

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО - КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Основы кроссплатформенного программирования**

**Отчет по лабораторной работе №3**

**«Условные операторы и циклы в Python3»**

Выполнил студент группы

ИТС-б-о-20-1(2)

Игнатова Е.С. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А.

\_\_\_\_\_  
(подпись)

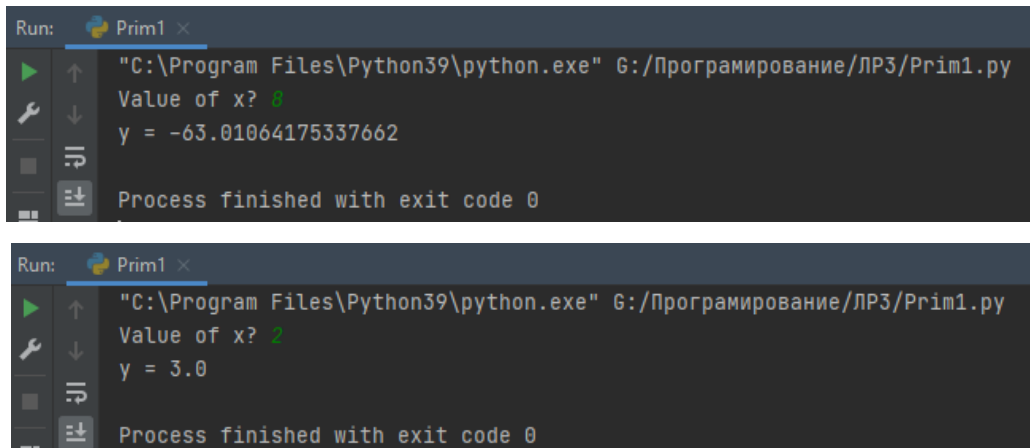
Ставрополь 2021

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

Ссылка на репозиторий: <https://github.com/Nebula139/Sky3/tree/main>

Ход работы:

### Пример 1

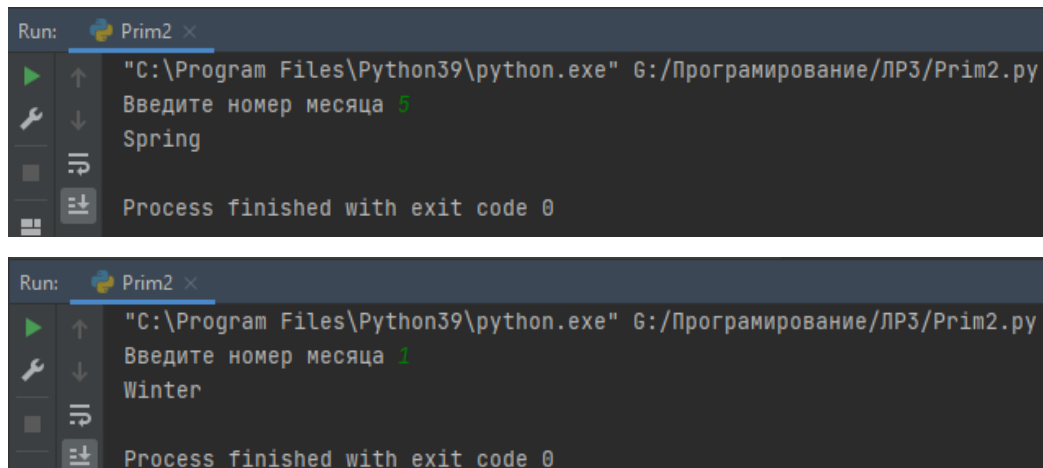


```
Run: Prim1 x
"C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim1.py
Value of x? 8
y = -63.01064175337662
Process finished with exit code 0

Run: Prim1 x
"C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim1.py
Value of x? 2
y = 3.0
Process finished with exit code 0
```

Рисунки 1 и 2. Результат выполнения программы с различными исходными данными (8 и 2 соответственно)

### Пример 2



```
Run: Prim2 x
"C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim2.py
Введите номер месяца 5
Spring
Process finished with exit code 0

Run: Prim2 x
"C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim2.py
Введите номер месяца 1
Winter
Process finished with exit code 0
```

Рисунки 3 и 4. Результат выполнения программы с различными исходными данными (5 и 1 соответственно)

### Пример 3

```
Run: Prim3 x
" C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim3.py
Value of n? 1
Value of x? 3
S= 1.0986122886681098
Process finished with exit code 0
```

```
Run: Prim3 x
" C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim3.py
Value of n? 2
Value of x? 5
S= 2.1850841856826118
Process finished with exit code 0
```

Рисунки 5 и 6. Результат выполнения программы с различными исходными данными (1;3 и 2;5 соответственно)

#### Пример 4

```
Run: Prim4 x
" C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim4.py
Value of a? 8
x = 2.82842712474619
X = 2.8284271247461903
Process finished with exit code 0
```

```
Run: Prim4 x
" C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim4.py
Value of a? 6
x = 2.449489742783178
X = 2.449489742783178
Process finished with exit code 0
```

Рисунки 7 и 8. Результат выполнения программы с различными исходными данными (8 и 6 соответственно)

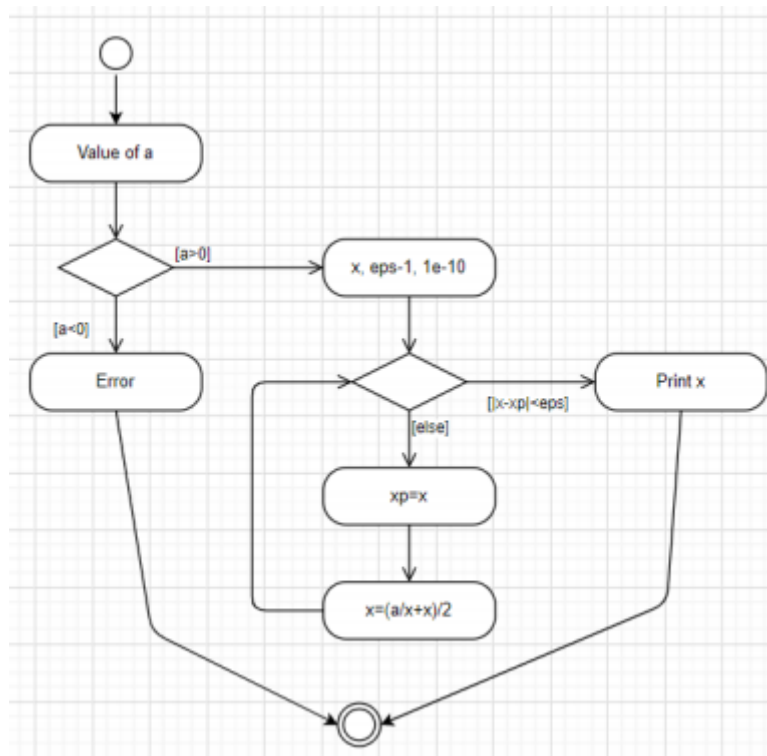


Рисунок 9. UML-диаграмма

### Пример 5.

```

Run: Prim5 x
"C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim5.py
Value of x? 5
Ei(5.0) = 40.18527535579794
Process finished with exit code 0

Run: Prim5 x
"C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/Prim5.py
Value of x? 6
Ei(6.0) = 85.98976214243285
Process finished with exit code 0
  
```

Рисунки 10 и 11. Результат выполнения программы с различными исходными данными (5 и 6 соответственно)

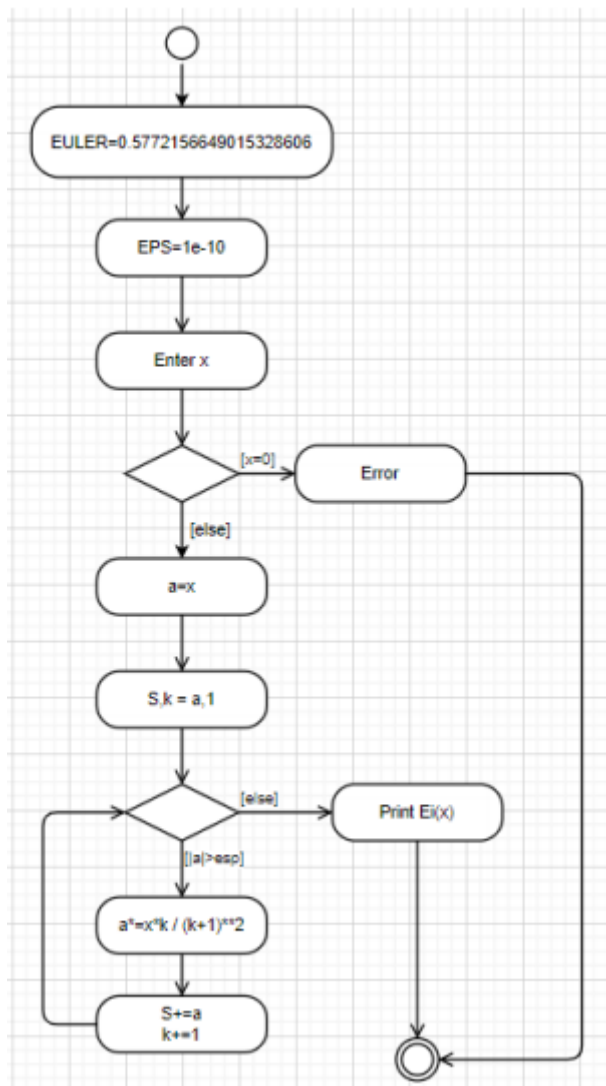


Рисунок 12. UML-диаграмма

### Индивидуальные задания:

**Задание 1.** С клавиатуры вводится цифра m (от 1 до 12). Вывести на экран название месяца, соответствующего цифре.

```

Run: ind x
"C:\Program Files\Python39\python.exe" G:/Программирование/ЛР3/ind.py
n = 3
Март
Process finished with exit code 0
  
```

Рисунок 13. Результат выполнения программы

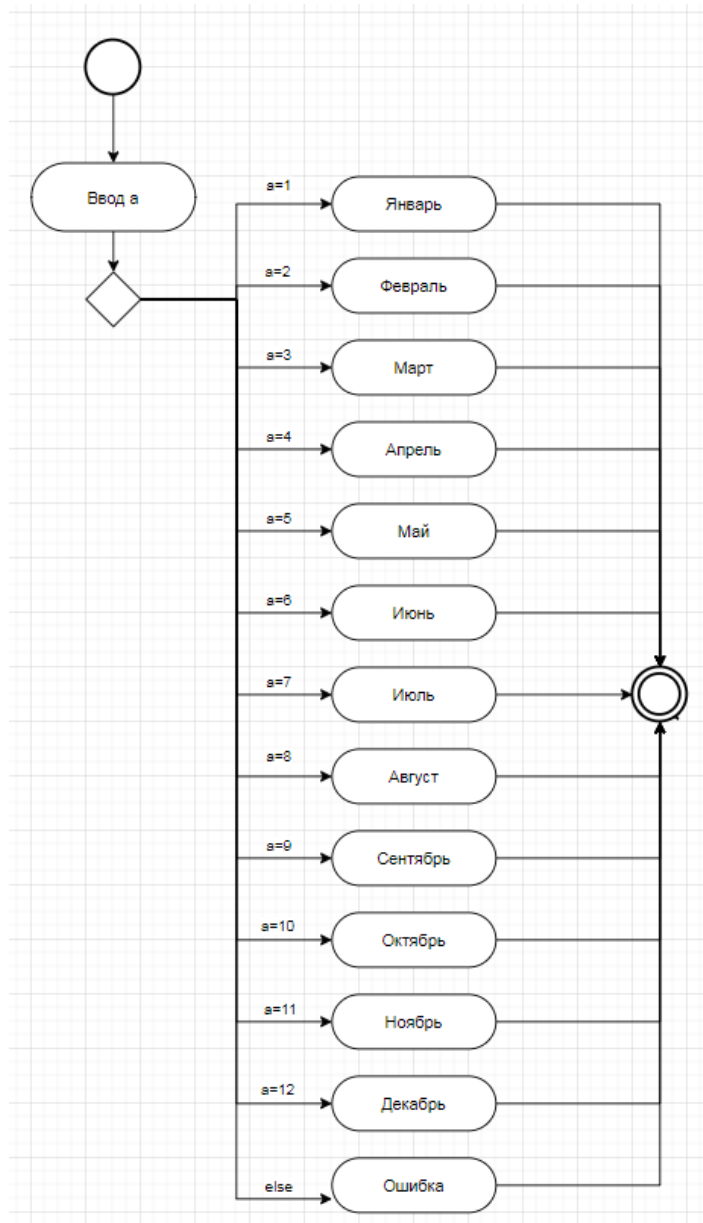


Рисунок 14. UML-диаграмма

**Задание 2.** Решить квадратное неравенство  $ax^2 + bx + c > 0 (a \neq 0)$ , где  $a$ ,  $b$  и  $c$  - действительные числа.

```

"C:\Program Files\Python39\python.exe" C:/Users/Елизавета/Downloads/ind2_1.py
Введите значение a= 0
Введите значение b= 3
Введите значение c= 2
Ошибка
  
```

Рисунок 15. Результат выполнения программы при  $a=0$

```
"C:\Program Files\Python39\python.exe" C:/Users/Елизавета/Downloads/ind2_1.py
Введите значение a= 3
Введите значение b= 0
Введите значение c= 5
Корней нет
```

Рисунок 16. Результат выполнения программы при  $D < 0$

```
"C:\Program Files\Python39\python.exe" C:/Users/Елизавета/Downloads/ind2_1.py
Введите значение a= 3
Введите значение b= 5
Введите значение c= 1
-0.2324081207560018
-1.434258545910665
```

Рисунок 17. Результат выполнения программы

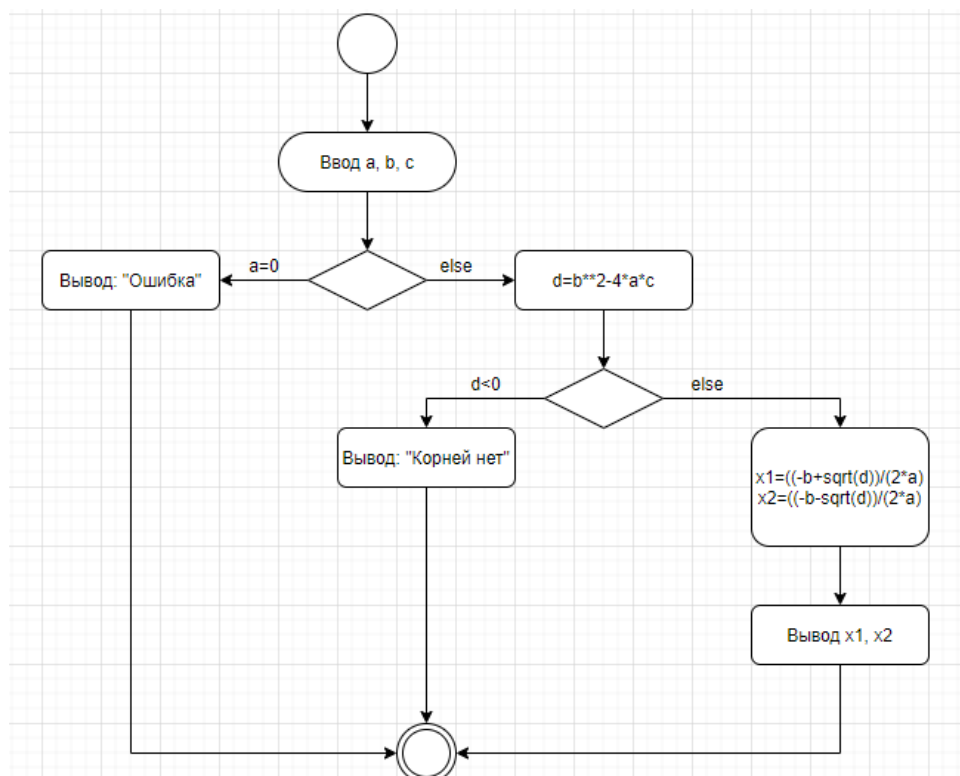


Рисунок 18. UML-диаграмма

### Задание 3.

```
"C:\Program Files\Python39\python.exe" G:/Програмирование/ЛР3/ind3.py
Введите время: 6
Через 6 часов будет 4 амёбы
```

Рисунок 19. Результат выполнения программы

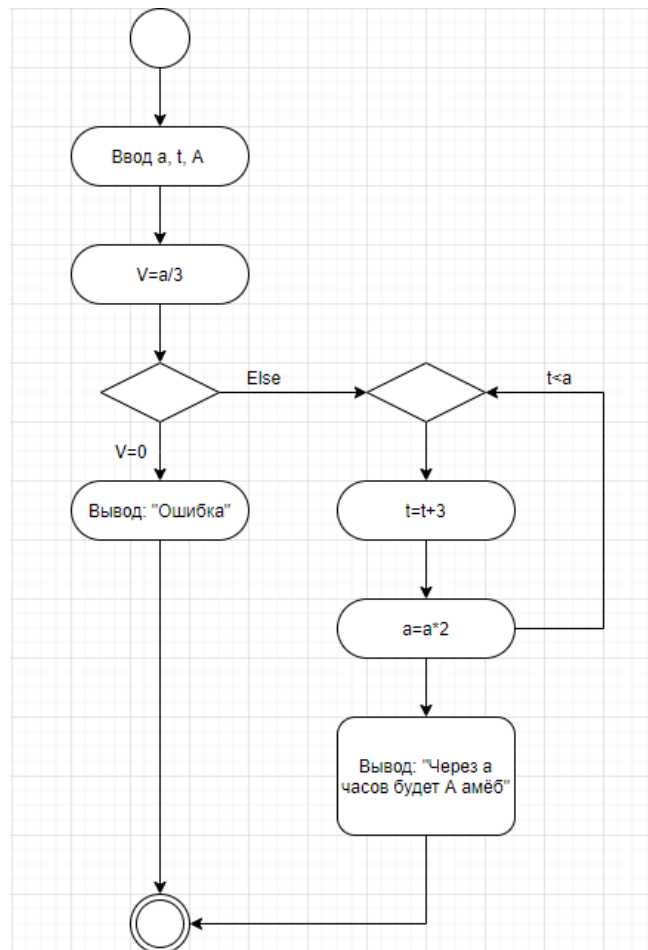


Рисунок 20. UML-диаграмма

**Вывод:** в ходе выполнения лабораторной работы были приобретены навыки программирования разветвляющихся алгоритмов и алгоритмов циклической структуры. Освоены операторы языка Python3 if, while, for, break и continue, позволяющие реализовывать разветвляющиеся алгоритмы и алгоритмы циклической структуры.

### Контрольные работы:

1. Для чего нужны диаграммы деятельности UML?

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?

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. 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?

В UML переход представляется простой линией со стрелкой. Точка ветвления представляется ромбом. В точку ветвления может входить ровно один переход, а выходить - два или более

## 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. 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?

В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в алгоритмическую структуру «ветвление» входит условие, в зависимости от выполнения или невыполнения которого реализуется та или иная последовательность команд.

#### 6. Что такое условный оператор? Какие существуют его формы?

Оператор ветвления `if` позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Возможны следующие варианты использования.

1. Конструкция `if`;
2. Конструкция `if – else`;
3. Конструкция `if – elif – else`.

#### 7. Какие операторы сравнения используются в Python?

Операторы сравнения используются для сравнения двух значений:

`==` Проверяет равны ли оба операнда. Если да, то условие становится истинным.

`!=` Проверяет равны ли оба операнда. Если нет, то условие становится истинным.

`<>` Проверяет равны ли оба операнда. Если нет, то условие становится истинным.

`>` Проверяет больше ли значение левого операнда, чем значение правого. Если да, то условие становится истинным.

`<` Проверяет меньше ли значение левого операнда, чем значение правого. Если да, то условие становится истинным.

`>=` Проверяет больше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным.

`<=` Проверяет меньше или равно значение левого операнда, чем значение правого. Если да, то условие становится истинным.

7. Что называется простым условием? Приведите примеры.

Простое условие — это два выражения, связанные одним из знаков отношений: = (равно), (больше), < (меньше), <= (меньше либо равно), >= (больше либо равно), != (не равно).

Если условие выполняется, то говорят, что условие истинно, в противном случае — условие ложно.

$a < 0$

$b$

$a + 2 * b = c / 3$

8. 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.

$(x = 0) \text{ or } (x = 8)$

$\text{not } (a = 0) \text{ or } (b = 0)$

9. Какие логические операторы допускаются при составлении сложных условий?

Сложное условие - состоит из двух или нескольких простых отношений (условий), которые объединяются с помощью логических операций: И - логическое умножение - на языке Python записывается как and, ИЛИ - логическое умножение - на языке Python записывается как or, НЕ - логическое умножение - на языке Python записывается как not.

10. Может ли оператор ветвления содержать внутри себя другие ветвления?

Ветви оператора if могут содержать любые допустимые к использованию внутри функции конструкции. Например, объявление переменных, ветвления и т.д. Ветвления могут быть вложены друг в друга.

11. Какой алгоритм является алгоритмом циклической структуры?

Циклический алгоритм – описание действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие. Перечень повторяющихся действий называют телом цикла.

12. Типы циклов в языке Python.

В Python есть два типа циклов: `for` и `while`.

13. Назовите назначение и способы применения функции `range`.

Функция `range` возвращает неизменяемую последовательность чисел в виде объекта `range`. `start` - с какого числа начинается последовательность. По умолчанию – 0 `stop` - до какого числа продолжается последовательность чисел. Указанное число не включается в диапазон `step` - с каким шагом растут числа. По умолчанию 1

14. Могут ли быть циклы вложенными?

Вложенными называются циклы, которые выполняются внутри других циклов.

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

Бесконечный цикл – это цикл, написанный таким образом, что он никогда не прекратит своё выполнение, так как никогда не выполнится условие выхода из этого цикла.

16. Для чего нужен оператор `break`?

В Python выражение `break` дает вам возможность выйти из цикла при активации внешнего условия.

17. Где употребляется оператор `continue` и для чего он используется?

Этот оператор используется в циклах `for`, `while` и необходим для того, чтобы прервать выполнение текущей итерации цикла с места его вызова и перейти к следующей итерации. Естественно, как и `break` оператор `continue` оператор является оператором завершающим тело условия.

18. Для чего нужны стандартные потоки `stdout` и `stderr`?

Стандартный вывод — программа пользователя записывает обычную информацию в этот дескриптор файла. Вывод возвращается через стандартный вывод (`stdout`).

Стандартная ошибка — программа пользователя записывает информацию об ошибке в этот дескриптор файла. Ошибки возвращаются через стандартную ошибку (`stderr`).

#### 19. Каково назначение функции `exit`?

Функция `exit()` модуля `sys` - выход из Python. Она реализуется путем вызова исключения `SystemExit`, поэтому выполняются действия по очистке, указанные в предложениях `finally` операторов `try` и можно перехватить попытку выхода на внешнем уровне.