

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования «СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.2

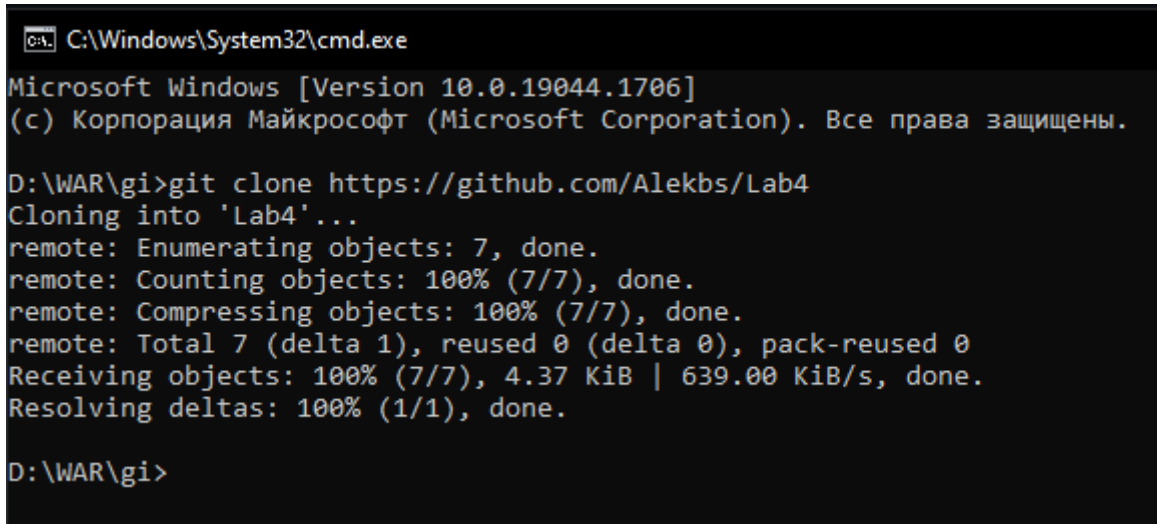
Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Условные операторы и циклы в языке Python»

Выполнил: студент 1 курса,
группы ИВТ-б-о-21-1
Богдан Александр Анатольевич

Ставрополь 2022

Цель: 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.

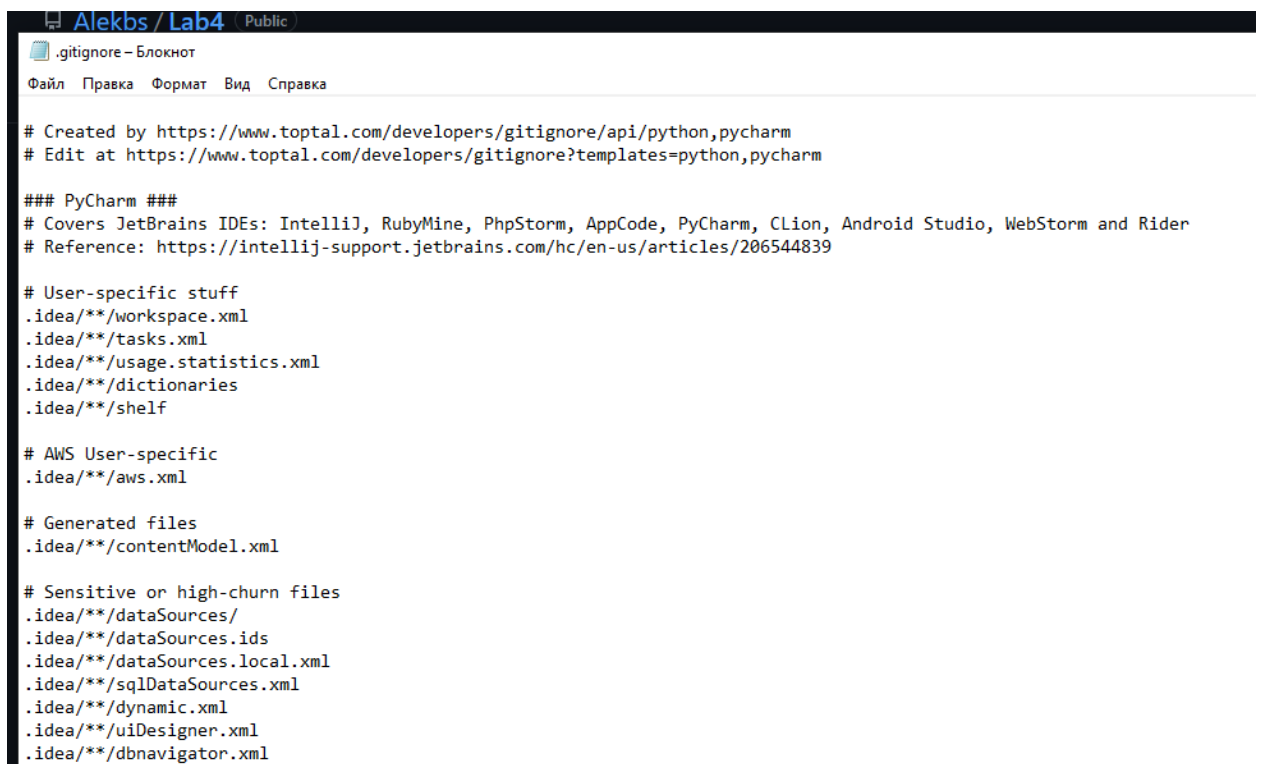


```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.1706]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

D:\WAR\gi>git clone https://github.com/Alekbs/Lab4
Cloning into 'Lab4'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), 4.37 KiB | 639.00 KiB/s, done.
Resolving deltas: 100% (1/1), done.

D:\WAR\gi>
```

Рисунок 1. Клонирование репозитория



```
Alekbs / Lab4 (Public)
.gitignore - Блокнот
Файл  Правка  Формат  Вид  Справка

# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
.idea/**/uiDesigner.xml
.idea/**/dbnavigator.xml
```

Рисунок 2. Изменения файла .gitignore

```
D:\WAR\gi\Lab4>git branch develop

D:\WAR\gi\Lab4>git flow init

Which branch should be used for bringing forth production releases?
- develop
- main
Branch name for production releases: [main] develop

Which branch should be used for integration of the "next release"?
- main
Branch name for "next release" development: [] main

How to name your supporting branch prefixes?
Feature branches? [feature/] features
Bugfix branches? [bugfix/] bug
Release branches? [release/] release
Hotfix branches? [hotfix/] hot
Support branches? [support/] support
Version tag prefix? [] pre
Hooks and filters directory? [D:/WAR/gi/Lab4/.git/hooks] dir

D:\WAR\gi\Lab4>git branch
  develop
* main

D:\WAR\gi\Lab4>git checkout develop
Switched to branch 'develop'
```

Рисунок 3. . Организация репозитория в соответствии с моделью git-flow

Изучил рекомендации к оформлению исходного кода на языке Python PEP-8.

Создал проект PyCharm в папке репозитория.

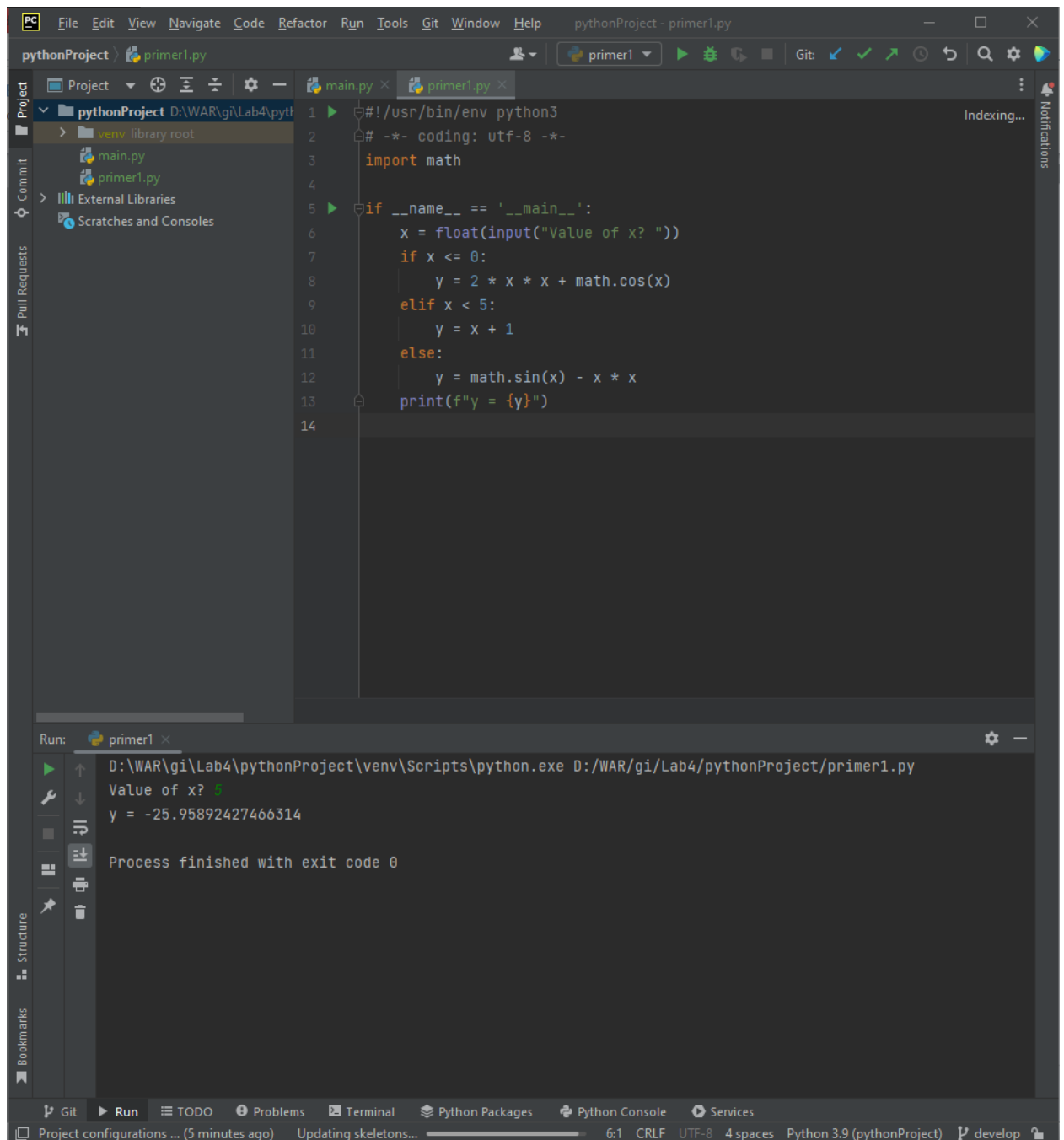


Рисунок 4. Пример 1

The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'pythonProject' with a 'venv' directory and files 'main.py', 'primer1.py', and 'primer2.py'. The code editor displays the content of 'primer2.py', which is a Python script that takes a month number as input and prints the corresponding season. The script uses a series of if-elif-else statements to check the input value against the range of months for each season. The terminal window at the bottom shows the command to run the script and the user's input '6', resulting in the output 'Лето' (Summer).

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

if __name__ == '__main__' > else

primer1 x primer2 x

D:\WAR\gi\Lab4\pythonProject\venv\Scripts\python.exe D:/WAR/gi/Lab4/pythonProject/primer2.py

Введите номер месяца: 6

Лето

Рисунок 5. Пример 2

The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'pythonProject' with a 'venv' directory and files 'main.py', 'primer1.py', 'primer2.py', and 'primer3.py'. The code editor displays the content of 'primer3.py', which is a Python script that calculates the sum of the series $S = \sum_{k=1}^n \frac{\log(k \cdot x)}{k^2}$. The script takes two inputs, 'n' and 'x', and prints the result. The terminal window at the bottom shows the command to run the script and the user's inputs '23' for 'n' and '2' for 'x', resulting in the output 'S = 1.871361905982464'.

```
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     S = 0.0
9     for k in range(1, n + 1):
10        a = math.log(k * x) / (k * k)
11        S += a
12    print(f"S = {S}")
```

if __name__ == '__main__'

primer1 x primer3 x

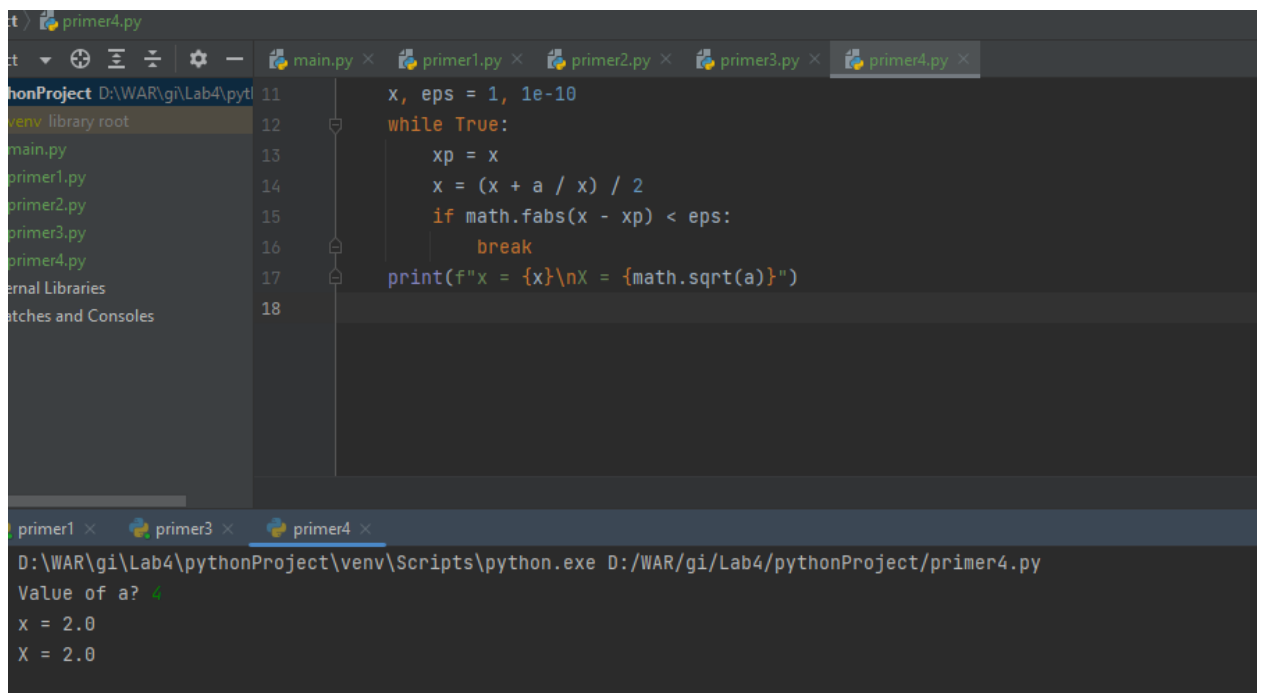
D:\WAR\gi\Lab4\pythonProject\venv\Scripts\python.exe D:/WAR/gi/Lab4/pythonProject/primer3.py

Value of n? 23

Value of x? 2

S = 1.871361905982464

Рисунок 6. Пример 3



```

11 x, eps = 1, 1e-10
12 while True:
13     xp = x
14     x = (x + a / x) / 2
15     if math.fabs(x - xp) < eps:
16         break
17 print(f"x = {x}\nX = {math.sqrt(a)}")
18

```

primer1 x primer3 x primer4 x

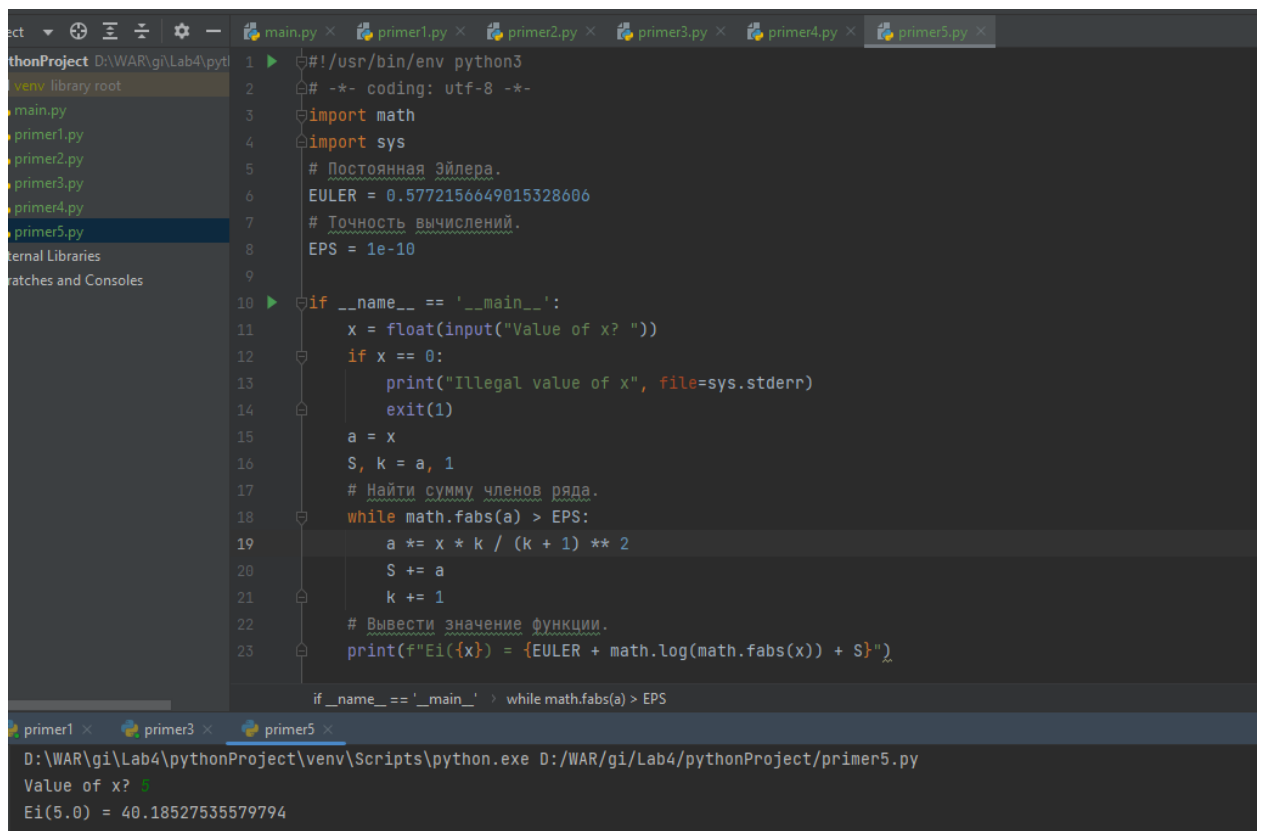
D:\WAR\gi\Lab4\pythonProject\venv\Scripts\python.exe D:/WAR/gi/Lab4/pythonProject/primer4.py

Value of a? 4

x = 2.0

X = 2.0

Рисунок 7. Пример 4



```

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

```

primer1 x primer3 x primer5 x

D:\WAR\gi\Lab4\pythonProject\venv\Scripts\python.exe D:/WAR/gi/Lab4/pythonProject/primer5.py

Value of x? 5

Ei(5.0) = 40.18527535579794

Рисунок 8. Пример 5

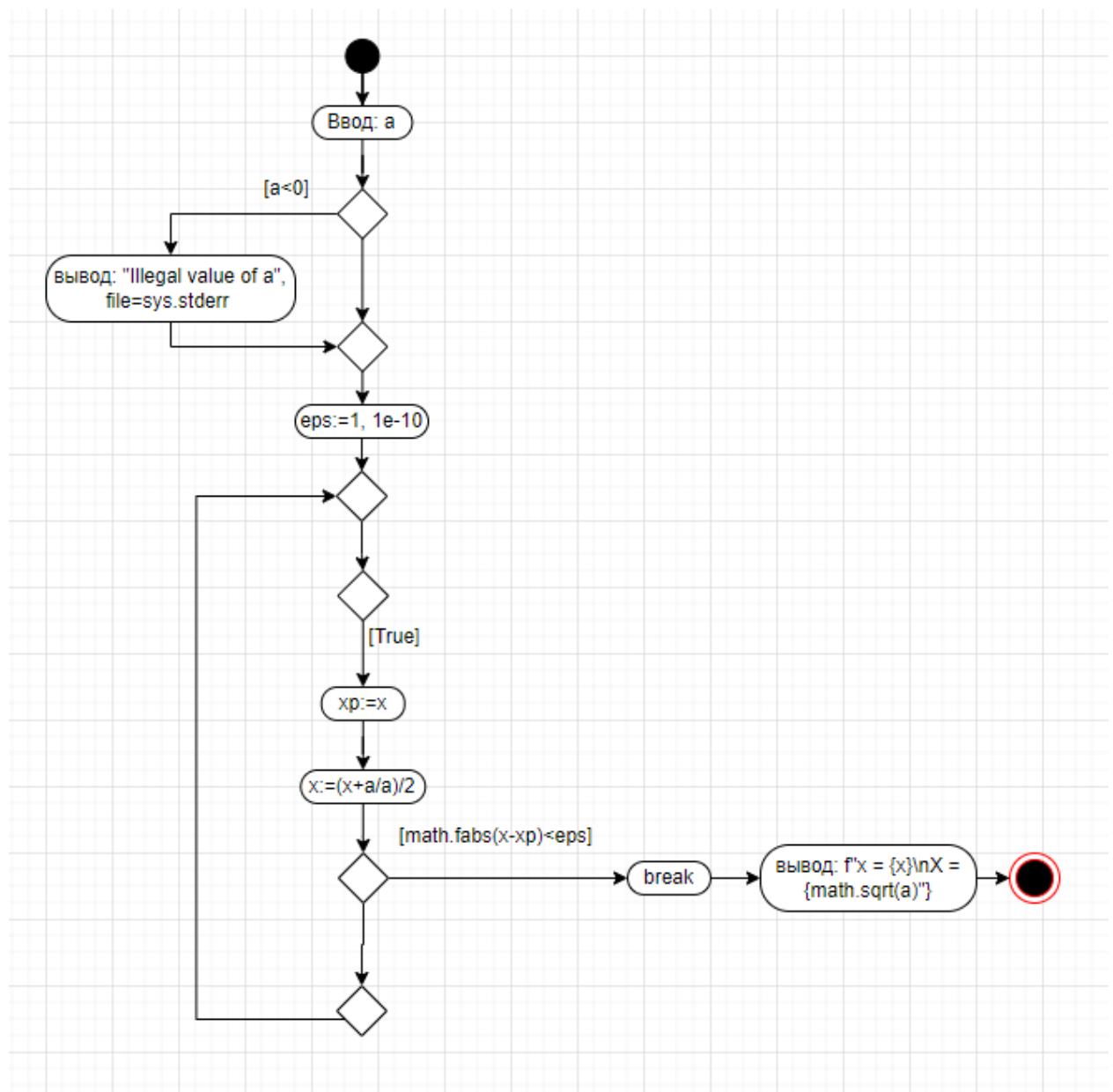


Рисунок 9. UML-диаграмма к примеру 4

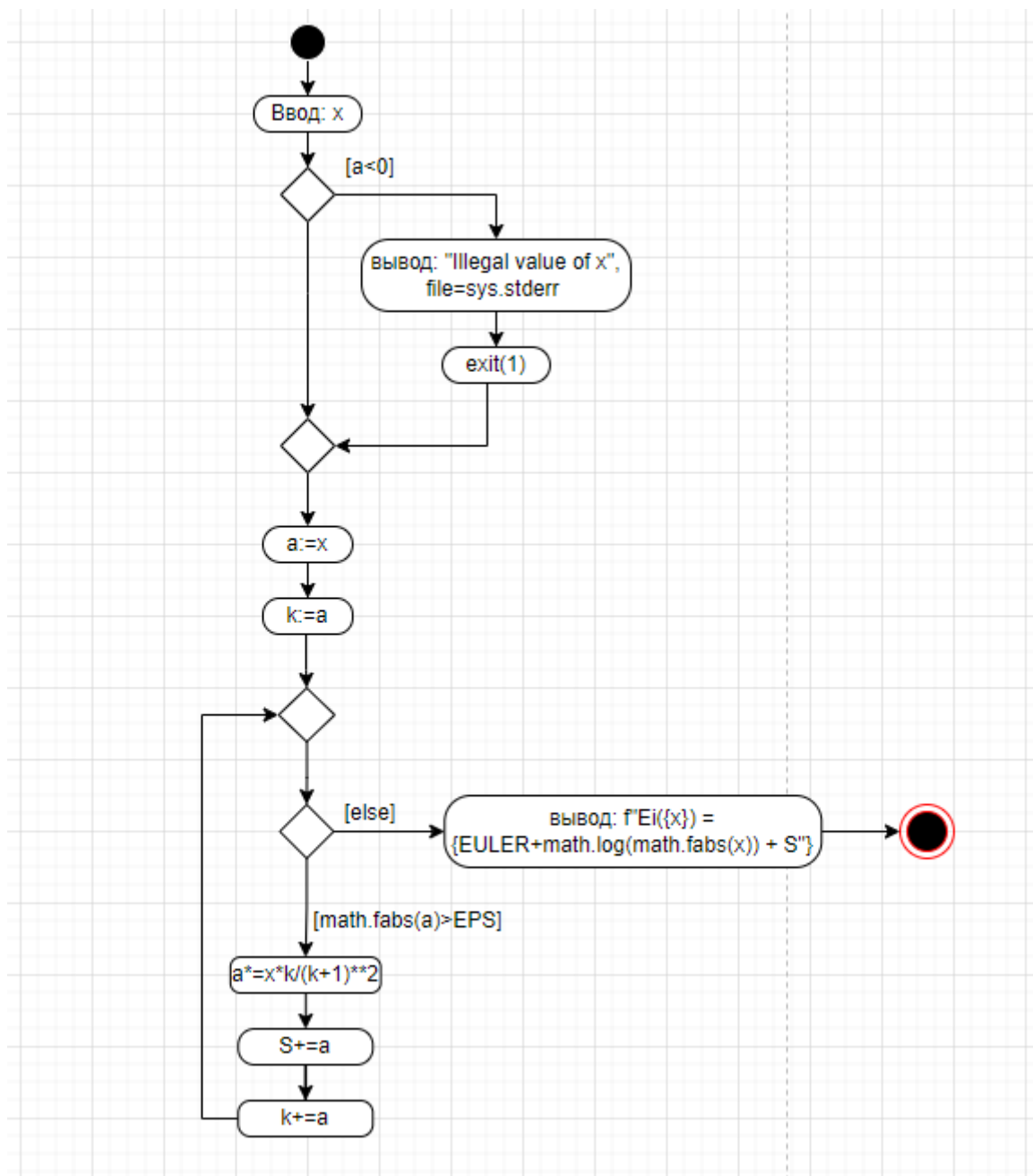


Рисунок 10. UML-диаграмма к примеру 5

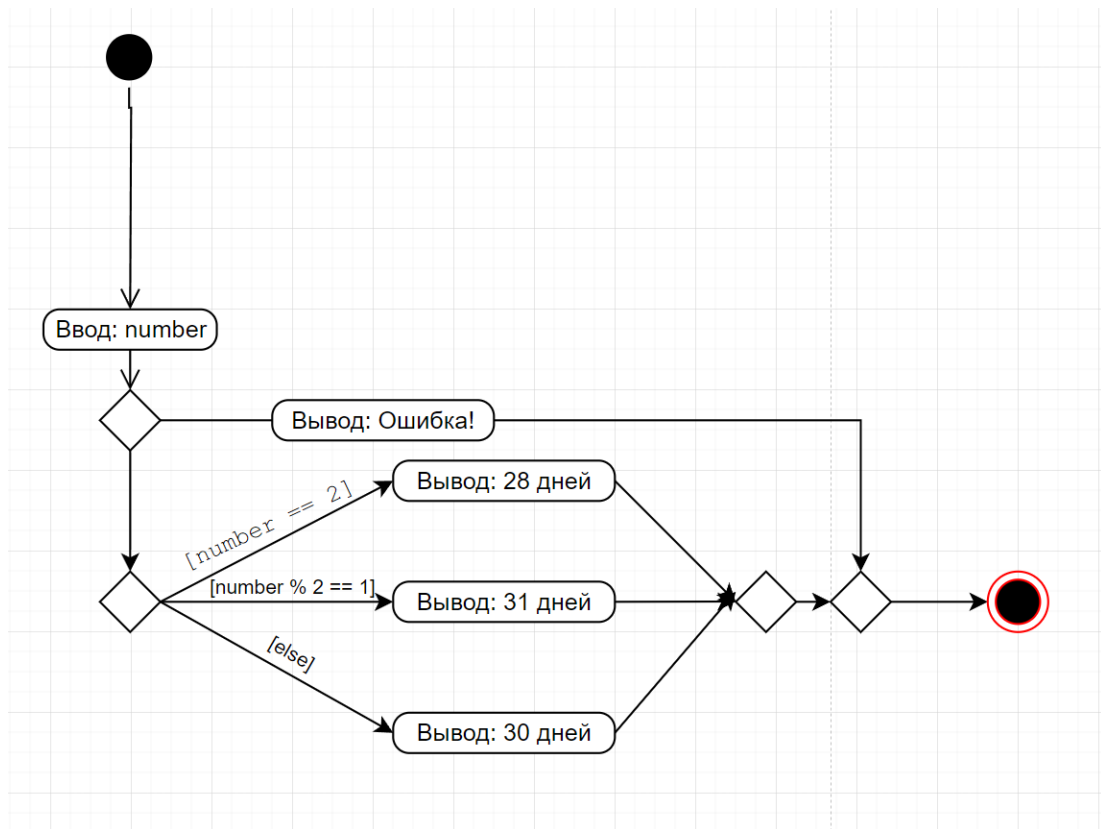


Рисунок 11. UML-диаграмма 1 задание

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      number = int(input("Введите номер месяца: "))
8      if number < 1 or number > 12:
9          print("Ошибка!", file=sys.stderr)
10         exit(1)
11     else:
12         if number == 2:
13             print("28 дней")
14         elif number % 2 == 1:
15             print("31 день")
16         elif number % 2 == 0:
17             print("30 дней")
18

```

ind1 x ind2 x

D:\WAR\gi\Lab4\pythonProject\venv\Scripts\python.exe D:\WAR\gi\Lab4\pythonProject\ind1.py

Введите номер месяца: 2

28 дней

Process finished with exit code 0

Рисунок 12. Индивидуальное задание 1 (Вариант 2)

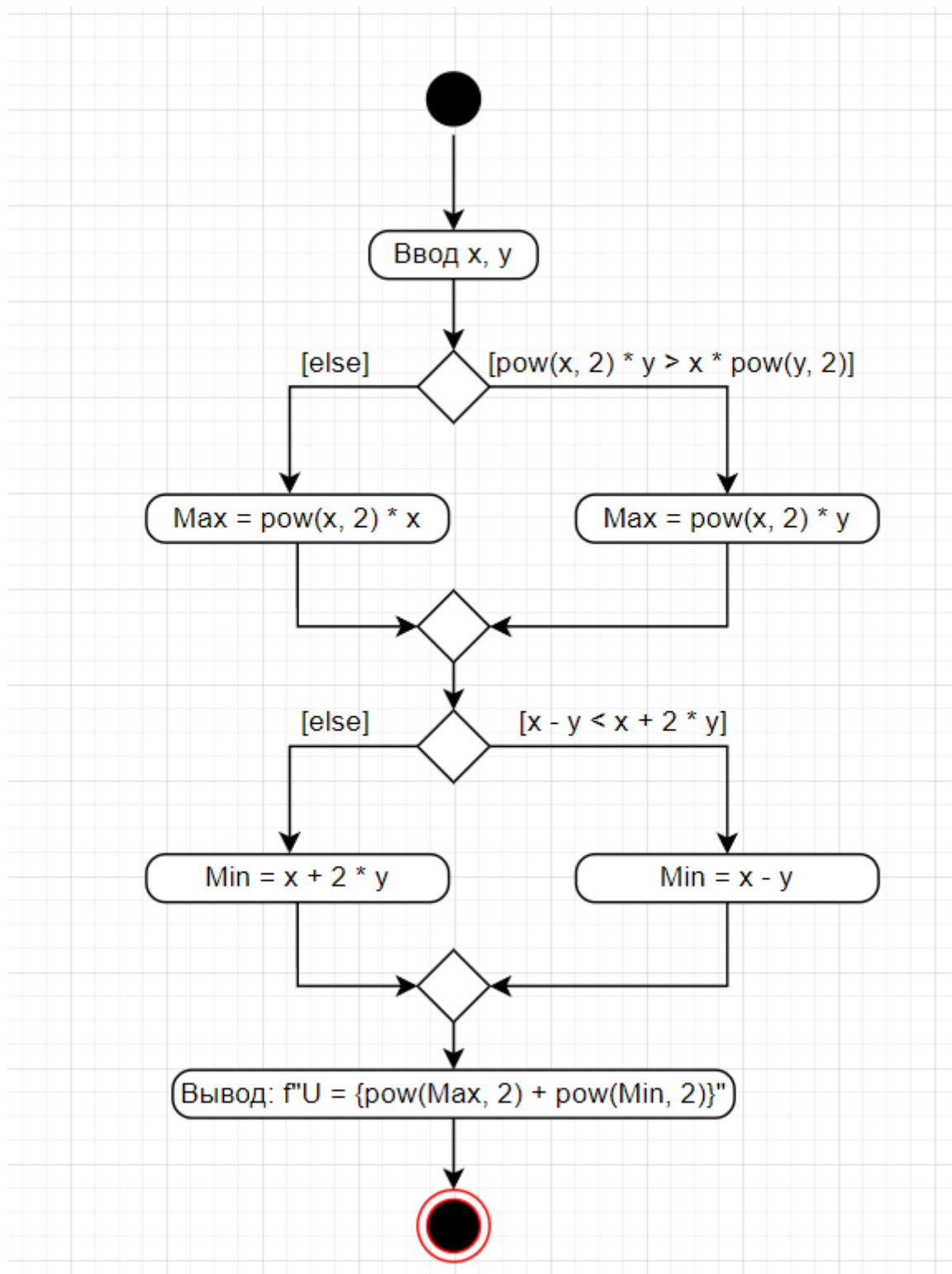


Рисунок 13. UML-диаграмма 2 задание

The image shows a Python IDE with multiple tabs. The active tab is `ind2.py`, which contains the following code:

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      x, y = int(input("Enter the x: ")), int(input("Enter the y: "))
6      if pow(x, 2) * y > x * pow(y, 2):
7          Max = pow(x, 2) * y
8      else:
9          Max = x * pow(y, 2)
10     if x - y < x + 2 * y:
11         Min = x - y
12     else:
13         Min = x + 2 * y
14     print(f"U = {pow(Max, 2) + pow(Min, 2)}")
15
```

The console output shows the execution of the script:

```
D:\WAR\gi\Lab4\pythonProject\venv\Scripts\python.exe D:\WAR\gi\Lab4\pythonProject\ind2.py
Enter the x: 5
Enter the y: 3
U = 325
Process finished with exit code 0
```

Рисунок 14. Индивидуальное задание 2 (Вариант 2)

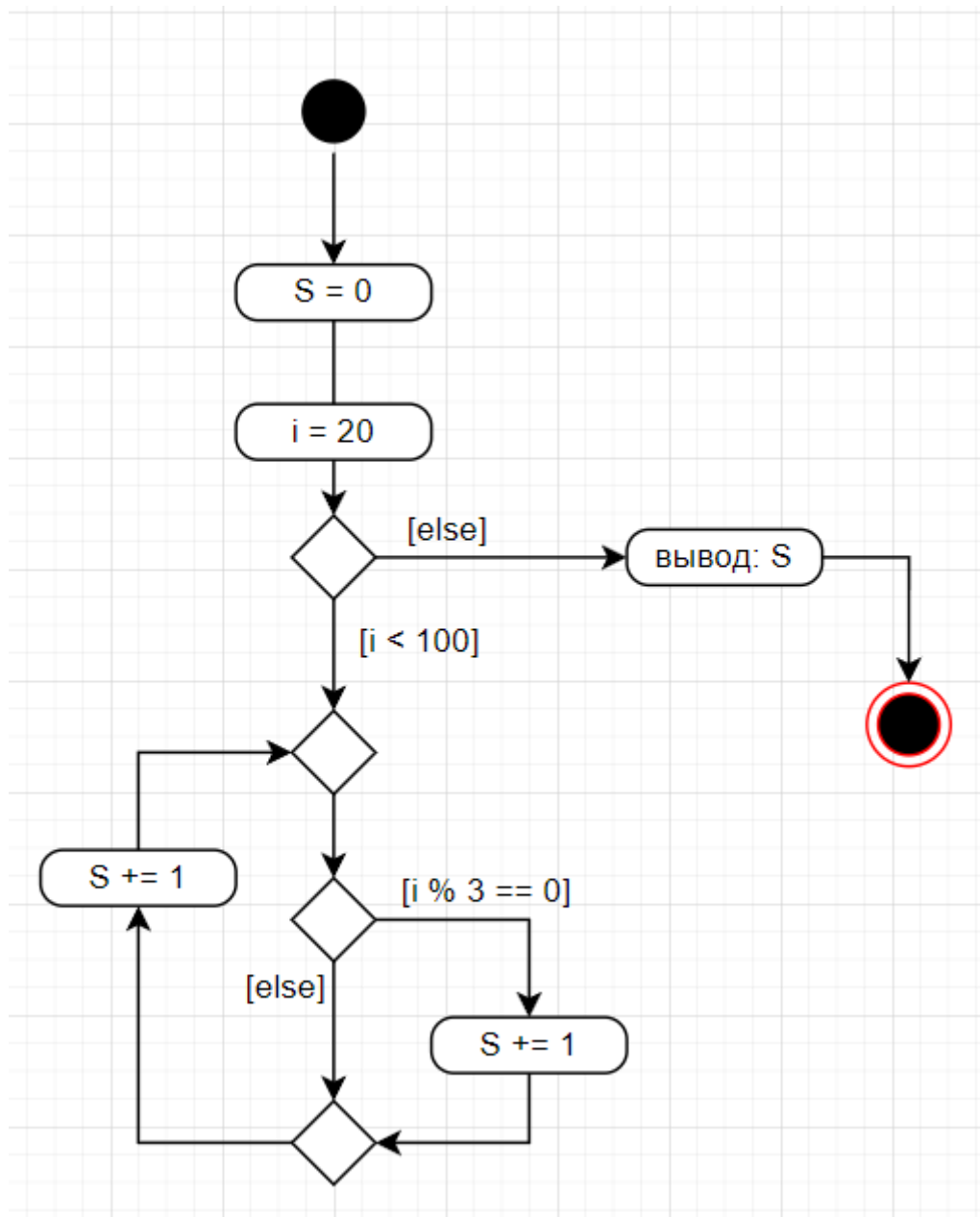


Рисунок 15. UML-диаграмма 3 задание

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      S = 0
6      for i in range(20, 100):
7          if i % 3 == 0:
8              S += i
9      print(f"Сумма = {S}")
  
```

ind1 x ind2 x ind3 x

D:\WAR\gi\Lab4\pythonProject\venv\Scripts\python.exe D:\WAR\gi\Lab4\pythonProject\ind3.py

Сумма = 1620

Рисунок 16. Индивидуальное задание 3 (Вариант 2)

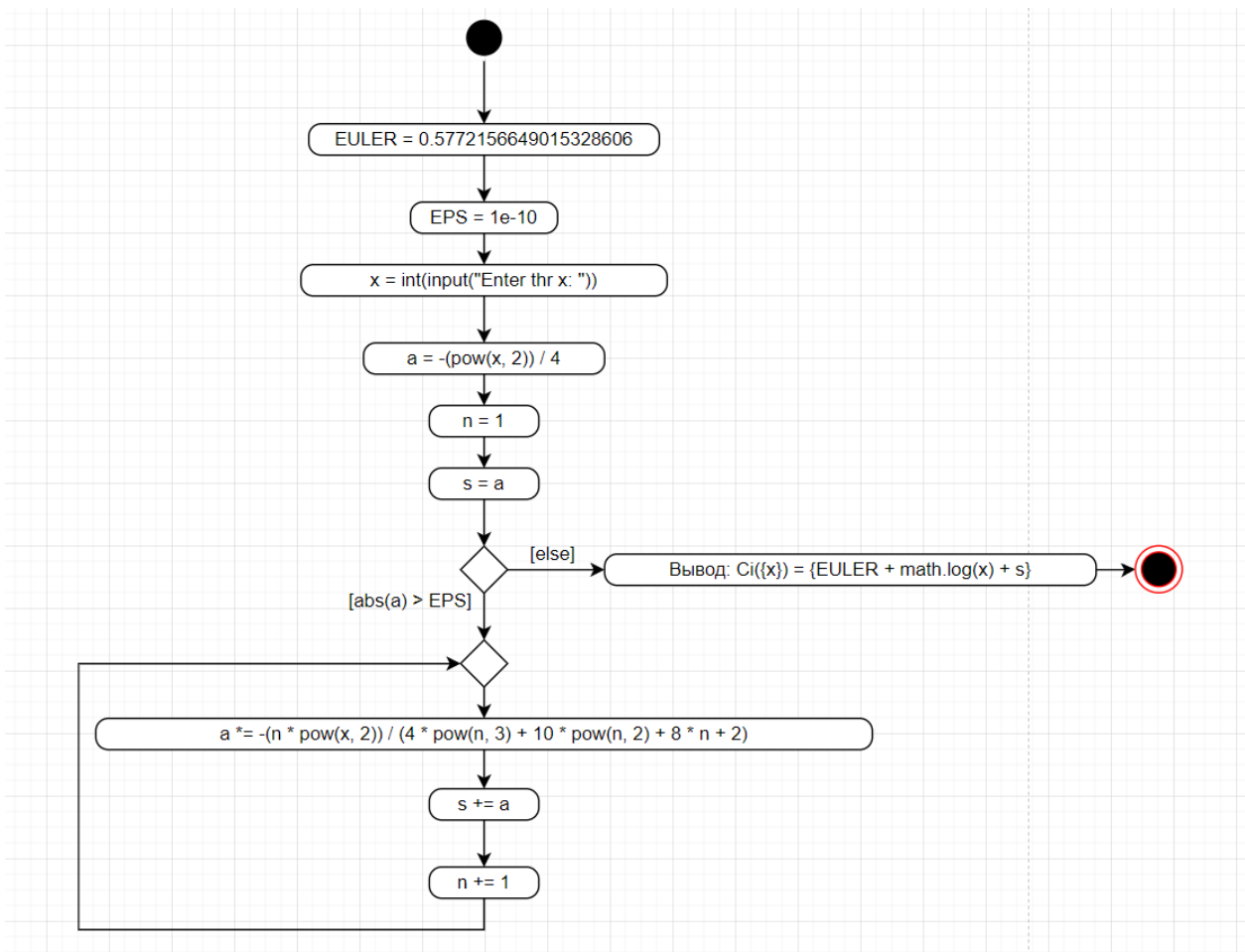


Рисунок 17. UML-диаграмма усложненное задание

```

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 = int(input("Enter thr x: "))
14     a = -(pow(x, 2)) / 4
15     n = 1
16     s = a
17     while abs(a) > EPS:
18         a *= -(n * x * x) / (4 * pow(n, 3) + 10 * n * n + 8 * n + 2)
19         s += a
20         n += 1
21     print(f"Ci({x}) = {EULER + math.log(x) + s}")
22
23 if __name__ == '__main__': while abs(a) > EPS
  
```

ind1 x ind2 x hard_ind1 x

D:\WAR\gi\Lab4\pythonProject\venv\Scripts\python.exe D:\WAR\gi\Lab4\pythonProject\hard_ind1.py

Enter thr x: 1

Ci(1) = 0.33740392290096516

Process finished with exit code 0

Рисунок 18. Усложненное задание 3 (Вариант 2)

Сверил полученное значение с помощью таблицы

x	$\text{Ci } x$	$\frac{\Delta}{\Delta x} \text{ Ci } x$	x	$\text{Ci } x$	$\frac{\Delta}{\Delta x} \text{ Ci } x$
0	$-\infty$		1,30	0,4457	0,253
0,005	-4,7211	138,7	1,40	0,4620	0,163
0,010	-4,0280	81,1	1,50	0,4704	0,084
0,015	-3,6225	57,5	1,571	0,4720	0,016
0,020	-3,3349	40,53	1,65	0,4701	-0,024
0,03	-2,9296	28,75	1,80	0,4568	-0,089
0,04	-2,6421	22,30	1,9264	0,4373	-0,154
0,05	-2,4191	18,20	2,00	0,4230	-0,194
0,06	-2,2371	14,35	2,225	0,3683	-0,243
0,08	-1,9501	11,11	2,50	0,2859	-0,300
0,10	-1,7279	9,07	2,750	0,2033	-0,330
0,12	-1,5466	7,37	3,00	0,1196	-0,335
0,15	-1,3255	6,00	3,384	0	-0,311
0,18	-1,1457	5,18	3,500	-0,0321	-0,149
0,20	-1,0422	4,35	3,75	-0,0931	-0,244
0,25	-0,8247	3,51	4,00	-0,1410	-0,192
0,30	-0,6492	2,92	4,25	-0,1746	-0,134
0,35	-0,5031	2,48	4,50	-0,1935	-0,0756
0,40	-0,3788	2,15	4,60	-0,1970	-0,035
0,45	-0,2715	1,87	4,712	-0,1984	-0,0125
0,50	-0,1778	1,65	5,00	-0,1900	+0,0292
0,55	-0,0953	1,46	5,50	-0,1420	+0,0960
0,60	-0,0223	1,35	6,00	-0,0680	+0,148
0,6165	0	1,27	6,427	0	+0,159
0,65	0,0426	1,16	6,50	0,0111	+0,152
0,70	0,1005	1,03	7,00	0,0767	+0,131
0,75	0,1522		7,854	0,1283	+0,0551
0,80	0,1983	0,92	8,50	0,0994	-0,0378
0,90	0,2761	0,778	9,00	0,0553	-0,0882
1,00	0,3374	0,613	9,526	0	-0,105
1,10	0,3849	0,475	10,00	-0,0454	-0,0958
1,20	0,4204	0,355	11,00	-0,0896	-0,0442

Рисунок 19. Таблица интегрального косинуса

```
D:\WAR\gi\Lab4>git push origin develop
Enumerating objects: 22, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 8 threads
Compressing objects: 100% (19/19), done.
Writing objects: 100% (21/21), 4.42 KiB | 754.00 KiB/s, done.
Total 21 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/Alekbs/Lab4/pull/new/develop
remote:
To https://github.com/Alekbs/Lab4
 * [new branch]   develop -> develop
D:\WAR\gi\Lab4>
```

Рисунок 20. Зафиксировал изменения

```

D:\WAR\gi\Lab4>git merge develop
Updating 9a10342..64b3f6d
Fast-forward
 pythonProject/.idea/.gitignore | 3 +++
 .../.idea/inspectionProfiles/profiles_settings.xml | 6 ++++++
 pythonProject/.idea/misc.xml | 4 ++++
 pythonProject/.idea/modules.xml | 8 ++++++++
 pythonProject/.idea/pythonProject.iml | 10 ++++++++
 pythonProject/.idea/vcs.xml | 6 ++++++
 pythonProject/hard_ind1.py | 20 ++++++
 pythonProject/ind1.py | 17 ++++++
 pythonProject/ind2.py | 14 ++++++
 pythonProject/ind3.py | 9 ++++++
 pythonProject/main.py | 16 ++++++
 pythonProject/primer1.py | 14 ++++++
 pythonProject/primer2.py | 18 ++++++
 pythonProject/primer3.py | 13 ++++++
 pythonProject/primer4.py | 18 ++++++
 pythonProject/primer5.py | 24 ++++++
16 files changed, 200 insertions(+)
create mode 100644 pythonProject/.idea/.gitignore
create mode 100644 pythonProject/.idea/inspectionProfiles/profiles_settings.xml
create mode 100644 pythonProject/.idea/misc.xml
create mode 100644 pythonProject/.idea/modules.xml
create mode 100644 pythonProject/.idea/pythonProject.iml
create mode 100644 pythonProject/.idea/vcs.xml
create mode 100644 pythonProject/hard_ind1.py
create mode 100644 pythonProject/ind1.py
create mode 100644 pythonProject/ind2.py
create mode 100644 pythonProject/ind3.py
create mode 100644 pythonProject/main.py
create mode 100644 pythonProject/primer1.py
create mode 100644 pythonProject/primer2.py
create mode 100644 pythonProject/primer3.py
create mode 100644 pythonProject/primer4.py
create mode 100644 pythonProject/primer5.py

```

Рисунок 21. Выполнил слияние ветки для разработки с веткой main

Ответы на вопросы:

1. Для чего нужны диаграммы деятельности UML?

Диаграммы деятельности - это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов поведения системы. Диаграмма деятельности - это, по существу, блок-схема, которая показывает, как поток управления переходит от одной деятельности к другой, однако, по сравнению с последней, у ней есть явные преимущества: поддержка многопоточности и объектно-ориентированного проектирования.

2. Что такое состояние действия и состояние деятельности?

Состояние действия и состояние деятельности. В потоке управления, моделируемом диаграммой деятельности, происходят различные события.

Вы можете вычислить выражение, в результате чего изменяется значение некоторого атрибута или возвращается некоторое значение. Также, например, можно выполнить операцию над объектом, послать ему сигнал или даже создать его или уничтожить. Все эти выполняемые атомарные вычисления называются состояниями действия, поскольку каждое из них есть состояние системы, представляющее собой выполнение некоторого действия.

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.

Поток управления должен где-то начинаться и заканчиваться (разумеется, если это не бесконечный поток, у которого есть начало, но нет конца). Как показано на рисунке, вы можете задать как начальное состояние (закрашенный кружок), так и конечное (закрашенный кружок внутри окружности).

4. Какой алгоритм является алгоритмом разветвляющейся структуры?

Состояние действия и состояние деятельности. В потоке управления, моделируемом диаграммой деятельности, происходят различные события. Вы можете вычислить выражение, в результате чего изменяется значение некоторого атрибута или возвращается некоторое значение. Также, например, можно выполнить операцию над объектом, послать ему сигнал или даже создать его или уничтожить. Все эти выполняемые атомарные вычисления называются состояниями действия, поскольку каждое из них

есть состояние системы, представляющее собой выполнение некоторого действия.

5. Чем отличается разветвляющийся алгоритм от линейного?

Линейный алгоритм - алгоритм, все этапы которого выполняются однократно и строго последовательно. Разветвляющийся алгоритм - алгоритм, содержащий хотя бы одно условие, в результате проверки которого ЭВМ обеспечивает переход на один из двух возможных шагов.

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

Условные операторы – это специальные конструкции, благодаря которым в программе происходит ветвление в зависимости от условий.

Условный оператор имеет полную и краткую формы.

7. Какие операторы сравнения используются в Python?

If, elif, else.

8. Что называется простым условием? Приведите примеры.

Условные операторы – это специальные конструкции, благодаря которым в программе происходит ветвление в зависимости от условий.

Пример: `a == b`

9. Что такое составное условие? Приведите примеры.

Составное условие – логическое выражение, содержащее несколько простых условий объединенных логическими операциями. Это операции not, and, or.

Пример: `(a == b or a == c)`

10. Какие логические операторы допускаются при составлении сложных условий?

Not, and, or.

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

Может.

12. Какой алгоритм является алгоритмом циклической структуры?

Циклический алгоритм — это вид алгоритма, в процессе выполнения которого одно или несколько действий нужно повторить.

13. Типы циклов в языке Python.

В Python есть 2 типа циклов: - цикл while, - цикл for.

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

Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно.

14. Назовите назначение и способы применения функции range .

Функция range генерирует серию целых чисел, от значения start до stop, указанного пользователем. Мы можем использовать его для цикла for и обходить весь диапазон как список.

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

```
for i in range(15, 0, -2).
```

16. Могут ли быть циклы вложенными?

Могут.

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

Бесконечный цикл в программировании — цикл, написанный таким образом, что условие выхода из него никогда не выполняется. Чтобы выйти из цикла нужно использовать оператор break.

18. Для чего нужен оператор break?

Используется для выхода из цикла.

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

Оператор `continue` используется только в циклах. В операторах `for` , `while` , `do while` , оператор `continue` выполняет пропуск оставшейся части кода тела цикла и переходит к следующей итерации цикла.

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

Ввод и вывод распределяется между тремя стандартными потоками: `stdin` — стандартный ввод (клавиатура), `stdout` — стандартный вывод (экран), `stderr` — стандартная ошибка (вывод ошибок на экран)

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

Указать в `print(..., file=sys.stderr)`.

22. Каково назначение функции `exit`?

Функция `exit()` модуля `sys` - выход из Python.

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