

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

**Отчет по лабораторной работе №4
Условные операторы и циклы в языке Python 3.
по дисциплине «Основы кроссплатформенного программирования»**

Выполнил студент группы ИВТ-б-о-20-1

Токарев В. А. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р. А. _____

(подпись)

Ставрополь 2021

Цель работы: приобретение навыков программирования разветвляющихся алгоритмов и циклических структур. Освоить операторы if, while, for, continue.

Ход работы:

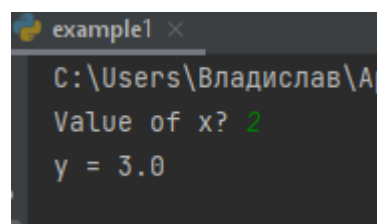
Ссылка на репозиторий: <https://github.com/UnDeR-The-mAsK/lab4.git>

1. Проработал примеры из методички.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import math

if __name__ == '__main__':
    x = float(input("Value of x? "))
    if x <= 0:
        y = 2 * x * x + math.cos(x)
    elif x < 5:
        y = x + 1
    else:
        y = math.sin(x) - x * x
    print(f"y = {y}")
```

Рисунок 1. Код первого примера



```
example1 x
C:\Users\Владислав\AppData\Local\Programs\Python\Python39\python.exe
Value of x? 2
y = 3.0
```

Рисунок 2. Результат выполнения программы

```

▶ #!/usr/bin/env python3
# -*- coding: utf-8 -*-

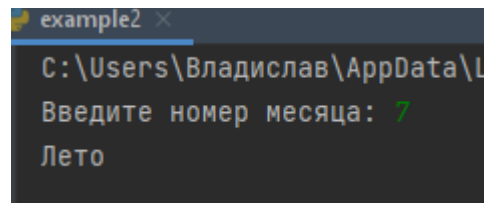
import sys

▶ 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)

```

Рисунок 3. Код второго примера



```

example2 x
C:\Users\Владислав\AppData\L
Введите номер месяца: 7
Лето

```

Рисунок 4. Результат выполнения программы

```

▶ #!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math

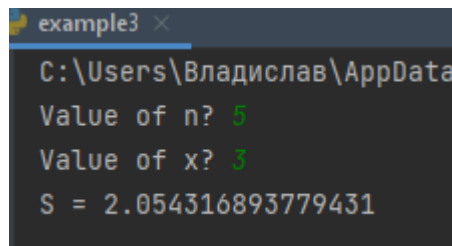
▶ if __name__ == '__main__':
    n = int(input("Value of n? "))
    x = float(input("Value of x? "))

    S = 0.0
    for k in range(1, n + 1):
        a = math.log(k * x) / (k * k)
        S += a

    print(f"S = {S}")

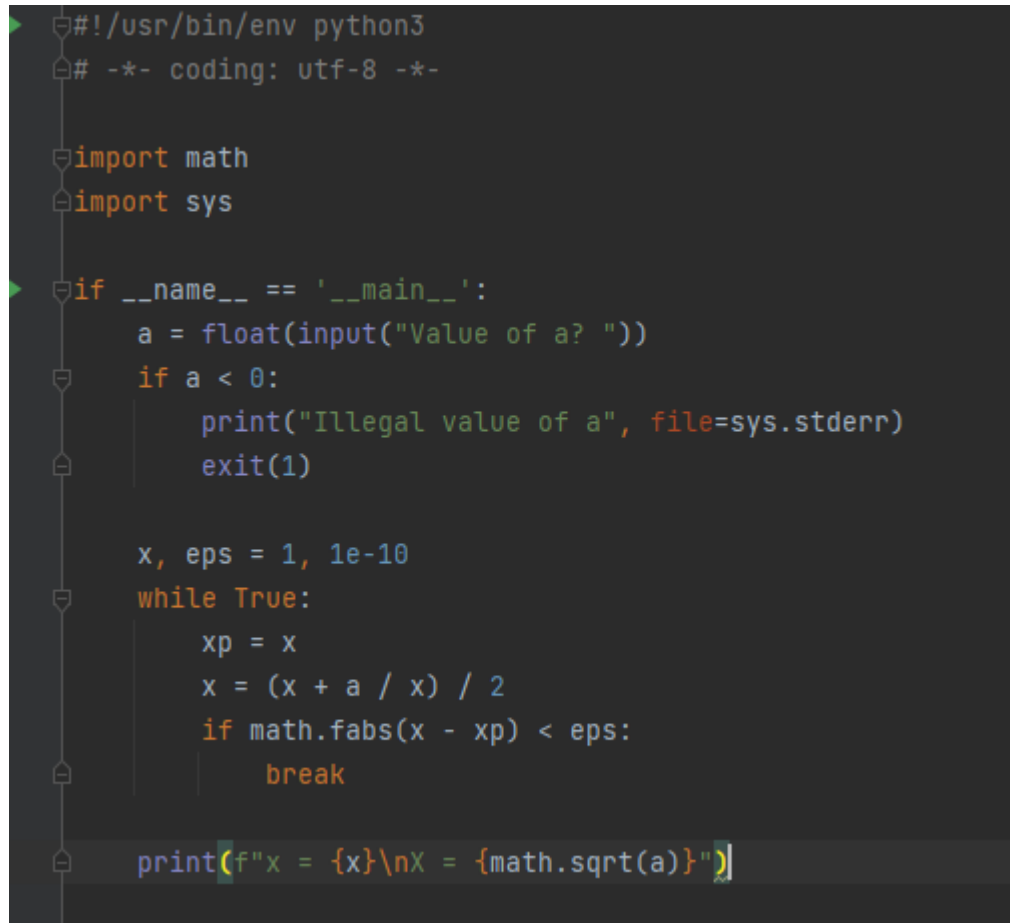
```

Рисунок 5. Код третьего примера



```
example3 x
C:\Users\Владислав\AppData
Value of n? 5
Value of x? 3
S = 2.054316893779431
```

Рисунок 6. Работа программы третьего примера



```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

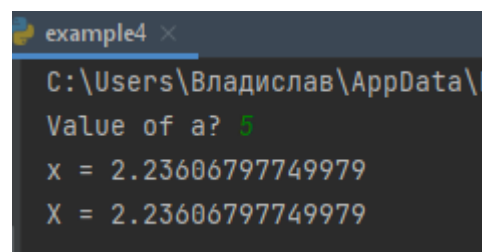
import math
import sys

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)}")
```

Рисунок 7. Код четвертого примера



```
example4 x
C:\Users\Владислав\AppData\I
Value of a? 5
x = 2.23606797749979
X = 2.23606797749979
```

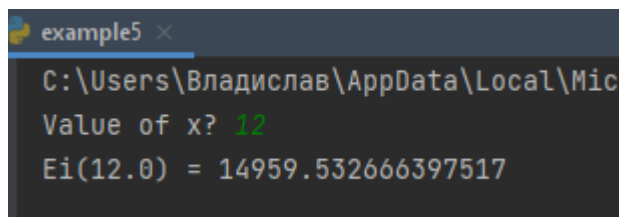
Рисунок 8. Работа кода четвертого примера

```

1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5 import sys
6
7 # Постоянная Эйлера.
8 EULER = 0.5772156649015328606
9 # Точность вычислений.
10 EPS = 1e-10
11
12 ▶ if __name__ == '__main__':
13     x = float(input("Value of x? "))
14     if x == 0:
15         print("Illegal value of x", file=sys.stderr)
16         exit(1)
17
18     a = x
19     S, k = a, 1
20
21     # Найти сумму членов ряда.
22     while math.fabs(a) > EPS:
23         a *= x * k / (k + 1) ** 2
24         S += a
25         k += 1
26
27     # Вывести значение функции.
28     print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")

```

Рисунок 9. Код пятого примера



```

example5 ×
C:\Users\Владислав\AppData\Local\Mic
Value of x? 12
Ei(12.0) = 14959.532666397517

```

Рисунок 10. Работоспособность кода пятого примера

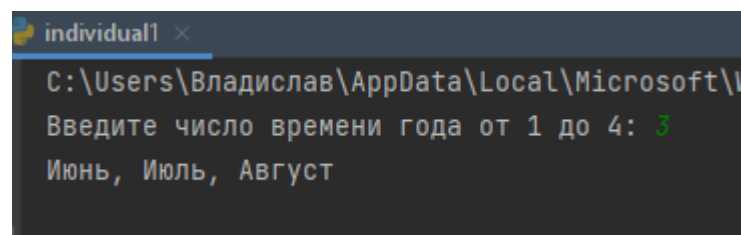
Индивидуальное задание № 1.

Вариант 5

Условие: Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```
1 ▶ 1 #!/usr/bin/env python3
2   2 # -*- coding: utf-8 -*-
3
4   3 import sys
5
6
7 ▶ 4 if __name__ == "__main__":
8     m = int(input("Введите число времени года от 1 до 4: "))
9
10    if m == 1:
11        print("Декабрь, Январь, Февраль")
12    elif m == 2:
13        print("Март, Апрель, Май")
14    elif m == 3:
15        print("Июнь, Июль, Август")
16    elif m == 4:
17        print("Сентябрь, Октябрь, Ноябрь")
18    else:
19        print("Неверное число", file=sys.stderr)
20    exit(1)
```

Рис. Индивидуальное задание 1



```
individual1 x
C:\Users\Владислав\AppData\Local\Microsoft\W
Введите число времени года от 1 до 4: 3
Июнь, Июль, Август
```

Рис. Результат инд. задания 1

Индивидуальное задание № 2.

Вариант 17

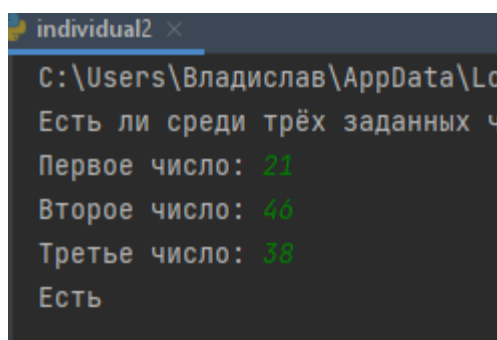
Условие: Определить, есть ли среди трёх заданных чисел нечётные.

```

1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5
6 ▶ if __name__ == "__main__":
7     print("Есть ли среди трёх заданных чисел нечётные?")
8     a1 = int(input("Первое число: "))
9     a2 = int(input("Второе число: "))
0     a3 = int(input("Третье число: "))
1
2     if a1 % 2 == 0 or a2 % 2 == 0 or a3 % 2 == 0:
3         print("Есть")
4     else:
5         print("Нет")
6     exit(1)
7

```

Рис. Индивидуальное задание 2



```

individual2 x
C:\Users\Владислав\AppData\Lo
Есть ли среди трёх заданных ч
Первое число: 21
Второе число: 46
Третье число: 38
Есть

```

Рис. Результат инд. задания 2

Индивидуальное задание № 3.

Вариант 16

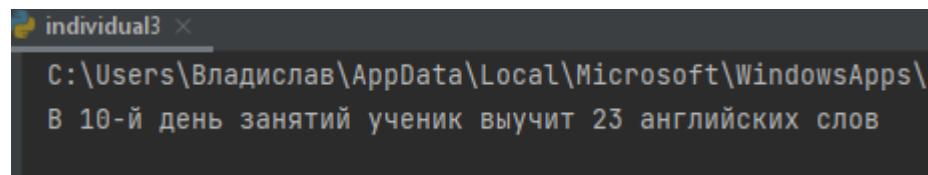
Условие: Ученик выучил в первый день 5 английских слов. В каждый следующий день он выучивал на 2 слова больше, чем в предыдущий. Сколько английских слов выучит ученик в 10-ый день занятий

```

1 ▶ ①#!/usr/bin/env python3
2 ①# -*- coding: utf-8 -*-
3
4 ①import math
5 ①import sys
6
7
8  a = 5
9 ▶ ①if __name__ == "__main__":
10     for i in range(9):
11         a = a + 2
12     print("В 10-й день занятий ученик выучит", a, "английских слов")
13 ①exit(1)
14

```

Рис. Индивидуальное задание 3



```

individual3 x
C:\Users\Владислав\AppData\Local\Microsoft\WindowsApps\p
В 10-й день занятий ученик выучит 23 английских слов

```

Рис. Результат инд. задания 3

Контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграмма UML — наглядное представление совокупности элементов модели в виде графа. Диаграммы UML применяются для визуализации разных

аспектов устройства или поведения моделируемой системы. UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Web-приложений.

2. Что такое состояние действия и состояние деятельности?

Состояние действия - операции, проводимые над объектами или различные вычисления.

В противоположность этому состояния деятельности могут быть подвергнуты дальнейшей декомпозиции, вследствие чего выполняемую деятельность можно представить с помощью других диаграмм деятельности. Состояния деятельности не являются атомарными, то есть могут быть прерваны. Предполагается, что для их завершения требуется заметное время. Можно считать, что состояние действия - это частный вид состояния деятельности, а конкретнее – такое состояние, которое не может быть

подвергнуто дальнейшей декомпозиции. А состояние деятельности можно представлять себе, как составное состояние, поток управления которого включает только другие состояния деятельности и действий.

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

Когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока используются переходы, показывающие путь из одного состояния действия или деятельности в другое.

В UML переход представляется простой линией со стрелкой

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм разветвляющейся структуры - это алгоритм, в котором вычислительный процесс осуществляется либо по одной, либо по другой ветви, в зависимости от выполнения некоторого условия.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный идет лишь в одном направлении, а в разветвляющемся возможны разные исходы.

6. Что такое условный оператор? Какие существуют его формы?
Условные операторы: `if` и `while`. Условный оператор позволяет выполнять действия в зависимости от логического значения условия.

7. Какие операторы сравнения используются в Python?
Операторы сравнения: `>` (больше), `<` (меньше), `>=` (больше или равно), `<=` (меньше или равно), `==` (равно), `!=` (не равно).

8. Что называется простым условием? Приведите примеры.
Простое условие - два выражения, связанные одним из знаков отношений. `If a>4: print('Hello')`

9. Что такое составное условие? Приведите примеры.

Составные условия - условия, состоящие из двух или более простых условий, соединенных с помощью логических операций `and` (и), `or` (или), `not` (не).

10. Какие логические операторы допускаются при составлении сложных условий?

В сложных условиях можно использовать операторы and (и), or (или), not (не).

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

Да операторы ветвления могут иметь внутри другие ветвления в виде множественного ветвления

12. Какой алгоритм является алгоритмом циклической структуры?

Оператор for выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе.

13. Типы циклов в языке Python.

Существует 2 вида циклов: while- Оператор цикла while выполняет

указанный набор инструкций до тех пор, пока условие цикла истинно. For- Оператор for выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе.

14. Назовите назначение и способы применения функции range.

Функция range возвращает неизменяемую последовательность чисел в виде объекта range. Функцию range можно использовать для генерации последовательности чисел.

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

```
list(range(15, 0, 2))
```

16. Могут ли быть циклы вложенными?

Python позволяет вкладывать циклы друг друга. Вложенный цикл – это цикл, который встречается внутри другого цикла.

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

Бесконечный цикл while — это цикл, в котором условие никогда не становится ложным. Это значит, что тело исполняется снова и снова, а цикл никогда не заканчивается. Прервать его можно с помощью break и continue

18. Для чего нужен оператор break.

Оператор break предназначен для досрочного прерывания работы цикла while.

19. Для чего нужен оператор continue?

Оператор continue запускает цикл заново, при этом код, расположенный после данного оператора, не выполняется.

20. Для чего нужны стандартные потоки stdout и stderr?

В операционной системе по умолчанию присутствуют стандартных потока вывода на консоль: буферизованный поток stdout для вывода данных и информационных сообщений, а также небуферизованный поток stderr для вывода сообщений об ошибках. По умолчанию функция print использует поток stdout. Для того, чтобы использовать поток stderr необходимо передать его в параметре file функции print. Само же определение потоков stdout и stderr находится в стандартном пакете Python sys. Хорошим стилем программирования является наличие вывода ошибок в стандартный поток stderr поскольку вывод в потоки stdout и stderr может обрабатываться как операционной системой, так и сценариями пользователя по-разному.

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

Для того, чтобы использовать поток stderr необходимо передать его в параметре file функции print.

22. Каково назначение функции exit ?

В Python завершить программу и передать операционной системе заданный код возврата можно посредством функции exit.

Вывод: в процессе выполнения лабораторной работы приобрел навыки программирования разветвляющихся алгоритмов и циклических структур. Освоил операторы if, while, for, continue.