size):
        print("|", end=" ")
        for j in range(size):
            print(board[i * size + j], end=" | ")
        print()
    print("-" * (size * 4 + 1))
```

```

def take_input(player_token, board, size):
    valid = False
    while not valid:
        player_answer = input(f"Куда поставим {player_token}? ")
        try:
            player_answer = int(player_answer)
        except:
            print("Некорректный ввод. Вы уверены, что ввели число?")
            continue
        if 1 <= player_answer <= size * size:
            if str(board[player_answer - 1]) not in "XO":
                board[player_answer - 1] = player_token
                valid = True
            else:
                print("Эта клетка уже занята!")
        else:
            print("Некорректный ввод. Введите число от 1 до", size * size)

```

```

def check_win(board, size):
    win_coord = []
    # Проверка горизонтальных и вертикальных линий
    for i in range(size):
        win_coord.append([i + j * size for j in range(size)])
        win_coord.append([i * size + j for j in range(size)])

    # Диагонали
    win_coord.append([i * (size + 1) for i in range(size)])
    win_coord.append([i * (size - 1) for i in range(1, size + 1)])

    for each in win_coord:
        if board[each[0]] == board[each[1]] == board[each[2]]:
            return board[each[0]]
    return False

```

```

def main():
    size = 5 # Размер поля 5x5 для двух игроков
    board = list(range(1, size * size + 1))
    counter = 0
    win = False

    while not win:
        draw_board(board, size)
        if counter % 2 == 0:
            take_input("X", board, size)
        else:
            take_input("O", board, size)
        counter += 1
        if counter > 4:
            tmp = check_win(board, size)
            if tmp:
                print(tmp, "выиграл!")

```

```

        win = True
        break
    if counter == size * size:
        print("Ничья!")
        break
draw_board(board, size)
input("Нажмите Enter для выхода!")

def game_with_computer_3x3():
    size = 3 # Поле 3x3 для игры с компьютером
    board = list(range(1, size * size + 1))
    counter = 0
    win = False

    while not win:
        draw_board(board, size)
        if counter % 2 == 0:
            take_input("X", board, size) # Игрок X
        else:
            computer_move(board, size) # Ход компьютера
        counter += 1
        if counter > 4:
            tmp = check_win(board, size)
            if tmp:
                print(tmp, "выиграл!")
                win = True
                break
            if counter == size * size:
                print("Ничья!")
                break
        draw_board(board, size)
        input("Нажмите Enter для выхода!")

def game_with_computer_5x5():
    size = 5 # Поле 5x5 для игры с компьютером
    board = list(range(1, size * size + 1))
    counter = 0
    win = False

    while not win:
        draw_board(board, size)
        if counter % 2 == 0:
            take_input("X", board, size) # Игрок X
        else:
            computer_move(board, size) # Ход компьютера
        counter += 1
        if counter > 4:
            tmp = check_win(board, size)
            if tmp:
                print(tmp, "выиграл!")
                win = True

```

```

        break
    if counter == size * size:
        print("Ничья!")
        break
    draw_board(board, size)
    input("Нажмите Enter для выхода!")

def computer_move(board, size):
    empty_cells = [i for i, x in enumerate(board) if str(x) not in "XO"]
    move = random.choice(empty_cells) + 1
    print(f"Компьютер ставит О в клетку {move}")
    board[move - 1] = "O"

def display_menu():
    print("Выберите режим игры:")
    print("1. Игра 5x5 для двух игроков")
    print("2. Игра 3x3 против компьютера")
    print("3. Игра 5x5 против компьютера")
    print("4. Выйти")

def main_menu():
    while True:
        display_menu()
        choice = input("Введите номер выбора: ")

        if choice == "1":
            print("\nВы выбрали игру 5x5 для двух игроков.\n")
            main()
        elif choice == "2":
            print("\nВы выбрали игру 3x3 против компьютера.\n")
            game_with_computer_3x3()
        elif choice == "3":
            print("\nВы выбрали игру 5x5 против компьютера.\n")
            game_with_computer_5x5()
        elif choice == "4":
            print("Выход из игры. До свидания!")
            break
        else:
            print("Некорректный ввод. Пожалуйста, выберите правильный номер.")

if __name__ == "__main__":
    main_menu()

```

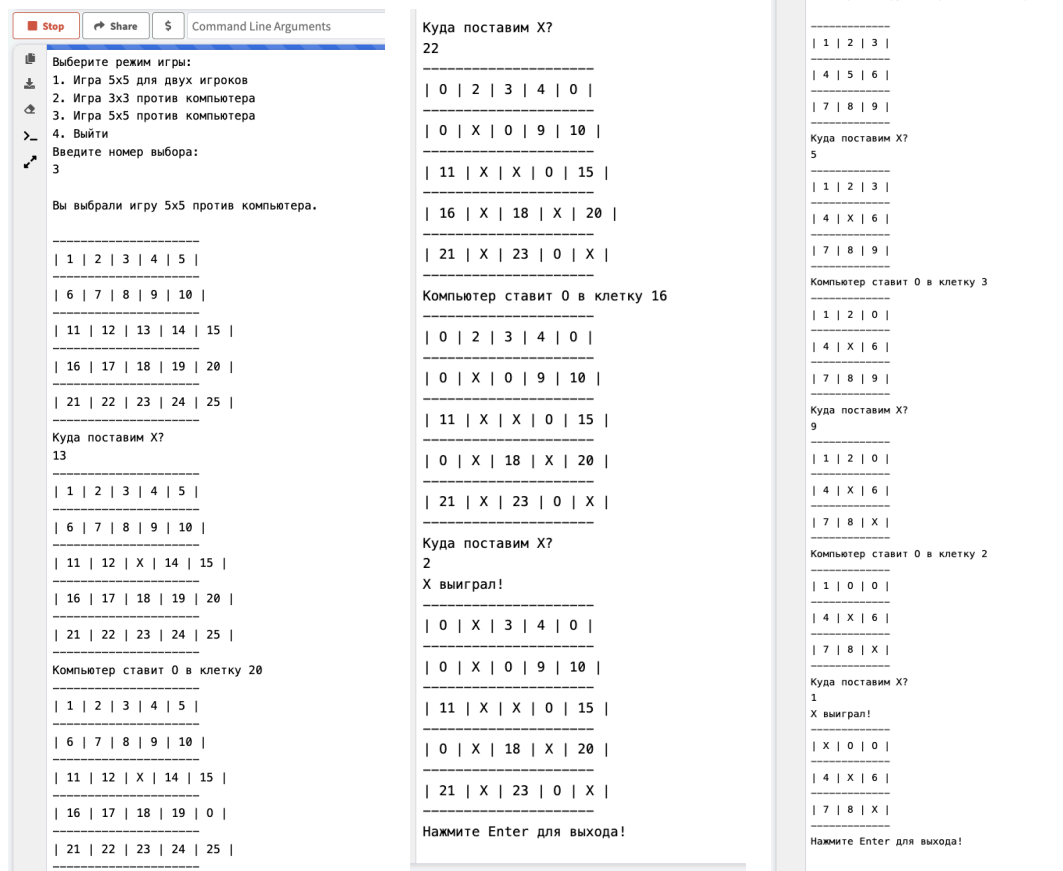


Рисунок 9.2 – Результат работы доработанной программы

Задача 10. Квадратное уравнение

Задание: Самостоятельно написать программу на Python вычисления корней квадратного уравнения $ax^2+bx+c=0$. Учесть все варианты значений коэффициентов a , b , c .

Программа будет использовать дискриминант для вычисления корней уравнения. Алгоритм:

Если $a = 0$ — уравнение становится линейным (не квадратным). Решение для линейного уравнения $bx+c=0$ будет: $x = -c/b$, если $b \neq 0$

Если $a \neq 0$ — решаем квадратное уравнение с использованием дискриминанта: $D = b^2 - 4ac$

- Если $D > 0$, то у уравнения два различных корня.
- Если $D = 0$, то у уравнения один корень (дважды повторяющийся).

- Если $D < 0$, то корней нет (решений нет в области действительных чисел).

```
import math
def solve_quadratic(a, b, c):
    if a == 0:
        # Это линейное уравнение:  $bx + c = 0$ 
        if b != 0:
            x = -c / b
            return f"Это линейное уравнение. Корень:  $x = \{x\}$ "
        elif c == 0:
            return "Уравнение имеет бесконечно много решений."
        else:
            return "Уравнение не имеет решений."

    # Если  $a \neq 0$ , решаем квадратное уравнение  $ax^2 + bx + c = 0$ 
    D = b**2 - 4*a*c # Дискриминант

    if D > 0:
        # два различных корня
        x1 = (-b + math.sqrt(D)) / (2 * a)
        x2 = (-b - math.sqrt(D)) / (2 * a)
        return f"Уравнение имеет два корня:  $x1 = \{x1\}$ ,  $x2 = \{x2\}$ "
    elif D == 0:
        # один корень (дважды повторяющийся)
        x = -b / (2 * a)
        return f"Уравнение имеет один корень:  $x = \{x\}$ "
    else:
        # нет решений
        return "Уравнение не имеет действительных корней."

# Ввод коэффициентов
a = float(input("Введите коэффициент a: "))
b = float(input("Введите коэффициент b: "))
c = float(input("Введите коэффициент c: "))

# Вывод решения
print(solve_quadratic(a, b, c))
```

Функция `solve_quadratic(a, b, c)`: Если $a = 0$, проверяется, является ли уравнение линейным (проверка на $b \neq 0$) и вычисляется корень линейного уравнения $bx+c=0$. Если $b=0$ и $c=0$, то у уравнения бесконечно много решений, если $c \neq 0$, решений нет.

Если $a \neq 0$, вычисляется дискриминант $D = b^2 - 4ac$, и в зависимости от значения дискриминанта:

- При $D > 0$ вычисляются два корня уравнения.
- При $D = 0$ вычисляется один корень.

- При $D < 0$ сообщается, что у уравнения нет действительных корней.

Ввод коэффициентов: Программа запрашивает у пользователя ввод коэффициентов a , b и c .

Вывод решения: После ввода коэффициентов программа вызывает функцию `solve_quadratic()` и выводит соответствующий результат.

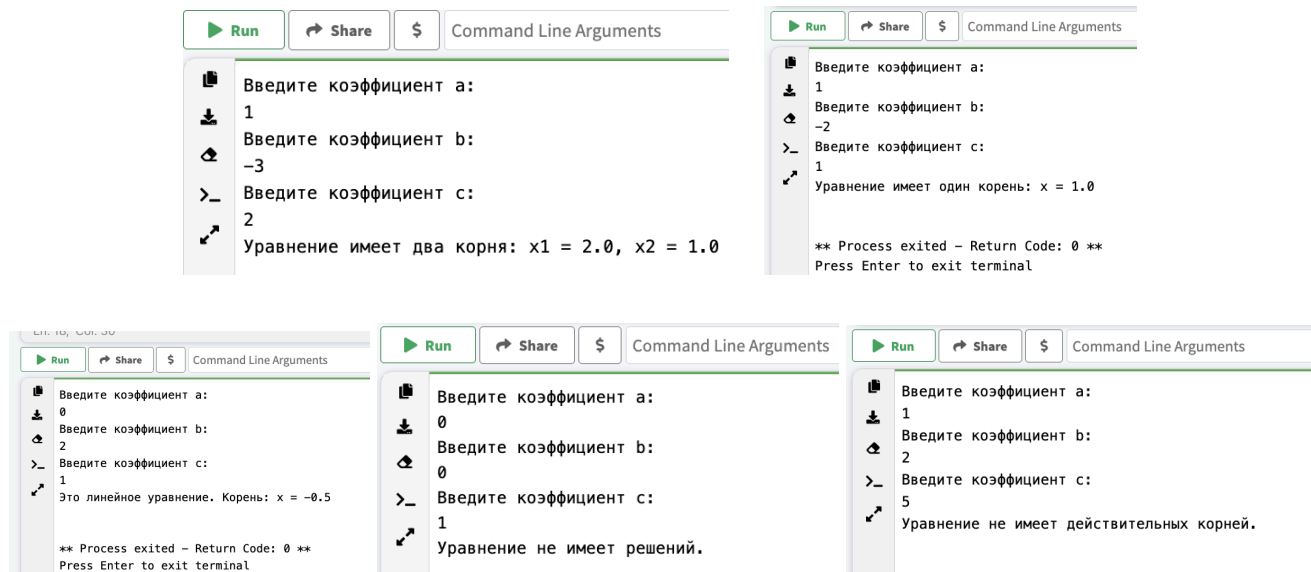


Рисунок 10.1 – Результат работы программы

Задача 11. Тип треугольника

Задание: Самостоятельно написать программу на Python определения типа треугольника. Учесть все возможные сочетания типов (например, равносторонний, равнобедренный-прямоугольный, равнобедренный-остроугольный и т.д.).

Программа должна определять тип треугольника по введенным сторонам. Для этого учитываются следующие типы треугольников:

1. По длинам сторон:
 - a. Равносторонний – все три стороны равны.
 - b. Равнобедренный – две стороны равны.
 - c. Разносторонний – все три стороны различны.
2. По углам (используем теорему Пифагора):
 - a. Прямоугольный – выполняется $c^2 = a^2 + b^2$ (где c – наибольшая

сторона).

b. Тупоугольный – выполняется $c^2 > a^2 + b^2$.

c. Остроугольный – выполняется $c^2 < a^2 + b^2$.

3. Дополнительные комбинации:

a. Равносторонний треугольник всегда остроугольный.

b. Равнобедренный треугольник может быть остроугольным, тупоугольным или прямоугольным.

```
import math

def check_triangle(a, b, c):
    """Проверяет существование треугольника"""
    return a + b > c and a + c > b and b + c > a

def triangle_type(a, b, c):
    """Определяет тип треугольника"""
    # Проверка существования треугольника
    if not check_triangle(a, b, c):
        return "Такого треугольника не существует!"
    # Сортировка сторон (c - гипотенуза для проверки углов)
    sides = sorted([a, b, c])
    a, b, c = sides
    # Проверка по длинам сторон
    if a == b == c:
        return "Равносторонний и остроугольный треугольник"
    elif a == b or b == c:
        side_type = "Равнобедренный"
    else:
        side_type = "Разносторонний"
    # Проверка по углам (теорема Пифагора)
    if math.isclose(c**2, a**2 + b**2): # Прямоугольный
        angle_type = "Прямоугольный"
    elif c**2 > a**2 + b**2: # Тупоугольный
        angle_type = "Тупоугольный"
    else: # Остроугольный
        angle_type = "Остроугольный"
    return f"{side_type} и {angle_type} треугольник"

a = float(input("Введите длину стороны a: "))
b = float(input("Введите длину стороны b: "))
c = float(input("Введите длину стороны c: "))
print(triangle_type(a, b, c))
```

- Функция `check_triangle(a, b, c)` – проверяет, существует ли треугольник, используя неравенство треугольника: $a + b > c$, $a + c > b$, $b + c > a$

Если хотя бы одно из условий не выполняется, треугольник невозможен.

- Функция `triangle_type(a, b, c)` – определяет вид треугольника:

Использует `check_triangle()` для проверки существования.

Сортирует стороны (чтобы *c* всегда была самой длинной) и проверяет:

- Равносторонний (всегда остроугольный).
 - Равнобедренный или разносторонний.
 - Прямоугольный, тупоугольный или остроугольный (по теореме Пифагора).
- Функция `math.isclose()` – используется для сравнения значений, чтобы избежать ошибок округления.

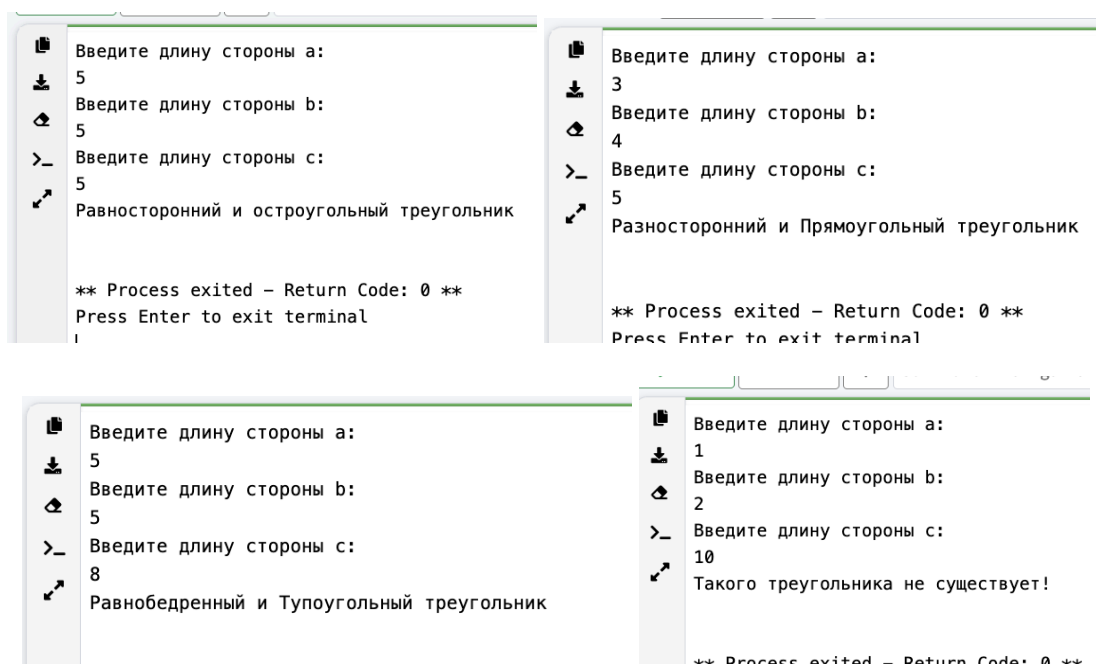


Рисунок 11.1 – Результат работы программы

Задача 12. Табулирование функции

Задание: Самостоятельно написать программу на Python табулирования функции (выбрать функцию самостоятельно) на интервале $[A, B]$ с шагом H . Использовать форматированный вывод.

Для табулирования функции на интервале $[A, B]$ с шагом H мы будем вычислять значения функции $f(x)$ для всех x в заданном диапазоне.

Выберем функцию: $f(x) = x^2 - 2x + 1$

Это квадратная функция, представляющая собой параболу.

```

import numpy as np

def f(x):
    """Функция  $f(x) = x^2 - 2x + 1$ """
    return x**2 - 2*x + 1

# Ввод параметров
A = float(input("Введите начало интервала A: "))
B = float(input("Введите конец интервала B: "))
H = float(input("Введите шаг H: "))

# Проверка корректности ввода
if H <= 0:
    print("Ошибка: Шаг H должен быть положительным числом.")
elif A > B:
    print("Ошибка: Начало интервала A должно быть меньше конца интервала B.")
else:
    # Заголовок таблицы
    print("\nТаблица значений функции  $f(x) = x^2 - 2x + 1$ :")
    print("+-----+-----+")
    print("|   x   | f(x) |")
    print("+-----+-----+")

    # Вычисление и вывод значений
    x = A
    while x <= B:
        print(f"| {x:^9.3f} | {f(x):^10.3f} |")
        x += H
    print("+-----+-----+")

```

Функция $f(x)$ вычисляет значения функции $f(x) = x^2 - 2x + 1$.

Ввод параметров A , B и H .

Проверка ввода:

- Шаг H должен быть положительным.
- Начало интервала A должно быть меньше конца B .

Форматированный вывод таблицы:

- Используется `while x <= B`, чтобы пройти по всем точкам.
- Числа выводятся с точностью до 3 знаков после запятой (.3f).
- Таблица красиво оформлена с границами.

Введите начало интервала A:	0
Введите конец интервала B:	1
Введите шаг H:	0.05
Таблица значений функции $f(x) = x^2 - 2x + 1$:	
x	f(x)
0.000	1.000
0.050	0.902
0.100	0.810
0.150	0.722
0.200	0.640
0.250	0.562
0.300	0.490
0.350	0.422
0.400	0.360
0.450	0.302
0.500	0.250
0.550	0.203
0.600	0.160
0.650	0.123
0.700	0.090
0.750	0.062
0.800	0.040
0.850	0.022
0.900	0.010
0.950	0.002

Введите начало интервала A:	-2
Введите конец интервала B:	3
Введите шаг H:	0.5
Таблица значений функции $f(x) = x^2 - 2x + 1$:	
x	f(x)
-2.000	9.000
-1.500	6.250
-1.000	4.000
-0.500	2.250
0.000	1.000
0.500	0.250
1.000	0.000
1.500	0.250
2.000	1.000
2.500	2.250
3.000	4.000

Рисунок 12.1 – Результат работы программы

Вывод

В ходе выполнения практических заданий по созданию различных программ на языке Python были освоены основы данного языка программирования, а также изучены популярные алгоритмы. Созданные программы протестированы и отлажены на различных наборах входных данных.