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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3
дисциплины «Программирование на Python»
Вариант 2

Выполнил:
Безруков Даниил Андреевич
2 курс, группа ИВТ-б-о-24-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2026 г.

Тема: Условные операторы и циклы в языке Python

Цель: приобретение навыков программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоить операторы языка Python версии 3.x if, while, for, break u continue, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

Порядок выполнения работы:

Ссылка на репозиторий: https://github.com/pokachtononame/python_3

1. Создаём публичный репозиторий
2. Клонируем репозиторий на компьютер
3. Создаём проект PyCharm и вносим изменения в файл .gitignore

```
#  option (not reco
.idea/
.venv/
```

Рисунок 1. Файл .gitignore

4. Построим UML-диаграмму деятельности для примера 4

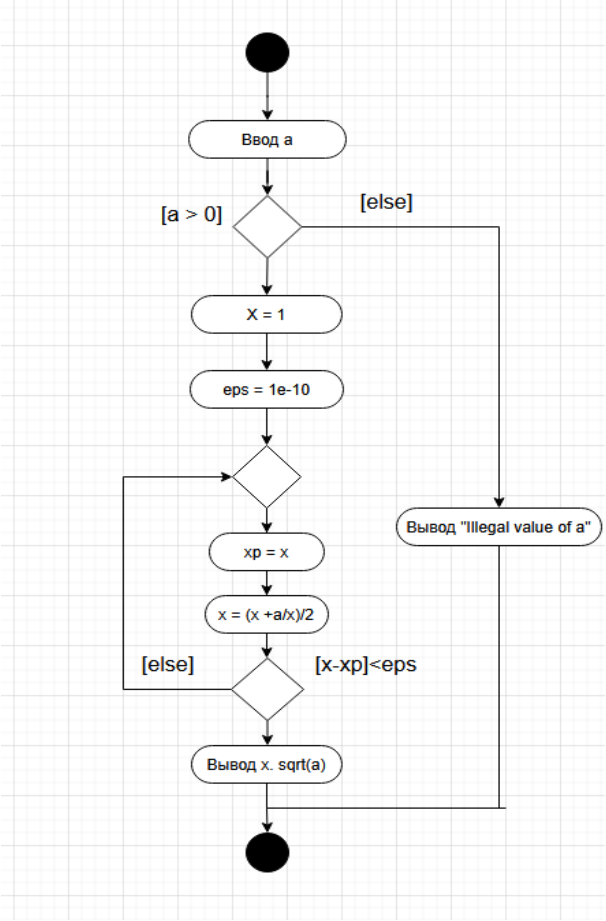

Рисунок 2. Диаграмма деятельности

```python
#!/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 3. Пример 4

5. Построим UML-диаграмму деятельности для примера 5

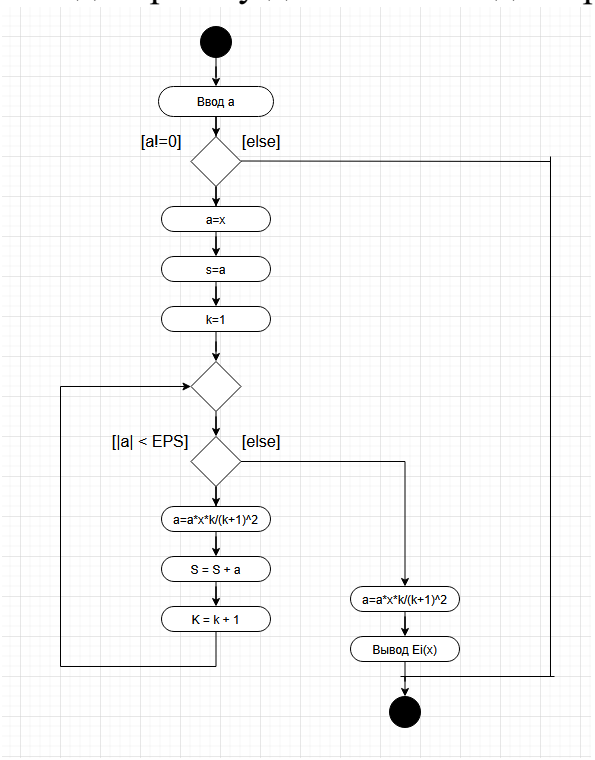

Рисунок 4. Диаграмма деятельности

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

import math
import sys

# Постоянная Эйлера.
EULER = 0.5772156649015328606
# Точность вычислений.
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

    # Найти сумму членов ряда.
    while math.fabs(a) > EPS:
        a *= x * k / (k + 1) ** 2
        S += a
        k += 1

    # Вывести значение функции.
    print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")

```

Рисунок 5. Пример 5

6. Выполняем индивидуальное задание 1 (вариант 2)

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

import sys

if __name__ == '__main__':
    # Ввод номера месяца
    month = int(input("Введите номер месяца (1-12): "))

    # Проверка и вывод количества дней в месяце
    if month in (1, 3, 5, 7, 8, 10, 12):
        print("В этом месяце 31 день")
    elif month in (4, 6, 9, 11):
        print("В этом месяце 30 дней")
    elif month == 2:
        print("В этом месяце 28 дней")
    else:
        print("Ошибка: номер должен быть от 1 до 12", file=sys.stderr)
        sys.exit(1)

```

Рисунок 6. Индивидуальное задание

```
D:\projects\python\py_lab_3\.venv\Scripts\python.exe D:\proje
Введите номер месяца (1-12): 5
В этом месяце 31 день

Process finished with exit code 0
```

Рисунок 7. Результат выполнения

7. Выполняем индивидуальное задание 2

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

if __name__ == '__main__':
    x = float(input("Введите x: "))
    y = float(input("Введите y: "))

    # Вычисление значения max^2(x^2 * y, x * y^2)
    if x ** 2 * y > x * y ** 2:
        maximum = (x ** 2 * y) ** 2
    else:
        maximum = (x * y ** 2) ** 2

    # Вычисление значения min^2(x - y, x + 2 * y)
    if x - y > x + 2 * y:
        minimum = (x + 2 * y) ** 2
    else:
        minimum = (x - y) ** 2

    u = maximum + minimum

    print(f'U = {u:.2f}')
```

Рисунок 8. Код индивидуального задания

```
D:\projects\python\python_3\.venv\Scripts\python.exe
Введите x: 3
Введите y: 4
U = 2305.00
```

Рисунок 9. Результат индивидуального задания

8. Выполняем индивидуальное задание 3

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

if __name__ == '__main__':

    # Нахождение суммы целых положительных чисел > 20 и < 100, кратных 3
    total = 0
    number = 21
    while number < 100:
        if number % 3 == 0:
            total += number
        number += 1
    print(f"Сумма чисел больше 20 и меньше 100, кратных 3, равна: {total}")
```

Рисунок 10. Индивидуальное задание

```
D:\projects\python\python_3\.venv\Scripts\python.exe D:\pro
Сумма чисел больше 20 и меньше 100, кратных 3, равна: 1620

Process finished with exit code 0
```

Рисунок 11. Результат выполнения задания

9. Составим UML-диаграммы деятельности для выполненных заданий

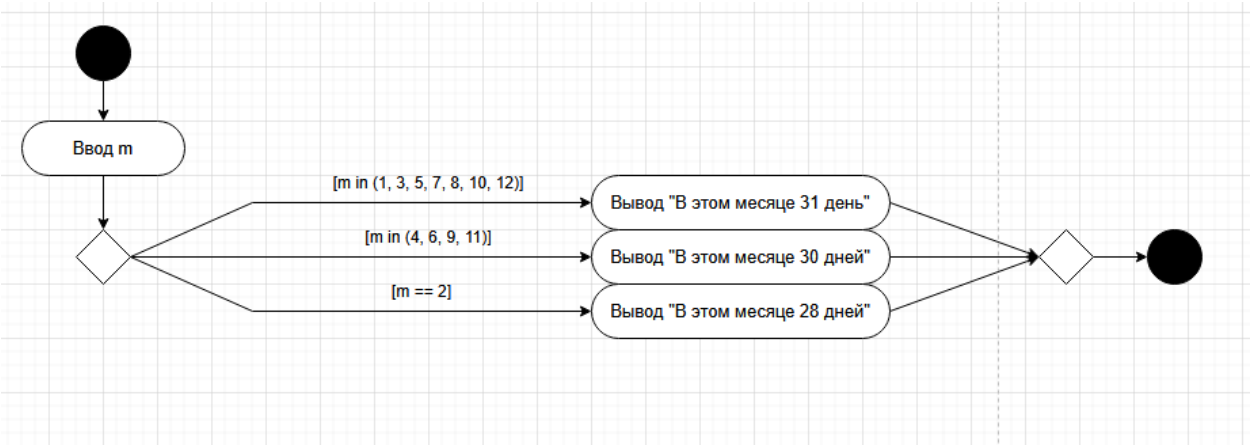

Рисунок 12. UML-диаграмма для кода задания 1

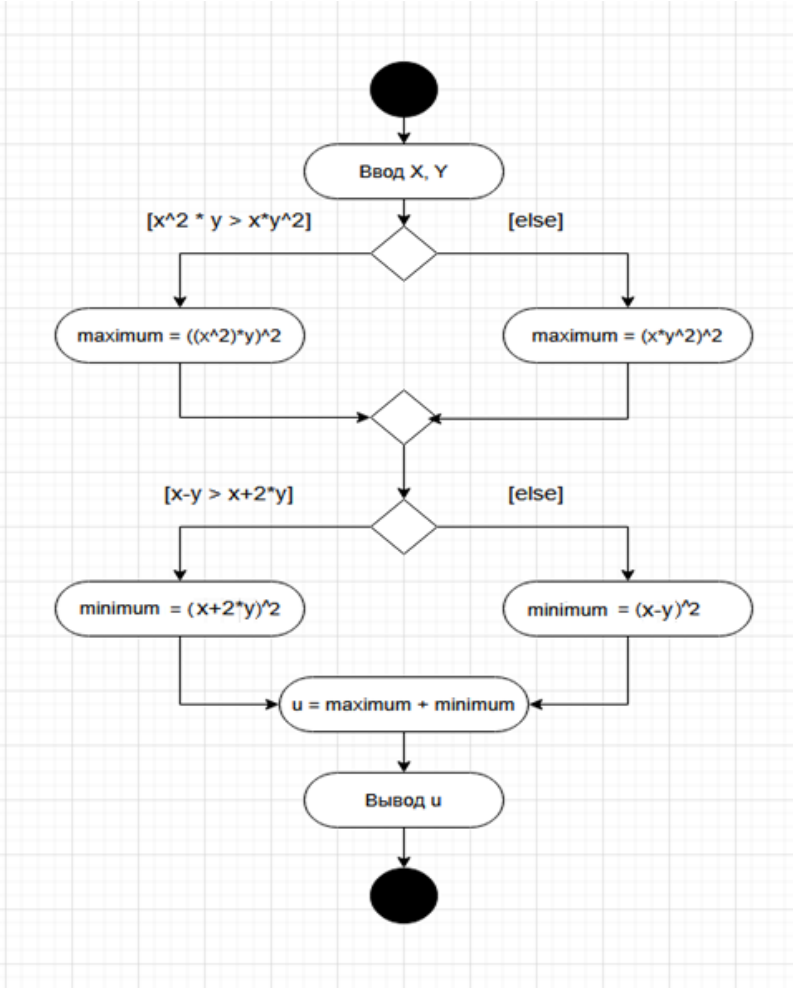

Рисунок 13. UML-диаграмма для задания 2

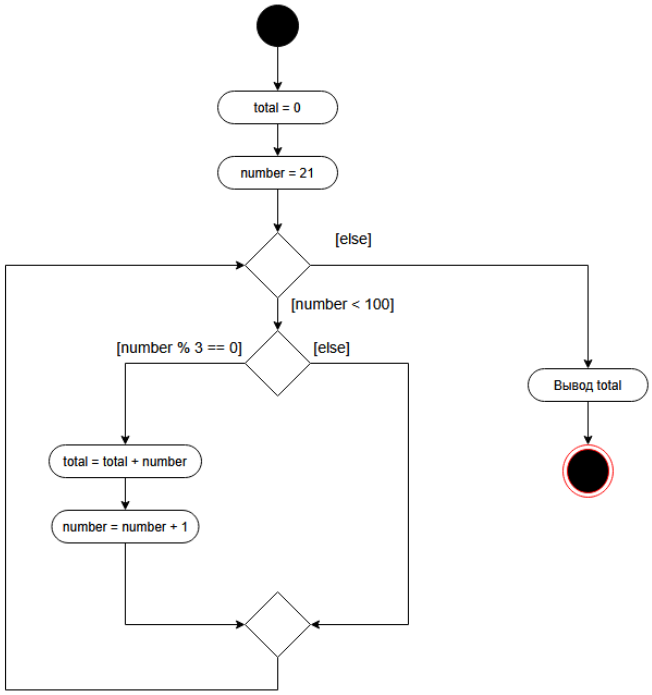

Рисунок 14. UML-диаграмма задания 3

10. Выполняем задание повышенной сложности и строим для него UML-диаграмму деятельности

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

import math

EPS = 1e-10

if __name__ == '__main__':
    x = float(input("Value of x: "))
    # Начальное значение при n=1
    # A1 = ((-1)^1 * x^2) / (2*1 * (2*1)!) = -x^2 / 4
    n = 1
    a = -(x ** 2) / 4
    total_sum = a
    while math.fabs(a) > EPS:
        # an+1 = an * (-x^2 * n) / (2 * (n+1)^2 * (2n + 1))
        multiplier = -(x ** 2 * n) / (2 * (n + 1) ** 2 * (2 * n + 1))
        a *= multiplier
        total_sum += a
        n += 1

    print("Result: ", total_sum)
```

Рисунок 15. Код задания повышенной сложности

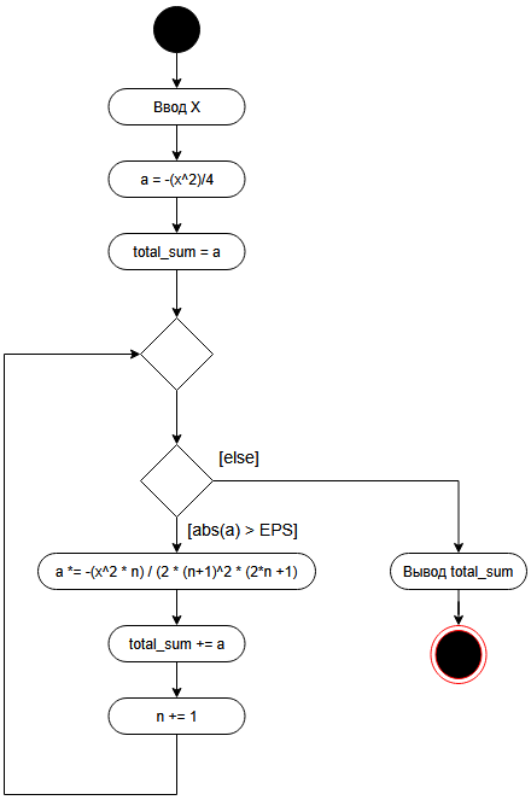

Рисунок 16. UML-диаграмма для задания 4

Ответы на контрольные вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности UML используются для моделирования бизнес-процессов и рабочих процессов, визуализации последовательности действий, решений и потоков управления в системе.

2. Что такое состояние действия и состояние деятельности?

- Состояние действия - атомарная операция, которая не может быть прервана (например, простое вычисление)
- Состояние деятельности - составная операция, которая может быть декомпозирована и прервана

3. Какие нотации существуют для обозначения переходов и ветвлений?

- Переходы - простые стрелки между действиями
- Ветвления - ромбы с условиями [условие]

- Начало/Конец - закрашенный круг и круг с границей

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Алгоритм, в котором вычислительный процесс осуществляется по разным ветвям в зависимости от выполнения условий (if-elif-else).

5. Чем отличается разветвляющийся алгоритм от линейного?

- Линейный - последовательное выполнение операций
- Разветвляющийся - выполнение разных операций в зависимости от условий.

6. Что такое условный оператор? Какие существуют его формы?

Формы оператора if:

if условие:

if условие: ... else:

if условие: ... elif: ... else:

7. Какие операторы сравнения используются в Python?

==, !=, <, >, <=, >=, is, is not, in, not in

8. Что называется простым условием?

Условие с одним логическим выражением:

возраст >= 18

9. Что такое составное условие?

Условие с несколькими выражениями, соединенными логическими операторами:

возраст >= 18 and возраст <= 65

10. Какие логические операторы допускаются?

and, or , not

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

Да, операторы if могут содержать другие операторы if внутри (вложенные условия).

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм, в котором многократно повторяется выполнение одной и той же последовательности операций.

13. Типы циклов в языке Python:

- while- выполняется пока условие истинно
- for-выполняется для каждого элемента последовательности

14. Назначение и способы применения функции range:

Функция range генерирует последовательности чисел для использования в циклах:

```
range(stop)
range(start, stop)
range(start, stop, step)
```

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

```
range(15, -1, -2)  # 15, 13, 11, 9, 7, 5, 3, 1
```

16. Могут ли быть циклы вложенными?

Да, циклы могут быть вложенными (один цикл внутри другого).

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

```
while True:  # Бесконечный цикл
    if условие:
        break  # Выход из цикла
```

18. Для чего нужен оператор break?

Для досрочного прерывания выполнения цикла.

19. Где употребляется оператор continue?

Используется в циклах для перехода к следующей итерации, пропуская оставшийся код текущей итерации.

20. Для чего нужны стандартные потоки stdout и stderr?

- stdout - для вывода обычных данных
- stderr - для вывода сообщений об ошибках

21. Как организовать вывод в stderr?

```
import sys
```

```
print("Ошибка!", file=sys.stderr)
```

22. Назначение функции exit?

Завершение выполнения программы с кодом возврата:

```
exit(0)  # Успешное завершение
exit(1)  # Завершение с ошибкой
```

Вывод: в ходе выполнения работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоены операторы языка Python версии 3.x if, while, for, break continue, позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.