

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Условные операторы и циклы в языке Python»

ОТЧЕТ
по лабораторной работе №5
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна

2 курс, группа ПИЖ-б-о-21-1,

09.03.04 «Программная инженерия»,

направленность (профиль) «Разработка

и сопровождение программного

обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

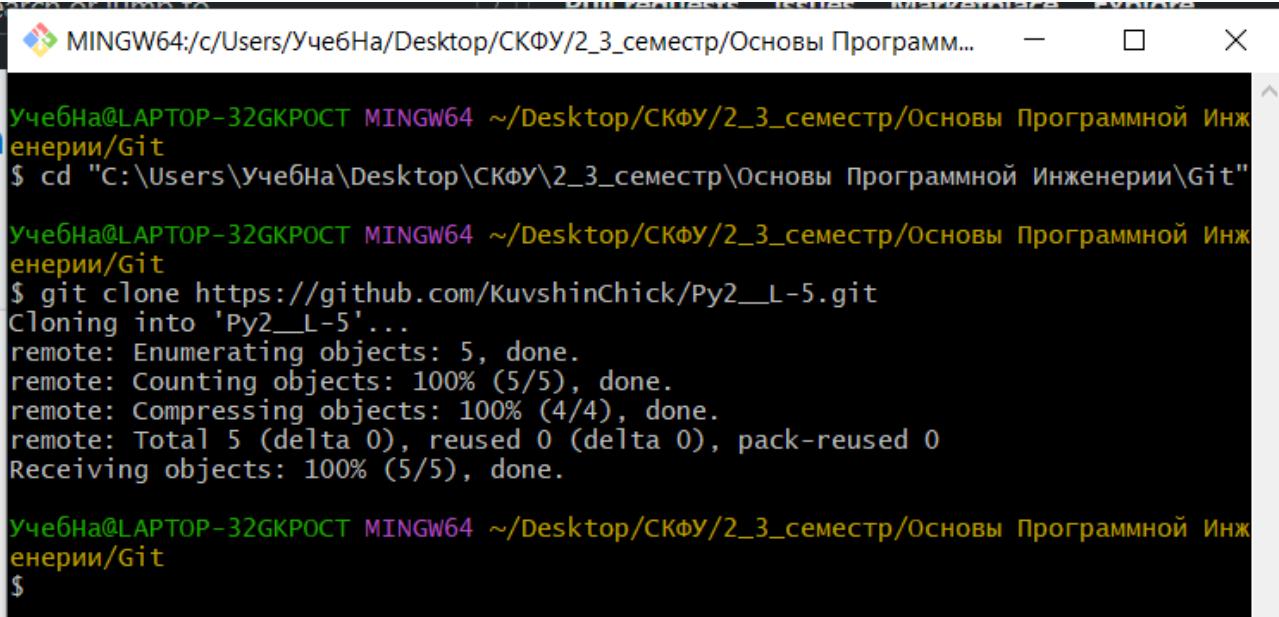
Ставрополь, 2022 г.

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.

Ссылка на репозиторий: https://github.com/KuvshinChick/Py2__L-5.git

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.

A screenshot of a Windows terminal window with a black background and white text. The window title is "MINGW64: c:/Users/Учебна/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Py2__L-5.git". The terminal shows the following commands and output:

```
Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd "C:\Users\Учебна\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git"

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git clone https://github.com/KuvshinChick/Py2__L-5.git
Cloning into 'Py2__L-5'...
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
Receiving objects: 100% (5/5), done.

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$
```

Рисунок 5.5.1 – клонирование репозитория

4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py2__L-5 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py2__L-5 (develop)
$ git branch
* develop
  main

```

Рисунок 5.5.2 – Создание ветки develop

```

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py2__L-5 (develop)
$ git add .gitignore

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py2__L-5 (develop)
$ git commit -m "modified .gitignore"
[develop 3bca29d] modified .gitignore
1 file changed, 2 insertions(+), 2 deletions(-)

Учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py2__L-5 (develop)
$ git log
commit 3bca29dfdda4b2c557f0210049e3b68e76145d3a (HEAD -> develop)
Author: KuvshinChick <kuvshinirina75@gmail.com>
Date: Fri Oct 28 21:03:39 2022 +0300

    modified .gitignore

commit 1d77b6b153ca6205b7cef0924f4a19bc50d02469 (origin/main, origin/HEAD, main)
Author: KuvshinChick <112752849+KuvshinChick@users.noreply.github.com>
Date: Fri Oct 28 20:19:31 2022 +0300

    Initial commit

```

Рисунок 5.3 – Обновление .gitignore

6. Самостоятельно изучите рекомендации к оформлению исходного кода на языке Python PEP-8 (<https://pep8.org/>). Выполните оформление исходного примеров лабораторной работы и индивидуальных созданий в соответствии с PEP-8.

7. Создайте проект PyCharm в папке репозитория.

8. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

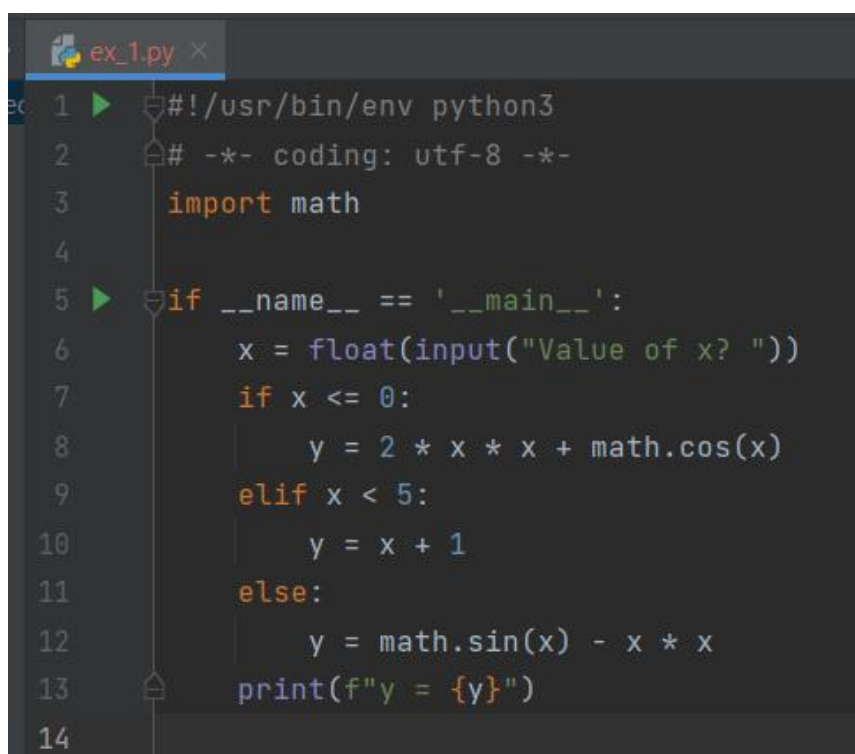
9. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры.

10. Для примеров 4 и 5 постройте UML-диаграмму деятельности. Для построения диаграмм деятельности использовать веб-сервис Google <https://www.diagrams.net/>.

Пример 1.

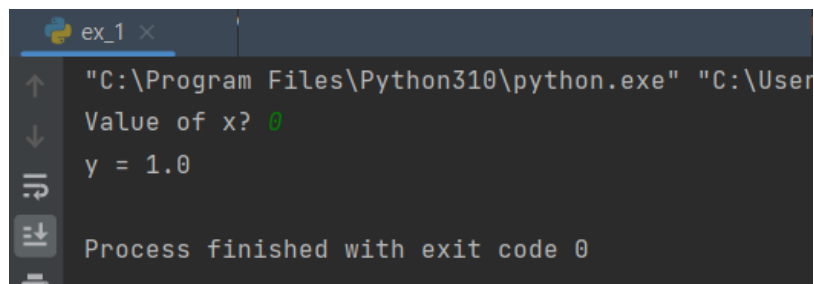
Составить UML-диаграмму деятельности и программу с использованием конструкции ветвления и вычислить значение функции.

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



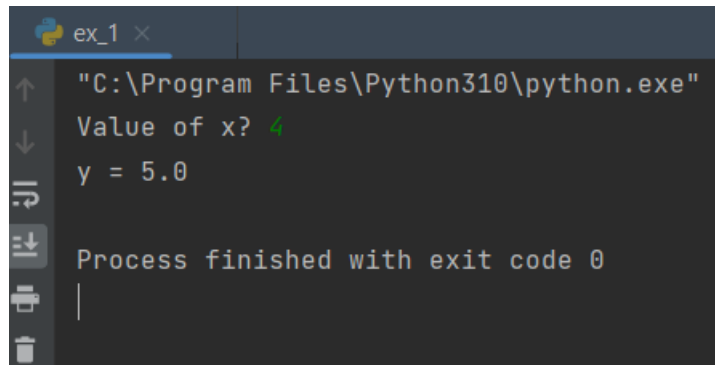
```
ex_1.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4
5  if __name__ == '__main__':
6      x = float(input("Value of x? "))
7      if x <= 0:
8          y = 2 * x * x + math.cos(x)
9      elif x < 5:
10         y = x + 1
11      else:
12         y = math.sin(x) - x * x
13         print(f"y = {y}")
14
```

Рисунок 5.3 – Код программы-примера



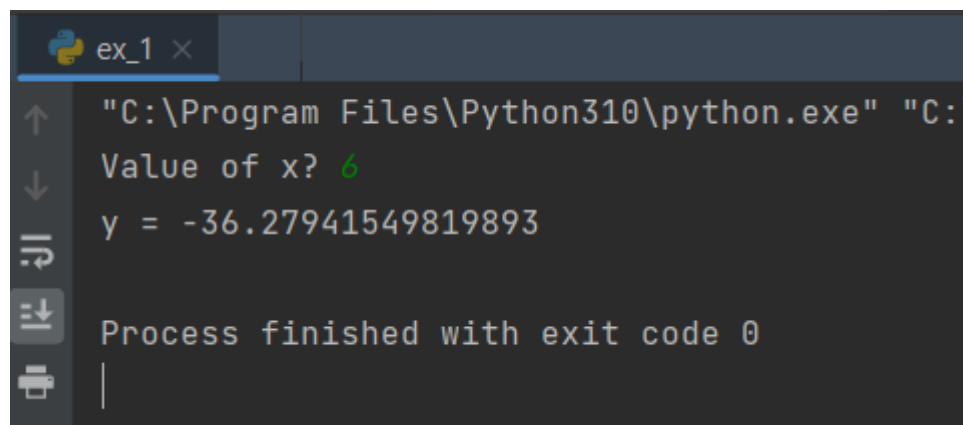
```
ex_1 x
"C:\Program Files\Python310\python.exe" "C:\User
Value of x? 0
y = 1.0
Process finished with exit code 0
```

Рисунок 5.4 – Результат программы при $x \leq 0$



```
"C:\Program Files\Python310\python.exe"
Value of x? 4
y = 5.0
Process finished with exit code 0
```

Рисунок 5.5 – Результат программы при $0 < x < 5$

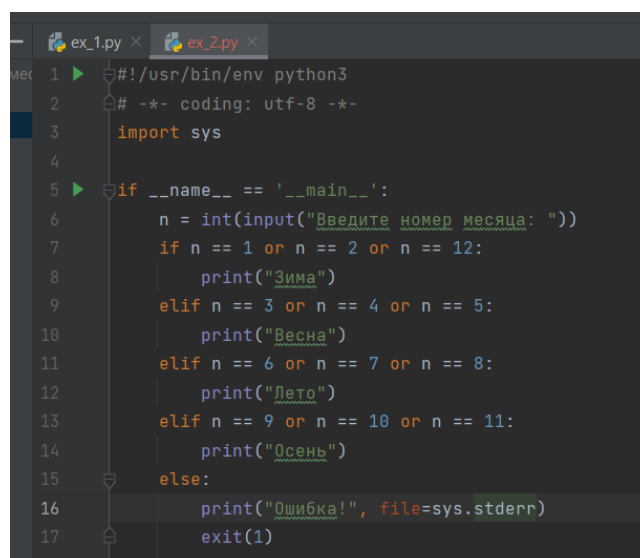


```
"C:\Program Files\Python310\python.exe" "C:\
Value of x? 6
y = -36.27941549819893
Process finished with exit code 0
```

Рисунок 5.6 – Результат программы при $x \geq 5$

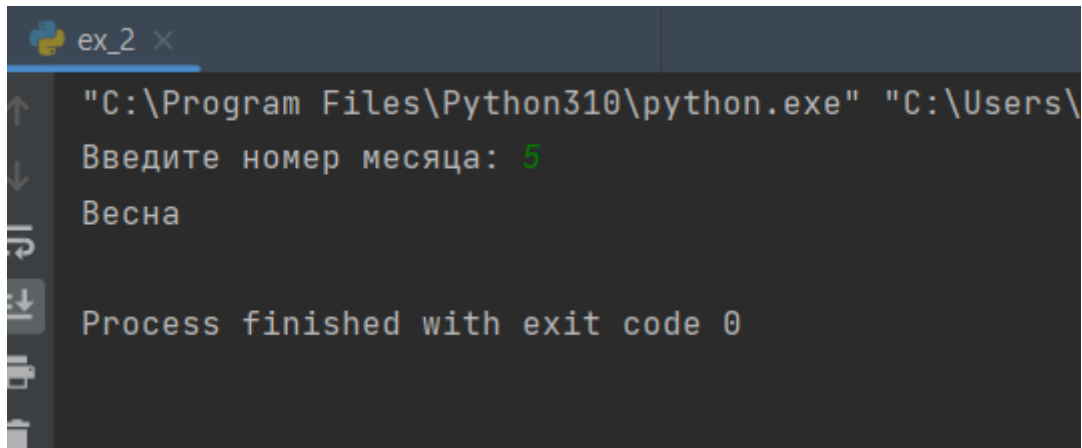
Пример 2.

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.



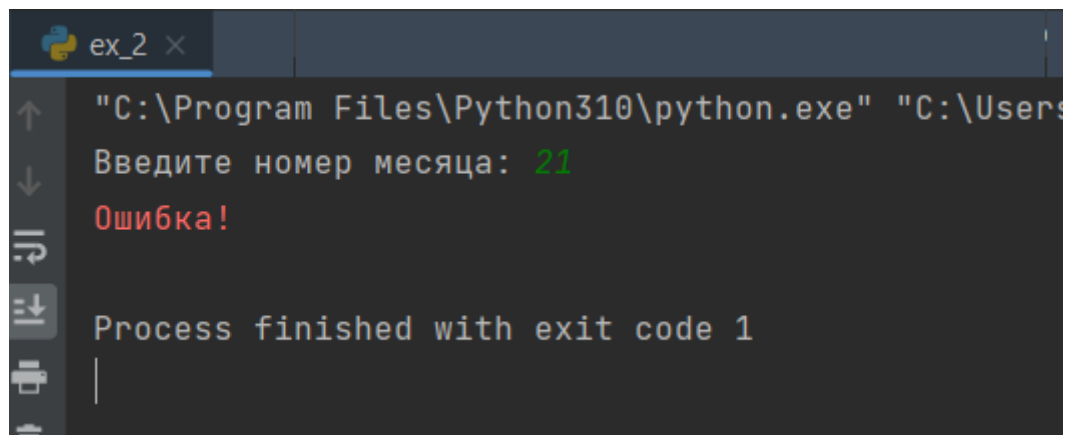
```
ex_1.py x ex_2.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 import sys
4
5 if __name__ == '__main__':
6     n = int(input("Введите номер месяца: "))
7     if n == 1 or n == 2 or n == 12:
8         print("Зима")
9     elif n == 3 or n == 4 or n == 5:
10        print("Весна")
11    elif 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)
```

Рисунок 5.7 – Код программы-примера



```
ex_2 x
"C:\Program Files\Python310\python.exe" "C:\Users\
Введите номер месяца: 5
Весна
Process finished with exit code 0
```

Рисунок 5.8 – Результат программы при n = 5



```
ex_2 x
"C:\Program Files\Python310\python.exe" "C:\Users\
Введите номер месяца: 21
Ошибка!
Process finished with exit code 1
|
```

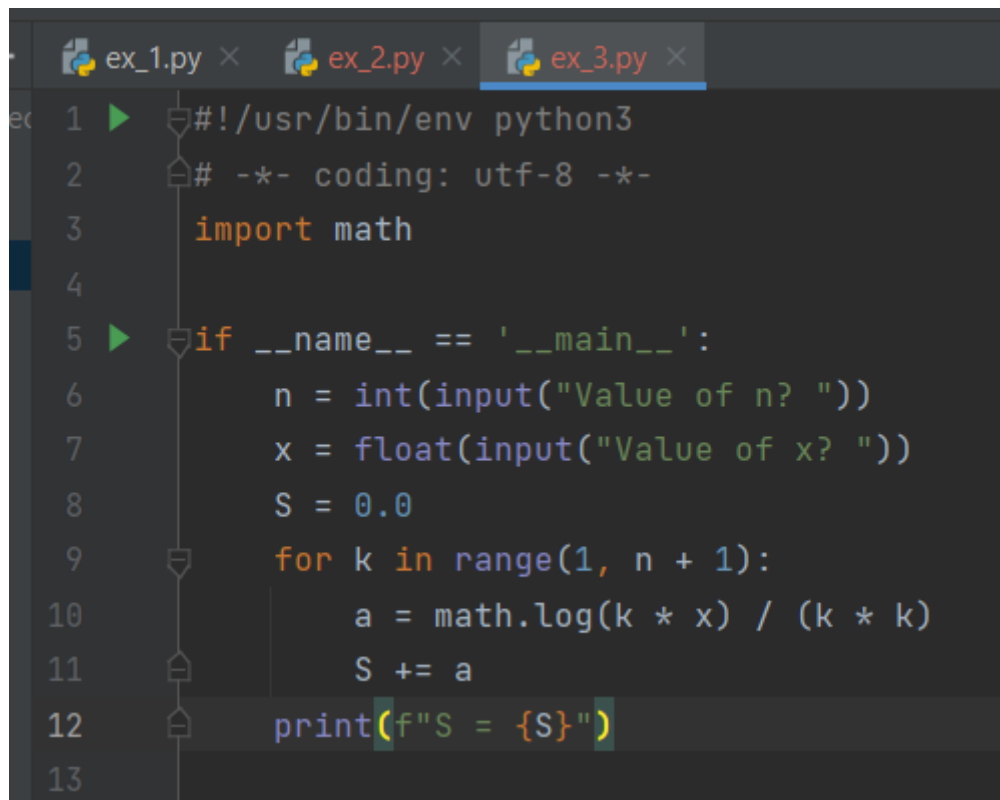
Рисунок 5.9 – Результат программы при n = 21 (ошибка)

Пример 3.

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

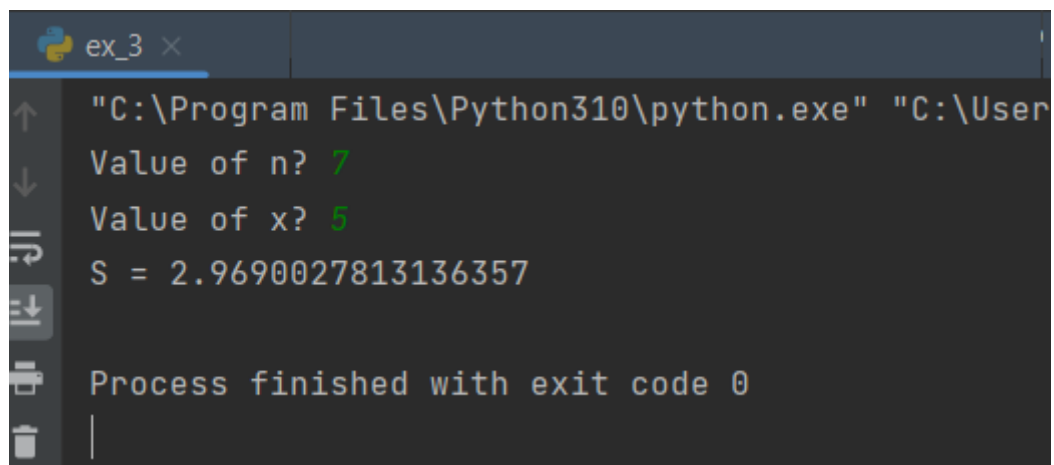
$$S = \sum_{k=1}^n \frac{\ln kx}{k^2},$$

где n и k вводятся с клавиатуры.



```
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}")
13
```

Рисунок 5.10 – Код программы-примера



```
ex_3 x
"C:\Program Files\Python310\python.exe" "C:\User
Value of n? 7
Value of x? 5
S = 2.9690027813136357
Process finished with exit code 0
```

Рисунок 5.11 – Результат программы при $n = 7$, $x = 5$

Пример 4.

Найти значение квадратного корня из положительного числа вводимого с клавиатуры, с некоторой заданной точностью с помощью рекуррентного соотношения:

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right).$$

В качестве начального значения примем $x_0 = 1$. Цикл должен выполняться до тех пор, пока не будет выполнено условие $|x_{n+1} - x_n| \leq \varepsilon$. Сравните со значением квадратного корня, полученным с использованием функций стандартной библиотеки. Значение $\varepsilon = 10^{-10}$.

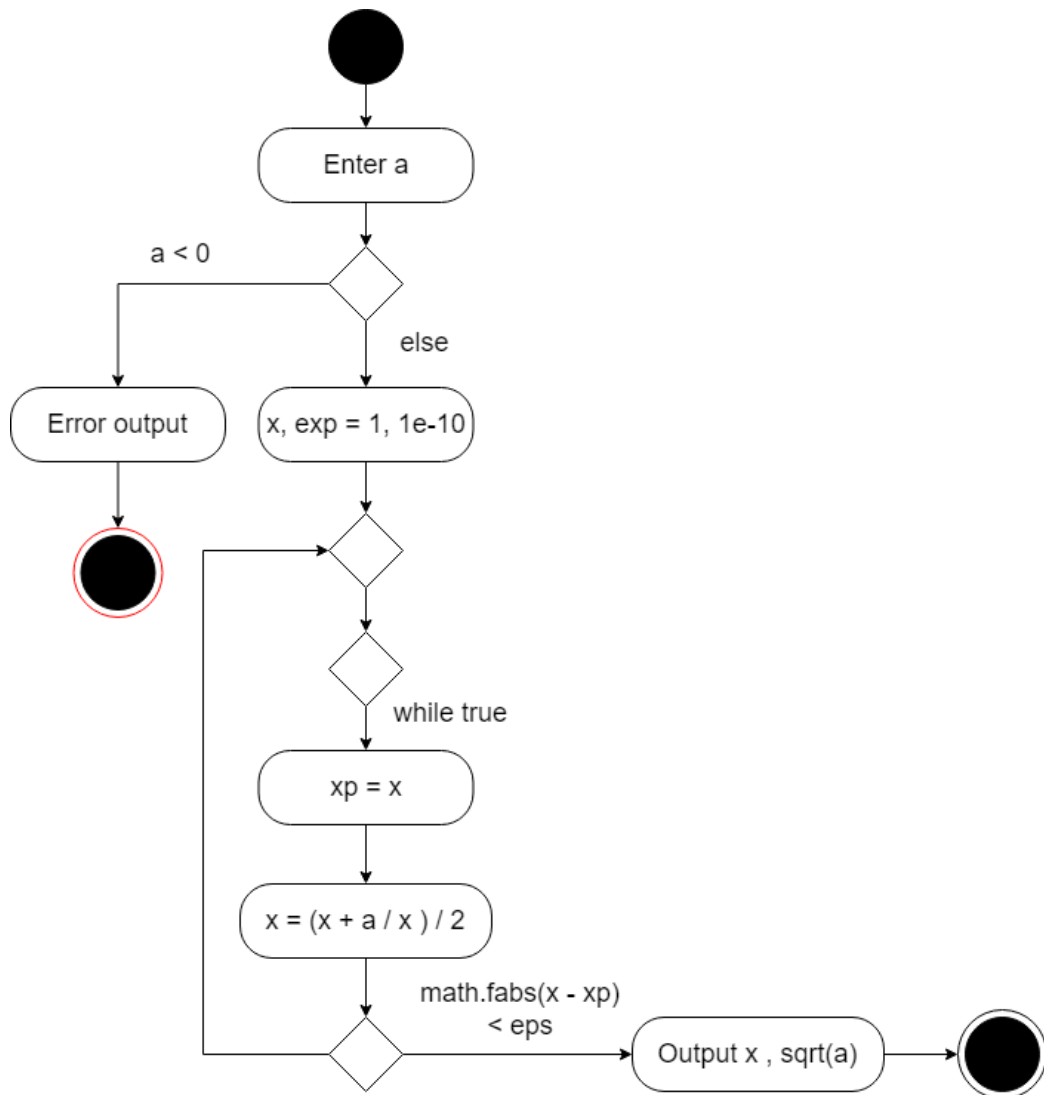


Рисунок 5.12 – UML диаграмма примера


```

1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3      import math
4      import sys
5      |
6      ▶  if __name__ == '__main__':
7          a = float(input("Value of a? "))
8          if a < 0:
9              print("Illegal value of a", file=sys.stderr)
10             exit(1)
11
12         x, eps = 1, 1e-10
13         while True:
14             xp = x
15             x = (x + a / x) / 2
16             if math.fabs(x - xp) < eps:
17                 break
18
19         print(f"x = {x}\nX = {math.sqrt(a)}")
20

```

Рисунок 5.13 – Код программы-примера

```

ex_4 x
↑ "C:\Program Files\Python310\python.exe" "C:\U
↓ Value of a? 8
| x = 2.82842712474619
| X = 2.8284271247461903
|⋮
| Process finished with exit code 0
|
|

```

Рисунок 5.14 – Результат программы при $a = 8$

```

in: ex_4 x
↑ "C:\Program Files\Python310\python.exe" "C:\Users\
↓ Value of a? -2
| Illegal value of a
|⋮
| Process finished with exit code 1
|
|

```

Рисунок 5.15 – Результат программы при $a = -2$ (ошибка)

Пример 5.

Вычислить значение специальной (интегральной показательной) функции

$$\text{Ei}(x) = \int_{-\infty}^x \frac{\exp t}{t} dt = \gamma + \ln x + \sum_{k=1}^{\infty} \frac{x^k}{k \cdot k!},$$

где $\gamma = 0.5772156649 \dots$ - постоянная Эйлера, по ее разложению в ряд с точностью $\varepsilon = 10^{-10}$, аргумент x вводится с клавиатуры.

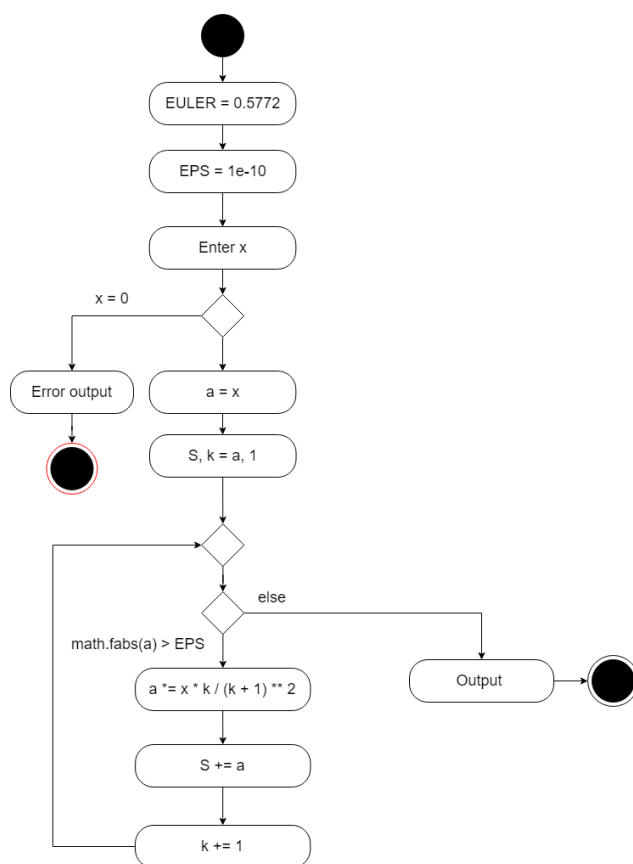


Рисунок 5.16 – UML диаграмма примера

```

ex_1.py x ex_2.py x ex_3.py x ex_4.py x ex_5.py x
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
16         a = x
17         S, k = a, 1
18
19         # Найти сумму членов ряда.
20         while math.fabs(a) > EPS:
21             a *= x * k / (k + 1) ** 2
22             S += a
23             k += 1
24
25         # Вывести значение функции.
26         print(f"Ei({x}) = {EULER + math.log(math.fabs(x)) + S}")
27

```

Рисунок 5.17 – Код программы-примера

```

ex_5 x
"C:\Program Files\Python310\python.exe"
Value of x? 13
Ei(13.0) = 37197.68849068905
Process finished with exit code 0

```

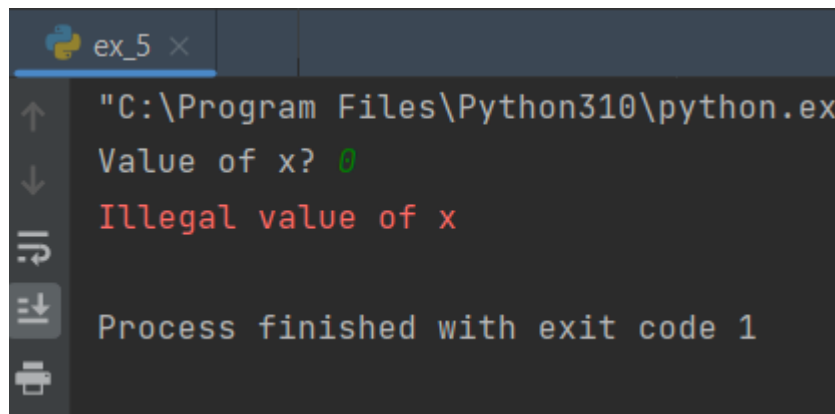
Рисунок 5.18 – Результат программы при $x = 13$

```

ex_5 x
"C:\Program Files\Python310\python.exe" "C
Value of x? -3
Ei(-3.0) = -0.013048381085575267
Process finished with exit code 0

```

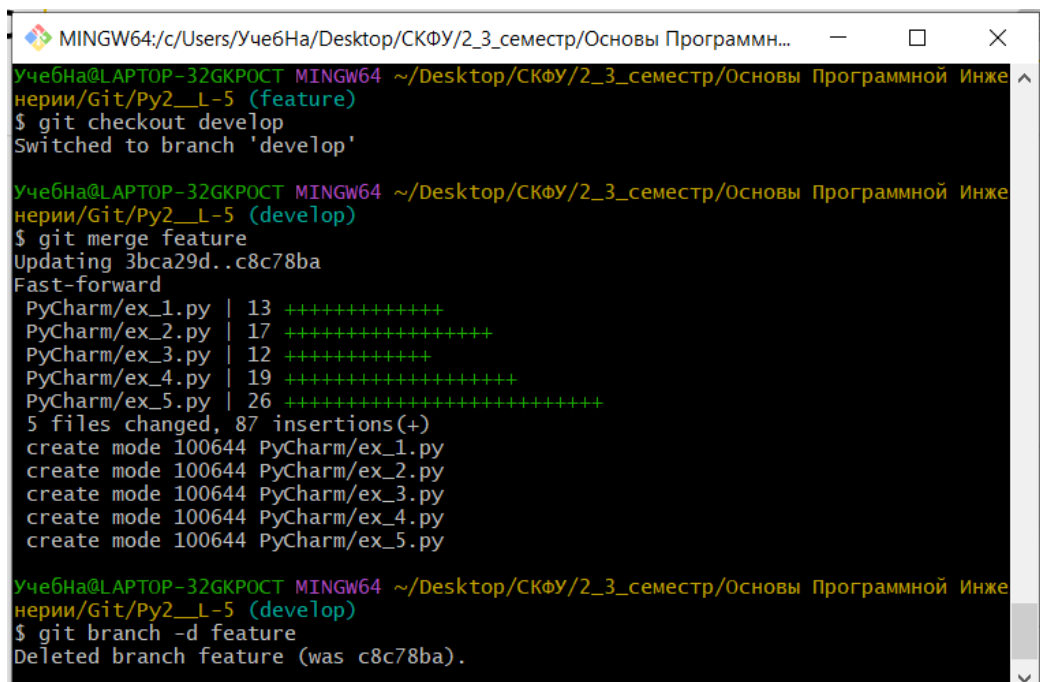
Рисунок 5.19 – Результат программы при $x = -3$



A screenshot of a Python console window titled 'ex_5'. The window shows the execution of a Python script. The first line is the command prompt 'C:\Program Files\Python310\python.exe'. The second line is the input 'Value of x? 0'. The third line is the output 'Illegal value of x' in red text. The fourth line is the output 'Process finished with exit code 1'.

```
ex_5 x
"C:\Program Files\Python310\python.exe"
Value of x? 0
Illegal value of x
Process finished with exit code 1
```

Рисунок 5.20 – Результат программы при $a = 0$ (ошибка)




A screenshot of a terminal window showing Git commands and their output. The window title is 'MINGW64; c:\Users\УчебНа\Desktop\СКФУ\2_3_семестр\Основы Программн...'. The output shows the user switching to the 'develop' branch, merging the 'feature' branch, and then deleting the 'feature' branch.

```
MINGW64; c:\Users\УчебНа\Desktop\СКФУ\2_3_семестр\Основы Программн...
УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py2__L-5 (feature)
$ git checkout develop
Switched to branch 'develop'

УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py2__L-5 (develop)
$ git merge feature
Updating 3bca29d..c8c78ba
Fast-forward
 PyCharm/ex_1.py | 13 ++++++++
 PyCharm/ex_2.py | 17 ++++++++
 PyCharm/ex_3.py | 12 ++++++++
 PyCharm/ex_4.py | 19 ++++++++
 PyCharm/ex_5.py | 26 ++++++++
 5 files changed, 87 insertions(+)
 create mode 100644 PyCharm/ex_1.py
 create mode 100644 PyCharm/ex_2.py
 create mode 100644 PyCharm/ex_3.py
 create mode 100644 PyCharm/ex_4.py
 create mode 100644 PyCharm/ex_5.py

УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py2__L-5 (develop)
$ git branch -d feature
Deleted branch feature (was c8c78ba).
```

Рисунок 5.21 – Merge ветки feature в develop



A screenshot of a terminal window showing Git commands and their output. The window title is 'УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже'. The output shows the user pushing the 'develop' branch to the origin and creating a pull request on GitHub.

```
УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инже
нерии/Git/Py2__L-5 (develop)
$ git push origin develop
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (14/14), 2.32 KiB | 1.16 MiB/s, done.
Total 14 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/KuvshinChick/Py2__L-5/pull/new/develop
remote:
To https://github.com/KuvshinChick/Py2__L-5.git
 * [new branch]      develop -> develop
```

Рисунок 5.22 – Отправка ветки develop в удаленный реп

11. Выполните индивидуальные задания, согласно своего варианта. Для заданий повышенной сложности номер варианта должен быть получен у преподавателя.

12. Приведите в отчете скриншоты работы программ и UML-диаграммы деятельности решения индивидуальных заданий.

Индивидуальные задания: (Вариант 15)

Задание 1.

2. Дано число m ($1 \leq m \leq 12$). Определить, сколько дней в месяце с номером m .

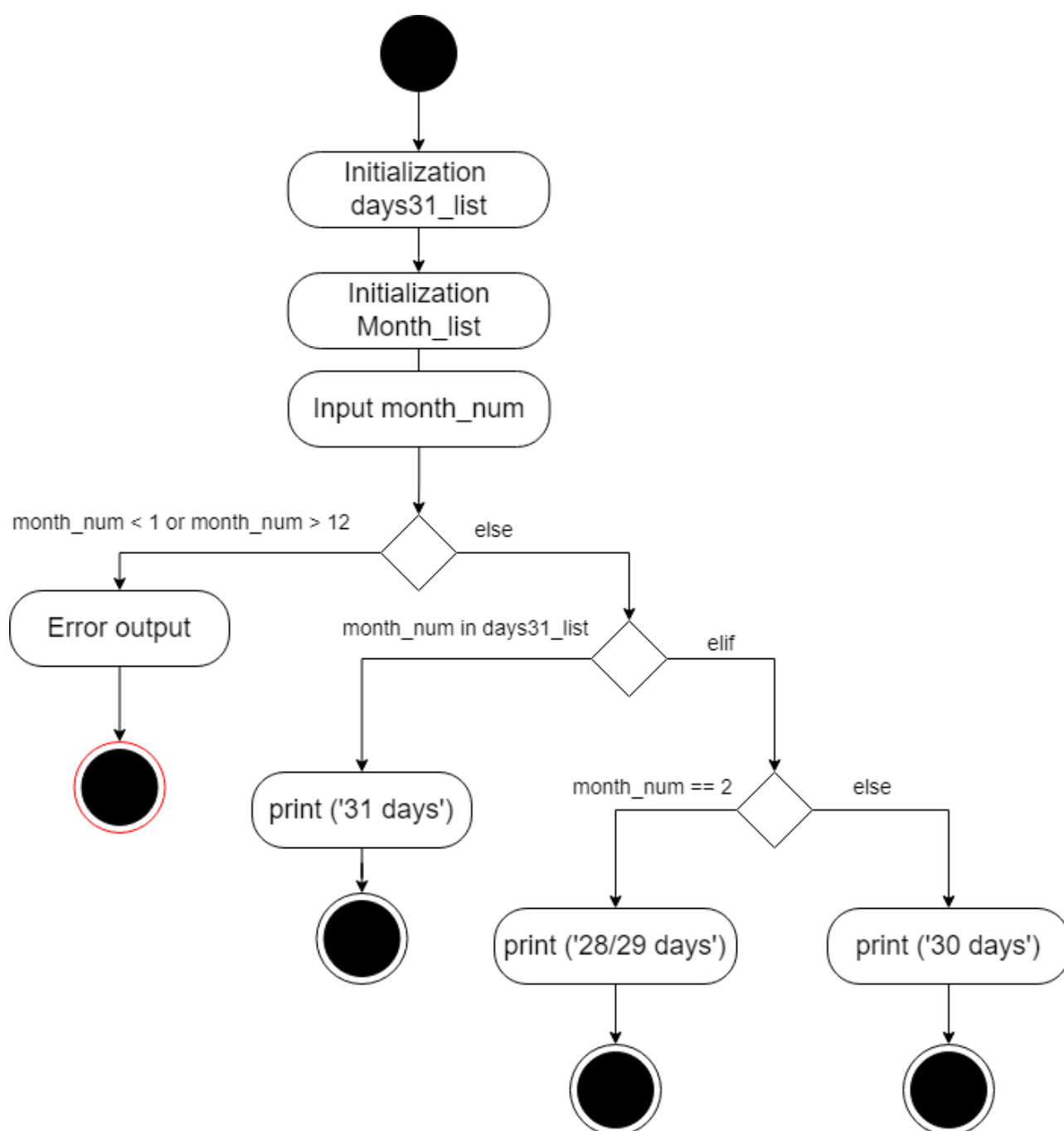
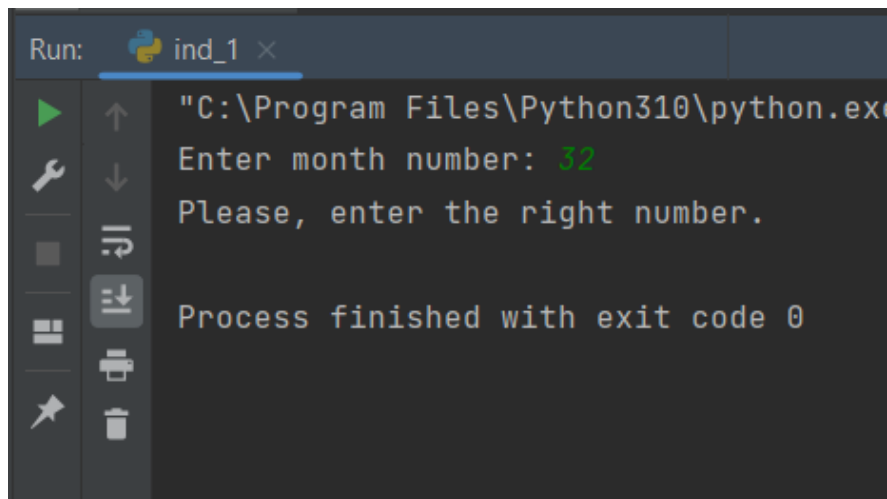


Рисунок 5.23 – UML диаграмма задачи

```
ex_1.py × ex_2.py × ex_3.py × ex_4.py × ex_5.py × ind_1.py × ind_2.py × ind_3.py × ind_up.py ×
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == "__main__":
5      # List of months & 31_days_list
6      days31_list = [1, 3, 5, 7, 8, 10, 12]
7      month_list = [
8          'January',
9          'February',
10         'March',
11         'April',
12         'May',
13         'June',
14         'July',
15         'August',
16         'September',
17         'October',
18         'November',
19         'December'
20     ]
21
22     # Input number of month
23     month_num = int(input("Enter month number: "))
24
25     # Output from the array check
26     if month_num < 1 or month_num > 12:
27         print("Please, enter the right number.")
28     else:
29         # Month output
30         if month_num in days31_list:
31             print(f'{month_list[month_num-1]}: 31 days')
```

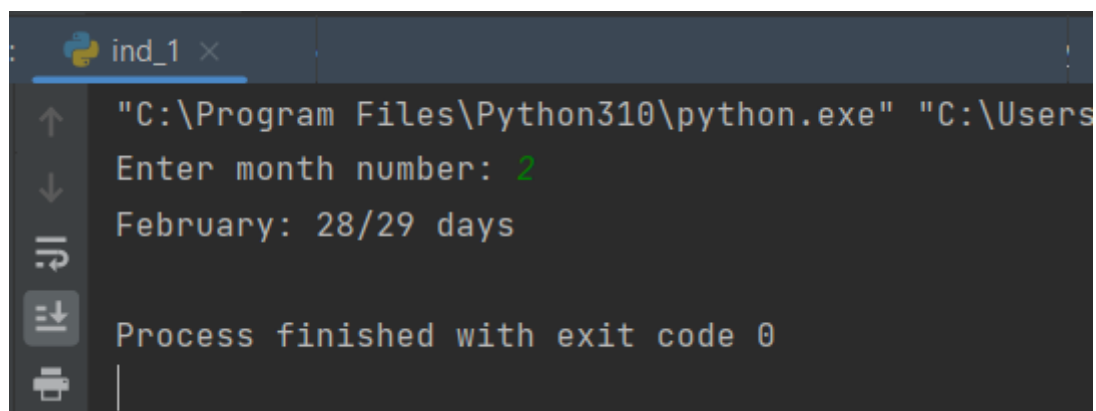
```
# Output from the array check
if month_num < 1 or month_num > 12:
    print("Please, enter the right number.")
else:
    # Month output
    if month_num in days31_list:
        print(f'{month_list[month_num-1]}: 31 days')
    elif month_num == 2:
        print(f'{month_list[month_num - 1]}: 28/29 days')
    else:
        print(f'{month_list[month_num - 1]}: 30 days')
```

Рисунок 5.24 – Код программы



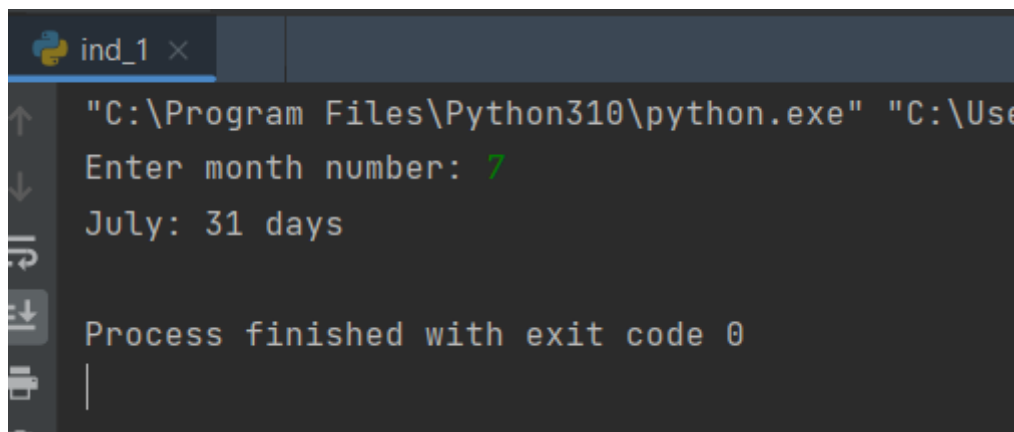
```
Run: ind_1 x
"C:\Program Files\Python310\python.exe"
Enter month number: 32
Please, enter the right number.
Process finished with exit code 0
```

Рисунок 5.25 – Результат программы при month_num = 32 (ошибка)



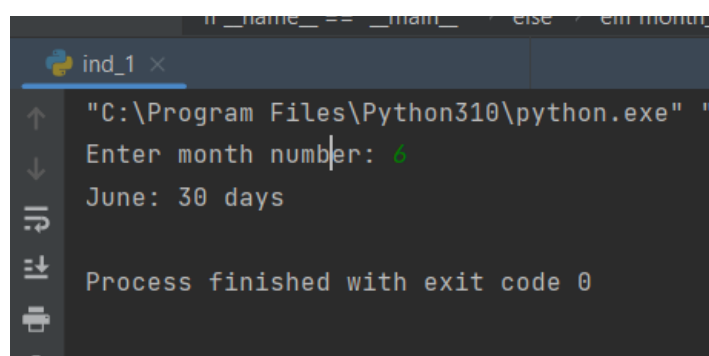
```
ind_1 x
"C:\Program Files\Python310\python.exe" "C:\Users
Enter month number: 2
February: 28/29 days
Process finished with exit code 0
```

Рисунок 5.26 – Результат программы при month_num = 2



```
ind_1 x
"C:\Program Files\Python310\python.exe" "C:\Use
Enter month number: 7
July: 31 days
Process finished with exit code 0
```

Рисунок 5.27 – Результат программы при month_num = 7



```
ind_1 x
"C:\Program Files\Python310\python.exe" "
Enter month number: 6
June: 30 days
Process finished with exit code 0
```

Рисунок 5.28 – Результат программы при month_num = 6

Задание 2.

15. Составить программу решения квадратного уравнения. Выводить также комплексные решения.

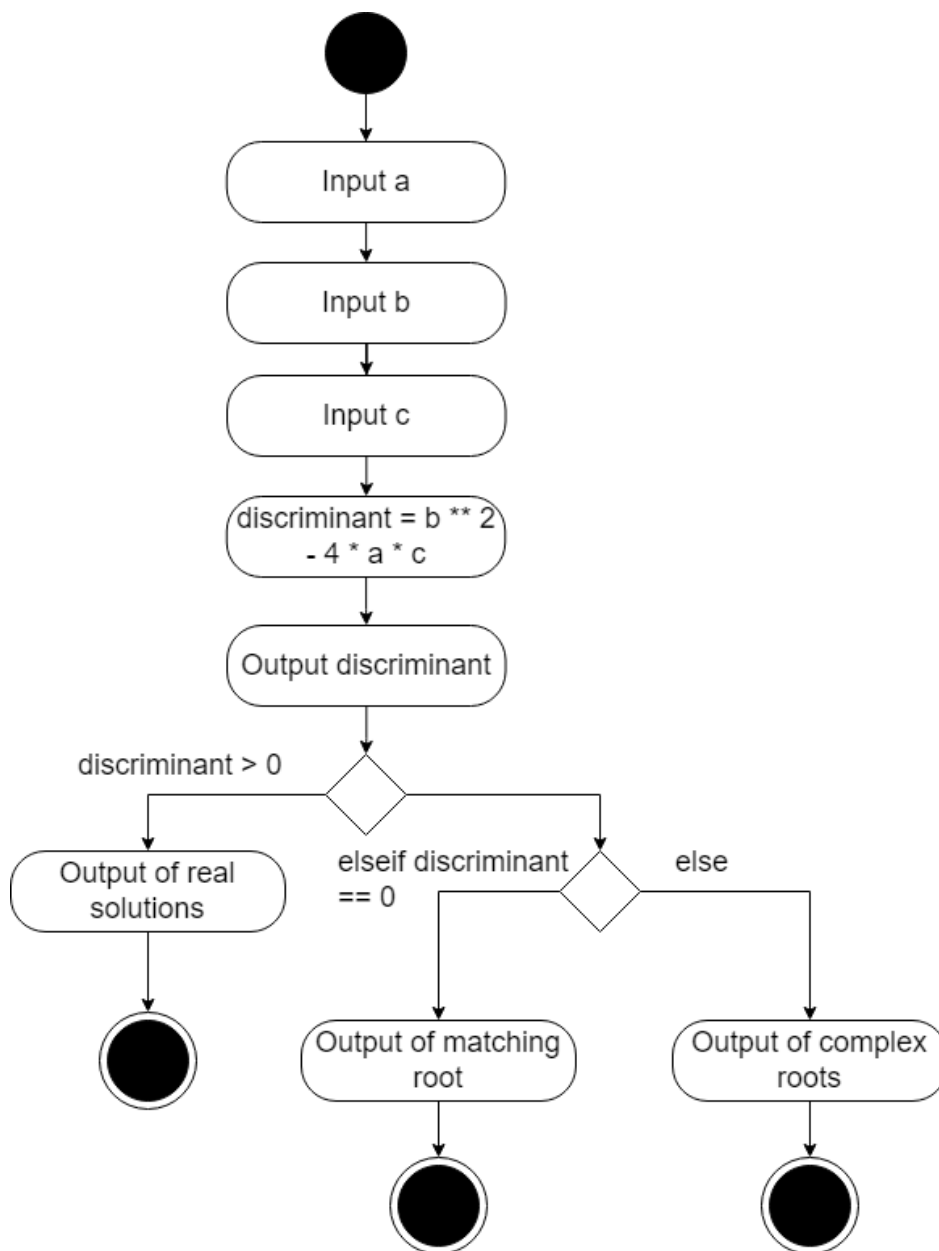


Рисунок 5.29 – UML диаграмма задачи


```
ex_1.py x ex_2.py x ex_3.py x ex_4.py x ex_5.py x ind_1.py x ind_2.py x ind_3.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 # import math module
5 import math
6
7 if __name__ == "__main__":
8     # coefficients input
9     print("Enter the coefficients for the equation")
10    print("ax^2 + bx + c = 0:")
11    a = float(input('Enter a: '))
12    b = float(input('Enter b: '))
13    c = float(input('Enter c: '))
14
15    # calculate the discriminant
16    discriminant = b ** 2 - 4 * a * c
17    print("Discriminant = %.2f" % discriminant)
18
19    # solution of the equation
20    if discriminant > 0:
21        x1 = (-b + math.sqrt(discriminant)) / (2 * a)
22        x2 = (-b - math.sqrt(discriminant)) / (2 * a)
23        print("x1 = %.2f \nx2 = %.2f" % (x1, x2))
24    elif discriminant == 0:
25        x = -b / (2 * a)
26        print("x = %.2f" % x)
27    else:
28        print('x1 = ', round(- b /(2 * a), 2), ' + ', round(math.sqrt(abs(discriminant))/(2 * a), 2), 'i')
29        print('x2 = ', round(- b /(2 * a), 2), ' - ', round(math.sqrt(abs(discriminant))/(2 * a), 2), 'i')
```

Рисунок 5.30 – Код программы

```
ind_2 x
"C:\Program Files\Python310\python.exe" "C:\Users\УчебНа\Desktop\ind_2.py"
Enter the coefficients for the equation
ax^2 + bx + c = 0:
Enter a: 2
Enter b: 5
Enter c: 2
Discriminant = 9.00
x1 = -0.50
x2 = -2.00
Process finished with exit code 0
```

Рисунок 5.31 – Результат программы при вещественных корнях

```
ind_2 x
"C:\Program Files\Python310\python.exe" "C:\Users\УчебНа\Desktop\ind_2.py"
Enter the coefficients for the equation
ax^2 + bx + c = 0:
Enter a: 1
Enter b: -6
Enter c: 9
Discriminant = 0.00
x = 3.00
Process finished with exit code 0
```

Рисунок 5.32 – Результат программы при совпадающем корне

```
ind_2 x
"C:\Program Files\Python310\python.exe"
Enter the coefficients for the equation
ax^2 + bx + c = 0:
Enter a: 7
Enter b: 5
Enter c: 2
Discriminant = -31.00
x1 = -0.36 + 0.4 i
x2 = -0.36 - 0.4 i

Process finished with exit code 0
```

Рисунок 5.33 – Результат программы при комплексных корнях

Задание 3.

15. Вычислить сумму всех n -значных чисел, кратных k ($1 \leq n \leq 4$).

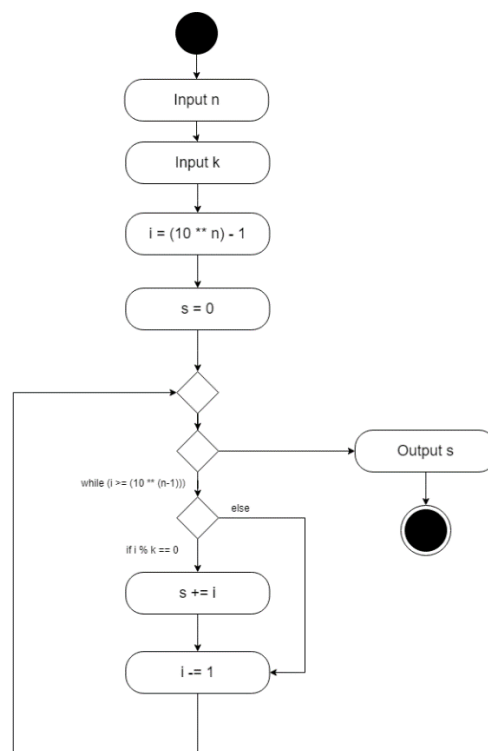


Рисунок 5.34 – UML диаграмма задачи

```

ex_1.py x ex_2.py x ex_3.py x ex_4.py x ex_5.py x ind_1.py x ind_2.py x ind_3.py x ind_up.py x
1  #!/usr/bin/env python3
2  #-*- coding: utf-8 -*-
3
4  if __name__ == "__main__":
5      # Input variables
6      n = int(input("Введите разряд чисел n: "))
7      k = int(input("Введите делитель k: "))
8      i = (10 ** n) - 1
9      s = 0
10     f = 10 ** (n-1)
11
12     # Divisibility check cycle
13     while i >= f:
14         if i % k == 0:
15             s += i
16             i -= 1
17     print("Сумма всех чисел кратных k:")
18     print(s)
19

```

Рисунок 5.35 – Код программы

```

Run: ind_3 x
"C:\Program Files\Python310\python.exe" "C:\
Введите разряд чисел n: 1
Введите делитель k: 2
Сумма всех чисел кратных k:
20
Process finished with exit code 0

```

Рисунок 5.36 – Результат программы при $n = 1$, $k = 2$

```

Run: ind_3 x
"C:\Program Files\Python310\python.exe" "C:\
Введите разряд чисел n: 2
Введите делитель k: 35
Сумма всех чисел кратных k:
105

```

Рисунок 5.37 – Результат программы при $n = 2$, $k = 35$

Задание повышенной сложности

Составить UML-диаграмму деятельности, программу и произвести вычисления вычисление значения специальной функции по ее разложению в ряд с точностью $\varepsilon = 10^{-10}$, аргумент функции x вводится с клавиатуры. Номер варианта необходимо получить у преподавателя.

6. Функция Бесселя первого рода $J_n(x)$, значение $n = 0, 1, 2, \dots$ также должно вводиться с клавиатуры

$$J_n(x) = \left(\frac{x}{2}\right)^n \sum_{k=0}^{\infty} \frac{(-x^2/4)^k}{k!(k+n)!}. \quad (16)$$

Handwritten derivation on green graph paper:

$$(k+n+1)! = (k+n+1) \cdot (k+n)!$$

$$\frac{(-x^2/4)^k}{(k+n)!} \cdot \frac{(-x^2/4)^{k+1}}{(k+n+1)!} = \frac{(-x^2/4)^k}{(k+n)!} \cdot \frac{(-x^2/4)^{k+1}}{(k+n+1) \cdot (k+n)!}$$

$$\frac{a_{k+1}}{a_k} = \frac{(-x^2/4)^{k+1}}{(k+n+1) \cdot (k+n)!} \cdot \frac{k! \cdot (k+n)!}{(-x^2/4)^k}$$

$$= \frac{-x^2/4}{(k+n+1)(k+1)}$$

$$a_1 = \frac{-x^2/4}{(1+n+1)(1+1)} = \frac{-x^2/4}{2(n+1)} = \frac{-x^2/4}{n+2}$$

Рисунок 5.38 – Работа с формулой

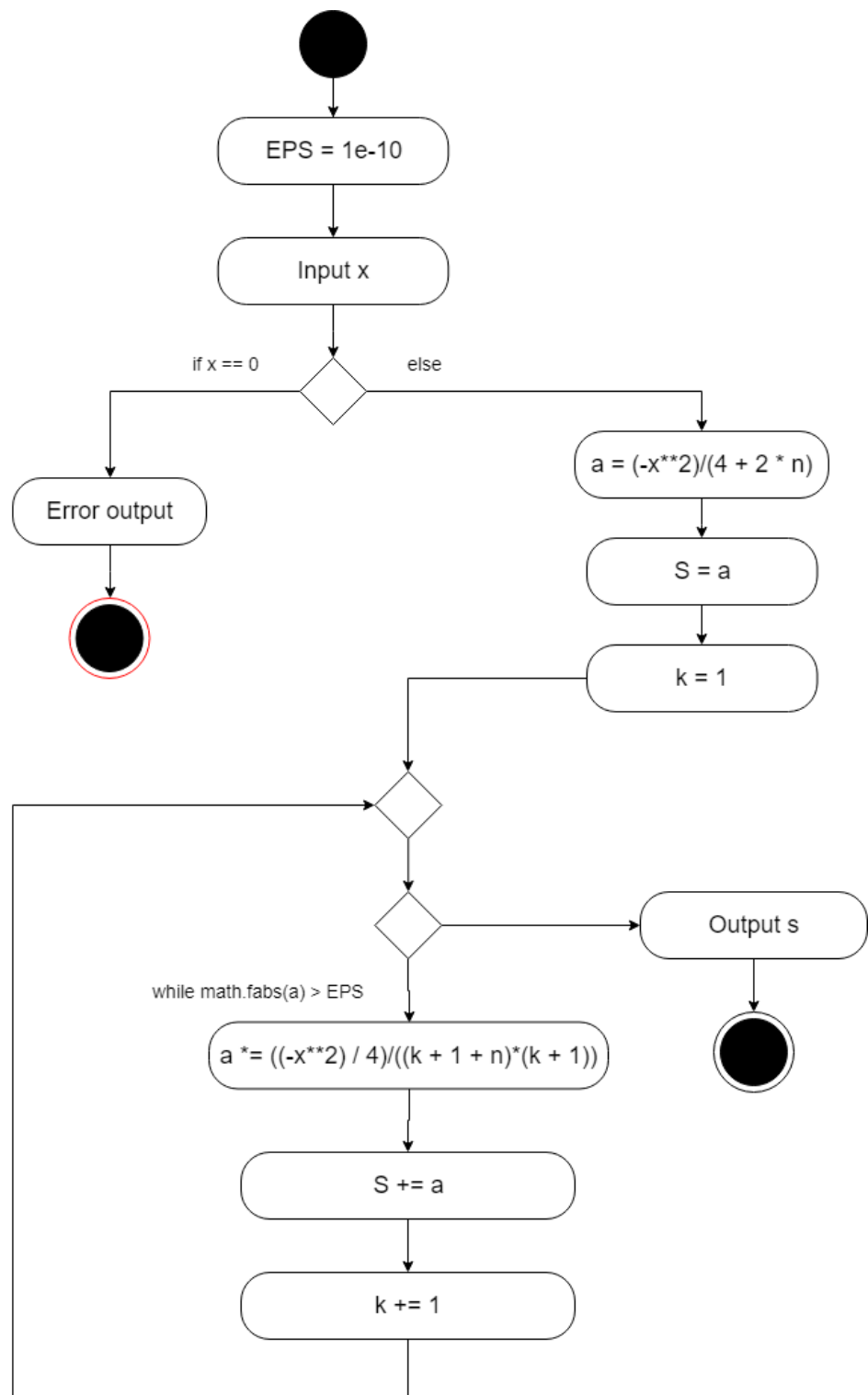


Рисунок 5.39 – UML диаграмма задачи

```
ex_1.py × ex_2.py × ex_3.py × ex_4.py × ex_5.py × ind_1.py × ind_2.py × ind_3.py × ind_up.py ×
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3  import math
4  import sys
5
6  # Точность вычислений.
7  EPS = 1e-10
8
9  if __name__ == '__main__':
10     x = float(input("Value of x? "))
11     n = float(input("Value of n? "))
12     if x == 0:
13         print("Illegal value of x", file=sys.stderr)
14         exit(1)
15
16     a = (-x**2)/(4 + 2 * n)
17     S = a
18     k = 1
19
20     # Найти сумму членов ряда.
21     while math.fabs(a) > EPS:
22         a *= ((-x**2) / 4)/((k + 1 + n)*(k + 1))
23         S += a
24         k += 1
25
26     # Вывести значение функции.
27     print("Sum: ", S)
28
```

Рисунок 5.40 – Код программы

```
ind_3 × ind_up ×
"C:\Program Files\Python310\python.exe"
Value of x? 5
Sum: 1.2681008071715392
Process finished with exit code 0
```

Рисунок 5.41 – Вывод программы при $x = 5$

13. Зафиксируйте сделанные изменения в репозитории.
14. Выполните слияние ветки для разработки с веткой main / master.
15. Отправьте сделанные изменения на сервер GitHub.
16. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

```
MINGW64:/c:/Users/УчебНа/Desktop/СКФУ/2_3_семестр/Основы Программ...
УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py2__L-5 (develop)
$ git commit -m "add individual task"
[develop 1623317] add individual task
1 file changed, 26 insertions(+)
create mode 100644 PyCharm/ind_up.py

УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py2__L-5 (develop)
$ git push origin develop
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (13/13), done.
Writing objects: 100% (13/13), 1.93 KiB | 1.93 MiB/s, done.
Total 13 (delta 7), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (7/7), completed with 2 local objects.
To https://github.com/KuvshinChick/Py2__L-5.git
d8151cd..1623317 develop -> develop
```

Рисунок 5.42 – Отправка ветки develop в удаленный реп

```
УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py2__L-5 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

УчебНа@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py2__L-5 (main)
$ git merge develop
Updating 1d77b6b..1623317
Fast-forward
 .gitignore | 4 ++--
 PyCharm/ex_1.py | 13 ++++++++
 PyCharm/ex_2.py | 17 ++++++++
 PyCharm/ex_3.py | 12 ++++++++
 PyCharm/ex_4.py | 19 ++++++++
 PyCharm/ex_5.py | 26 ++++++++
 PyCharm/ind_1.py | 24 ++++++++
 PyCharm/ind_2.py | 29 ++++++++
 PyCharm/ind_3.py | 17 ++++++++
 PyCharm/ind_up.py | 26 ++++++++
10 files changed, 185 insertions(+), 2 deletions(-)
create mode 100644 PyCharm/ex_1.py
create mode 100644 PyCharm/ex_2.py
create mode 100644 PyCharm/ex_3.py
create mode 100644 PyCharm/ex_4.py
create mode 100644 PyCharm/ex_5.py
create mode 100644 PyCharm/ind_1.py
create mode 100644 PyCharm/ind_2.py
create mode 100644 PyCharm/ind_3.py
create mode 100644 PyCharm/ind_up.py
```

Рисунок 5.21 – Merge ветки develop в main

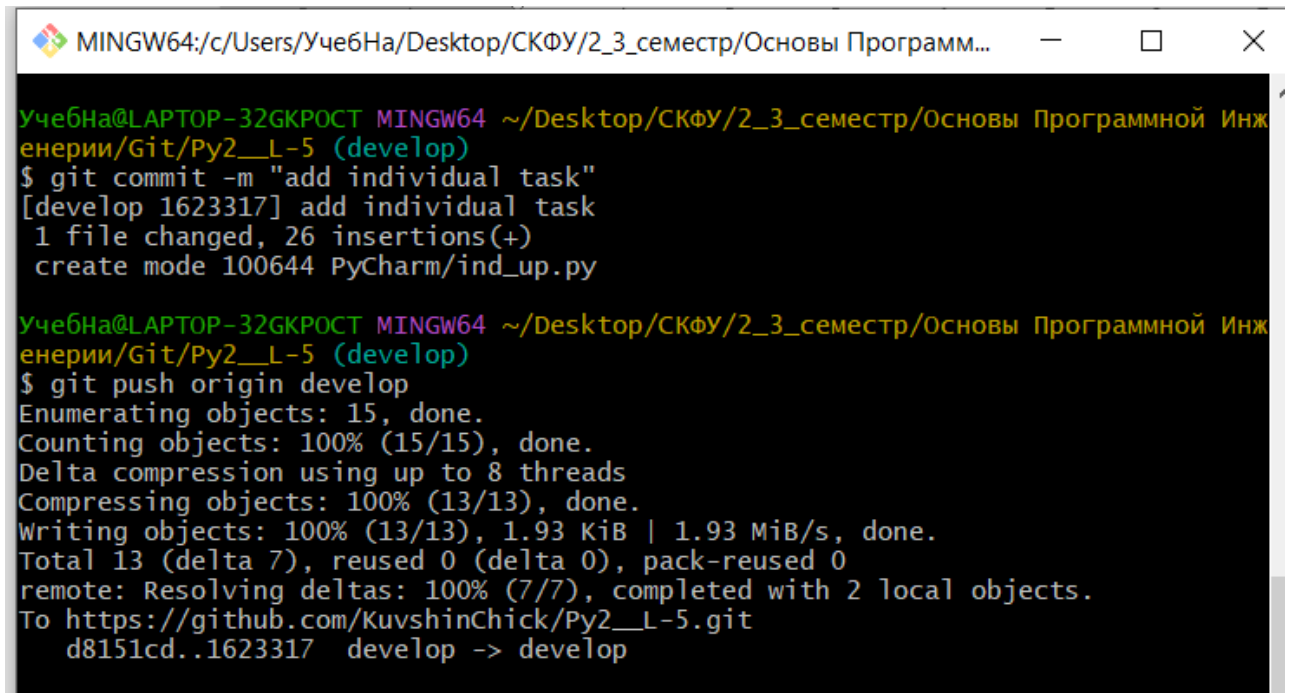
A screenshot of a terminal window with a black background and green and white text. The window title is 'MINGW64:/c:/Users/Учебна/Desktop/СКФУ/2_3_семестр/Основы Программн...'. The prompt is 'учебна@LAPTOP-32GKPOCT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py2__L-5 (develop)'. The first command is '\$ git commit -m "add individual task"', followed by its output: '[develop 1623317] add individual task', '1 file changed, 26 insertions(+)', and 'create mode 100644 PyCharm/ind_up.py'. The second command is '\$ git push origin develop', followed by its output: 'Enumerating objects: 15, done.', 'Counting objects: 100% (15/15), done.', 'Delta compression using up to 8 threads', 'Compressing objects: 100% (13/13), done.', 'Writing objects: 100% (13/13), 1.93 KiB | 1.93 MiB/s, done.', 'Total 13 (delta 7), reused 0 (delta 0), pack-reused 0', 'remote: Resolving deltas: 100% (7/7), completed with 2 local objects.', 'To https://github.com/KuvshinChick/Py2__L-5.git', and 'd8151cd..1623317 develop -> develop'.

Рисунок 5.42 – Отправка ветки main в удаленный реп

Вопросы для защиты работы

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?

Состояния действия не могут быть подвергнуты декомпозиции. Кроме того, они атомарны. Это значит, что внутри них могут происходить различные события, но выполняемая в состоянии действия работа не может быть прервана. Обычно предполагается, что длительность одного состояния действия занимает неощутимо малое время. 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 алгоритм выполняется последовательно, а алгоритм ветвления выполняется по разным алгоритмам/ветвям в зависимости от выполнения/невыполнения некоторого условия.

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

Условный оператор или оператор ветвления - это оператор, конструкция языка программирования, обеспечивающая выполнение определённой команды (набора команд) только при условии истинности некоторого логического выражения, либо выполнение одной из нескольких команд (наборов команд) в зависимости от значения некоторого выражения.

If, elif, else

7. Какие операторы сравнения используются в Python?

Оператор	<	«меньше»;
Оператор	<=	«меньше или равно»;
Оператор	==	«равно»;
Оператор	!=	«не равно»;
Оператор	>	«больше»;
Оператор	>=	«больше или равно».

8. Что называется простым условием? Приведите примеры.

Логические выражения типа `kByte >= 1023` являются простыми, так как в них выполняется только одна логическая операция.

9. Что такое составное условие? Приведите примеры

В составных условиях используется 2 и более логические операции.

`y < 15 and x > 8`

10. Какие логические операторы допускаются при составлении сложных условий?

Широко используются два оператора – так называемые логические И (and) и ИЛИ (or).

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

+

12. Какой алгоритм является алгоритмом циклической структуры?

Алгоритм циклической структуры - это алгоритм, в котором происходит многократное повторение одного и того же участка программы. Такие повторяемые участки вычислительного процесса называются циклами.

13. Типы циклов в языке Python.

Цикл «while» и цикл «for».

14. Назовите назначение и способы применения функции range .

Функция `range`

Функция `range` возвращает неизменяемую последовательность чисел в виде объекта *range*.

Синтаксис функции:

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

Параметры функции:

- **start** - с какого числа начинается последовательность. По умолчанию - 0
- **stop** - до какого числа продолжается последовательность чисел. Указанное число не включается в диапазон
- **step** - с каким шагом растут числа. По умолчанию 1

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

`range(0, 15, -2)`

16. Могут ли быть циклы вложенными?

+

Циклы называются вложенными (т. е. один цикл находится внутри другого), если внутри одного цикла во время каждой итерации необходимо выполнить другой цикл.

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

Пример бесконечного цикла.

```
a = 0
while a == 0:
    print("A")
```

Написать `a += 1` в теле цикла

18. Для чего нужен оператор `break` ?

Оператор `break` предназначен для досрочного прерывания работы цикла `while`.

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

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

Пример.

```
a = -1
while a < 10:
    a += 1
    if a >= 7:
        continue
    print("A")
```

При запуске данного кода символ "А" будет напечатан 7 раз, несмотря на то, что всего будет выполнено 11 проходов цикла.

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

В операционной системе по умолчанию присутствуют стандартных потоков вывода на консоль: буферизованный поток `stdout` для вывода данных и информационных сообщений, а также небуферизованный поток `stderr` для вывода сообщений об ошибках.

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

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.

```
print("Illegal value of a", file=sys.stderr)
```

22. Каково назначение функции `exit` ?

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