иМинистерство 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»
Вариант 28

Выполнил:
Репкин Александр Павлович
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., канд. техн. наук,
доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: Условные операторы и циклы в языке Python.

Цель: приобрести навыки 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. Выполнен первый пример. В нём вычислялось значение функции y в зависимости от введённого значения x.

```
Good day! We need you to give us X's value. X = -2
X <= 0. It means, that Y =  7.583853163452858
```

Рисунок 1. Полученный результат примера №1.

2. Выполнен второй пример. В нём относительно введённого номера месяца определялся текущий сезон.

```
Good day! We need you to give us N's value. N = 10
Well, it seems to us, that Autumn is outdoors!
```

Рисунок 2. Полученный результат примера №2.

3. Выполнен третий пример. В нём требовалось: увеличить строку до требуемого размера путём равномерного увеличения количества пробелов между словами.

```
Good day! We need you to give us N's and X's values. N = 3
Great, and X = 5
According to X's and N's values, S =  2.485978652471746
```

Рисунок 3. Полученный результат примера №3.

4. Выполнен четвёртый пример. В нём вычислялся квадратный корень введённого значения функцией sqrt библиотеки math и вручную при помощи цикла.

```
Good day! We need you to give us A's value. A = 99
It seems to us that in this case x = 9.9498743710662. Using sqrt we got answer: sqrt(a) = 9.9498743710662.
```

Рисунок 4. Полученный результат примера №4.

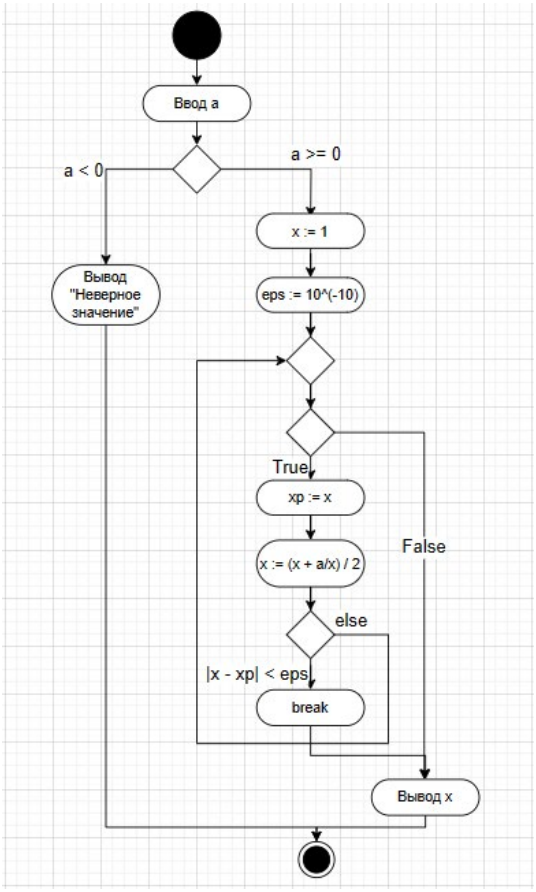

Рисунок 5. UML-диаграмма для примера №4.

5. Выполнен пятый пример. В нём вычислялось значение специальной (интегральной показательной) функции.

```
Good day! We need you to give us X's value. X = 6
According to our calculations, Ei(6.0) = 85.98976214243285
```

Рисунок 6. Полученный результат примера №5.

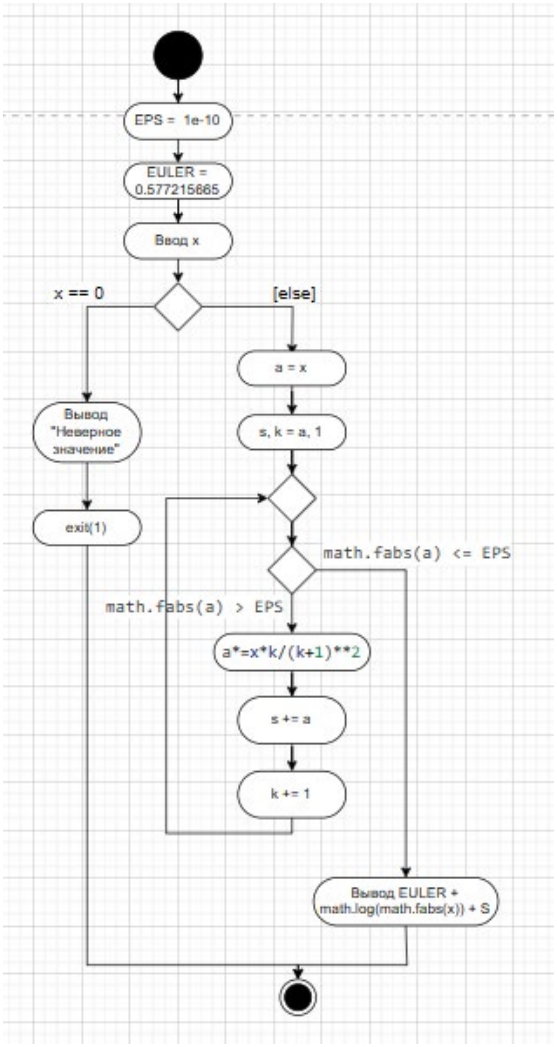

Рисунок 7. UML-диаграмма для примера №5.

6. Выполнено индивидуальное задание №1 – необходимо на основе введённого номера месяца вывести число дней в нём.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys


# Вариант - 28. Задание № 1, пункт №2.
if __name__ == '__main__':
    month = int(input("Good day! Which month you want us to analise? Month - "))
    if 1 > month or month > 12:
        print("Oops! There is no such a month!", file=sys.stderr)
    elif month == 4 or month == 6 or month == 9 or month == 11:
        print("Very well, answer is - 30.")
    elif month == 2:
        print("Oof... There are two variants, in February it can be 28 or 29 days.")
    else:
        print("Very well, answer is - 31.")

```

Рисунок 8. Полученный код индивидуального задания №1.

```
Good day! Which month you want us to analise? Month - 13
Oops! There is no such a month!
```

Рисунок 9. Полученный результат индивидуального задания №1.

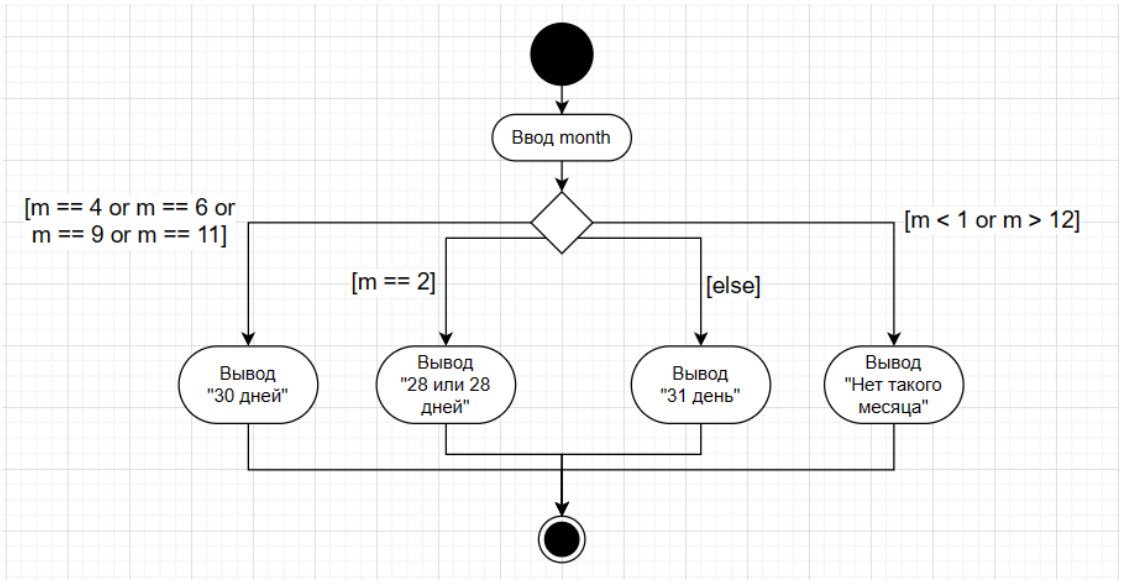

Рисунок 10. Полученная UML-диаграмма индивидуального задания №1.

7. Выполнено индивидуальное задание №2 – необходимо на основе введённых координат определить, лежит ли точка в кольце между окружностями x^2+y^2 = 1 и x^2+y^2 = 0,25.

```
1   #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4   # Вариант - 28. Задание № 2, пункт №5.
5   if __name__ == '__main__':
6       a = float(input("Good day! Please, enter coordinates of A, a - "))
7       b = float(input("And now b - "))
8       useful = a * a + b * b
9       if 1 > useful > 0.25:
10          print("Yep! It's somewhere between 1 and 0.25.")
11      elif 1 == useful or useful == 0.25:
12          print("Oof! It's exactly on a border of given ring!")
13      else:
14          print("Oops! It's not inside.")
15
```

Рисунок 11. Полученный код индивидуального задания №2.

```
Good day! Please, enter coordinates of A, a - 1
And now b - 0
Oof! It's exactly on a border of given ring!
```

Рисунок 12. Полученный результат индивидуального задания №2.

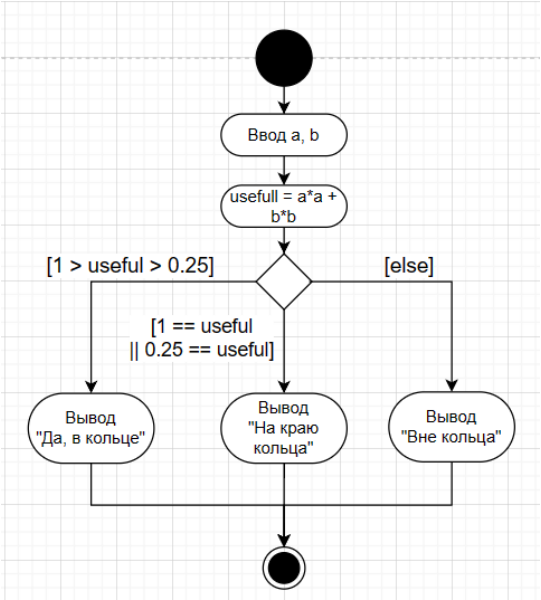

Рисунок 13. Полученная UML-диаграмма индивидуального задания №2.

8. Выполнено индивидуальное задание №3 – программа должна выводить таблицу перевода из килограммов в фунты.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# Вариант - 28. Задание № 3, пункт №6.
pounds = int(input("Good day! How many pounds do we need to convert? - "))
print("Very well:\n|     pounds     |       kg       |\n" + "-"*35)
for i in range(1, pounds + 1):
    calculation = round(i * 0.4, 1)
    # print(i, "pounds are equal to ", round(i * 0.4, 1), "kg.")
    print("|", " "*(7 - len(str(i))), i, "      |", " "*(7 - len(str(calculation))),
          calculation, "      |\n" + "-"*35)

```

Рисунок 14. Полученный код индивидуального задания №3.

```
Good day! How many pounds do we need to convert? - 12
Very well:
|     pounds     |       kg       |
-----------------------------------
|       1        |      0.4       |
-----------------------------------
|       2        |      0.8       |
-----------------------------------
|       3        |      1.2       |
-----------------------------------
|       4        |      1.6       |
-----------------------------------
|       5        |      2.0       |
-----------------------------------
|       6        |      2.4       |
-----------------------------------
|       7        |      2.8       |
-----------------------------------
|       8        |      3.2       |
-----------------------------------
|       9        |      3.6       |
-----------------------------------
|      10        |      4.0       |
-----------------------------------
|      11        |      4.4       |
-----------------------------------
|      12        |      4.8       |
-----------------------------------
```

Рисунок 15. Полученный результат индивидуального задания №3.

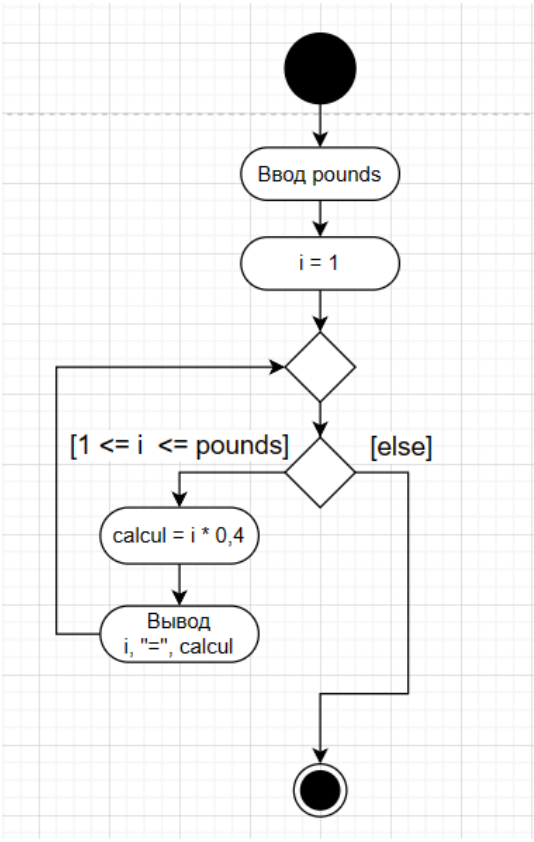

Рисунок 16. UML-диаграмма индивидуального задания №3.

9. Выполнено задание повышенной сложности, №1 – Интегральный синус.

$$\mathrm{Si}(x) = \int_0^x \frac{\sin t}{t}\,dt = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)(2n+1)!}.$$

Точность	10^(-10)

Чтобы вычислить сумму ряда найдем рекуррентное соотношение, позволяющее определить следующий член ряда исходя из значения текущего. Для этого разделим следующий член ряда на текущий. Текущий член ряда:

an = ((-1)^n * x^(2n+1))/((2n+1)*(2n+1)!) = $\frac{(-1)^n x^{2n+1}}{(2n+1)(2n+1)!}$

Следующий член ряда:

a(n+1) = ((-1)^(n+1) * x^(2n+3))/((2n+3)*(2n+3)!) = $\frac{(-1)^{n+1} x^{2n+3}}{(2n+3)(2n+3)!}$

Так как (2n+3)! = (2n+2)! * (2n+3) = (2n+1)! * (2n+3) * (2n+2) => a(n+1) = $\frac{(-1)^{n+1} x^{2n+3}}{(2n+3)(2n+3)(2n+2)(2n+1)!}$

Соотношение a(n+1)/an:

$$\frac{((-1)^{(n+1)} * x^{(2n+3)})/((2n+3)*(2n+1)! * (2n+3) * (2n+2))}{((-1)^n * x^{(2n+1)})/((2n+1)*(2n+1)!)} \quad \frac{-1 * ((-1)^n x^{2n+3})\,(2n+1)(2n+1)!}{(2n+3)(2n+3)(2n+2)(2n+1)!\,(-1)^n x^{2n+1}}$$

Полученное соотношение:

$$a(n+1) = \frac{-1 * (x^2)\,(2n+1)}{(2n+3)(2n+3)(2n+2)} * an$$

Помимо выражения, связывающего an и a(n+1), для вычисления значения рекуррентного соотношения необходимо найти значение первого члена ряда. В данном случае:

$$a(0) = \frac{x}{1} = x$$

Рисунок 17. Полученные расчёты для выполнения задания.

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import math
import sys


# Вариант - 28. Задание Повышенной сложности, №1 - Интегральный синус.
if __name__ == '__main__':
    x = int(input("Good day! Please, enter x - "))
    eps = 1e-10
    a = x  # a, когда n = 0.
    cycle_sum, n = a, 0
    while math.fabs(a) > eps:
        a *= -((x**2 * (2*n + 1))/((2*n + 3)**2 * (2*n + 2)))
        cycle_sum += a
        n += 1
    print("According to our calculations, Si(", x, ") = ", cycle_sum)

```

Рисунок 18. Полученный код усложнённого задания.

```
Good day! Please, enter x - 5
According to our calculations, Si( 5 ) =  1.5499312449439657
```

Рисунок 19. Полученный результат усложнённого задания.

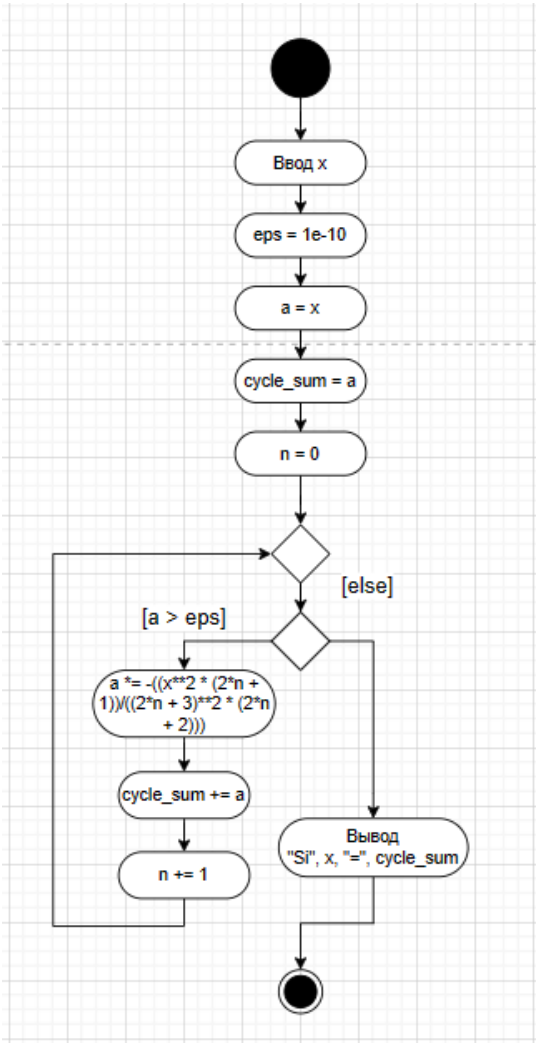

Рисунок 20. UML-диаграмма усложнённого задания.

10. Ответы на вопросы.

1) Для чего нужны диаграммы деятельности UML?

Ответ: 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. Диаграммы деятельности — это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов поведения системы. Диаграмма деятельности — это 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. Эта диаграмма показывает поток переходов от одной деятельности к другой (Деятельность (Activity) — это продолжающийся во времени неатомарный шаг вычислений в автомате). Деятельности в конечном счете приводят к выполнению некоего действия (Action), составленного из выполняемых атомарных вычислений, каждое из которых либо изменяет состояние системы, либо возвращает какое-то значение. Действие может заключаться в вызове другой операции, посылке сигнала, создании или уничтожении объекта либо в простом вычислении. Графически диаграмма деятельности представляется в виде графа, имеющего вершины и ребра.

2) Что такое состояние действия и состояние деятельности?

Ответ: в потоке управления, моделируемом диаграммой деятельности, происходят различные события. Все выполняемые атомарные вычисления (Вычислить выражение, в результате чего изменяется значение некоторого атрибута или возвращается некоторое значение; выполнить операцию над объектом - послать ему сигнал или даже создать его или уничтожить)

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 (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?

Ответ: когда действие или деятельность в некотором состоянии завершается, поток управления сразу переходит в следующее состояние действия или деятельности. Для описания этого потока используются переходы, показывающие путь из одного состояния действия или деятельности в другое. В UML переход представляется простой линией со стрелкой. Поток управления должен где-то начинаться и заканчиваться. Начальное состояние – закрашенный круг, в то время как конечное состояние – закрашенный круг внутри окружности.

4) Какой алгоритм является алгоритмом разветвляющейся структуры?

Ответ: простые последовательные переходы встречаются наиболее часто, но их одних недостаточно для моделирования любого потока управления. Как и в блок-схеме, можно включить в модель ветвление, которое описывает различные пути выполнения в зависимости от значения некоторого булевского выражения. Точка ветвления представляется ромбом, в неё 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. Для удобства разрешается использовать ключевое слово else для пометки того из исходящих переходов, который должен быть выбран в случае, если условия, заданные для всех остальных переходов, не выполнены. Реализовать итерацию можно, если ввести два состояния действия - в первом устанавливается значение счетчика, во втором оно увеличивается - и точку ветвления, вычисление в которой показывает, следует ли прекратить итерации.

5) Чем отличается разветвляющийся алгоритм от линейного?

Ответ: разветвляющийся алгоритм – 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 хотя бы один условный оператор. Для программной реализации условия используется логическое выражение. В сложных структурах с большим числом ветвей применяют оператор выбора. Линейный алгоритм - алгоритм, все этапы которого выполняются однократно и строго последовательно.

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

Ответ: условные операторы –специальные конструкции, благодаря которым в программе происходит ветвление в зависимости от условий. Могут быть представлены в полной и краткой формах.

7) Какие операторы сравнения используются в Python?

Ответ: в Python присутствуют такие операторы сравнения: if, else, elif.

8) Что называется простым условием? Приведите примеры.

Ответ: простое условие - выражение, которое анализирует только одно условие. Оно используется для принятия решений в коде на основе значения переменных или других условий. Примеры: if True/ if smth > 37/ if bool_thing.

9) Что такое составное условие? Приведите примеры.

Ответ: составное условие – логическое выражение, содержащее несколько простых условий, объединенных логическими операциями (not, and, or). Пример: (m == “smth” or m != “not_smth”).

10) Какие логические операторы допускаются при составлении сложных условий?

Ответ: В сложных условиях можно использовать операторы and, or, not.

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

Ответ: Оператор ветвления может быть вложен в другой оператор ветвления.

12) Какой алгоритм является алгоритмом циклической структуры?

Ответ: алгоритм циклической структуры — 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?

Ответ: в Python есть цикл for (выполняет указанный набор инструкций заданное количество раз) и цикл while (выполняет указанный набор инструкций до тех пор, пока условие цикла истинно).

14) Назовите назначение и способы применения функции range.

Ответ: range(“начало”, “конец”, “шаг”) возвращает неизменяемую последовательность чисел в виде объекта range, где начало - с какого числа начинается последовательность (по умолчанию = 0); конец - до какого числа (не включительно) продолжается последовательность чисел; шаг - с каким шагом растут числа (по умолчанию 1).

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

Ответ: команда будет выглядеть: for smth in range(15, 0, -2).

16) Могут ли быть циклы вложенными?

Ответ: Цикл может быть вложен в другой цикл.

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

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

18) Для чего нужен оператор break?

Ответ: break предназначен для досрочного прерывания работы цикла.

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

Ответ: continue завершает текущий круг цикла, переходя к следующему, при этом не выполняя следующий после этого оператора код.

20) Для чего нужны стандартные потоки stdout и stderr?

Ответ: в Python есть стандартные потоки: stdout — стандартный вывод (экран), stderr — стандартная ошибка (вывод ошибок на экран).

21) Как в Python организовать вывод в стандартный поток stderr?

Ответ: необходимо наличие библиотеки sys и использовать команды print(“сообщение”, file=sys.stderr).

22) Каково назначение функции exit?

Ответ: функция exit завершает выполнение программы в Python.

Вывод: в ходе выполнения лабораторной работы были исследованы операторы языка Python версии 3.x: if, while, for, break и continue, позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической

структуры. Также, приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры.