

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФГАОУ ВО
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт цифрового развития

Кафедра инфокоммуникаций

дисциплина «Основы кроссплатформенного программирования»

Отчет по лабораторной работе №3

Условные операторы и циклы в языке Python

Выполнил: студент группы ИТС-б-о-21-1
Джу Алексей

(подпись)

Проверил: кандидат технических наук,

Доцент кафедры инфокоммуникаций

Роман Александрович Воронкин

(подпись)

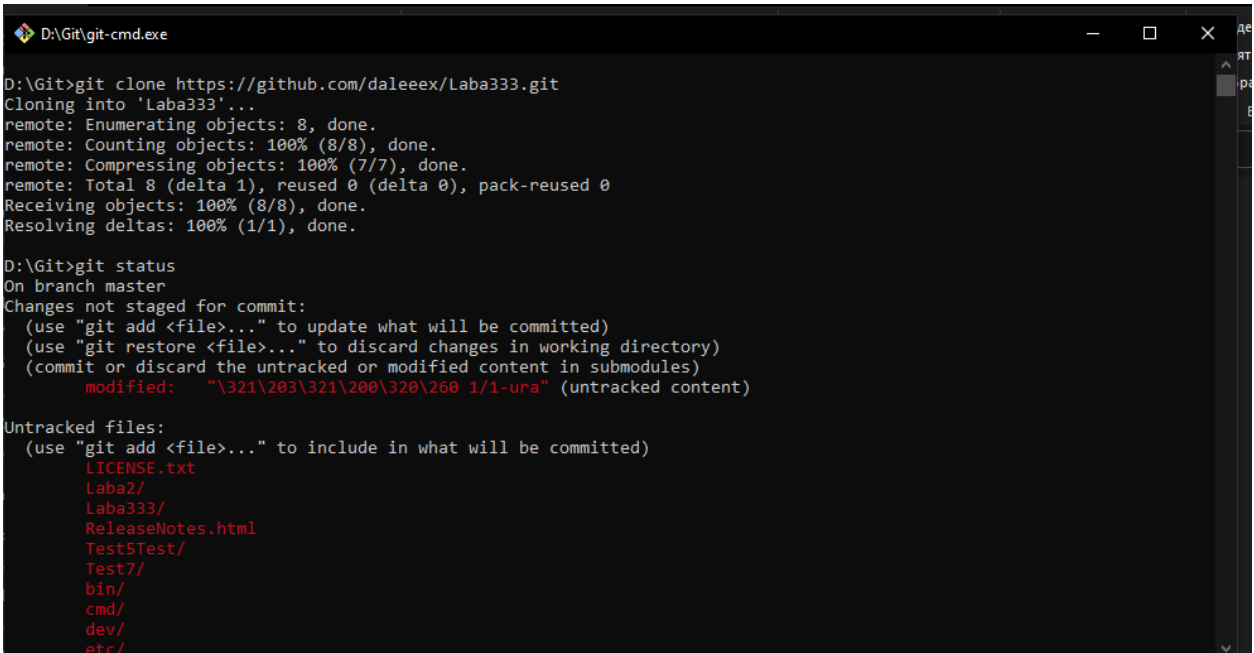
Ставрополь, 2022

Лабораторная работа №3. Условные операторы и циклы в языке Python

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.

Ход работы:

Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python. 3. Выполнил клонирование созданного репозитория.



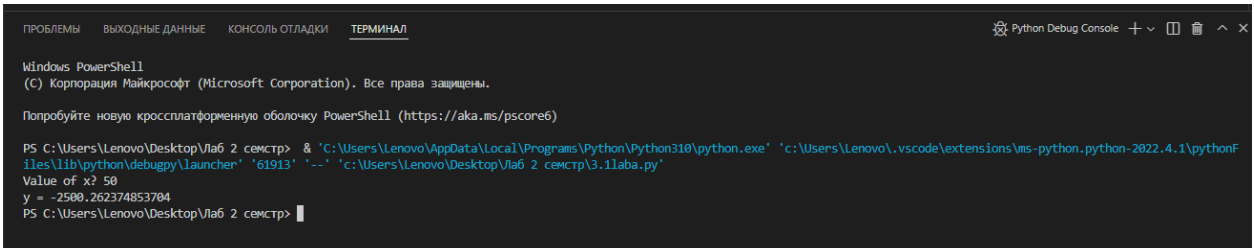
```
D:\Git>git clone https://github.com/daleeex/Laba333.git
Cloning into 'Laba333'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.

D:\Git>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
  (commit or discard the untracked or modified content in submodules)
        modified:   "\321\203\321\200\320\260 1/1-ura" (untracked content)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        LICENSE.txt
        Laba2/
        Laba333/
        ReleaseNotes.html
        Test5Test/
        Test7/
        bin/
        cmd/
        dev/
        etc/
```

Рисунок 1.1

Пример 1



```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ
Python Debug Console + - [ ] [x] ^ x

Windows PowerShell
(C) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

Попробуйте новую кроссплатформенную оболочку PowerShell (https://aka.ms/pscore6)

PS C:\Users\Lenovo\Desktop\Лаб 2 семстр> & 'C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '61913' '-.' 'c:\Users\Lenovo\Desktop\Лаб 2 семстр\3.1laba.py'
Value of x? 50
y = -2500.262374853704
PS C:\Users\Lenovo\Desktop\Лаб 2 семстр> |
```

Рисунок 1.2

Пример 2

```
Введите номер месяца 9
Autumn
PS C:\Users\Lenovo\Desktop>
```

Рисунок 1.3

Пример 3

```
PS C:\Users\Lenovo\Desktop\Лаб 2 семстр> & 'C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '61871' '--' 'c:\Users\Lenovo\Desktop\Лаб 2 семстр\3.31aba.py'
Value of n? 10
Value of x? 5
S= 3.1127575860454884
PS C:\Users\Lenovo\Desktop\Лаб 2 семстр>
```

Рисунок 1.4

Пример 4

```
PS C:\Users\Lenovo\Desktop\Лаб 2 семстр> & 'C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '62030' '--' 'c:\Users\Lenovo\Desktop\Лаб 2 семстр\3.41aba.py'
Value of a? 9
x = 3.0
X = 3.0
PS C:\Users\Lenovo\Desktop\Лаб 2 семстр>
```

Рисунок 1.5

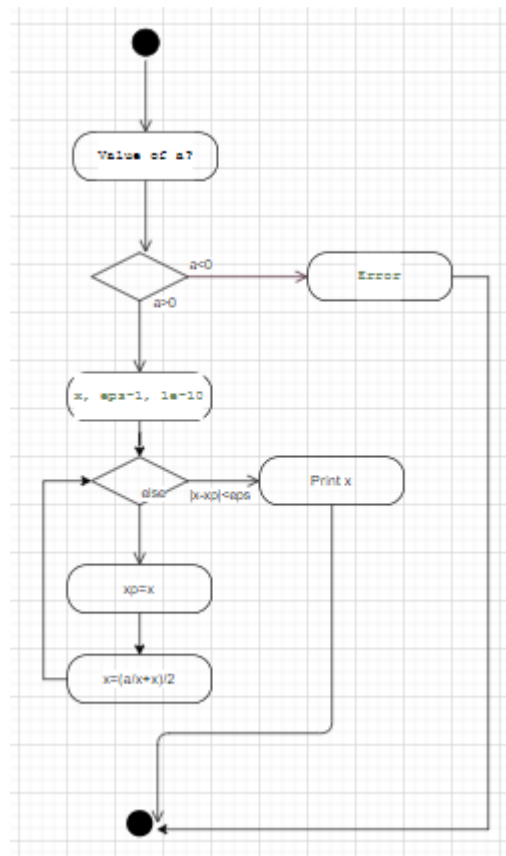


Рисунок 1.6

Пример 5

```
PS C:\Users\Lenovo\Desktop\Лаб 2 семстр> & 'C:\Users\Lenovo\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\Lenovo\.vscode\extensions\ms-python.python-2022.4.1\pythonFiles\lib\python\debugpy\launcher' '62851' '--' 'c:\Users\Lenovo\Desktop\Лаб 2 семстр\3.51aba.py'  
Value of x? 7  
Ei(7,0) = 191.50474333549477  
PS C:\Users\Lenovo\Desktop\Лаб 2 семстр> |
```

Рисунок 1.7

Индивидуальное задание:

12 - вариант

1) При покупке товара на сумму от 200 до 500 руб предоставляется скидка 3%, при покупке товара на сумму от 500 до 800 - скидка 5%, при покупке товара на сумму от 800 до 1000 руб - скидка 7%, свыше 1000 руб - скидка 9%. Покупатель приобрел 8 рулонов обоев по цене x_1 и две банки краски по цене x_2 . Сколько он заплатил?

```
Сумма покупки 300  
291.0  
291.0  
PS C:\Users\Lenovo\Desktop\Лаб 2 семстр> |
```

Рисунок 1.8

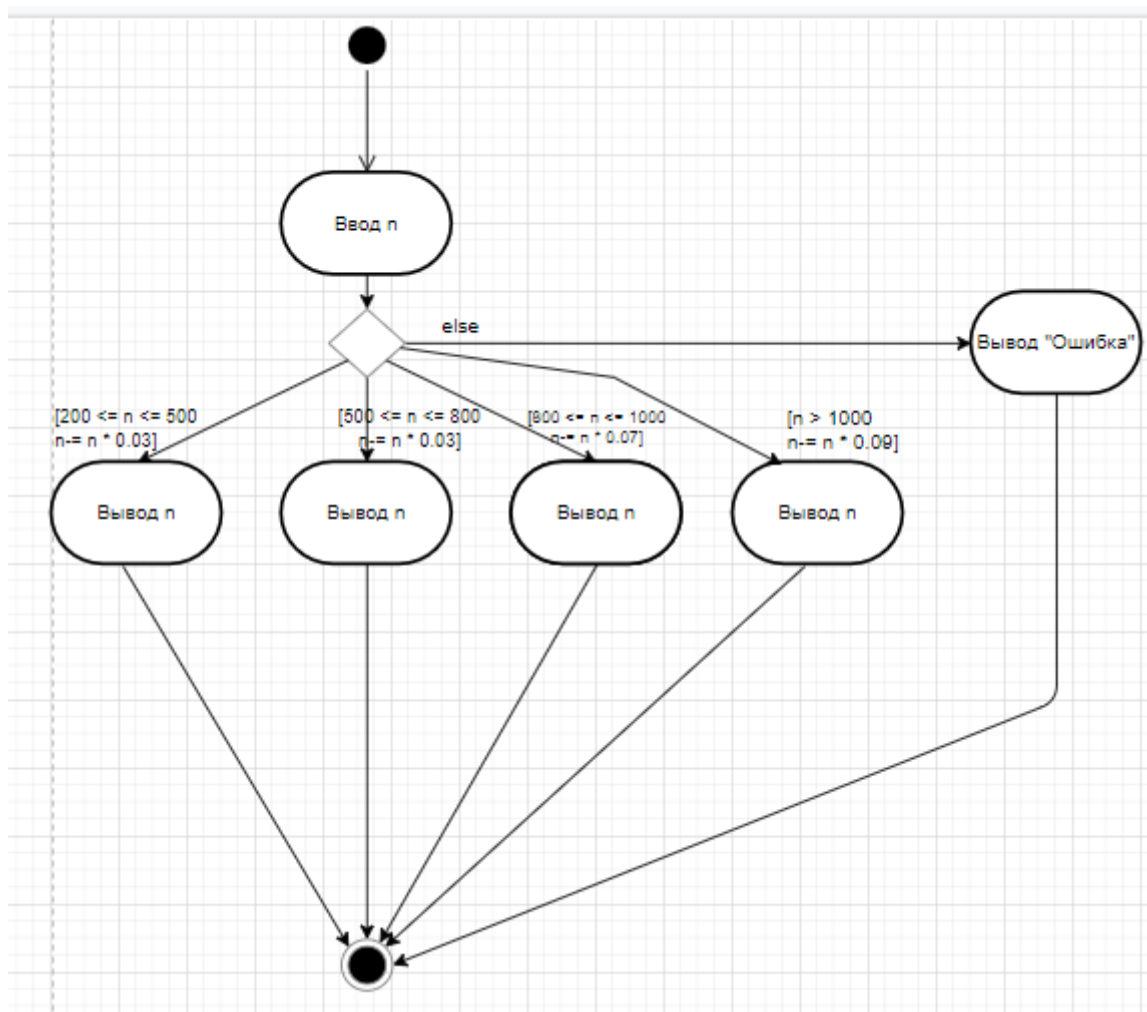


Рисунок 1.9

2) Две окружности заданы координатами центра и радиусами. Сколько точек пересечения имеют эти окружности?

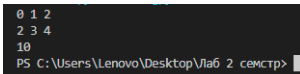


Рисунок 2

Ответы на вопросы:

1) Диаграммы деятельности - это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов поведения системы. Диаграмма деятельности - это, по существу, 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.

2) 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.

3) 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. В UML переход представляется простой линией со стрелкой

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. Как видно из рис. 4.3, точка ветвления представляется ромбом. В точку ветвления может входить ровно один переход, а выходить - два или более

4) 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) 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) Условные операторы в Python 3 иногда называют операторами ветвления. Они созданы, чтобы программа могла выбрать, какой инструкции стоит следовать при определенном значении заданной переменной. Встречаются следующие формы условного оператора: Условный оператор с одной ветвью, условный оператор с двумя ветвями условный оператор с несколькими условиями Реализация.

7) Операторы сравнения. Перечень

`==`, `!=` – операторы (операции) проверки на равенство (наивысший приоритет); `<`, `>`, `<=`, `>=` – операторы сравнения соответственно меньше, больше, меньше или равно, больше или равно

8) 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: $x < y$; $k \leq \sqrt{c} + \text{abs}(a+b)$; $9 \neq 11$; 'мама' <> 'папа'.

9) 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.

Пример составных условий:

$(a < 5) \text{ and } (b > 8)$

$(x \geq 0) \text{ or } (x < -3)$

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

10) В таких случаях используются специальные операторы, объединяющие два и более простых логических выражения. Широко используются два оператора – так называемые логические И (and) и ИЛИ (or).

11) В программе может присутствовать ветвление, которое реализуется условным оператором – особой конструкцией языка программирования.

12) Алгоритм циклической структуры - это алгоритм, в котором происходит многократное повторение одного и того же участка программы. Такие повторяемые участки вычислительного процесса называются циклами. Программа циклической структуры содержит один или несколько циклов. Различают детерминированные циклы с заранее известным числом повторений и итерационные циклы, в которых число повторений заранее неизвестно.

13) В мире Python есть два типа циклов:

Цикл for. Оператор for выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе.

Цикл while. Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно. Истинность условия определяется также как и в операторе if

14) Функция `range` возвращает неизменяемую последовательность чисел в виде объекта `range`. Синтаксис функции:

```
range(stop)
range(start, stop[, step])
```

Параметры функции:

`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

15) Если заданы два, то числа генерируются от первого до второго, не включая его. Если заданы три, то третье число – это шаг.

16) Python позволяет также создавать вложенные циклы. Так, сначала программа запустит внешний и в первой его итерации перейдет во вложенный. Затем она снова вернется к началу внешнего и снова вызовет внутренний. Это будет происходить до тех пор, пока последовательность не завершится или не прервется.

17) В большинстве случаев, бесконечные циклы появляются из-за логических ошибок программиста (например, когда условие цикла `while` при любых вариантах равно `True`). Поэтому следует внимательно следить за условием, при котором цикл будет завершаться. Чтобы остановить выполнение такого скрипта - в shell нужно нажать `Ctrl+C`.

18) Оператор `break` завершает выполнение ближайшего включающего цикла или условного оператора, в котором он отображается. Управление передается оператору, который расположен после оператора, при его наличии.

19) Оператор `continue` запускает цикл заново, при этом код, расположенный после данного оператора, не выполняется.

```
a = -1
while a < 10:
    a += 1
    if a >= 7:
        continue
    print("A")
```


20) В операционной системе по умолчанию присутствуют стандартных потока вывода на консоль: буферизованный поток `stdout` для вывода данных и информационных сообщений, а также небуферизованный поток `stderr` для вывода сообщений об ошибках. По умолчанию функция `print` использует поток `stdout`. Для того, чтобы использовать поток `stderr` необходимо передать его в параметре `file` функции `print`. Само же определение потоков `stdout` и `stderr` находится в стандартном пакете Python `sys`. Хорошим стилем программирования является наличие вывода ошибок в стандартный поток `stderr` поскольку вывод в потоки `stdout` и `stderr` может обрабатываться как операционной системой, так и сценариями пользователя по-разному.

21) Помимо вывода сообщения об ошибке необходимо как-то информировать операционную систему о некорректном завершении программы. Сделать это можно, передав операционной системе код возврата. Если программа завершается успешно, то она передает код возврата равный 0 (любая программа на Python делает это по умолчанию). Если в процессе выполнения программы произошли ошибки, программа должна передать операционной системе код возврата отличный от нуля. В Python завершить программу и передать операционной системе заданный код возврата можно посредством функции `exit`. В данном примере выполняется вызов `exit(1)`, что приводит к немедленному завершению программы и операционной системе передается 1 в качестве кода возврата, что говорит о том, что в процессе выполнения программы произошли ошибки.

22) В Python завершить программу и передать операционной системе заданный код возврата можно посредством функции `exit`.