

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии  
Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №5**  
**дисциплины**  
**«Основы кроссплатформенного программирования»**

Выполнил:  
Сопов Максим Игоревич  
2 курс, группа ИТС-б-о-23-1,  
11.03.02 «Инфокоммуникационные  
технологии и системы связи»,  
очная форма обучения

---

(подпись)

Проверил:  
Доцент департамента цифровых,  
робототехнических систем и  
электроники Воронкин Р.А.

---

(подпись)

Ставрополь, 2024 г.

# ТЕМА: УСЛОВНЫЕ ОПЕРАТОРЫ И ЦИКЛЫ В ЯЗЫКЕ PYTHON

**Цель:** исследовать условные операторы и циклы в языке Python

**Порядок выполнения работы:**

1. Изучил теоретический материал.
2. Приступил к выполнению заданий.
3. Создал общедоступный репозиторий.

**Ссылка на GitHub:** <https://github.com/MaxITS-kurwa/lb5>

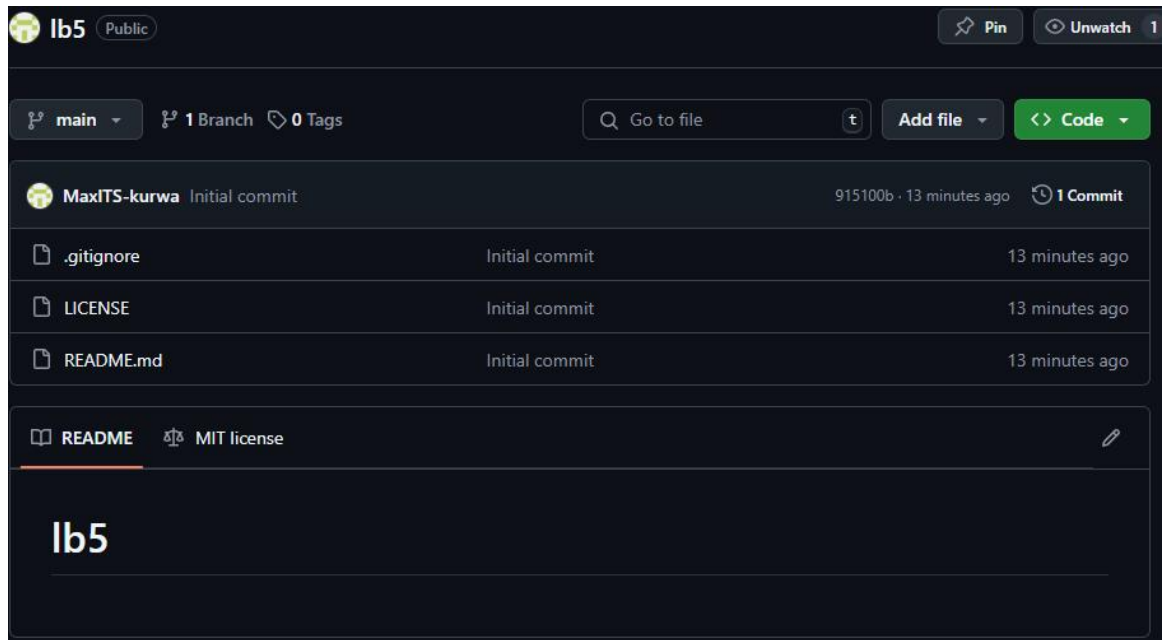


Рисунок 1. Репозиторий

4. Выполнил клонирование репозитория

```
C:\Users\sopov>git clone https://github.com/MaxITS-kurwa/lb5.git
Cloning into 'lb5'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 2. Клонирование

5. Изучил рекомендации PEP-8
6. Создал проект
7. Приступил к проработке примеров

```
Value of x? 4
y = 5.0

Process finished with exit code 0
```

Рисунок 3. Результат первого примера

```
import sys

if __name__ == '__main__':
    n = int(input("Введите номер месяца: "))
    if n == 1 or n == 2 or n == 12:
        print("Зима")
    elif n == 3 or n == 4 or n == 5:
        print("Весна")
    elif n == 6 or n == 7 or n == 8:
        print("Лето")
    elif n == 9 or n == 10 or n == 11:
        print("Осень")
    else:
        print("Ошибка!", file=sys.stderr)
        exit(1)
```

```
Введите номер месяца: 5
Весна

Process finished with exit code 0
```

Рисунок 4. Результат второго примера

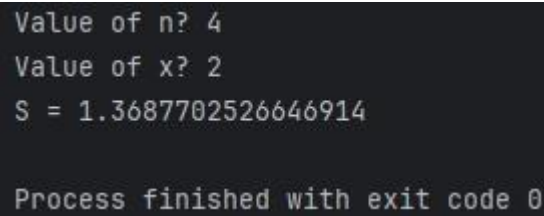
```
import math

if __name__ == '__main__':
    n = int(input("Value of n? "))
    x = float(input("Value of x? "))
    S = 0.0
```

```

for k in range(1, n + 1):
    a = math.log(k * x) / (k * k)
    S += a
print(f'S = {S}')

```



```

Value of n? 4
Value of x? 2
S = 1.3687702526646914

Process finished with exit code 0

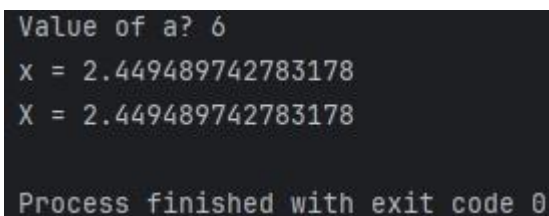
```

Рисунок 5. Результат третьего примера

```

import math
import sys
if __name__ == '__main__':
    a = float(input("Value of a? "))
    if a < 0:
        print("Illegal value of a", file=sys.stderr)
        exit(1)
    x, eps = 1, 1e-10
    while True:
        xp = x
        x = (x + a / x) / 2
        if math.fabs(x - xp) < eps:
            break
    print(f'x = {x}\nX = {math.sqrt(a)}')

```



```

Value of a? 6
x = 2.449489742783178
X = 2.449489742783178

Process finished with exit code 0

```

Рисунок 6. Результат четвертого примера

```

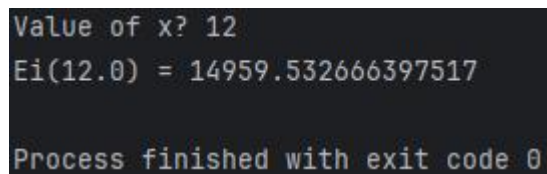
import math
import sys

EULER = 0.5772156649015328606
EPS = 1e-10

if __name__ == '__main__':
    x = float(input("Value of x? "))
    if x == 0:
        print("Illegal value of x", file=sys.stderr)
        exit(1)
    a = x
    S, k = a, 1

    while math.fabs(a) > EPS:
        a *= x * k / (k + 1) ** 2
        S += a
        k += 1
    print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")

```



```

Value of x? 12
Ei(12.0) = 14959.532666397517
Process finished with exit code 0

```

Рисунок 7. Результат пятого примера

## 8. Зафиксировал изменения в репозитории

```

C:\Users\sopov\lb5>git add .
C:\Users\sopov\lb5>git commit -m "Добавление файлов"
[main 0267711] Добавление файлов
5 files changed, 73 insertions(+)
create mode 100644 1.py
create mode 100644 2.py
create mode 100644 3.py
create mode 100644 4.py
create mode 100644 5.py
C:\Users\sopov\lb5>git push
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 1.49 KiB | 762.00 KiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/MaxITS-kurwa/lb5.git
915100b..0267711 main -> main

```

Рисунок 8. Коммит

9. Приступил к выполнению индивидуального задания

```
month = int(input("Enter the month number (1 to 12): "))
```

```
if 1 <= month <= 6:
```

```
    half_year = "first half of the year"
```

```
elif 7 <= month <= 12:
```

```
    half_year = "second half of the year"
```

```
else:
```

```
    print("Invalid month number.")
```

```
    exit()
```

```
if month in [1, 3, 5, 7, 8, 10, 12]:
```

```
    days = 31
```

```
elif month in [4, 6, 9, 11]:
```

```
    days = 30
```

```
elif month == 2:
```

```
    days = 28
```

```
print(f"The month is in the {half_year}, and it has {days} days.")
```

```

Enter the month number (1 to 12): 5
The month is in the first half of the year, and it has 31 days.

Process finished with exit code 0

```

Рисунок 9. Результат задания 1

```
a = float(input("Enter the value of a: "))
```

```

import math

def solve_inequality(a):
    solutions = []

    x_max = a # x <= a
    coeff_a = 1
    coeff_b = -3
    coeff_c = 4 - a

    discriminant = coeff_b**2 - 4 * coeff_a * coeff_c

    if discriminant < 0:
        return "No solutions for the inequality."

    x1 = (-coeff_b - math.sqrt(discriminant)) / (2 * coeff_a)
    x2 = (-coeff_b + math.sqrt(discriminant)) / (2 * coeff_a)

    solutions.append(max(min(x1, x2), -math.inf))
    solutions.append(min(max(x1, x2), x_max))

    return f"x ∈ [{solutions[0]:.2f}, {solutions[1]:.2f}]"

result = solve_inequality(a)
print("Solution:", result)

```

```

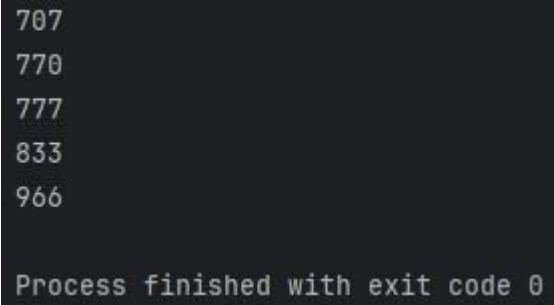
Enter the value of a: 12
Solution: x ∈ [-1.70, 4.70]

Process finished with exit code 0

```

Рисунок 10. Результат задания 2

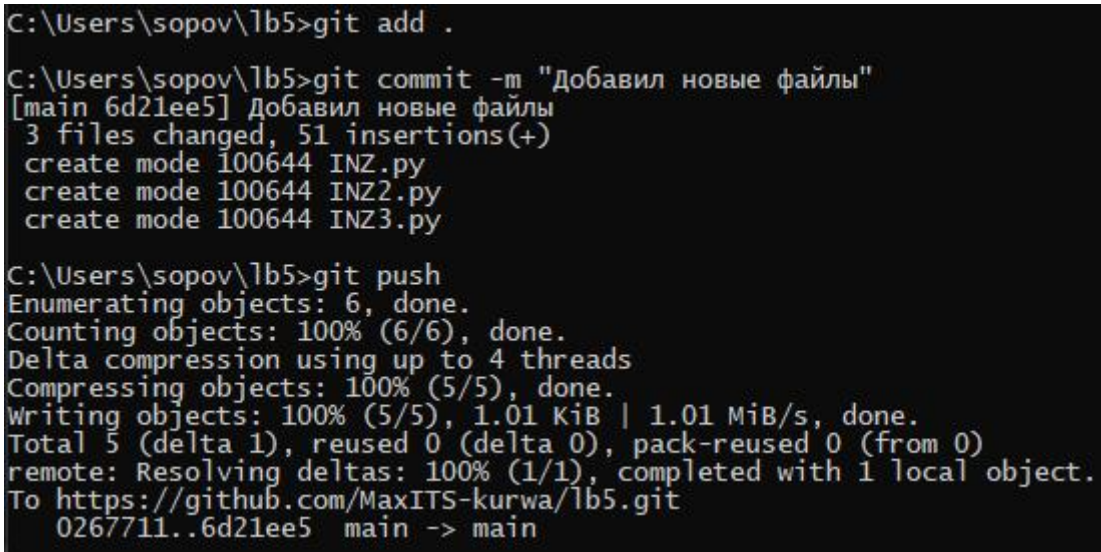
```
for number in range(100, 1000):  
    digits_sum = sum(int(digit) for digit in str(number))  
    if digits_sum % 7 == 0 and number % 7 == 0:  
        print(number)
```



```
707  
770  
777  
833  
966  
  
Process finished with exit code 0
```

Рисунок 11. Результат задания 3

#### 10. Зафиксировал изменения



```
C:\Users\sopov\lb5>git add .  
  
C:\Users\sopov\lb5>git commit -m "Добавил новые файлы"  
[main 6d21ee5] Добавил новые файлы  
3 files changed, 51 insertions(+)  
create mode 100644 INZ.py  
create mode 100644 INZ2.py  
create mode 100644 INZ3.py  
  
C:\Users\sopov\lb5>git push  
Enumerating objects: 6, done.  
Counting objects: 100% (6/6), done.  
Delta compression using up to 4 threads  
Compressing objects: 100% (5/5), done.  
Writing objects: 100% (5/5), 1.01 KiB | 1.01 MiB/s, done.  
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/MaxITS-kurwa/lb5.git  
0267711..6d21ee5 main -> main
```

Рисунок 12. Изменения



## **Ответы на контрольные вопросы:**

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности UML используются для моделирования бизнес-процессов и алгоритмов. Они помогают визуализировать последовательность действий, условия и потоки управления в системе.

2. Что такое состояние действия и состояние деятельности?

Состояние действия — это конкретное действие или операция, выполняемая в процессе. Состояние деятельности — это более общее состояние, которое может включать в себя несколько действий.

3. Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности?

Переходы обозначаются стрелками. Ветвления обозначаются ромбами, где условия перехода указываются на стрелках.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры — это алгоритм, который принимает решение на основе условий и выполняет разные действия в зависимости от результата.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный алгоритм выполняет действия последовательно, без условий, а разветвляющийся алгоритм включает условия, которые определяют, какие действия выполнять.

6. Что такое условный оператор? Какие существуют его формы? Условный оператор позволяет выполнять разные действия в зависимости от истинности условия. Основные формы: if if-else if-elif-else

7. Какие операторы сравнения используются в Python?

= равно, != не равно, >, <, >=, <=.

8. Что называется простым условием?

Простое условие — это условие, состоящее из одного логического выражения.

9. Что такое составное условие?

Составное условие — это условие, состоящее из нескольких логических выражений, объединенных логическими операторами.

10. Какие логические операторы допускаются при составлении сложных условий?

And, or , not

11. Может ли оператор ветвления содержать внутри себя другие ветвления?

Да, оператор ветвления может содержать другие ветвления, создавая вложенные условия.

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры выполняет одно и то же действие несколько раз, пока выполняется определенное условие.

13. Типы циклов в языке Python.

for — цикл, который перебирает элементы последовательности. while — цикл, который выполняется, пока истинно заданное условие.

14. Назовите назначение и способы применения функции range.

Функция range используется для генерации последовательности чисел. Она часто применяется в циклах для итерации по числовым диапазонам.

15. Как с помощью функции range организовать перебор значений от 15 до 0 с шагом 2?

```
for i in range(15, -1, -2)
```

16. Могут ли быть циклы вложенными?

Да, циклы могут быть вложенными, что позволяет создавать более сложные структуры итерации.

17. Как образуется бесконечный цикл и как выйти из него?

Бесконечный цикл образуется, когда условие цикла всегда истинно. Выйти из него можно с помощью оператора break или прерывания программы.

18. Для чего нужен оператор break?

Оператор `break` используется для немедленного выхода из цикла, прекращая его выполнение.

19. Где употребляется оператор `continue` и для чего он используется?

Оператор `continue` используется для пропуска текущей итерации цикла и перехода к следующей.

20. Для чего нужны стандартные потоки `stdout` и `stderr`?

`stdout` используется для вывода информации, `stderr` используется для вывода сообщений об ошибках.

21. Как в Python организовать вывод в стандартный поток `stderr`?

через модуль `sys`

22. Каково назначение функции `exit`?

Функция `exit` используется для завершения программы. Она может принимать код завершения, который указывает на успешное или неуспешное завершение.

**Вывод:** в ходе работы были исследованы условные операторы и циклы в языке Python.