

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 ЛАБОРАТОРНОЙ РАБОТЕ №5
дисциплины «Основы программной инженерии»

Выполнил:
Мелтонян Одиссей
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил: Воронкин Р. А.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

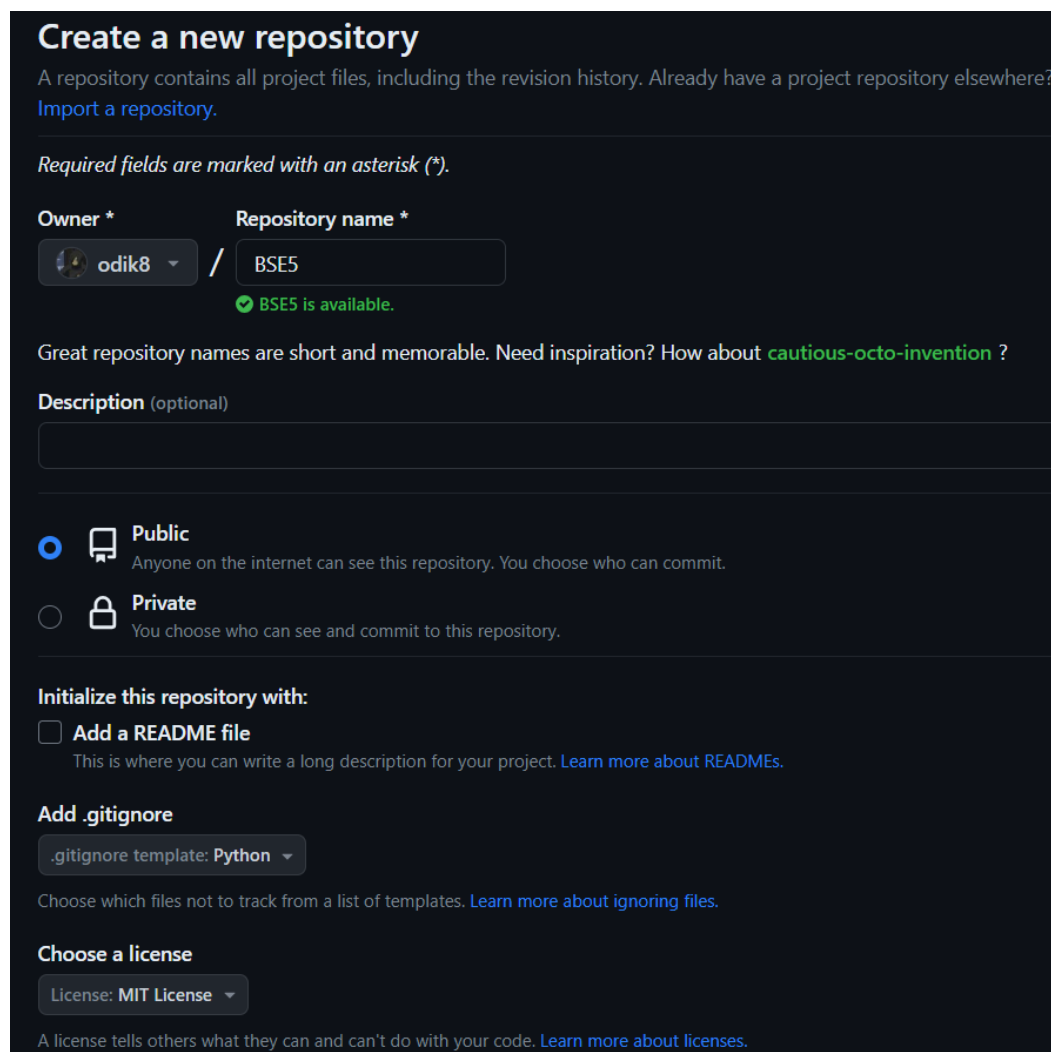
Ставрополь, 2023 г

Тема: Условные операторы и циклы в языке Python

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

Ход работы:

1. Изучил теоретический материал
2. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.




Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*


Owner * / Repository name *


 odik8 / BSE5

✔ BSE5 is available.

Great repository names are short and memorable. Need inspiration? How about [cautious-octo-invention](#) ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **Python**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

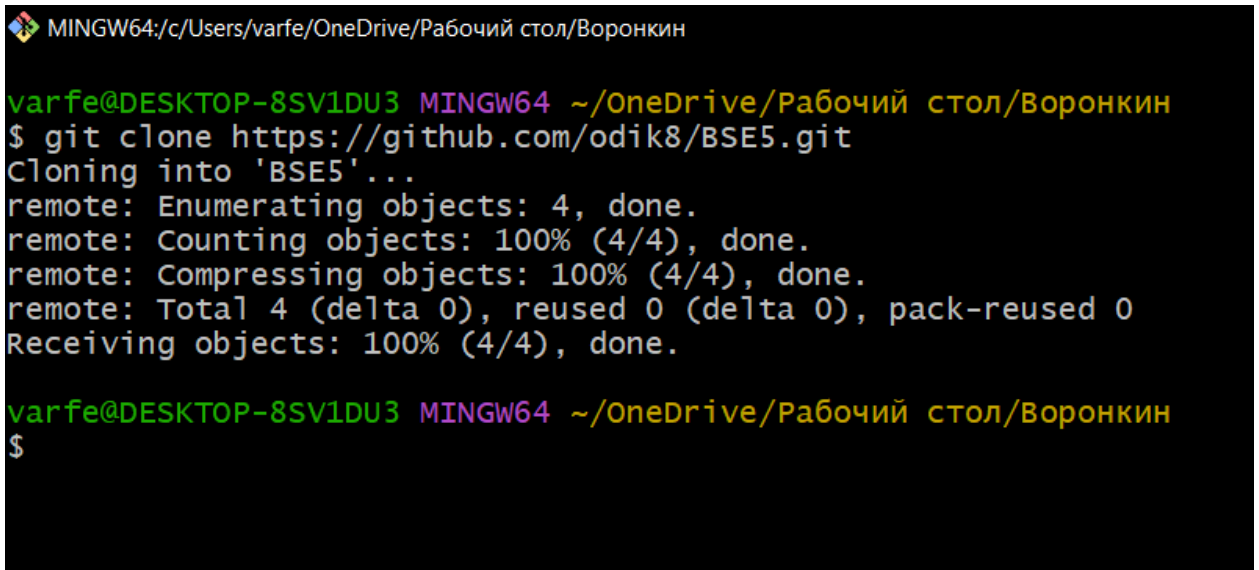
Choose a license

License: **MIT License**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

Рисунок 1. – Создание репозитория

3. Клонировал репозиторий

A terminal window with a black background and white text. The title bar shows the path 'MINGW64:/c/Users/varfe/OneDrive/Рабочий стол/Воронкин'. The user 'varfe@DESKTOP-8SV1DU3' is in the 'MINGW64' environment. The command 'git clone https://github.com/odik8/BSE5.git' is entered. The output shows the cloning process: 'Cloning into 'BSE5'...', 'remote: Enumerating objects: 4, done.', 'remote: Counting objects: 100% (4/4), done.', 'remote: Compressing objects: 100% (4/4), done.', 'remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0', and 'Receiving objects: 100% (4/4), done.'.

```
MINGW64:/c/Users/varfe/OneDrive/Рабочий стол/Воронкин
varfe@DESKTOP-8SV1DU3 MINGW64 ~/OneDrive/Рабочий стол/Воронкин
$ git clone https://github.com/odik8/BSE5.git
Cloning into 'BSE5'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
varfe@DESKTOP-8SV1DU3 MINGW64 ~/OneDrive/Рабочий стол/Воронкин
$
```

Рисунок 2. – Клонирование репозитория

4. Дополнил файл .gitignore

A screenshot of a text editor window titled '.gitignore - Блокнот'. The editor shows the content of a .gitignore file. The file lists various directories and files to be ignored, including environment files, IDE settings, documentation, and caches. The status bar at the bottom indicates 'Стр 1, стр 1', '100%', 'Windows (CRLF)', and 'UTF-8'.

```
.gitignore - Блокнот
Файл Правка Формат Вид Справка
*.sage.py

# Environments
.env
.venv
env/
venv/
ENV/
env.bak/
venv.bak/

# Spyder project settings
.spyderproject
.spyproject

# Rope project settings
.ropeproject

# makedocs documentation
/site

# mypy
.mypy_cache/
.dmypy.json
dmypy.json

# Pyre type checker
.pyre/

# pytype static type analyzer
.pytype/

# Cython debug symbols
cython_debug/

# PyCharm
# JetBrains specific template is maintained in a separate JetBrains.gitignore that can
# be found at https://github.com/github/gitignore/blob/main/Global/JetBrains.gitignore
# and can be added to the global gitignore or merged into this file. For a more nuclear
# option (not recommended) you can uncomment the following to ignore the entire idea folder.
.idea/
```

Рисунок 3. – Файл .gitignore

5. Организовал свой репозиторий в соответствии с моделью ветвления git-flow. Для этого создал ветку develop в которую будут сливаться ветки features.

```
varfe@DESKTOP-8SV1DU3 MINGW64 ~/OneDrive/Рабочий стол/Воронкин/bse5 (main)
$ git branch develop
```

Рисунок 4 – Создание ветки develop

6. Изучил рекомендации к оформлению исходного кода на языке Python PEP-8. Выполнил оформление исходного примеров лабораторной работы и индивидуальных созданий в соответствии с PEP-8.

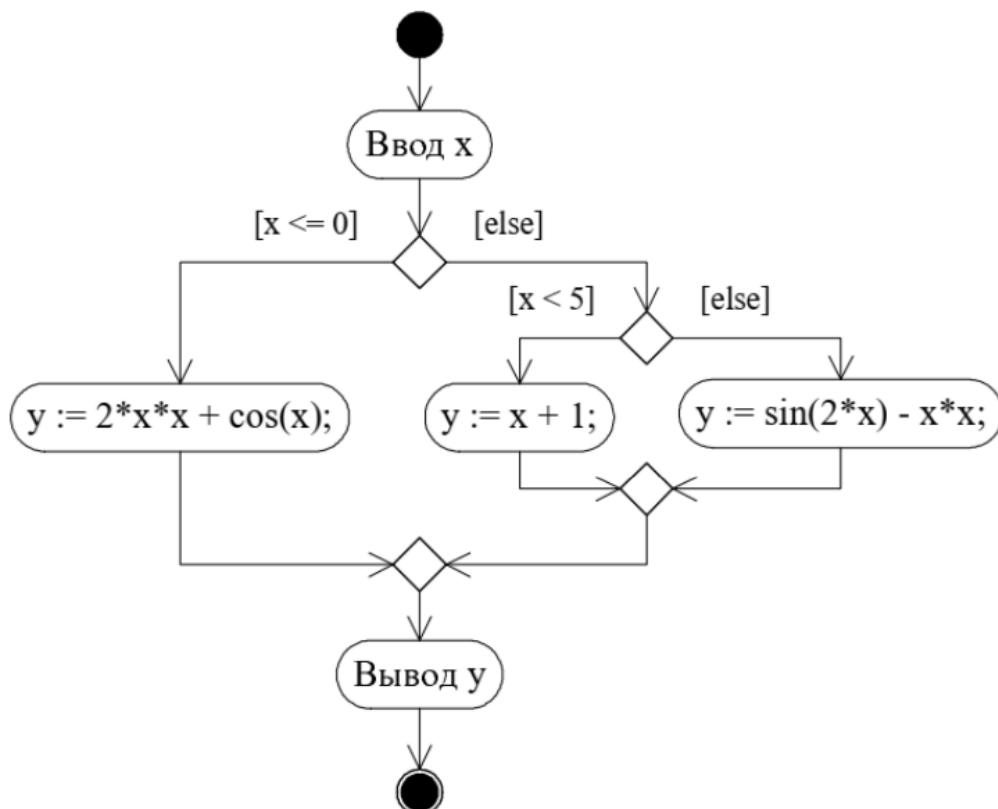
7. Создал проект PyCharm в папке репозитория

8. Проработал примеры лабораторной работы.

Пример 1. Составить UML-диаграмму деятельности и программу с использованием конструкции ветвления и вычислить значение функции

$$y = \begin{cases} 2x^2 + \cos x, & x \leq 3.5, \\ x + 1, & 0 < x < 5, \\ \sin 2x - x^2, & x \geq 5. \end{cases}$$

UML-диаграмма:



```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4
5  if __name__ == '__main__':
6      x = float(input("Value of x? "))
7      if x <= 0:
8          y = 2 * x * x + math.cos(x)
9      elif x < 5:
10         y = x + 1
11     else:
12         y = math.sin(x) - x * x
13     print(f"y = {y}")

```

Рисунок 5 – Код примера

```

C:\Users\varfe\AppData\Local\Programs\Python\Python39-32\python.exe
Value of x? -1
y = 2.5403023058681398

```

Рисунок 6 – Результат выполнения программы

```

C:\Users\varfe\AppData\Local\Programs\Python\Python39-32\python.exe
Value of x? 3
y = 4.0

```

Рисунок 7 – Результат выполнения программы

```

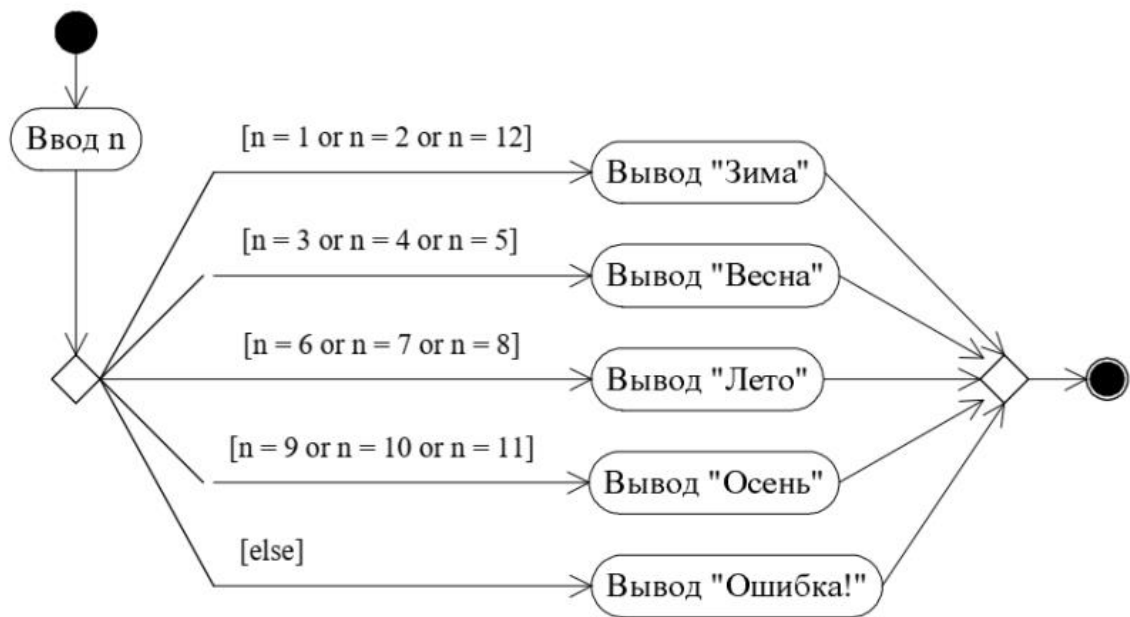
C:\Users\varfe\AppData\Local\Programs\Python\Python39-32\python.exe
Value of x? 7
y = -48.34301340128121

```

Рисунок 8 – Результат выполнения программы

Пример 2. Составить UML-диаграмму деятельности и программу для решения задачи: с клавиатуры вводится номер месяца от 1 до 12, необходимо для этого номера месяца вывести наименование времени года.

UML-диаграмма:



9. Для примеров 4 и 5 построил UML-диаграммы деятельности.

Uml-диаграмма для примера 4:

```

1      #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3      import sys
4      if __name__ == '__main__':
5          n = int(input("Введите номер месяца: "))
6          if n == 1 or n == 2 or n == 12:
7              print("Зима")
8          elif n == 3 or n == 4 or n == 5:
9              print("Весна")
10         elif n == 6 or n == 7 or n == 8:
11             print("Лето")
12         elif n == 9 or n == 10 or n == 11:
13             print("Осень")
14         else:
15             print("Ошибка!", file=sys.stderr)
16         exit(1)
  
```

Рисунок 10 – Код второго примера

```
C:\Users\varfe\AppData\Local\Pr
Введите номер месяца: 12
Зима
```

Рисунок 11 – Результат выполнения программы

```
C:\Users\varfe\AppData\Local\Progr
Введите номер месяца: 4
Весна
```

Рисунок 12 – Результат выполнения программы

```
C:\Users\varfe\AppData\Local\Pr
Введите номер месяца: 8
Лето
```

Рисунок 13 – Результат выполнения программы

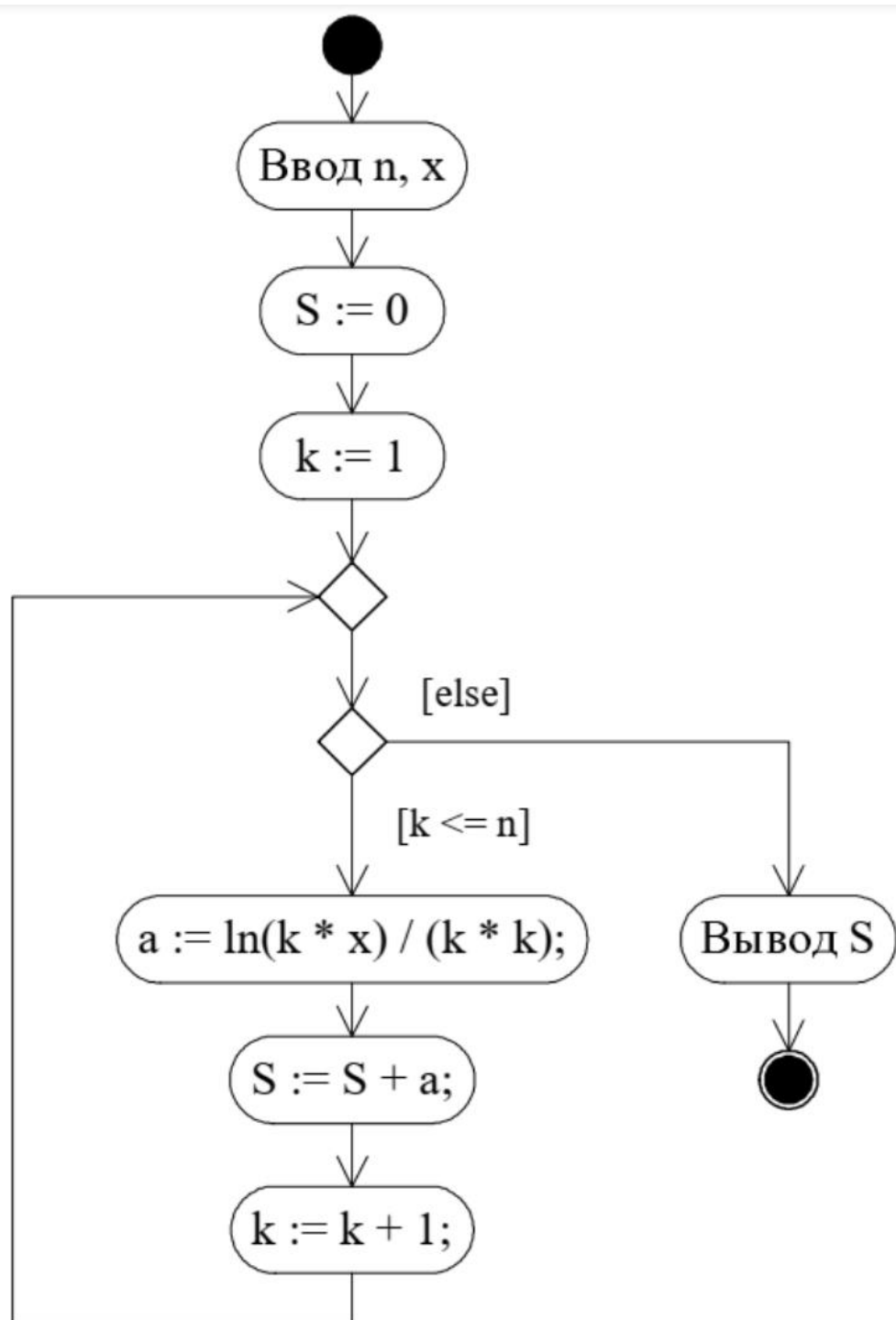
```
C:\Users\varfe\AppData\Local\Pr
Введите номер месяца: 13
Ошибка!
```

Рисунок 14 – Результат выполнения программы

Пример 3. Составить UML-диаграмму деятельности и написать программу, позволяющую вычислить конечную сумму:

$$S = \sum_{k=1}^n \frac{\ln kx}{k^2},$$

UML-диаграмма:




```

1      #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3      import math
4
5      if __name__ == '__main__':
6          n = int(input("Value of n? "))
7          x = float(input("Value of x? "))
8
9          S = 0.0
10
11         for k in range(1, n + 1):
12             a = math.log(k * x) / (k * k)
13             S += a
14         print(f"S = {S}")
15

```

Рисунок 15 – Код третьего примера

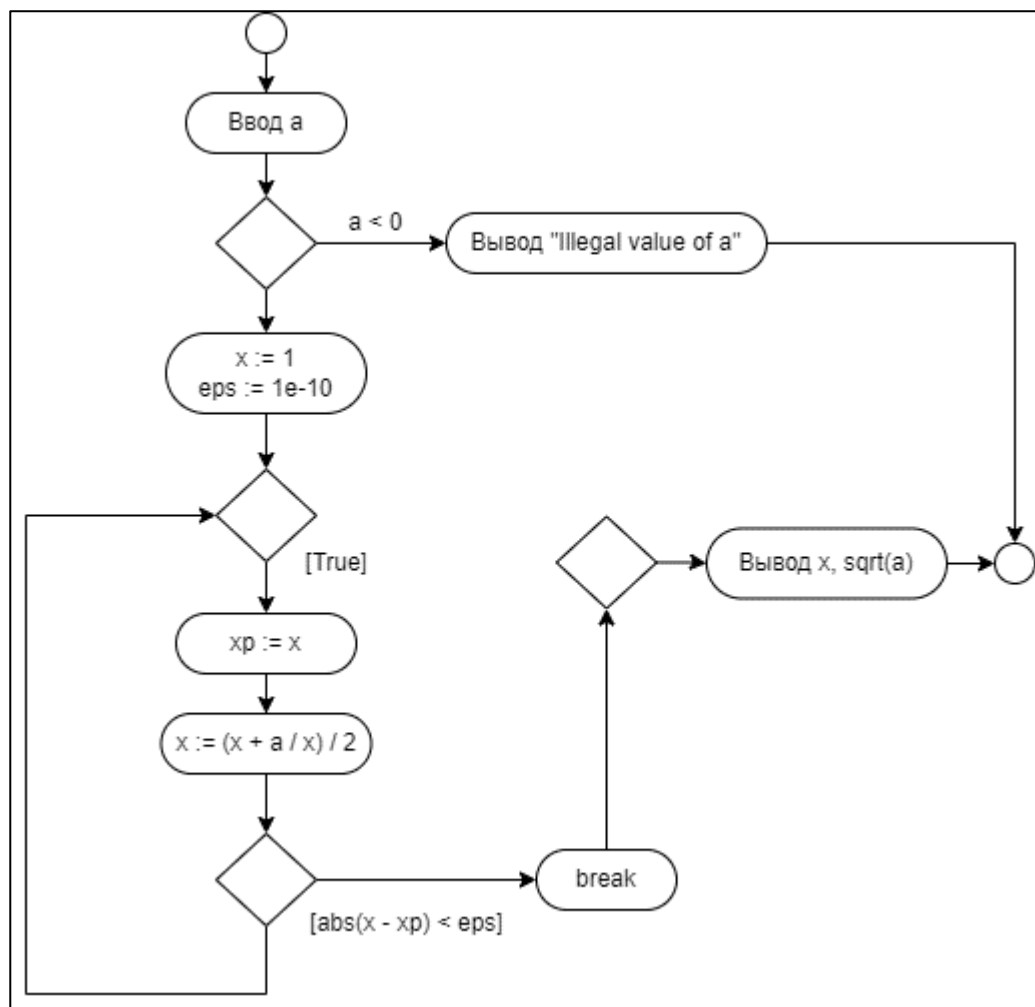
```

C:\Users\varfe\AppData\Lo
Value of n? 1
Value of x? 3
S = 1.0986122886681098

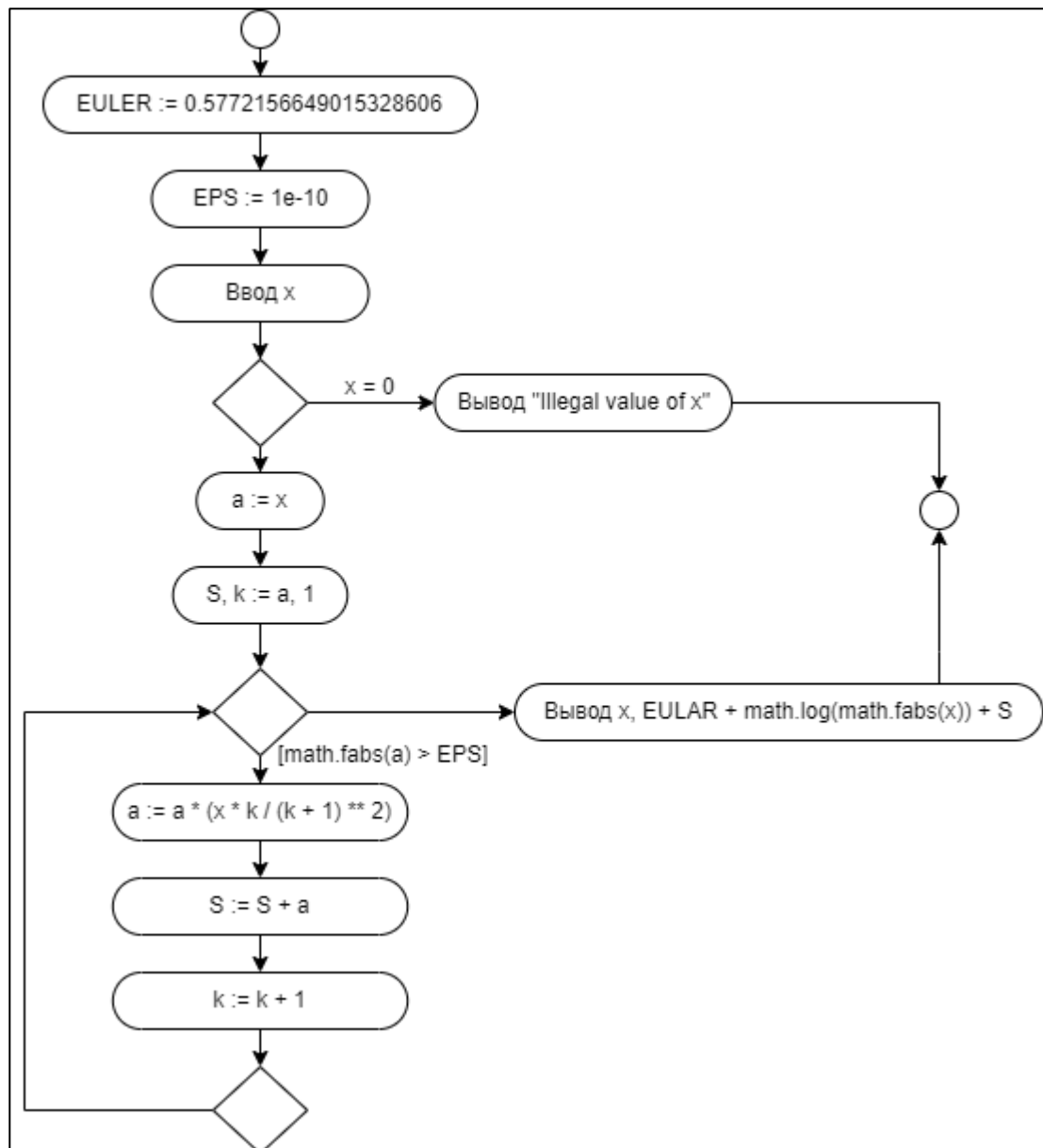
```

Рисунок 16 – Результат выполнения программы

Uml-диаграмма для примера 4:



Uml-диаграмма для примера 5:



Индивидуальные задания:

Вариант 11

11_1. Компания по снабжению электроэнергией взимает плату с клиентов по тарифу:

- 7 р. за 1 кВт/ч за первые 250 кВт/ч;
- 17 р. за кВт/ч, если потребление свыше 250, но не превышает 300 кВт/ч;
- 20 р. за кВт/ч, если потребление свыше 300 кВт/ч.
- Потребитель израсходовал n кВт/ч. Подсчитать плату.

Код:

```

1.py > ...
1  def main():
2      n = int(input("Enter kW: "))
3      payment = 0
4      for i in range(1, n+1):
5          if i <= 250: payment += 7
6          elif 250 < i <= 300: payment += 17
7          else: payment += 20
8
9      return payment
10
11
12  if __name__ == "__main__":
13      print(main())

```

Рисунок 17 – Код решения

Результаты при различных исходных данных:

```

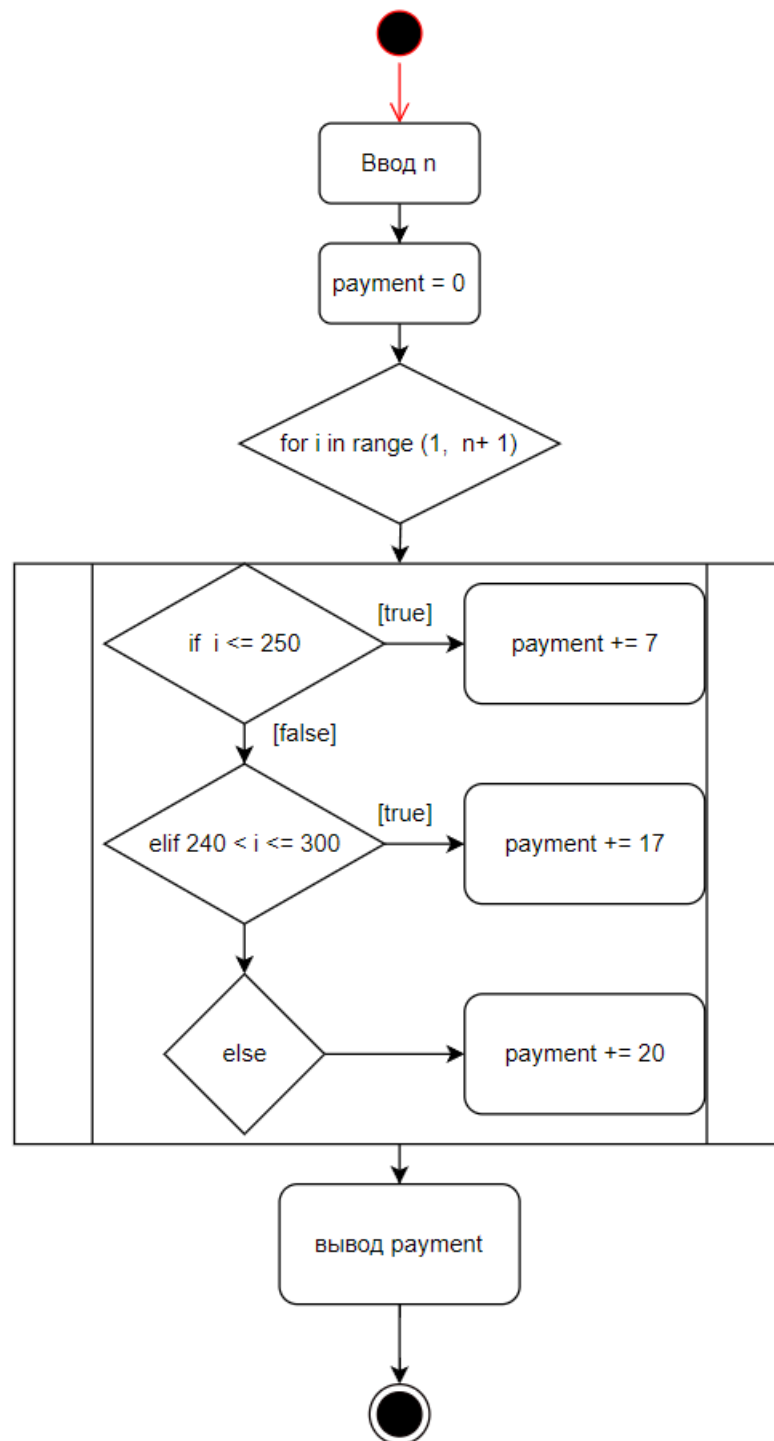
ПРОБЛЕМЫ    ВЫХОДНЫЕ ДАННЫЕ    КОНСОЛЬ ОТЛАДКИ    ТЕРМИНАЛ    ПОРТЫ

PS C:\Users\varfe\OneDrive\Рабочий стол\Воронкин\BSE5> & C:/Users/varfe/AppData/Локальные данные/Воронкин/BSE5/1.py
Enter kW: 100
700
PS C:\Users\varfe\OneDrive\Рабочий стол\Воронкин\BSE5> & C:/Users/varfe/AppData/Локальные данные/Воронкин/BSE5/1.py
Enter kW: 270
2090
PS C:\Users\varfe\OneDrive\Рабочий стол\Воронкин\BSE5> & C:/Users/varfe/AppData/Локальные данные/Воронкин/BSE5/1.py
Enter kW: 400
4600
PS C:\Users\varfe\OneDrive\Рабочий стол\Воронкин\BSE5> 

```

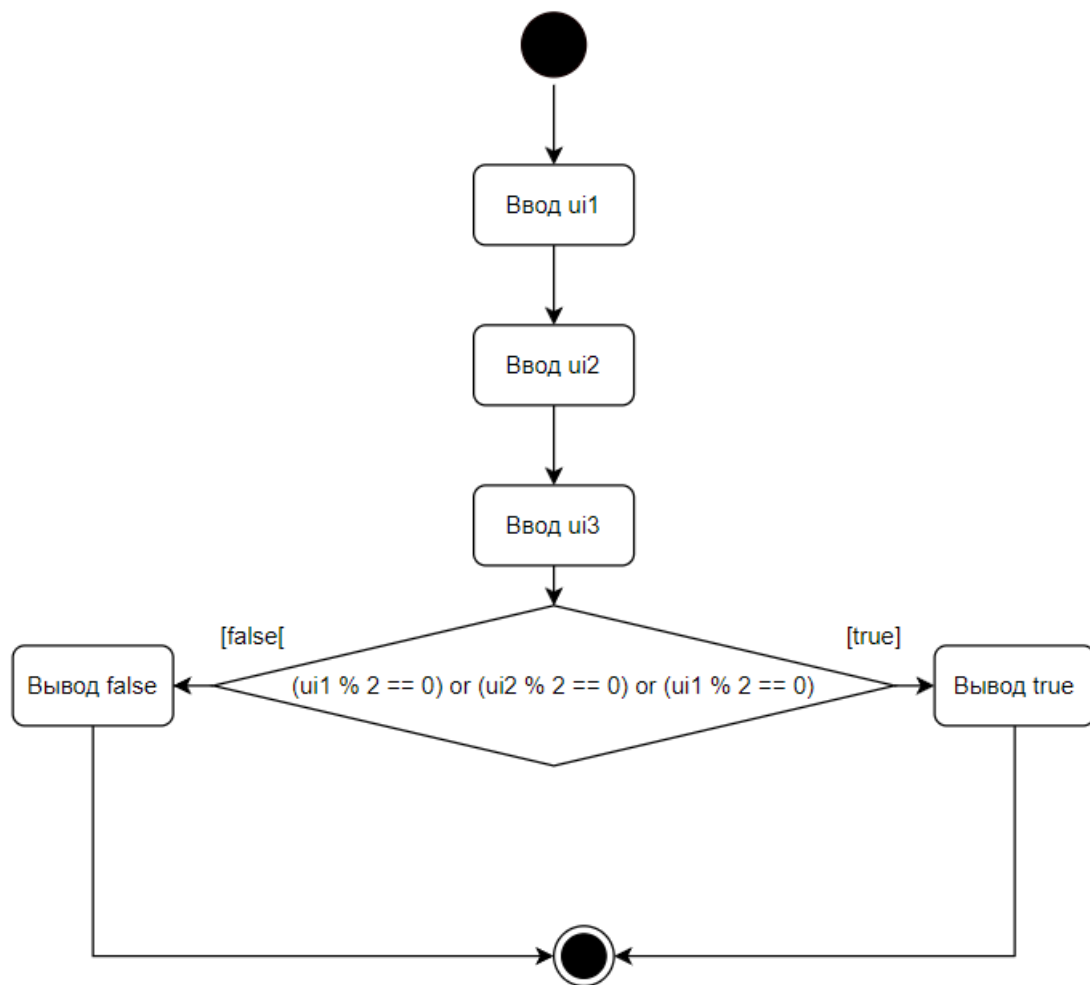
Рисунок 18 – Результат выполнения команды

Uml-диаграмма для примера 11_1:



11_2. Определить, есть ли среди трёх заданных чисел чётные.

Код решения:



11_3. Составьте программу, которая печатает таблицу умножения натуральных чисел в десятичной системе счисления.

Код:

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  usage  Odyssey Meltonyan
5  def multiplication_table(n):
6      print(" ", end='\t')
7      for i in range(1, n + 1):
8          print(f'{i}', end='\t')
9      print("\n-----")
10
11     for i in range(1, n + 1):
12         print(f'{i} |', end='\t')
13         for j in range(1, n + 1):
14             print(f'{i * j}', end='\t')
15         print()
16
17 if __name__ == "__main__":
18     n = 9
19     multiplication_table(n)
20

```

Рисунок 21 Код решения третьей задачи

	1	2	3	4	5	6	7	8	9

1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Рисунок 22 – Результат выполнения кода

Вопросы для защиты работы

1. Для чего нужны диаграммы деятельности UML? Диаграммы деятельности в UML (Unified Modeling Language) используются для визуального представления процессов и деятельности в системе. Они

помогают моделировать бизнес-процессы, анализировать и проектировать системы, понимать последовательность действий и взаимодействия между объектами.

2. Что такое состояние действия и состояние деятельности?

- Состояние действия (Action State): представляет мгновенные действия, которые не имеют продолжительности. Например, отправка сообщения.
- Состояние деятельности (Activity State): представляет длительные действия, которые могут продолжаться некоторое время. Например, обработка данных.

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

- Переходы могут быть обозначены стрелками с условиями.
- Ветвления отображаются с использованием ромбов, указывающих разветвление потока выполнения.

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

- Алгоритм разветвляющейся структуры часто ассоциируется с условным оператором (if-else).

5. Чем отличается разветвляющийся алгоритм от линейного?

- Разветвляющийся алгоритм имеет различные пути выполнения в зависимости от условий.
- Линейный алгоритм выполняется последовательно, без разветвлений.

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

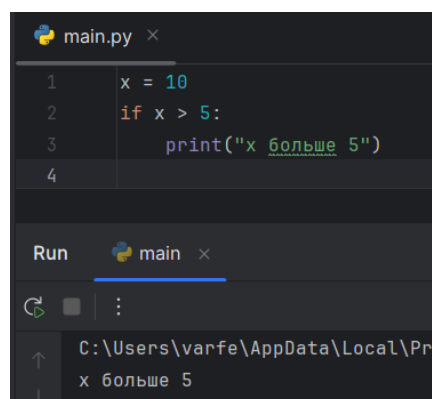
- Условный оператор в программировании выполняет различные блоки кода в зависимости от условия. Формы:
 - **if**: Выполняется, если условие истинно.
 - **else**: Выполняется, если условие в **if** ложно.
 - **elif**: Используется для проверки дополнительных условий вместе с **if**.

7. Какие операторы сравнения используются в Python?

- **==** (равно)
- **!=** (не равно)
- **<** (меньше)
- **>** (больше)
- **<=** (меньше или равно)
- **>=** (больше или равно)

8. Что называется простым условием? Приведите примеры.

- Простое условие — это выражение, которое может быть истинным или ложным.

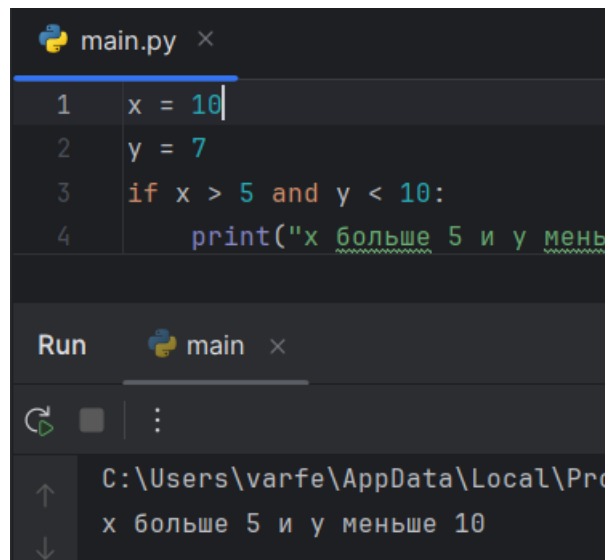


```
main.py x
1 x = 10
2 if x > 5:
3     print("x больше 5")
4
Run main x
C:\Users\varfe\AppData\Local\Pro
x больше 5
```

The screenshot shows a Python IDE window titled 'main.py'. The code contains a simple conditional statement: `x = 10`, `if x > 5:`, and `print("x больше 5")`. Below the code editor, there is a 'Run' button and a terminal window showing the output: `x больше 5`.

9. Что такое составное условие? Приведите примеры.

- Составное условие — это комбинация нескольких простых условий с использованием логических операторов.



```
main.py x
1 x = 10
2 y = 7
3 if x > 5 and y < 10:
4     print("x больше 5 и y меньше 10")

Run main x
C:\Users\varfe\AppData\Local\Pro
x больше 5 и y меньше 10
```

10.Какие логические операторы допускаются при составлении сложных условий?

- **and** (логическое И)
- **or** (логическое ИЛИ)
- **not** (логическое НЕ)

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

- Да, оператор ветвления может содержать внутри себя другие ветвления, создавая вложенные структуры **if-else**.

12.Какой алгоритм является алгоритмом циклической структуры?

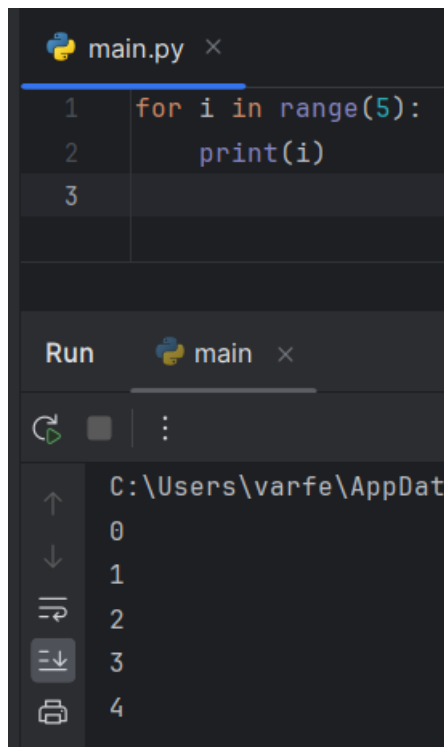
- Алгоритм циклической структуры часто ассоциируется с циклическими операторами, такими как **for** и **while**.

13.Типы циклов в языке Python.

- **for** - цикл со счетчиком.
- **while** - цикл с предусловием.

14.Назовите назначение и способы применения функции **range**.

- **range** используется для создания последовательности чисел и обычно применяется в циклах **for**. Пример:



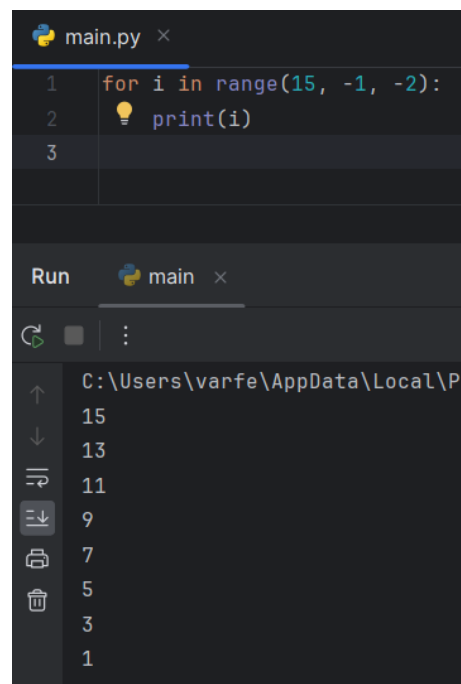
The screenshot shows a Python IDE with a file named `main.py`. The code contains a `for` loop that iterates over the range from 0 to 4. The output of the program is displayed in the console, showing the numbers 0, 1, 2, 3, and 4 on separate lines.

```
1 for i in range(5):  
2     print(i)  
3
```

Run main

C:\Users\varfe\AppData\Local\Programs\Python\Python39-64\python.exe
0
1
2
3
4

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



The screenshot shows a Python IDE with a file named `main.py`. The code contains a `for` loop that iterates over the range from 15 down to 1 with a step of -2. The output of the program is displayed in the console, showing the numbers 15, 13, 11, 9, 7, 5, 3, and 1 on separate lines.

```
1 for i in range(15, -1, -2):  
2     print(i)  
3
```

Run main

C:\Users\varfe\AppData\Local\Programs\Python\Python39-64\python.exe
15
13
11
9
7
5
3
1

16. Могут ли быть циклы вложенными?

- Да, циклы могут быть вложенными, т.е. один цикл может находиться внутри другого.

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

- Бесконечный цикл может возникнуть, если условие цикла всегда истинно. Выход из бесконечного цикла можно осуществить с использованием оператора **break** или изменением условия цикла.

18. Для чего нужен оператор **break**?

- **break** используется для выхода из цикла досрочно, даже если условие цикла остается истинным.

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

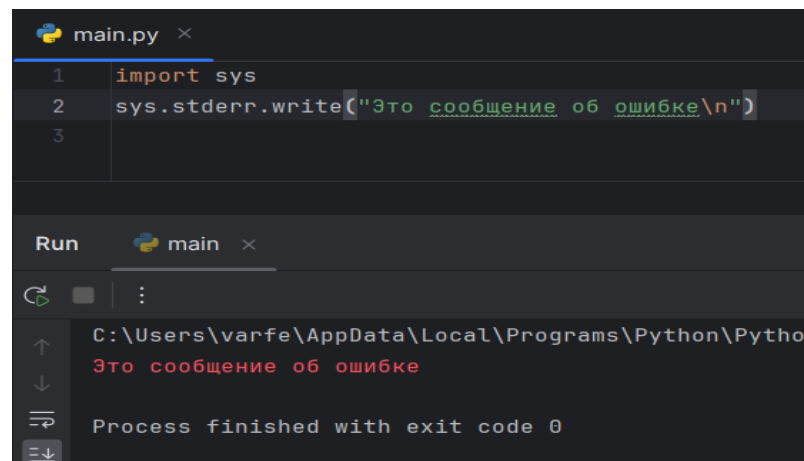
- **continue** используется для перехода к следующей итерации цикла, пропуская оставшуюся часть кода в текущей итерации.

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

- **stdout** (стандартный вывод) используется для вывода обычных данных, а **stderr** (стандартный вывод ошибок) используется для вывода сообщений об ошибках.

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

- Можно использовать **sys.stderr** из модуля **sys**:



```
main.py x
1 import sys
2 sys.stderr.write("Это сообщение об ошибке\n")
3

Run main x
C:\Users\varfe\AppData\Local\Programs\Python\Python
Это сообщение об ошибке
Process finished with exit code 0
```

22. Каково назначение функции **exit**?

- **exit()** используется для выхода из программы. Она принимает необязательный аргумент, который может быть использован как код возврата (exit code).