

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 №2.2
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Хачатрян Владимир Владимирович
1 курс, группа ИТС-б-о-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., доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Условные операторы и циклы в языке Python

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

Ход работы:

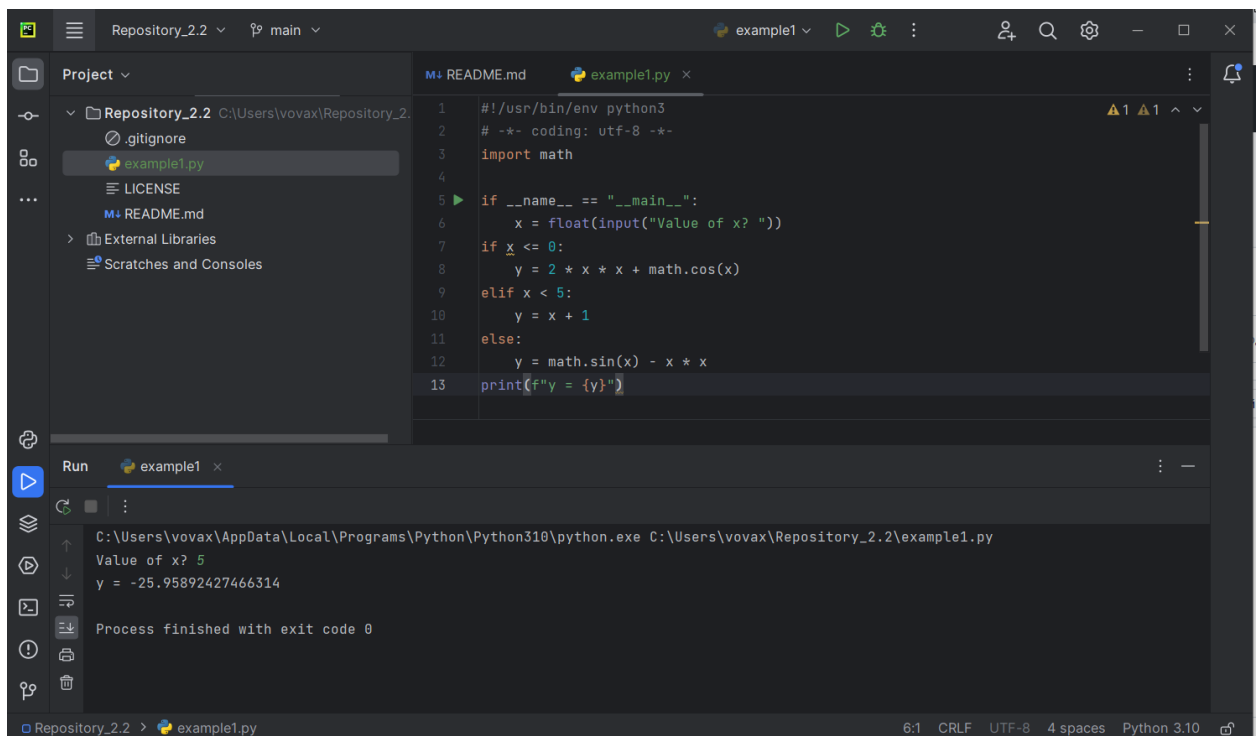
Создал репозиторий на GitHub, добавил файл .gitignore с необходимыми правилами. Клонировал репозиторий на свой компьютер.

 Git CMD

```
C:\Users\vovax>git clone https://github.com/GitVolodya/Repository_2.2.git
Cloning into 'Repository_2.2'...
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.
```

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

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



The screenshot shows the Visual Studio Code editor with a project named 'Repository_2.2'. The file explorer on the left shows the project structure: Repository_2.2 (containing .gitignore, example1.py, LICENSE, README.md, External Libraries, and Scratches and Consoles). The main editor window displays the code for example1.py:

```
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}")
```

The Run and Debug console at the bottom shows the execution of the script:

```
C:\Users\vovax\AppData\Local\Programs\Python\Python310\python.exe C:\Users\vovax\Repository_2.2\example1.py
Value of x? 5
y = -25.95892427466314
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.

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.

The screenshot shows a Visual Studio Code editor with a project named 'Repository_2.2'. The file explorer on the left shows 'example2.py' selected. The main editor displays the following Python code:

```

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)

```

The Run and Debug console at the bottom shows the execution of 'example2.py'. The input was '5', and the output was 'Весна'. The process finished with exit code 1.

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

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

The screenshot shows a Visual Studio Code editor with a project named 'Repository_2.2'. The file explorer on the left shows 'example3.py' selected. The main editor displays the following Python code:

```

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}")

```

The Run and Debug console at the bottom shows the execution of 'example3.py'. The input for 'Value of n?' was 2, and for 'Value of x?' was 3. The output was 'S = 0.44793986730701374'. The process finished with exit code 0.

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 $|x_{n+1} - x_n| \leq \varepsilon$. Сравните со значением квадратного корня, полученным с использованием функций стандартной библиотеки. Значение $\varepsilon = 10^{-10}$.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import math
5 import sys
6
7 if __name__ == "__main__":
8     a = float(input("Value of a? "))
9     if a < 0:
10         print("Illegal value of a", file=sys.stderr)
11         exit(1)
12
13     x, eps = 1, 1e-10
14     while True:
15         xp = x
16         x = (x + a / x) / 2
17         if math.fabs(x - xp) < eps:
18             break
19
20     print(f"x = {x}\nX = {math.sqrt(a)}")
21
22 if __name__ == "__main__":
```

Run example4

C:\Users\vovax\AppData\Local\Programs\Python\Python310\python.exe C:\Users\vovax\Repository_2.2\example4.py

Value of a? 5

x = 2.23606797749979

X = 2.23606797749979

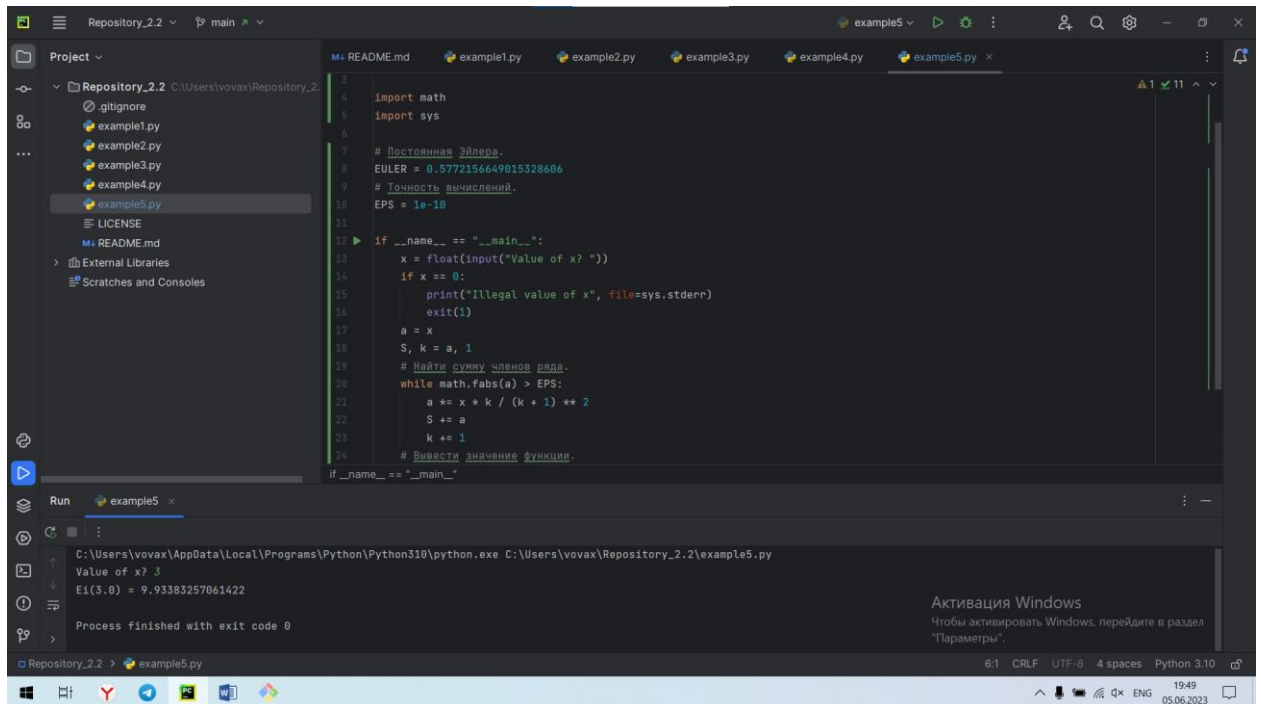
Process finished with exit code 0

Активация Windows
Чтобы активировать Windows, перейдите в раздел "Параметры".

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.



```
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 = float(input("Value of x? "))
14     if x == 0:
15         print("Illegal value of x", file=sys.stderr)
16         exit(1)
17     a = x
18     S, k = a, 1
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     # Вывести значения функции.
25
26 if __name__ == "__main__":
```

Run example5

C:\Users\vovax\AppData\Local\Programs\Python\Python310\python.exe C:\Users\vovax\Repository_2.2\example5.py

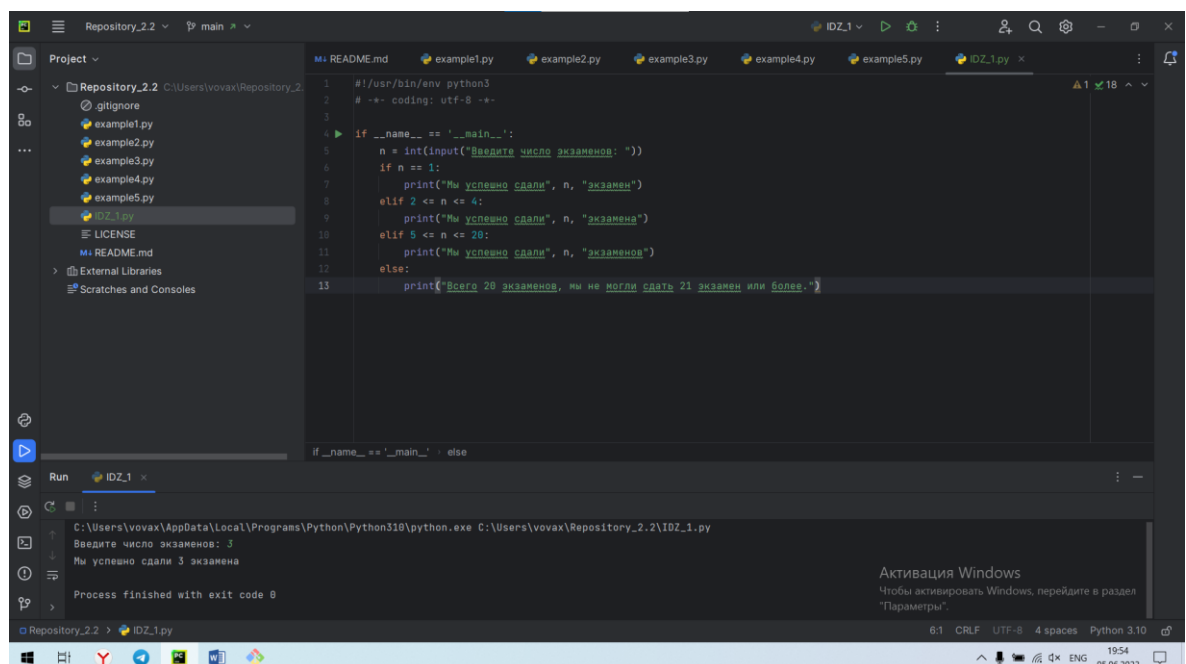
Value of x? 3

Ei(3.0) = 9.93383257061422

Process finished with exit code 0

Индивидуальное задание (вариант 9)

9. Вводится число экзаменов $N \leq 20$. Напечатать фразу **мы успешно сдали N экзаменов**, согласовав слово "экзамен" с числом N .



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     n = int(input("Введите число экзаменов: "))
6     if n == 1:
7         print("Мы успешно сдали", n, "экзамен")
8     elif 2 <= n <= 4:
9         print("Мы успешно сдали", n, "экзамена")
10    elif 5 <= n <= 20:
11        print("Мы успешно сдали", n, "экзаменов")
12    else:
13        print("Всего 20 экзаменов, мы не могли сдать 21 экзамен или более.")
14
15 if __name__ == '__main__': else
```

Run IDZ_1

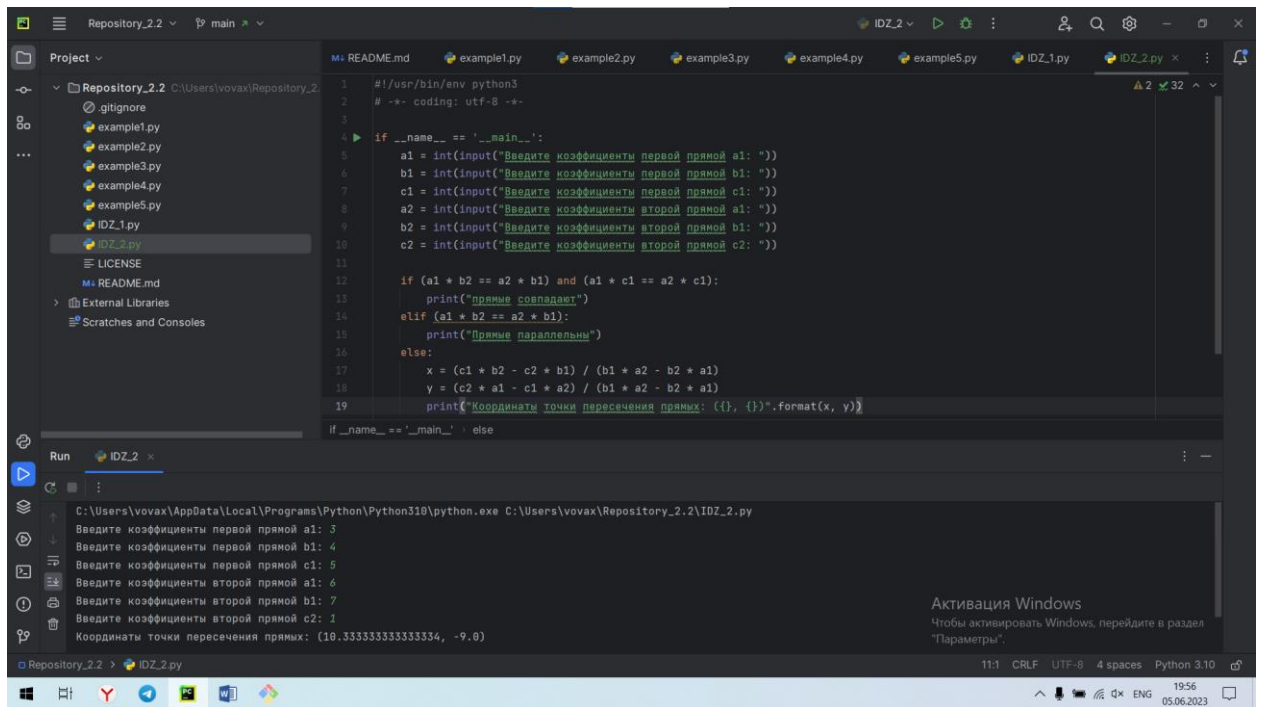
C:\Users\vovax\AppData\Local\Programs\Python\Python310\python.exe C:\Users\vovax\Repository_2.2\IDZ_1.py

Введите число экзаменов: 3

Мы успешно сдали 3 экзамена

Process finished with exit code 0

9. Найти координаты точки пересечения прямых заданных уравнениями $a_1x + b_1y + c_1 = 0$ и $a_2x + b_2y + c_2 = 0$, либо сообщить совпадают, параллельны или не существуют.



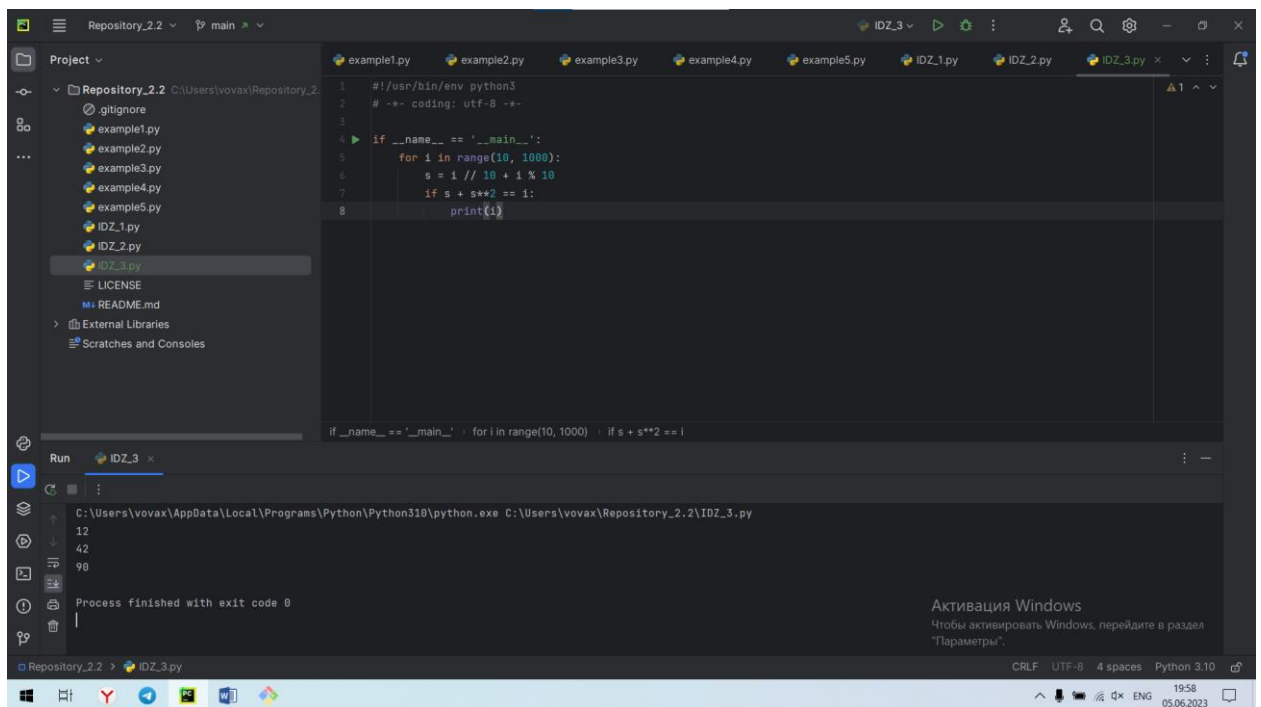
```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     a1 = int(input("Введите коэффициенты первой прямой a1: "))
6     b1 = int(input("Введите коэффициенты первой прямой b1: "))
7     c1 = int(input("Введите коэффициенты первой прямой c1: "))
8     a2 = int(input("Введите коэффициенты второй прямой a1: "))
9     b2 = int(input("Введите коэффициенты второй прямой b1: "))
10    c2 = int(input("Введите коэффициенты второй прямой c2: "))
11
12    if (a1 * b2 == a2 * b1) and (a1 * c1 == a2 * c1):
13        print("Прямые совпадают")
14    elif (a1 * b2 == a2 * b1):
15        print("Прямые параллельны")
16    else:
17        x = (c1 * b2 - c2 * b1) / (b1 * a2 - b2 * a1)
18        y = (c2 * a1 - c1 * a2) / (b1 * a2 - b2 * a1)
19        print("Координаты точки пересечения прямых: ({}, {})".format(x, y))
20
21 if __name__ == '__main__': else
```

Run IDZ_2

C:\Users\vovax\AppData\Local\Programs\Python\Python310\python.exe C:\Users\vovax\Repository_2.2\IDZ_2.py

Введите коэффициенты первой прямой a1: 3
Введите коэффициенты первой прямой b1: 4
Введите коэффициенты первой прямой c1: 5
Введите коэффициенты второй прямой a1: 6
Введите коэффициенты второй прямой b1: 7
Введите коэффициенты второй прямой c2: 1
Координаты точки пересечения прямых: (10.333333333333334, -9.0)

9. Если к сумме цифр двузначного числа прибавить квадрат этой суммы, то снова получится это двузначное число. Найти все эти числа.



```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 if __name__ == '__main__':
5     for i in range(10, 1000):
6         s = i // 10 + i % 10
7         if s + s**2 == i:
8             print(i)
9
10 if __name__ == '__main__': for i in range(10, 1000) if s + s**2 == i
```

Run IDZ_3

C:\Users\vovax\AppData\Local\Programs\Python\Python310\python.exe C:\Users\vovax\Repository_2.2\IDZ_3.py

12
42
90
Process finished with exit code 0

Ссылка на репозиторий: https://github.com/GitVolodya/Repository_2.2.git

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 Web приложений 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.

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?

Диаграммы деятельности могут использовать следующие нотации для обозначения переходов и ветвлений:

Стрелки - это общая нотация для обозначения переходов между действиями в диаграмме деятельности. Стрелка обычно указывает направление движения в диаграмме. Если стрелка указывает на другое действие, это обозначает переход к этому действию.

Условные дуги - это нотация, которая используется для обозначения условных переходов в диаграммах деятельности. Условная дуга имеет условие, которое определяет, какой путь следует выбрать при переходе.

Решетка - это нотация, которая используется для обозначения параллельных ветвлений. Решетка имеет несколько ветвей, которые выполняются параллельно друг другу.

Декомпозиция - это нотация, которая используется для разделения действий на более мелкие части. Декомпозиция может быть выполнена в виде разделения диаграммы на поддиаграммы или разделения действия на более мелкие действия.

Выбор конкретной нотации зависит от контекста и задачи, которую необходимо решить с помощью диаграммы деятельности.

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 <условие> 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: $xy > 10$; $k \leq \text{sqr}(c) + \text{abs}(a+b)$.

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<5) and (b>8)

(x>=0) 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 <условие> 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: While() — Цикл с предусловием; 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 зацклилась.

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 данной работы я приобрел навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоил операторы языка Python версии 3.x `if`, `while`, `for`, `break` и `continue`, позволяющих реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.