

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
дисциплины «Программирование на Python»

Выполнил:
Гайчук Дарья Дмитриевна
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика
и вычислительная техника»,
направленность (профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р.А.-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.

Порядок выполнения работы

1. Создала общедоступный репозиторий на GitHub, в котором будет использоваться лицензия MIT.

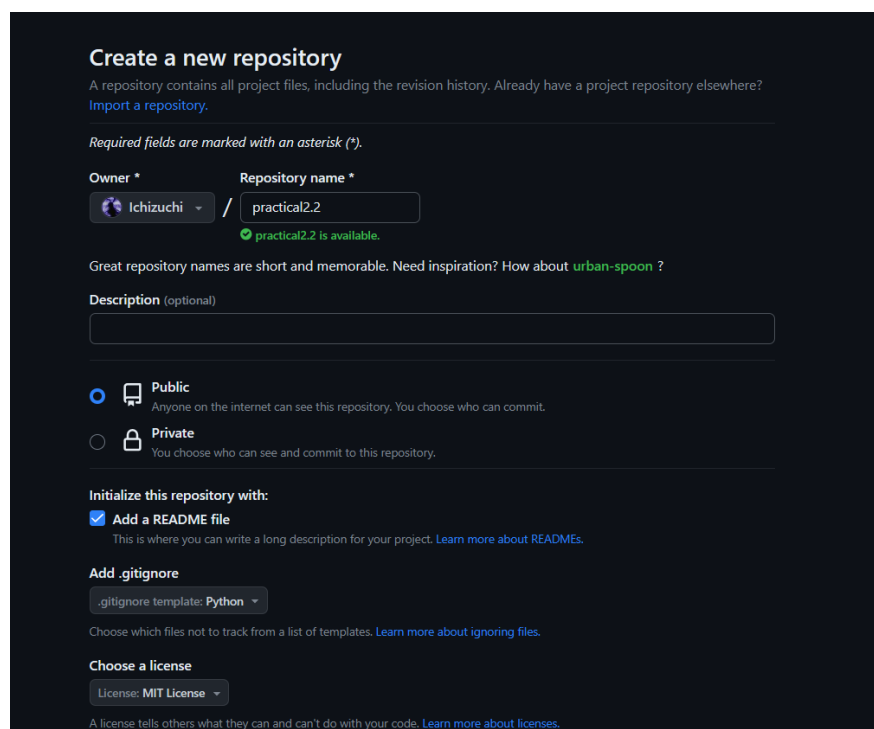


Рисунок 1. Новый репозиторий

2. Скопировала репозиторий на свой компьютер.

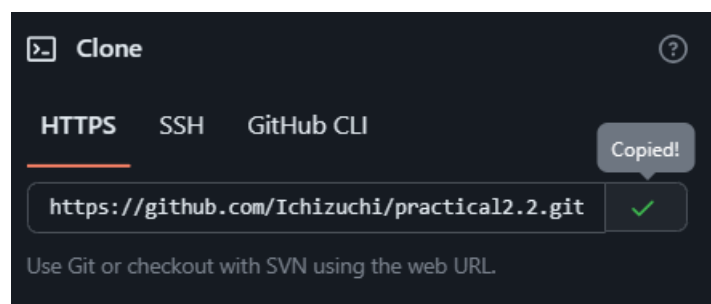


Рисунок 2. Клонирование репозитория

3. Использовала систему ветвления git-flow



Рисунок 3. Ветка develop

4. Работа с примером 1.

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

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

```

1  import math
2
3  if __name__ == '__main__':
4      x = float(input("Чему равен x? "))
5      if x <= 0:
6          y = 2 * x * x + math.cos(x)
7      elif x < 5:
8          y = x + 1
9      else:
10         y = math.sin(x) - x * x
11         print(f"y={y}")
12     exit(1)
  
```

Run ex1 x

```

C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe
Чему равен x? 6
y=-36.27941549819893
  
```

Рисунок 4. Программа примера 1

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.

```
1 import sys
2
3 if __name__ == '__main__':
4     n = int(input("Введите номер месяца: "))
5     if n == 1 or n == 2 or n == 12:
6         print("Зима")
7     elif n == 3 or n == 4 or n == 5:
8         print("Весна")
9     elif n == 6 or n == 7 or n == 8:
10        print("Лето")
11    elif n == 9 or n == 10 or n == 11:
12        print("Зима")
13    else:
14        print("Ошибка!", file=sys.stderr)
15    exit(1)
```

Run ex2 x

C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe

Введите номер месяца: 4

Весна

Process finished with exit code 1

Рисунок 5. Программа примера 2

6. Работа с примером 3

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

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

где n и k вводятся с клавиатуры.

The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure for 'practical2.2' with files 'ex1.py', 'ex2.py', 'ex3.py', 'LICENSE', and 'README.md'. The code editor shows the following Python code:

```
1 import math
2
3 if __name__ == "__main__":
4     n = int(input("Чему равен n? "))
5     x = float(input("Чему равен x? "))
6     S = 0.0
7     for k in range(1, n + 1):
8         a = math.log(k * x) / (k * k)
9     S += a
10    print(f"S = {S}")
11
```

Below the code editor, there is a 'Run' button and a terminal window. The terminal window shows the output of the program:

```
C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe
Чему равен n? 7
Чему равен x? 4
S = 0.06800417367704498
Process finished with exit code 0
```

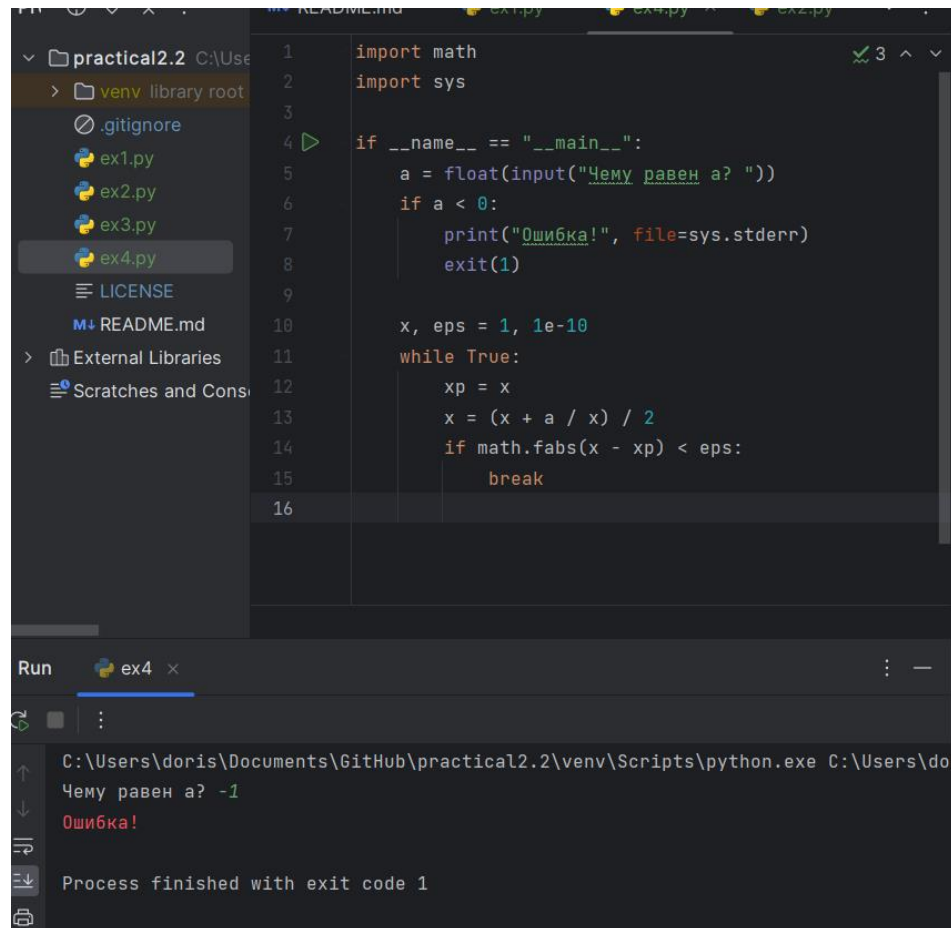
Рисунок 6. Программа примера 3

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 $\epsilon = 10^{-10}$.



The screenshot shows a code editor with a file explorer on the left and a terminal at the bottom. The file explorer shows a directory named 'practical2.2' containing files 'ex1.py', 'ex2.py', 'ex3.py', 'ex4.py', 'LICENSE', and 'README.md'. The code editor displays a Python script with the following content:

```
1 import math
2 import sys
3
4 if __name__ == "__main__":
5     a = float(input("Чему равен а? "))
6     if a < 0:
7         print("Ошибка!", file=sys.stderr)
8         exit(1)
9
10 x, eps = 1, 1e-10
11 while True:
12     xp = x
13     x = (x + a / x) / 2
14     if math.fabs(x - xp) < eps:
15         break
16
```

The terminal at the bottom shows the execution of the script 'ex4'. The command prompt is 'C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe C:\Users\do'. The input is 'Чему равен а? -1'. The output is 'Ошибка!'. The process finished with exit code 1.

Рисунок 7. Программа примера 4

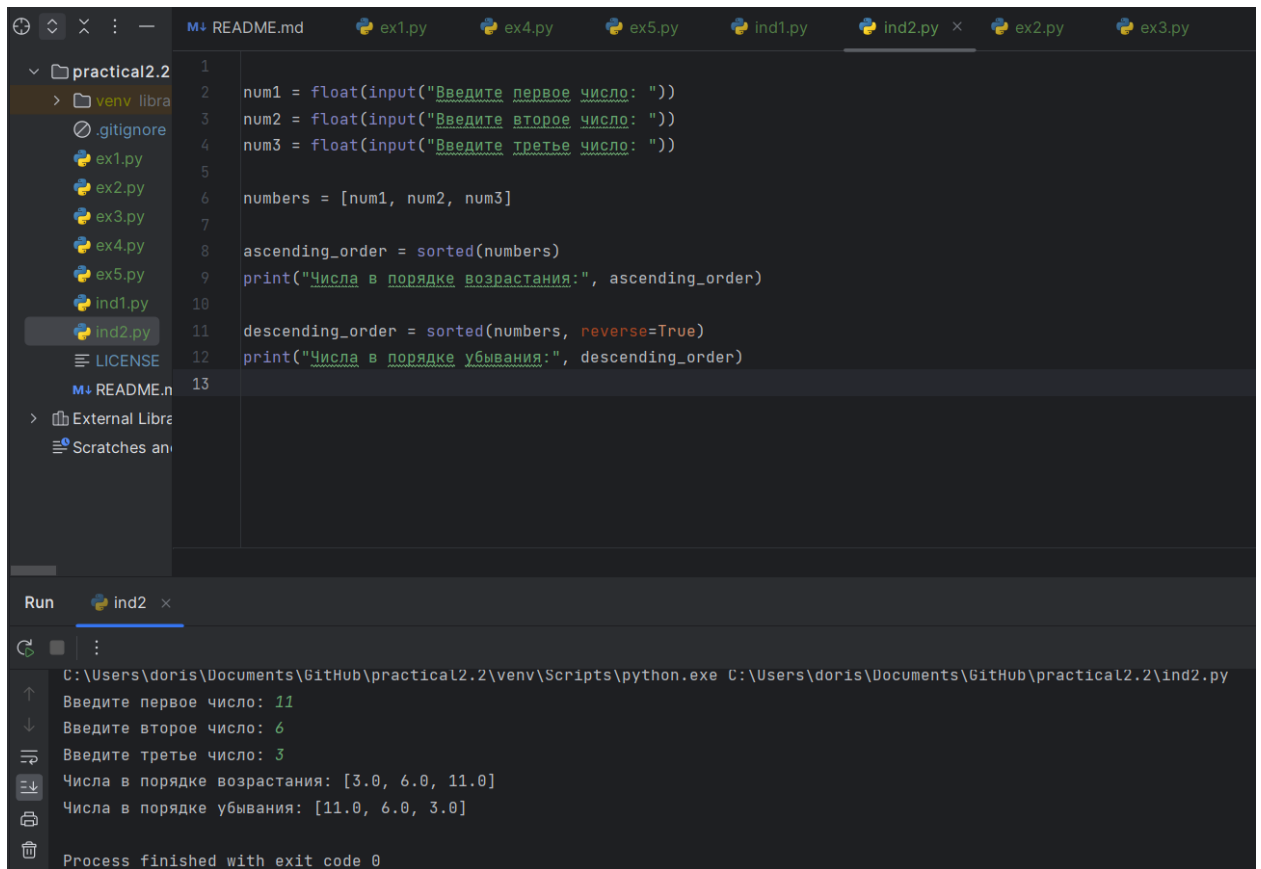
8. Работа с примером 5.

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 $\epsilon = 10^{-10}$, аргумент x вводится с клавиатуры.

4. Напечатать три данных действительных числа a , b и c сначала в порядке их возрастания, затем - в порядке убывания.



```
1 num1 = float(input("Введите первое число: "))
2 num2 = float(input("Введите второе число: "))
3 num3 = float(input("Введите третье число: "))
4
5 numbers = [num1, num2, num3]
6
7 ascending_order = sorted(numbers)
8 print("Числа в порядке возрастания:", ascending_order)
9
10 descending_order = sorted(numbers, reverse=True)
11 print("Числа в порядке убывания:", descending_order)
12
13
```

Run ind2 x

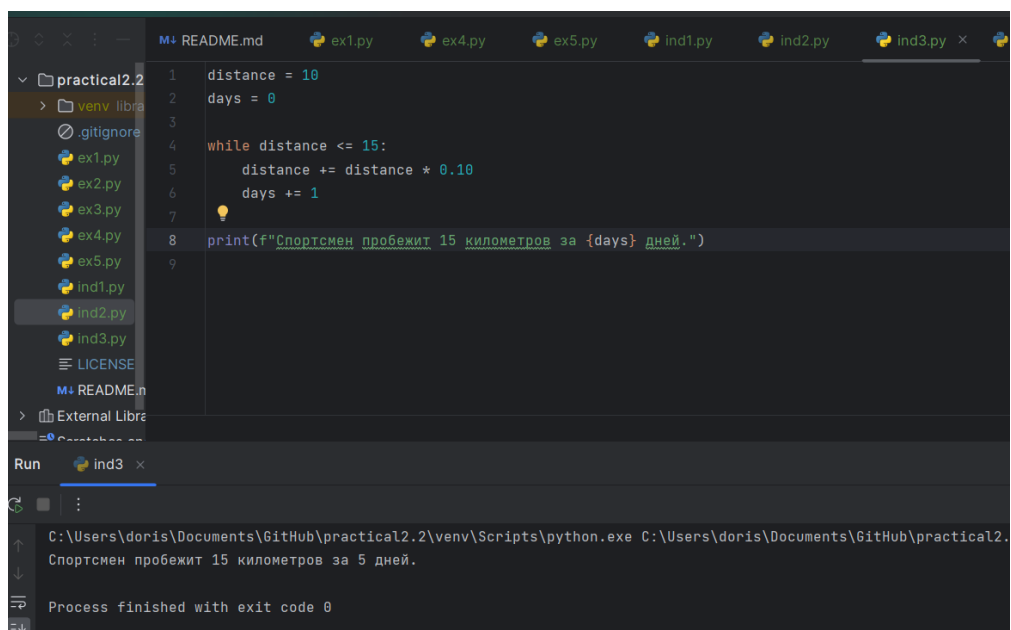
C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe C:\Users\doris\Documents\GitHub\practical2.2\ind2.py

Введите первое число: 11
Введите второе число: 6
Введите третье число: 3
Числа в порядке возрастания: [3.0, 6.0, 11.0]
Числа в порядке убывания: [11.0, 6.0, 3.0]
Process finished with exit code 0

Рисунок 10. Программа индивидуального задания 2

- 11.Выполнила индивидуальное задание 3 согласно своему варианту:

4. Через сколько дней спортсмен из задания 1 будет пробегать в день больше 15 км?



```
1 distance = 10
2 days = 0
3
4 while distance <= 15:
5     distance += distance * 0.10
6     days += 1
7
8 print(f"Спортсмен пробежит 15 километров за {days} дней.")
9
```

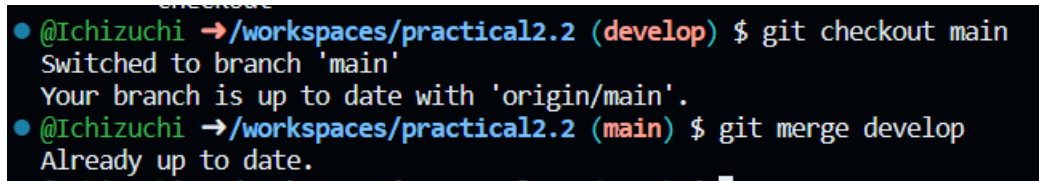
Run ind3 x

C:\Users\doris\Documents\GitHub\practical2.2\venv\Scripts\python.exe C:\Users\doris\Documents\GitHub\practical2.2\ind3.py

Спортсмен пробежит 15 километров за 5 дней.
Process finished with exit code 0

Рисунок 11. Программа индивидуального задания 3

12. Слила ветку develop в главную main.



```
@Ichizuchi →/workspaces/practical2.2 (develop) $ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
@Ichizuchi →/workspaces/practical2.2 (main) $ git merge develop
Already up to date.
```

Рисунок 12. Слияние веток

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. 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 – это алгоритм, все этапы которого выполняются однократно и строго последовательно. Разветвляющийся алгоритм — это 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: $x-y>10$; $k \leq \sqrt{c} + \text{abs}(a+b)$; $9 \in 11$; 'мама' \in '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: $(a \geq 8)$ $(x \geq 0) \text{ or } (x < -3) \text{ not } (a = 0) \text{ or } (b = 0)$.

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:

- 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?

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` 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.

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