

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 РАБОТЕ №2.2
дисциплины «Основы кроссплатформенного программирования»
Вариант 3

Выполнил:
Баканов Артем Вадимович
1 курс, группа ИТС-6-0-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: основы ветвления Git.

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: <https://github.com/temacteklyannayapuwka/cross-platform-programming-v.2.2>

Порядок выполнения работы:

1. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT.

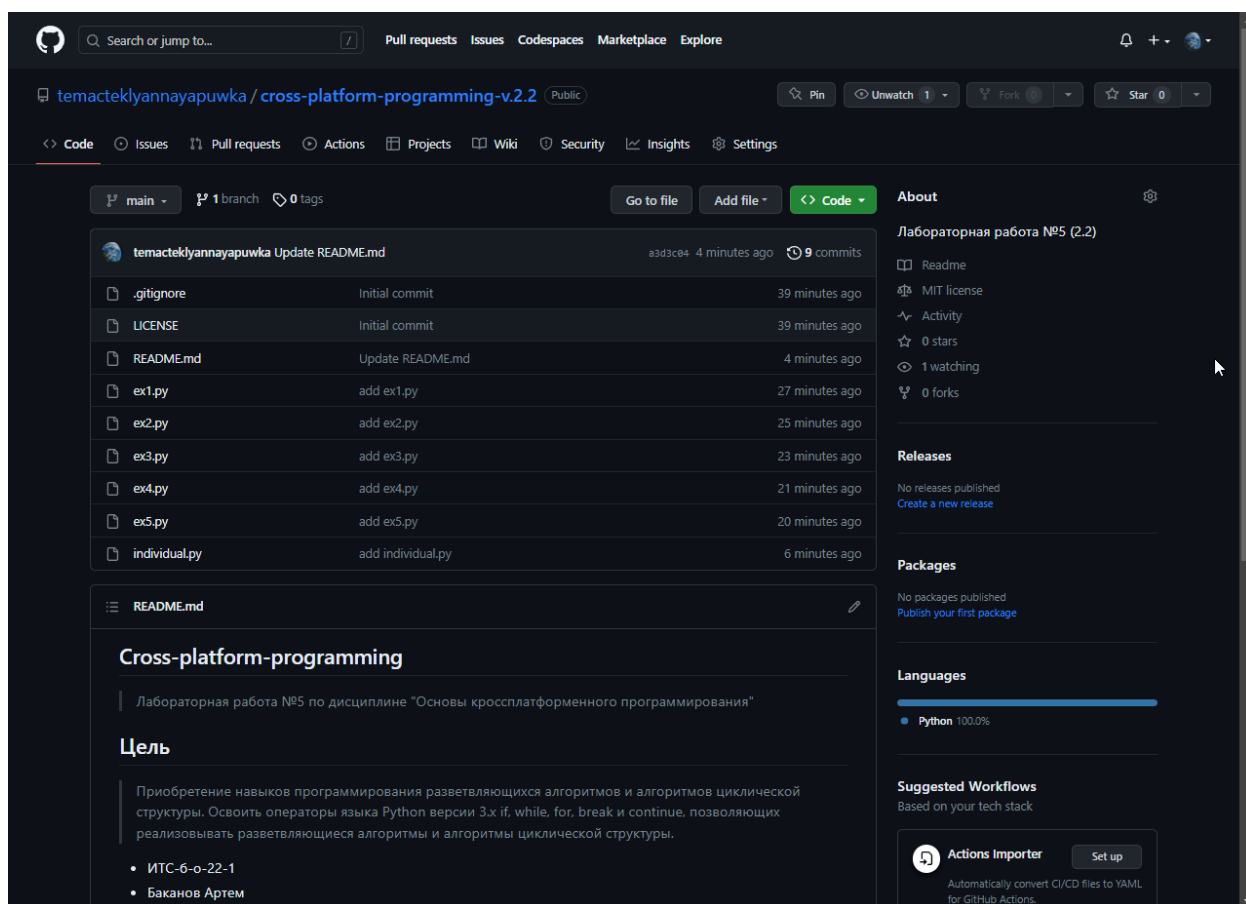


Рис. 1. Новый репозиторий.

2. Проклонировал свой репозиторий на свой компьютер.

```
PS C:\Users\Administrator> cd D:\Crosslabs
PS D:\Crosslabs> git clone https://github.com/temacteklyannayapuwka/cross-platform-programming-v.2.2.git
Cloning into 'cross-platform-programming-v.2.2'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
Receiving objects: 100% (8/8), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Resolving deltas: 100% (1/1), done.
PS D:\Crosslabs> cd cross-platform-programming-v.2.2
```

Рис. 2. Клон репозитория.

3. Использовал такую модель ветвления как git-flow.

```
PS D:\Crosslabs\cross-platform-programming-v.2.2> git flow init

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

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/Crosslabs/cross-platform-programming-v.2.2/.git/hooks]
PS D:\Crosslabs\cross-platform-programming-v.2.2> git branch
* develop
main
```

Рис. 3. Модель ветвления git-flow.

4. Работа с примером 1.

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

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

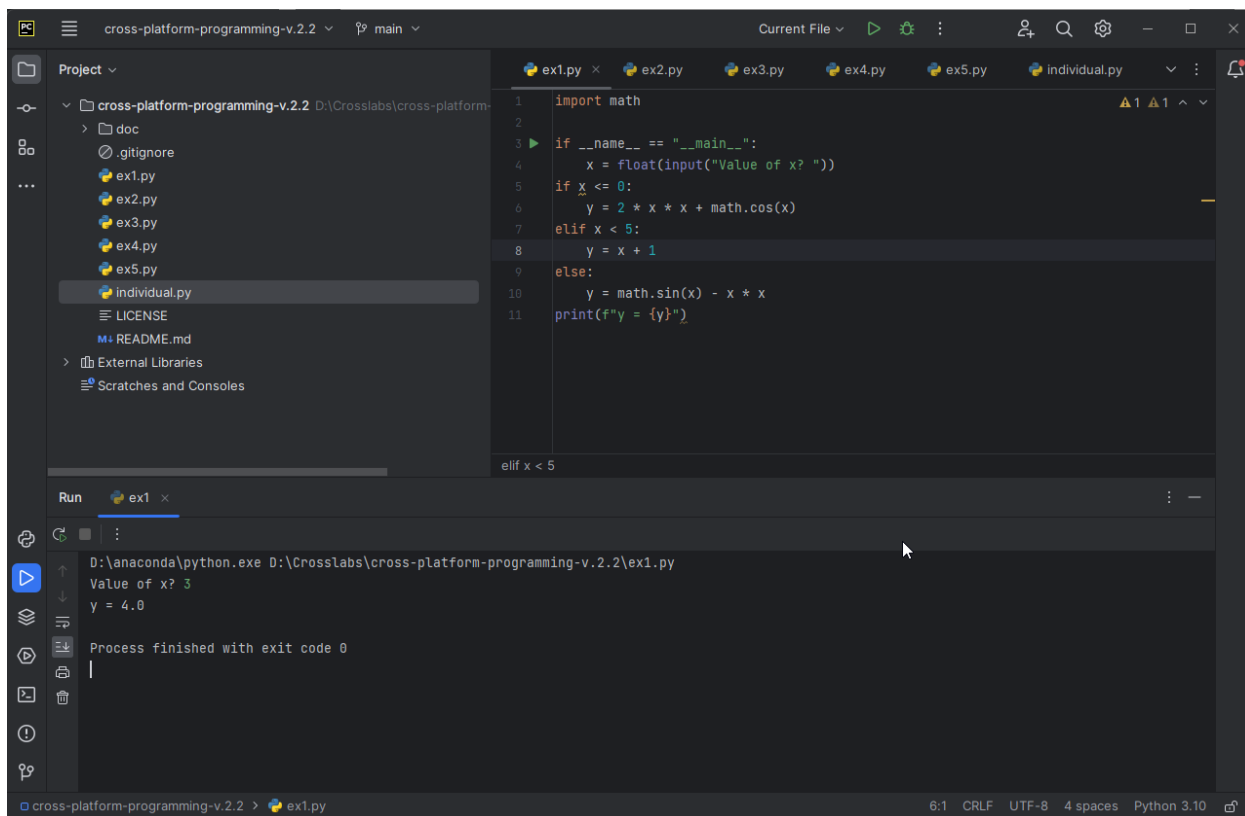


Рис. 4. Программа и ее результат.

5. Работа с примером 2.

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.

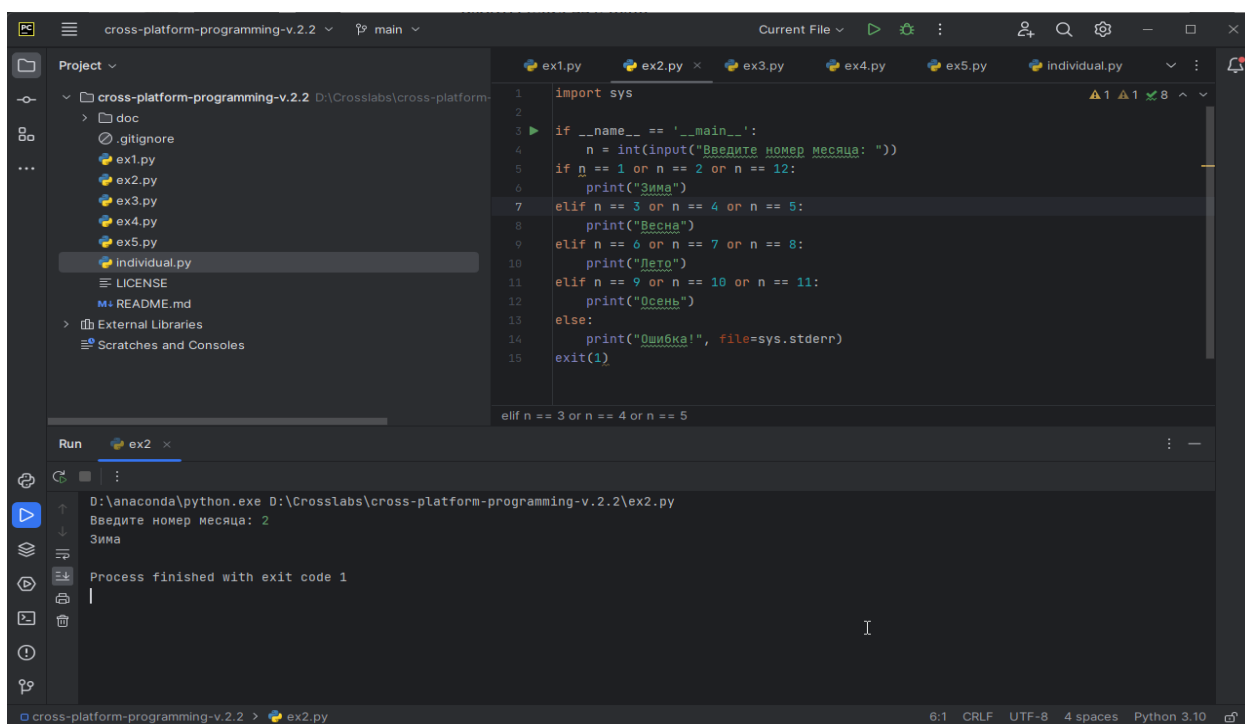


Рис. 5. Программа и ее результат.

6. Работа с примером 3.

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.

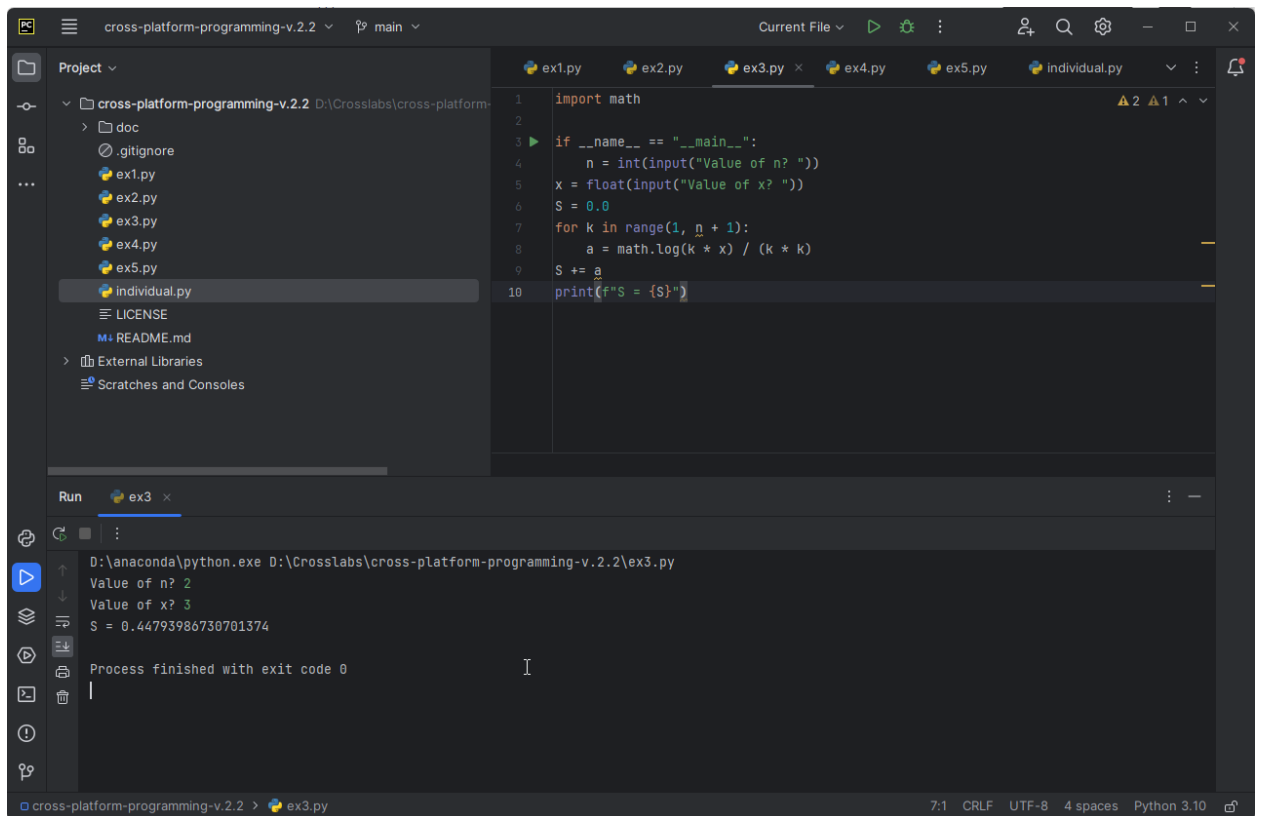


Рис. 6. Программа и ее результат.

7. Работа с примером 4.

Пример 4. Найти значение квадратного корня $x = \sqrt{a}$ из положительного числа a вводимого с клавиатуры, с некоторой заданной точностью ϵ с помощью рекуррентного соотношения:

$$x_{n+1} = \frac{1}{2} \cdot \left(x_n + \frac{a}{x_n} \right). \quad (3)$$

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

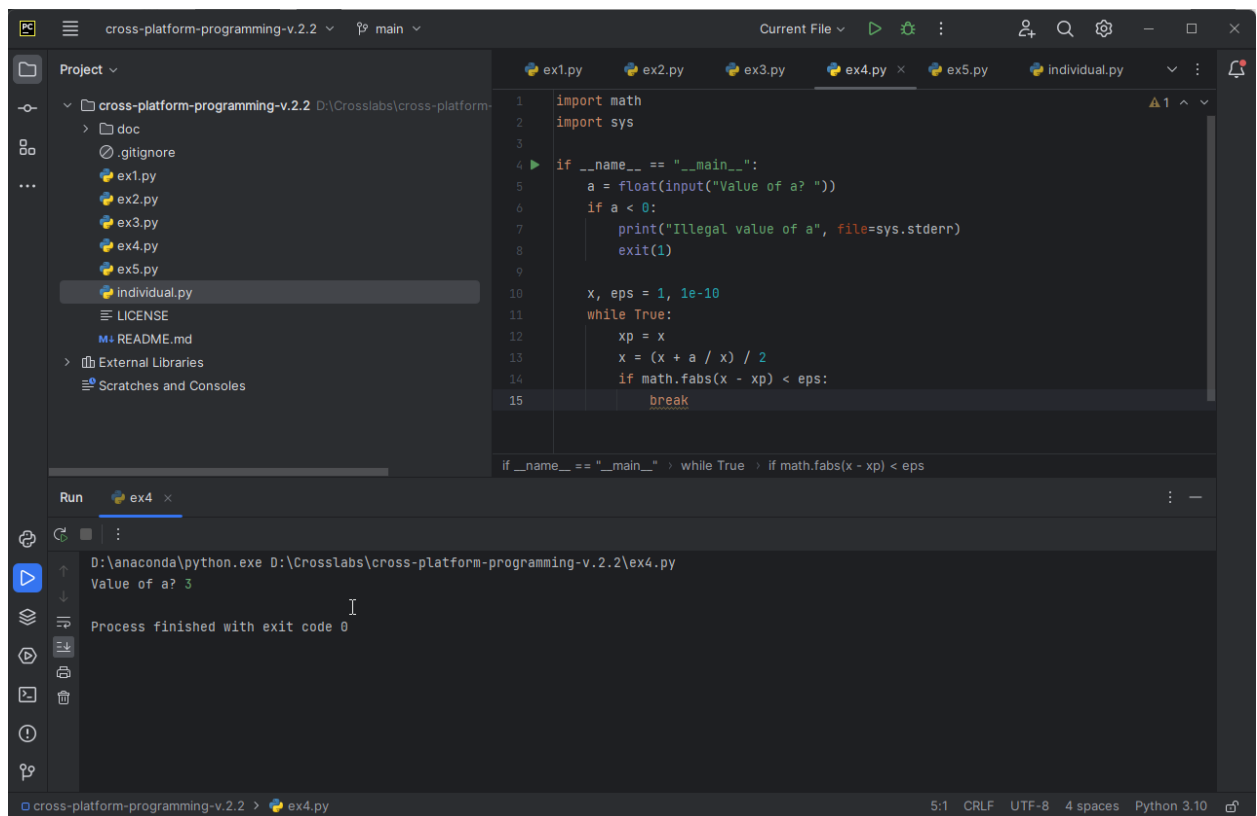


Рис. 7. Программа и ее результат.

8. Работа с примером 5.

Пример 5. Вычислить значение специальной (интегральной показательной) функции

$$\text{Ei}(x) = \int_{-\infty}^x \frac{\exp t}{t} dt = \gamma + \ln x + \sum_{k=1}^{\infty} \frac{x^k}{k \cdot k!}, \quad (4)$$

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

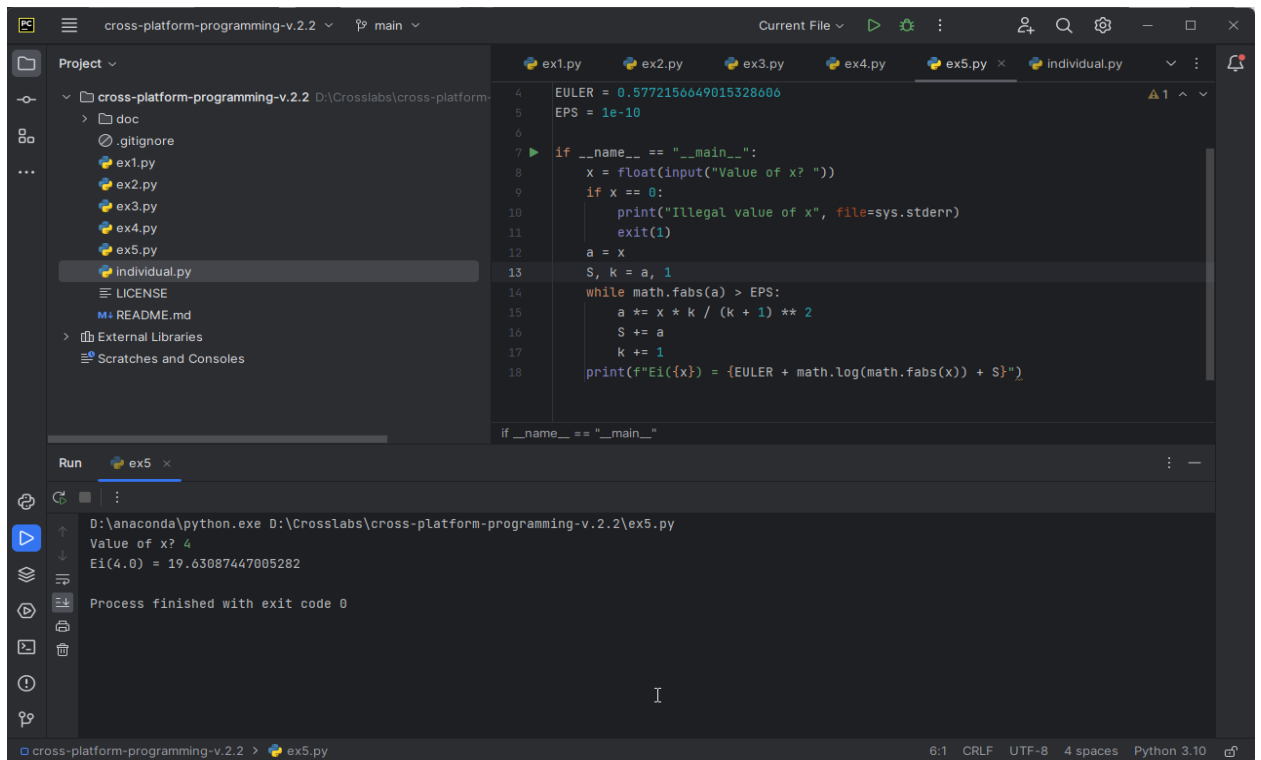


Рис. 8. Программа и ее результат.

9. Работа с индивидуальным заданием 1.

Вариант 3

Условие задания:

3. Дано число m ($1 \leq m \leq 7$). Вывести на экран название дня недели, который соответствует этому номеру.

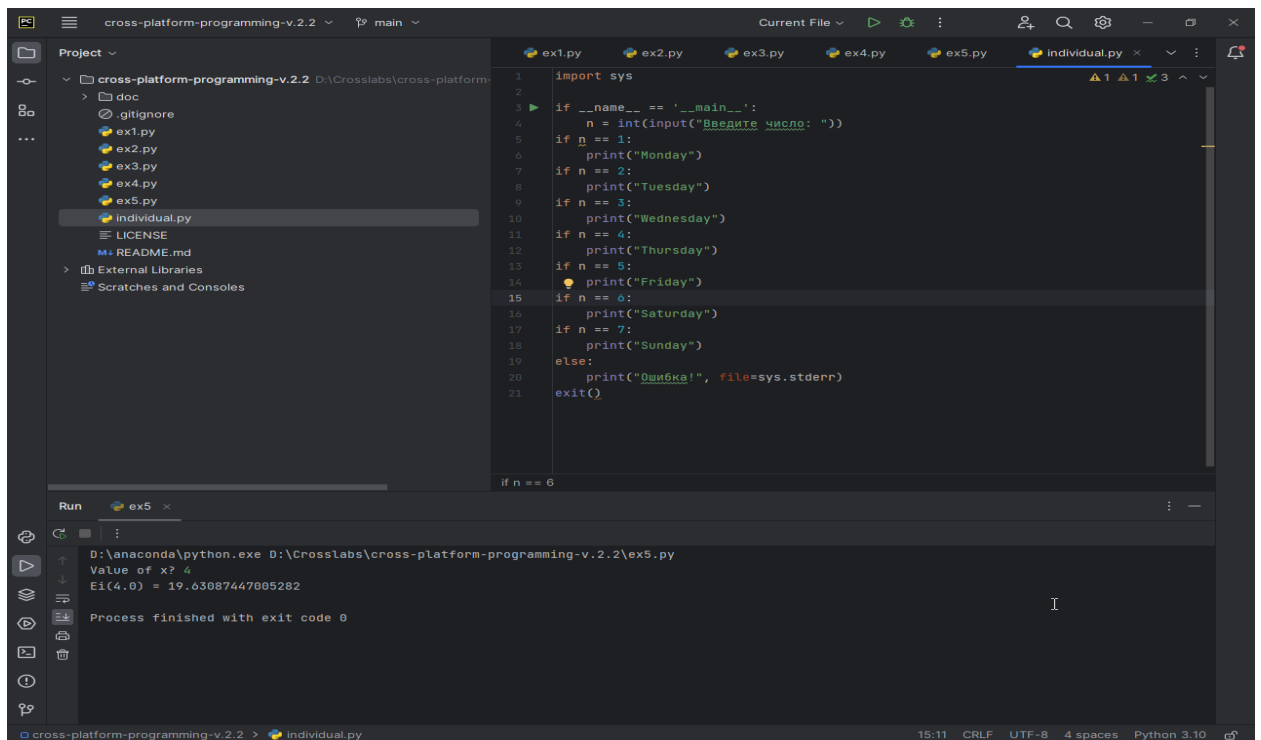


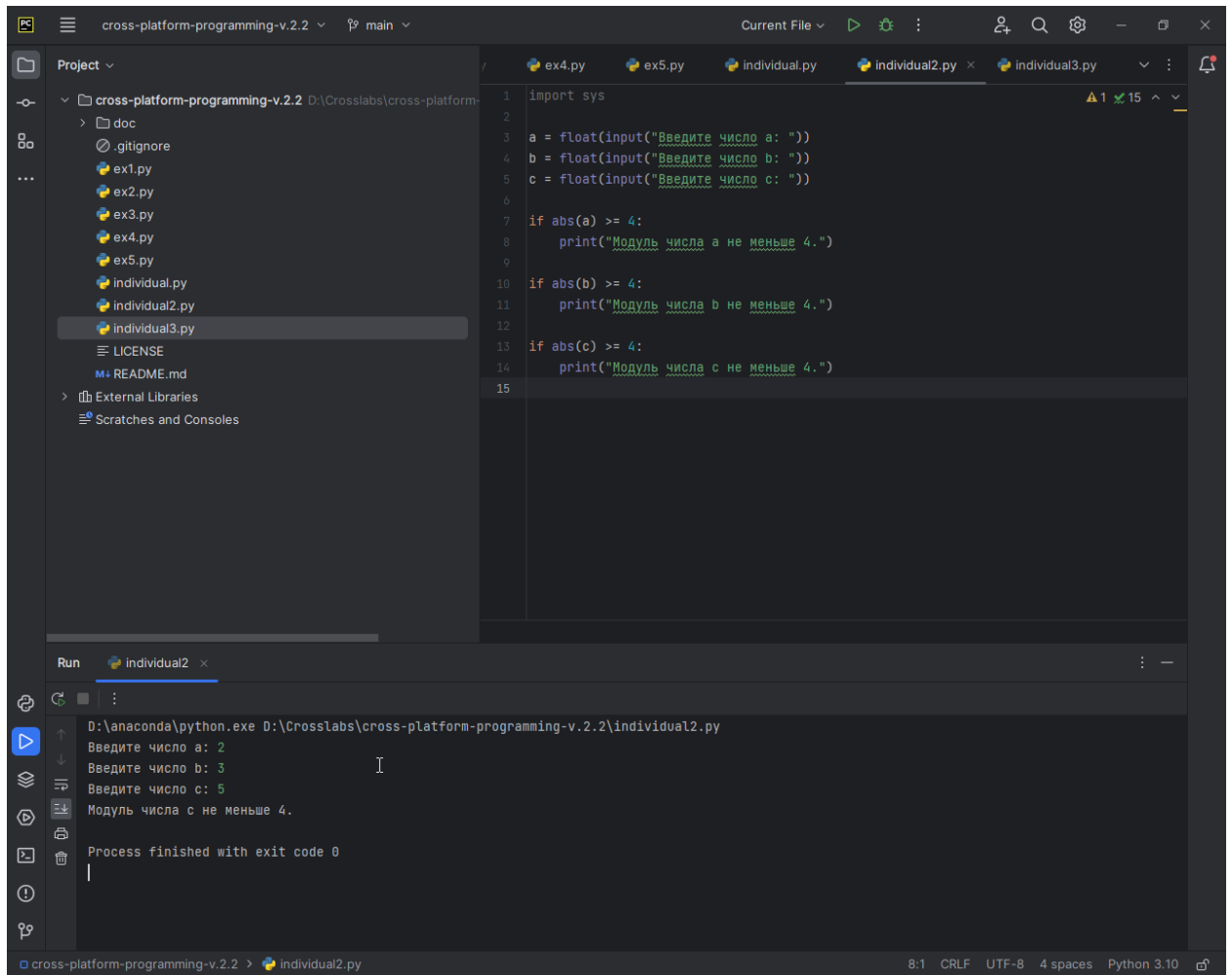
Рис. 9. Программа и ее результат.

10. Работа с индивидуальным заданием 2.

Вариант 3

Условие задания:

3. Из трех действительных чисел a , b и c выбрать те, модули которых не меньше 4.



The screenshot shows a code editor with a project named 'cross-platform-programming-v.2.2'. The project structure includes a 'doc' folder, a '.gitignore' file, and several Python files: 'ex1.py', 'ex2.py', 'ex3.py', 'ex4.py', 'ex5.py', 'individual.py', 'individual2.py', and 'individual3.py'. The 'individual2.py' file is open in the editor, showing the following code:

```
1 import sys
2
3 a = float(input("Введите число a: "))
4 b = float(input("Введите число b: "))
5 c = float(input("Введите число c: "))
6
7 if abs(a) >= 4:
8     print("Модуль числа a не меньше 4.")
9
10 if abs(b) >= 4:
11     print("Модуль числа b не меньше 4.")
12
13 if abs(c) >= 4:
14     print("Модуль числа c не меньше 4.")
15
```

The 'Run' panel at the bottom shows the execution of 'individual2.py' using 'D:\anaconda\python.exe'. The output is as follows:

```
D:\anaconda\python.exe D:\Crosslabs\cross-platform-programming-v.2.2\individual2.py
Введите число a: 2
Введите число b: 3
Введите число c: 5
Модуль числа c не меньше 4.
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, the line ending is CRLF, and the Python version is 3.10.

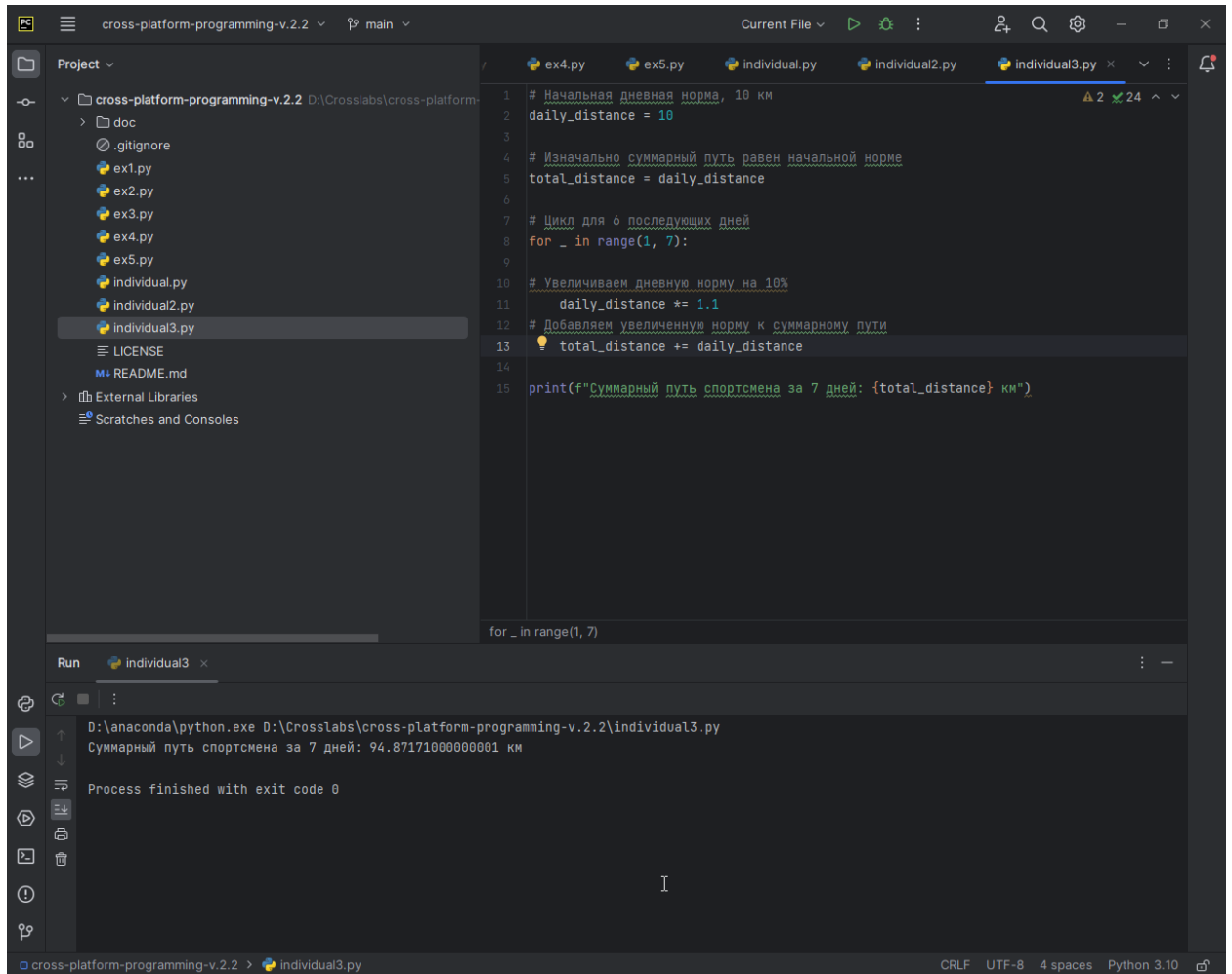
Рис. 10. Программа и ее результат.

11. Работа с индивидуальным заданием 3.

Вариант 3

Условие задания:

3. Начав тренировки, спортсмен пробежал 10 км. Каждый следующий день он увеличивал дневную норму на 10% от нормы предыдущего дня. Какой суммарный путь пробежит спортсмен за 7 дней?



The screenshot shows a code editor with a project named 'cross-platform-programming-v.2.2'. The file explorer on the left lists several Python files, including 'individual3.py' which is currently selected. The editor displays the following Python code:

```
1 # Начальная дневная норма, 10 км
2 daily_distance = 10
3
4 # Изначально суммарный путь равен начальной норме
5 total_distance = daily_distance
6
7 # Цикл для 6 последующих дней
8 for _ in range(1, 7):
9
10 # Увеличиваем дневную норму на 10%
11     daily_distance *= 1.1
12 # Добавляем увеличенную норму к суммарному пути
13     total_distance += daily_distance
14
15 print(f"Суммарный путь спортсмена за 7 дней: {total_distance} км")
```

Below the code editor, the 'Run' panel shows the execution of 'individual3.py'. The output is:

```
D:\anaconda\python.exe D:\Crosslabs\cross-platform-programming-v.2.2\individual3.py
Суммарный путь спортсмена за 7 дней: 94.87171000000001 км
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, using 4 spaces for indentation, and the Python version is 3.10.

Рис. 11. Программа и ее результат.

11. Слил ветку develop с веткой main, Отправка на гитхаб.

```
PS D:\Crosslabs\cross-platform-programming-v.2.2> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS D:\Crosslabs\cross-platform-programming-v.2.2> git merge develop
Updating 2051214..3a3fd37
Fast-forward
 ex1.py      | 11 ++++++++
 ex2.py      | 15 ++++++++
 ex3.py      | 10 ++++++++
 ex4.py      | 15 ++++++++
 ex5.py      | 18 ++++++++
 individual.py | 21 ++++++++
 6 files changed, 90 insertions(+)
 create mode 100644 ex1.py
 create mode 100644 ex2.py
 create mode 100644 ex3.py
 create mode 100644 ex4.py
 create mode 100644 ex5.py
 create mode 100644 individual.py
PS D:\Crosslabs\cross-platform-programming-v.2.2> git push
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 4 threads
Compressing objects: 100% (18/18), done.
Writing objects: 100% (18/18), 2.49 KiB | 637.00 KiB/s, done.
Total 18 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (5/5), done.
To https://github.com/temacteklyannayayuwka/cross-platform-programming-v.2.2.git
 2051214..3a3fd37  main -> main
```

Рис. 11. Слияние веток/отправка на гитхаб.

Ответы на контрольные вопросы:

1) Для чего нужны диаграммы деятельности UML? Диаграммы деятельности. Унифицированный язык моделирования (UML) является стандартным инструментом для создания «чертежей» программного обеспечения. С помощью UML можно визуализировать, специфицировать, конструировать и документировать артефакты программных систем. UML пригоден для моделирования любых систем: от информационных систем масштаба предприятия до распределенных Webприложений и даже встроенных систем реального времени. Это очень выразительный язык, позволяющий рассмотреть систему со всех точек зрения, имеющих отношение к ее разработке и последующему развертыванию. Несмотря на обилие

выразительных возможностей, этот язык прост для понимания и использования.

2) Что такое 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. Состояния действия изображаются прямоугольниками с закругленными краями. Внутри такого символа можно записывать произвольное выражение. 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.

3) Какие нотации существуют для обозначения переходов и ветвлений в диаграммах деятельности? Переходы. 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. 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). Ветвление. Простые последовательные переходы встречаются наиболее часто, но их одних недостаточно для моделирования любого потока управления. Как и в блок-схеме, вы можете включить в модель ветвление, которое описывает различные пути выполнения в зависимости от

значения некоторого булевского выражения. Точка ветвления представляется ромбом. В точку ветвления может входить ровно один переход, а выходить – два или более. Для каждого исходящего перехода задается булевское выражение, которое вычисляется только один раз при входе в точку ветвления. Ни для каких двух исходящих переходов эти сторожевые условия не должны одновременно принимать значение «истина», иначе поток управления окажется неоднозначным. Но эти условия должны покрывать все возможные варианты, иначе поток остановится.

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. Программа разветвляющейся структуры реализует такой алгоритм. В программе разветвляющейся структуры имеется один или несколько условных операторов. Для программной реализации условия используется логическое выражение. В сложных структурах с большим числом ветвей применяют оператор выбора.

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? Условный оператор состоит из трёх слов IF ELSE THEN. Здесь просто помещает значение на вершину стека, IF анализирует флаг, и если: он не равен нулю, то выполняются выражения до ELSE или THEN ; если он равен нулю, то выполняется выражения между ELSE и THEN . Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно. Истинность условия определяется также как и в операторе if. Оператор break предназначен для досрочного прерывания работы цикла while.

Оператор `continue` запускает цикл заново, при этом код, расположенный после данного оператора, не выполняется. Оператор `for` выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе. Функция `range` возвращает неизменяемую последовательность чисел в виде объекта `range`.

7) Какие операторы сравнения используются в Python? В Python есть шесть операций сравнения. Все они имеют одинаковый приоритет, который выше, чем у логических операций. Разрешенные операции сравнения:

- $x < y$ – строго x меньше y ,
- $x \leq y$ – x меньше или равно y ,
- $x > y$ – строго x больше y ,
- $x \geq y$ – x больше или равно y ,
- $x == y$ – x равно y ,
- $x != y$ – x не равно y .

8) Что называется простым условием? Приведите примеры. Простым условием (отношением) называется выражение, составленное из двух арифметических выражений или двух текстовых величин (иначе их еще называют операндами), связанных одним из знаков: $<$ - меньше, чем...

Например, простыми отношениями являются следующие:

$x-y > 10$; $k \leq \text{sqrt}(c) + \text{abs}(a+b)$; $9 < 11$; `'мама' <> 'папа'`.

9) Что такое составное условие? Приведите примеры. Составные условия — это условия, состоящие из двух или более простых условий, соединенных с помощью логических операций: `and`, `or`, `not`. Простые условия при этом заключаются в скобки. Примеры составных условий: $(a \geq 8) \text{ or } (x < -3)$ `not (a=0) or (b=0)`

10) Какие логические операторы допускаются при составлении сложных условий? Используются специальные операторы, объединяющие два и более простых логических выражения. Широко используются два оператора – так называемые логические И (`and`) и ИЛИ (`or`).

11) Может ли оператор ветвления содержать внутри себя другие ветвления? Да. Такой оператор ветвления называется вложенным. В этом случае важно не перепутать какая ветвь кода к какому оператору относится. Поэтому рекомендуется соблюдать отступы в исходном коде программы, чтобы не запутаться. `if then if< условие> then< оператор 1>;` Вложенный условный оператор.

12) Какой алгоритм является алгоритмом циклической структуры? Например, перевод текста с иностранного языка (прочитать первое предложение, перевести, записать и т.д.) Построение графика функции по точкам (взять первый аргумент, вычислить значение функции, построить точку и т.д.)

13) Типы циклов в языке Python. Цикл — это блок, элементы которого повторяют своё действие заданное количество раз. Бывают. Практически все современные алгоритмы содержат в себе циклы. В языке Python существуют следующие типы циклов: 1. `While()` — Цикл с предусловием 2. `for()` — Цикл с чётким количеством проходов.

14) Назовите назначение и способы применения функции `range`. Функция `range` является одной из встроенных функций, доступных в Python. Он генерирует серию целых чисел, от значения `start` до `stop`, указанного пользователем. Мы можем использовать его для цикла `for` и обходить весь диапазон как список. Функция `range ()` принимает один обязательный и два необязательных параметра. Это работает по-разному с различными комбинациями аргументов.

15) Как с помощью функции `range` организовать перебор значений от 15 до 0 с шагом 2? `range(15, 0, -2)`

16) Могут ли циклы быть вложенными? Чисто технически во вложенных циклах нет ничего особенного. Их можно вкладывать внутрь любого блока и друг в друга сколько угодно раз. Но прямой связи между внешним и вложенным циклами нет. Внутренний цикл может использовать результаты внешнего, а может и работать по своей собственной логике независимо.

Вложенные циклы коварны. Их наличие может резко увеличить сложность кода, так как появляется множество постоянно изменяющихся переменных.

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. О программе, вошедшей в бесконечный цикл, иногда говорят, что она зациклилась.

18) Для чего нужен оператор `break`? Выражение `break` заставляет программу выйти из цикла.

19) Где употребляется оператор `continue` и для чего он используется? Выражение `Continue`. Выражение `continue` дает возможность пропустить часть цикла, где активируется внешнее условие, но при этом выполнить остальную часть цикла. При этом прерывается текущая итерация цикла, но программа возвращается к началу цикла. Выражение `continue` размещается в блоке кода под выражением цикла, обычно после условного выражения `if`.

20) Для чего нужны стандартные потоки `stdout` и `stderr`? `STDOUT` - стандартный вывод - то, куда выводят данные команды `echo/print`, консоль или сокет, отправляющий данные браузеру. `STDERR` – поток сообщений об ошибках.

21) Каково назначение функции `exit`? Функция `exit()` вызывает немедленное нормальное завершение программы.

Вывод: в ходе выполнения лабораторной работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Также изучены операторы языка Python.3 `if`, `while`, `for`, `break`, `continue`, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.