

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

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №5
дисциплины
«Основы кроссплатформенного программирования»

Выполнил:
Бакулин Вадим Романович
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи», очная
форма обучения

(подпись)

Проверил:
Доцент департамента цифровых,
робототехнических систем и
электроники Воронкин Р.А.

(подпись)

Отчет защищен с оценкой _____
защиты _____

Дата

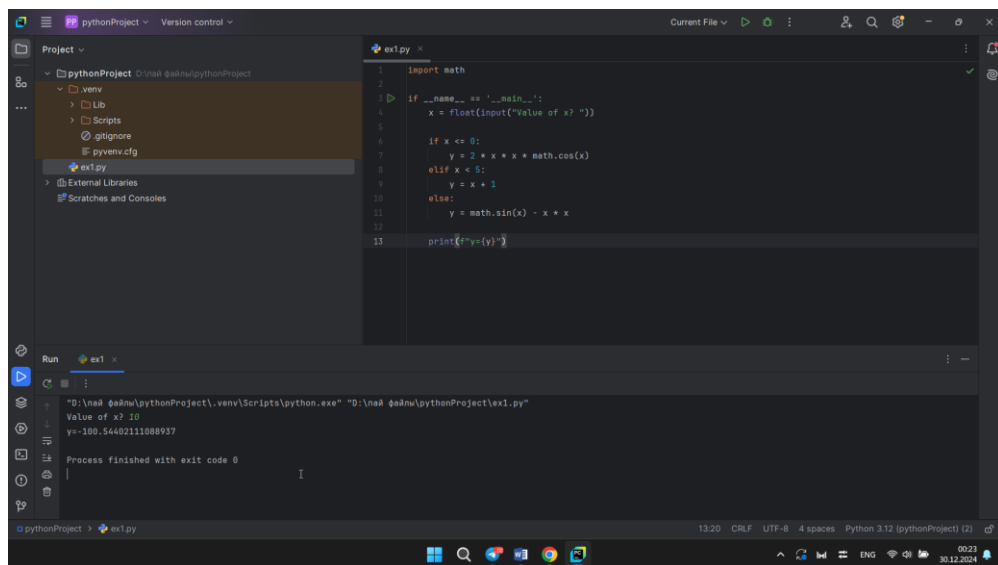
Ставрополь, 2024 г.

Тема: Условные операторы и циклы в языке Python

Цель: исследовать условные операторы и циклы в языке Python

Порядок выполнения работы:

1. Изучил теоретический материал.
2. Приступил к выполнению заданий.
3. Создал общедоступны репозиторий.
4. Выполнил клонирование репозитория
5. Изучил рекомендации PEP-8
6. Приступил к проработке примеров

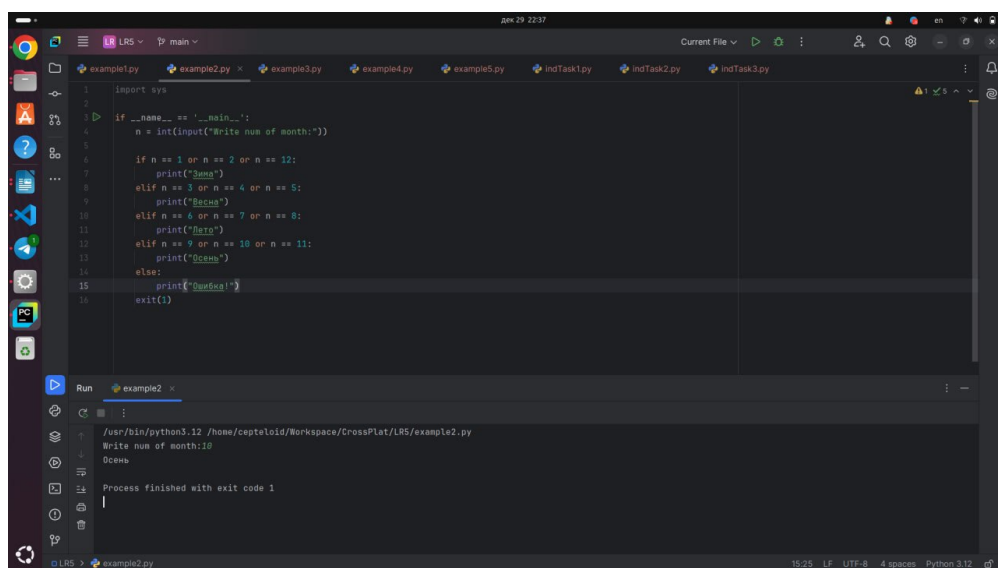


```
1 import math
2
3 if __name__ == '__main__':
4     x = float(input("Value of x? "))
5
6     if x <= 0:
7         y = 2 * x * x + math.cos(x)
8     elif x < 5:
9         y = x + 1
10    else:
11        y = math.sin(x) - x * x
12
13    print(f"y={y}")
```

Run ex1 x

Output: Value of x? 10
y=-100.54402111088937
Process finished with exit code 0

Рисунок 1. Первый пример

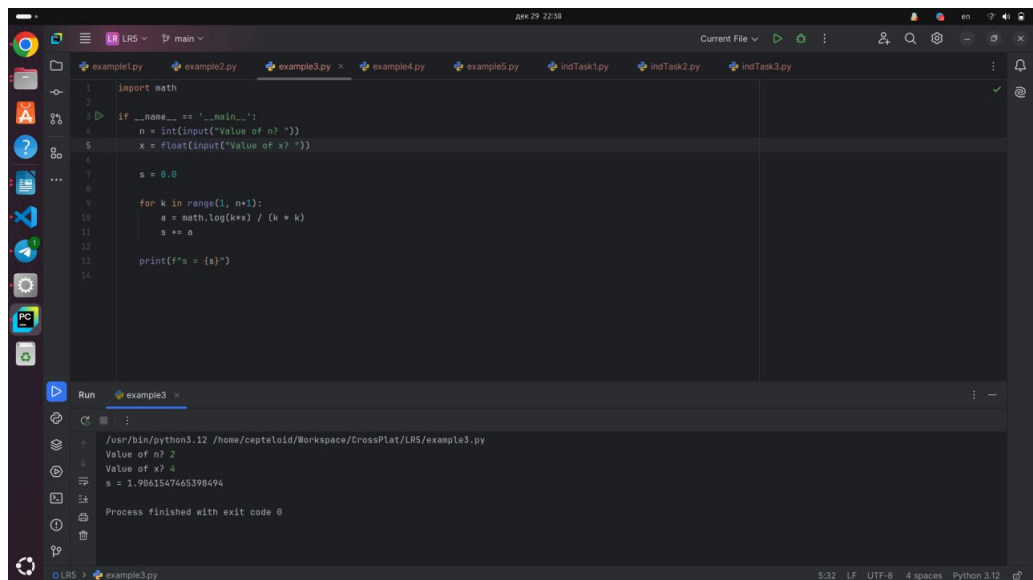


```
1 import sys
2
3 if __name__ == '__main__':
4     n = int(input("Write num of month:"))
5
6     if n == 1 or n == 2 or n == 12:
7         print("3mes")
8     elif n == 3 or n == 4 or n == 5:
9         print("4mes")
10    elif n == 6 or n == 7 or n == 8:
11        print("6mes")
12    elif n == 9 or n == 10 or n == 11:
13        print("5mes")
14    else:
15        print("Error")
16    exit(1)
```

Run example2

Output: Write num of month:10
0mes
Process finished with exit code 1

Рисунок 2. Второй пример

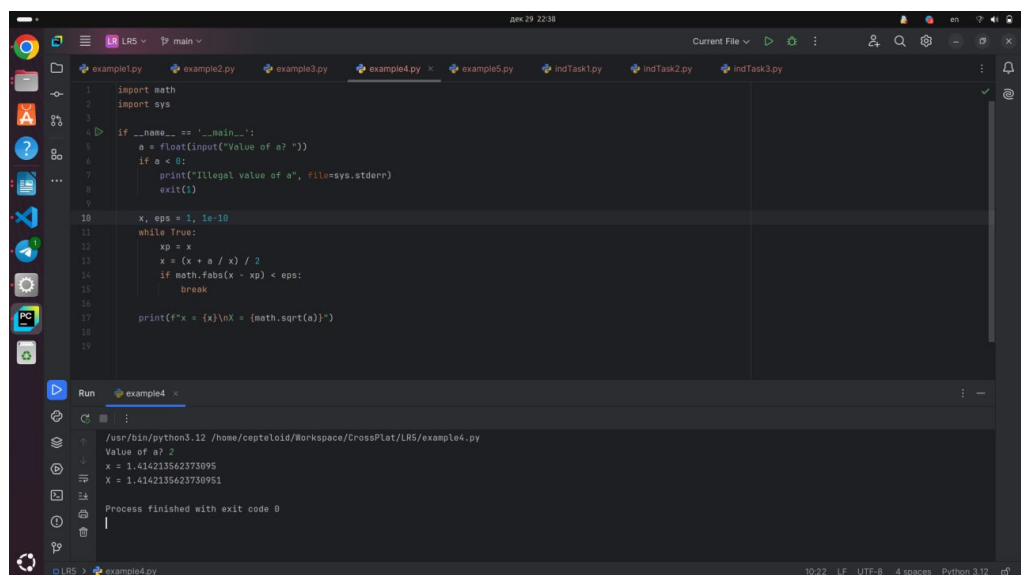


```
1 import math
2
3 if __name__ == '__main__':
4     n = int(input("Value of n? "))
5     x = float(input("Value of x? "))
6
7     s = 0.0
8
9     for k in range(1, n+1):
10         a = math.log(k*x) / (k + x)
11         s += a
12
13     print(f"s = {s}")
14
```

Run example3

```
/usr/bin/python3.12 /home/cepteloid/Workspace/CrossPlat/LRS/example3.py
Value of n? 2
Value of x? 4
s = 1.9861547465398494
Process finished with exit code 0
```

Рисунок 3. Третий пример

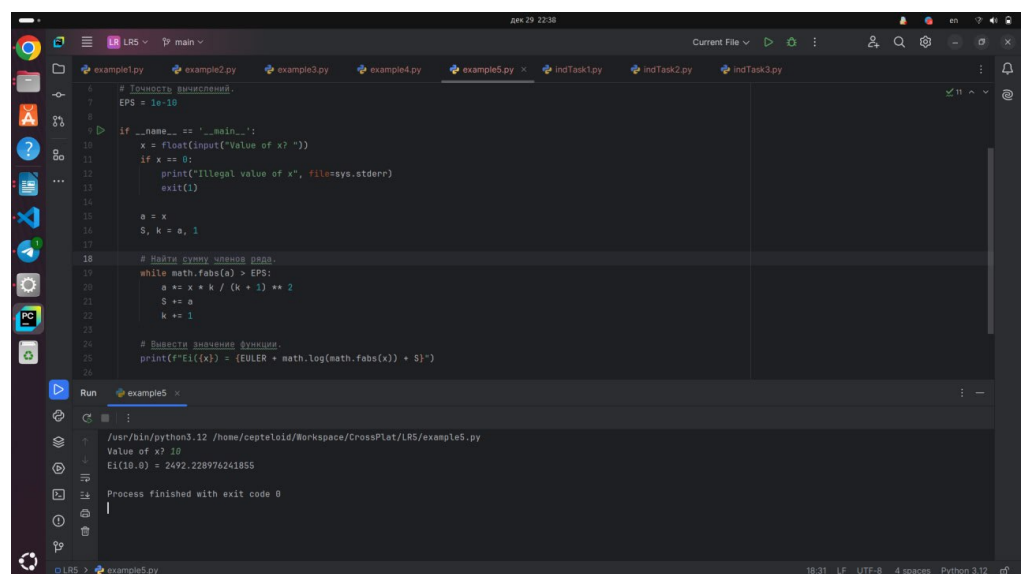


```
1 import math
2 import sys
3
4 if __name__ == '__main__':
5     a = float(input("Value of a? "))
6     if a < 0:
7         print("Illegal value of a", file=sys.stderr)
8         exit(1)
9
10     x, eps = 1, 1e-10
11     while True:
12         xp = x
13         x = (x + a / x) / 2
14         if math.fabs(x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

Run example4

```
/usr/bin/python3.12 /home/cepteloid/Workspace/CrossPlat/LRS/example4.py
Value of a? 2
x = 1.414213562373095
X = 1.4142135623730951
Process finished with exit code 0
```

Рисунок 4. Четвертый пример



```
1 # Константа Эйлера-Маскерони.
2 EPS = 1e-10
3
4 if __name__ == '__main__':
5     x = float(input("Value of x? "))
6     if x == 0:
7         print("Illegal value of x", file=sys.stderr)
8         exit(1)
9
10     a = x
11     S, k = a, 1
12
13     # Найти сумму членов ряда.
14     while math.fabs(a) > EPS:
15         a = x * k / (k + 1) ** 2
16         S += a
17         k += 1
18
19     # Вывести значение функции.
20     print(f"Ei({x}) = (EULER + math.log(math.fabs(x)) + S)")
21
```

Run example5

```
/usr/bin/python3.12 /home/cepteloid/Workspace/CrossPlat/LRS/example5.py
Value of x? 10
Ei(10.0) = 2492.228976241855
Process finished with exit code 0
```

Рисунок 5. Пятый пример

7. В ходе выполнения лабораторной работы были выполнены индивидуальные задания:

```
def age_phrase(n):  
    if 11 <= n % 100 <= 19:  
        word = "лет"  
    else:  
        last_digit = n % 10  
        if last_digit == 1:  
            word = "год"  
        elif 2 <= last_digit <= 4:  
            word = "года"  
        else:  
            word = "лет"  
  
    print(f"Мне {n} {word}")
```

Пример использования

```
n = int(input("Введите натуральное число меньше 100: "))  
if 1 <= n < 100:  
    age_phrase(n)  
else:  
    print("Число должно быть от 1 до 99!")
```

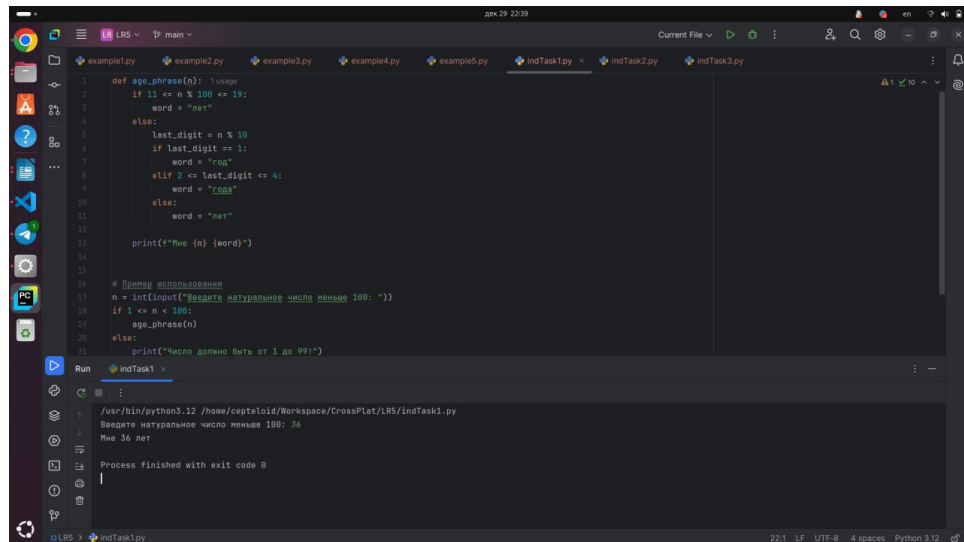


Рисунок 6. Вывод программы первого индивидуального задания

def japanese_calendar(year):

Названия животных в подцикле

```

animals = [
    "мышь", "корова", "тигр", "заяц", "дракон",
    "змея", "лошадь", "овца", "обезьяна", "курица",
    "собака", "свинья"
]

```

Названия цветов

```

colors = ["зеленый", "красный", "желтый", "белый", "черный"]

```

Начальный год цикла

```

base_year = 1984

```

Вычисляем смещение от начала цикла

```

offset = (year - base_year) % 60

```

Определяем животное и цвет

```

animal = animals[offset % 12]
color = colors[(offset // 12) % 5]

```

```
return f"{year} - год {color} {animal}"
```

Пример использования

```
year = int(input("Введите год: "))
```

```
if year >= 1984:
```

```
    print(japanese_calendar(year))
```

```
else:
```

```
    print("Год должен быть не меньше 1984!")
```

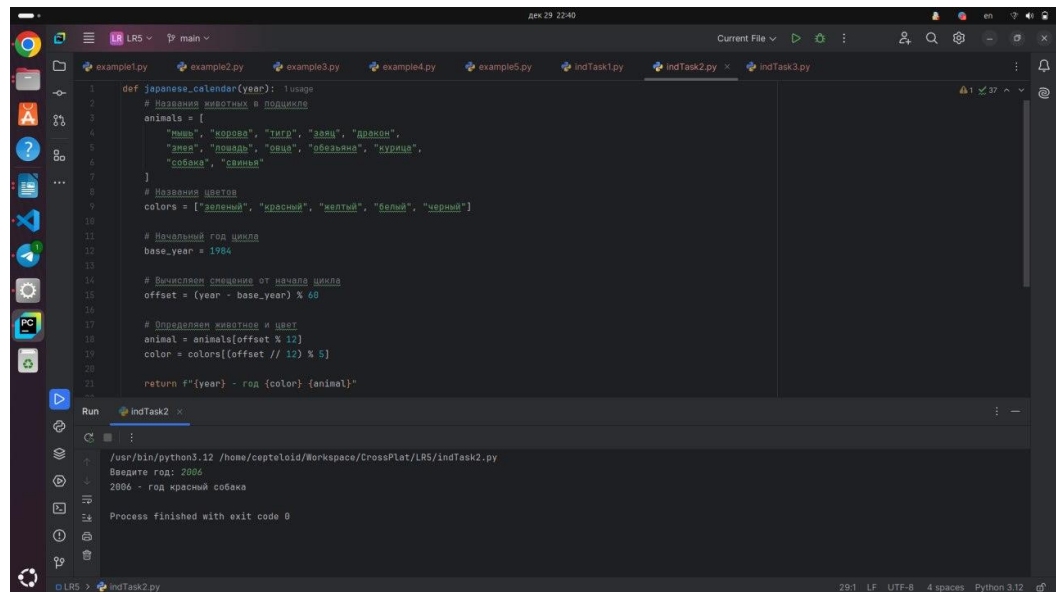


Рисунок 7. Вывод программы второго индивидуального задания

```
def find_numbers():
```

```
    results = []
```

```
    for number in range(100, 1000):
```

```
        # Разбиваем число на цифры
```

```
        a = number // 100 # Сотни
```

```
        b = (number // 10) % 10 # Десятки
```

```
        c = number % 10 # Единицы
```

```
        # Проверяем условие
```

```
        if a + b + c == a * b * c:
```

```
            results.append(number)
```

return results

Выводим результат

numbers = find_numbers()

print("Числа, удовлетворяющие условию:")

print(numbers)

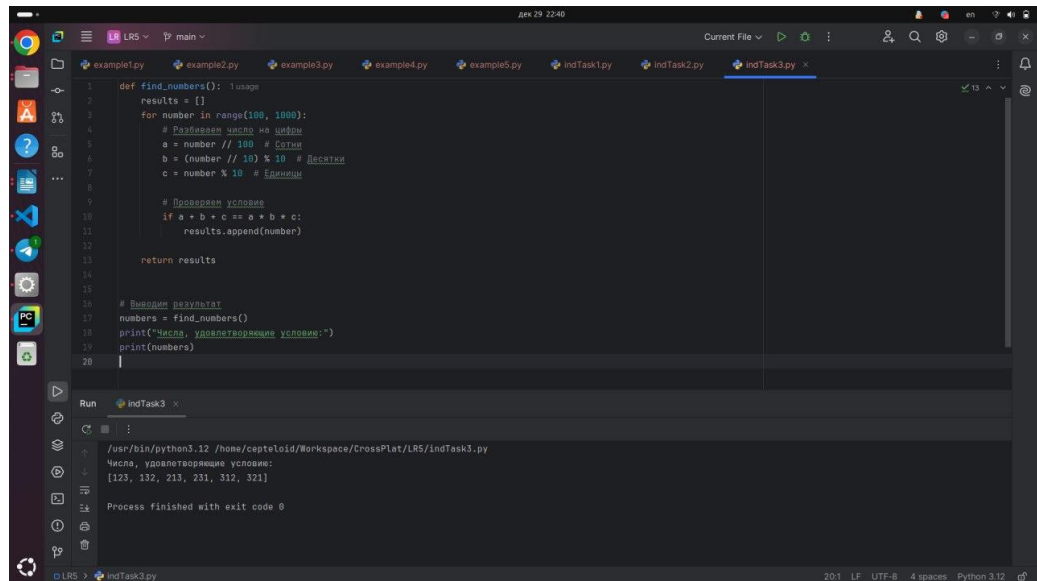


Рисунок 8. Вывод программы третьего индивидуального задания

8. В ходе выполнения лабораторной работы были проиндексированы файлы:

```
cepteloid@cepteloid-BoDE-WXX9:~/Workspace/CrossPlat/LR5$ git add .
cepteloid@cepteloid-BoDE-WXX9:~/Workspace/CrossPlat/LR5$ git commit -m "commit all files"
[main c385db9] commit all files
14 files changed, 191 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/LR5.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 example1.py
create mode 100644 example2.py
create mode 100644 example3.py
create mode 100644 example4.py
create mode 100644 example5.py
create mode 100644 indTask1.py
create mode 100644 indTask2.py
create mode 100644 indTask3.py
```

Рисунок 9. Проиндексированные файлы

9. Отправка на удаленный сервер:

```
and the repository exists.  
cepteloid@cepteloid-BoDE-WXX9:~/Workspace/CrossPlat/LR5$ git push  
Enumerating objects: 19, done.  
Counting objects: 100% (19/19), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (16/16), done.  
Writing objects: 100% (18/18), 3.93 KiB | 1.31 MiB/s, done.  
Total 18 (delta 0), reused 0 (delta 0), pack-reused 0  
To github.com:zepteloid/LR5.git  
351bd2d..c385db9 main -> main  
cepteloid@cepteloid-BoDE-WXX9:~/Workspace/CrossPlat/LR5$
```

Рисунок 10. Отправка на удаленный сервер

Вывод: в процессе выполнения изучены конструкции ветвлений и циклы в языке программирования Python.

1. Для чего применяются UML-диаграммы активности?

Эти диаграммы используют для отображения бизнес-процессов и алгоритмов. Они позволяют наглядно представить последовательность действий, условия и управление потоками в системе.

2. Что представляют собой состояния действия и активности?

Состояние действия обозначает конкретную операцию или задачу, которая выполняется. Состояние активности — это более общее понятие, которое может включать несколько действий.

3. Какие обозначения используются для переходов и ветвлений на диаграммах?

Переходы изображают стрелками, а ветвления — ромбами, где условия указываются рядом со стрелками.

4. Что такое алгоритм с разветвляющейся структурой?

Это алгоритм, в котором выбор действий определяется условиями, выполняющимися в зависимости от ситуации.

5. В чем отличие линейного алгоритма от разветвляющегося?

Линейный алгоритм выполняет шаги по порядку, без условий. Разветвляющийся алгоритм предполагает наличие условий, влияющих на выполнение различных действий.

6. Что называют условным оператором и какие его разновидности существуют?

Условный оператор дает возможность выполнять действия в зависимости от истинности условий. Основные формы: if, if-else, if-elif-else.

7. Какие операторы сравнения применяются в Python?

==, !=, >, <, >=, <=.

8. Что понимают под простым условием?

Это условие, выражающееся одним логическим утверждением.

9. Что такое составное условие?

Составное условие объединяет несколько логических утверждений через операторы and, or, not.

10. Можно ли вложить одно ветвление в другое?

Да, допускаются вложенные ветвления, что позволяет создавать сложные условия.

11. Как работает алгоритм с циклической структурой?

Такой алгоритм многократно повторяет действия, пока выполняется заданное условие.

12. Какие типы циклов есть в Python?

- for — для обхода элементов последовательности.
- while — выполняется, пока истинно условие.

13. Для чего предназначена функция range?

Эта функция генерирует последовательности чисел, часто используемые в циклах.

14. Как организовать перебор от 15 до 0 с шагом 2 через range?
for i in range(15, -1, -2):.

15. Допускаются ли вложенные циклы?

Да, вложенные циклы позволяют создавать сложные алгоритмы повторений.

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

Бесконечный цикл возникает, если условие выполнения всегда истинно. Завершить его можно оператором break или прерыванием программы.

17. Для чего используется оператор break?

Он прекращает выполнение цикла немедленно, вне зависимости от текущего состояния.

18. Когда применяют оператор `continue`?

Для пропуска текущей итерации цикла с переходом к следующей.

19. Какие стандартные потоки существуют для вывода данных?

`stdout` — для вывода информации, `stderr` — для вывода сообщений об ошибках.

20. Как выполнить вывод в стандартный поток `stderr`?

Через модуль `sys` (например, `sys.stderr.write`).

21. Какова цель функции `exit`?

Функция завершает выполнение программы, при этом можно указать код завершения (успех или ошибка).

Ссылка на GitHub: <https://github.com/zepteloid/LR5>