Ссылка на урок: https://coreapp.ai/app/player/lesson/6751ae82f4de11afca0ad139

(для учеников)

Занятие 9

Тема: **Ветвления. Условный оператор**

1. Теоретическая часть

Возможности, которые мы рассмотрели до этого, позволяют писать *линейные* программы, в которых команды (действия) выполняются последовательно друг за другом. Но в большинстве реальных задач порядок выполнения команд может изменяться, например, в зависимости от исходных данных или от от результатов выполнения предыдущих этапов решения задачи.

То есть в алгоритм решения задачи вводится некоторое условие, в зависимости от результата проверки которого предусмотрен выбор одной из двух последовательностей действий (ветвей). Такой алгоритм называется **разветвляющимся**, а алгоритмическая конструкция (структура) – **ветвлением** (рис.1).

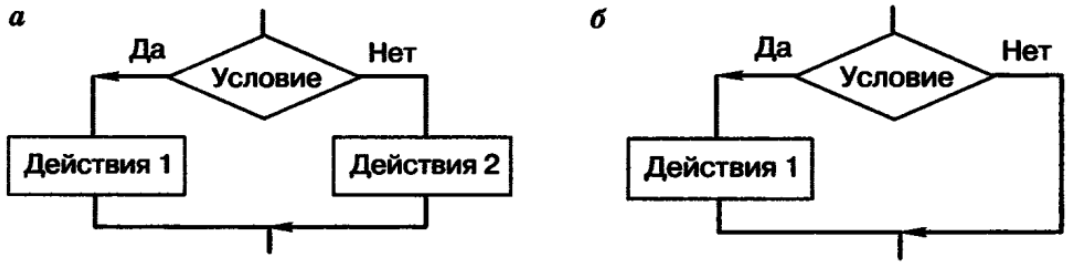


Рис. 1. Структура «ветвление»: а – полная форма ветвления; б – неполная форма ветвления

Ветвление в Python реализуется с помощью **условного оператора** или, по-другому, **оператора ветвления**. Его общий вид:

```
if <условие>:
      <блок_операторов_1>
else:
      <блок_операторов_2>
```

Слова **if** и **else** начинаются на одном уровне, а все команды ветвей «Да» или «Нет» (внутренних блоков) сдвинуты относительно этого уровня вправо на одно и то же расстояние. Начало и конец блока, который выполняется при истинности (ложности) условия, определяется именно этими сдвигами.

Обратите внимание! В Python сдвиги (отступы) операторов относительно левой границы влияют на работу программы. Для отступа можно использовать пробелы (обычно не меньше двух) или символы табуляции (вставляются при нажатии на клавишу *Tab).*

Если в условном операторе в ветвях «Да», «Нет» предполагается выполнить только по одному действию, то он записывается так:

```
if <условие>: <оператор_1>
else: <оператор_2>
```

Как видим, операторы в ветвях записываются без сдвигов.

Для записи **неполных ветвлений** используется неполная форма условного оператора:

```
if <условие>:
<оператор>
```

В качестве условий используются *логические выражения*: *простые* – записанные с помощью операций отношения <, >, >=, <=, != (не равно), == (равно) и *составные* – записанные с помощью логических операций **and, or, not**.

В языке Python разрешены двойные неравенства, например, A < B < C. Запись

```
if A < B < C:
```

означает то же самое, что и

```
if A < B and B < C:
```

2. Примеры программ с использованием условного оператора

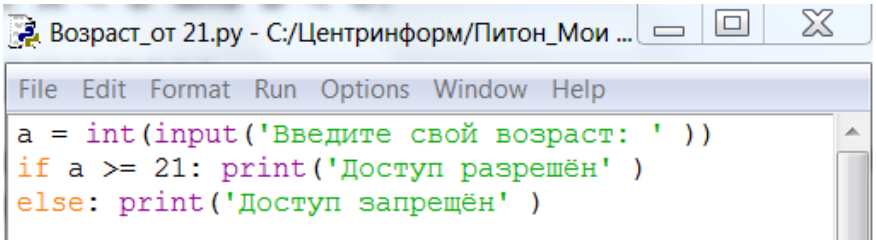
Задача 1. Написать программу, разрешающую доступ только для тех, чей возраст старше 21 года.

Решение.

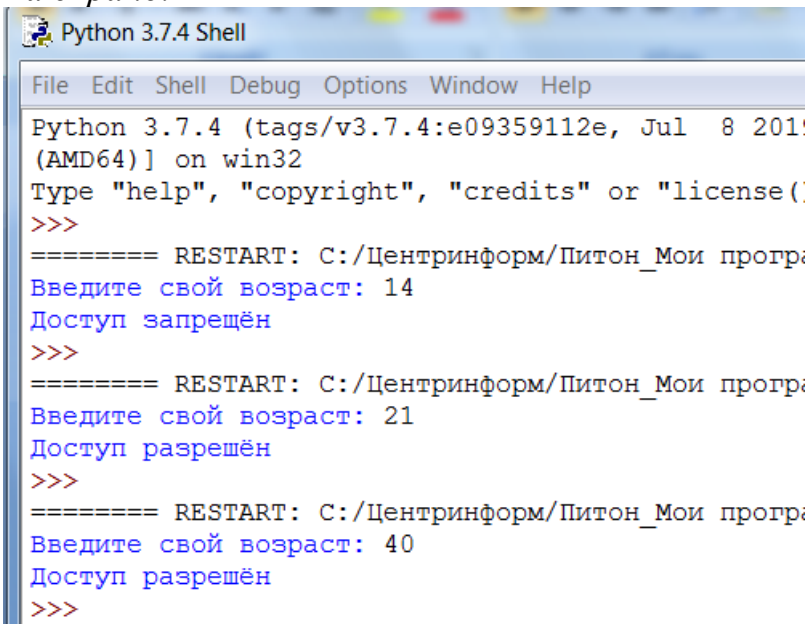
```
a = int(input('Введите свой возраст: ' ))
if a >= 21:
     print('Доступ разрешён' )
else:
     print('Доступ запрещён' )
```

ИЛИ

```
a = int(input('Введите свой возраст: ' ))
if a >= 21: print('Доступ разрешён' )
else: print('Доступ запрещён' )
```



На экране:

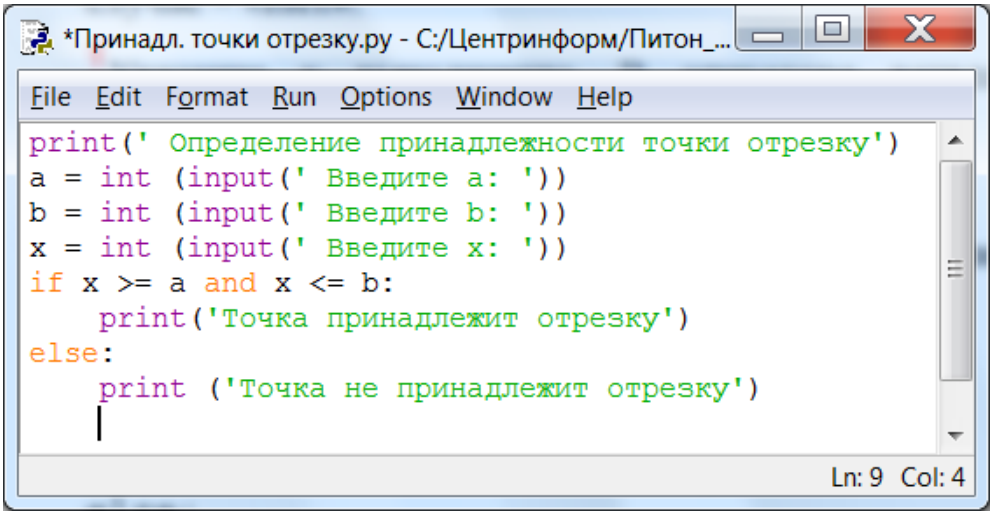


Задача 2. Написать программу, определяющую принадлежность точки x отрезку [a, b]. Если точка принадлежит данному отрезку, вывести ответ «Да», в противном случае – «Нет».

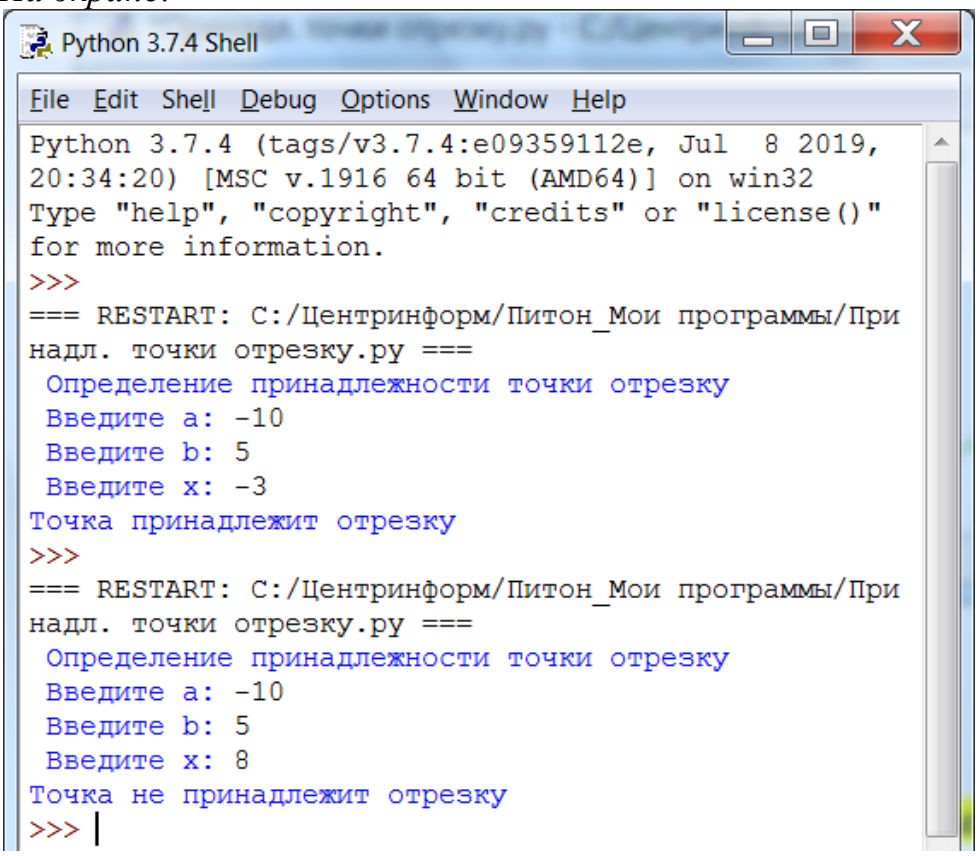
!Указание к выполнению. В операторе ветвления следует использовать составное условие.

Решение.

```
print(' Определение принадлежности точки отрезку')
a = int (input(' Введите a: '))
b = int (input(' Введите b: '))
x = int (input(' Введите x: '))
if x >= a and x <= b:
     print('Точка принадлежит отрезку')
else:
     print ('Точка не принадлежит отрезку')
```



На экране:

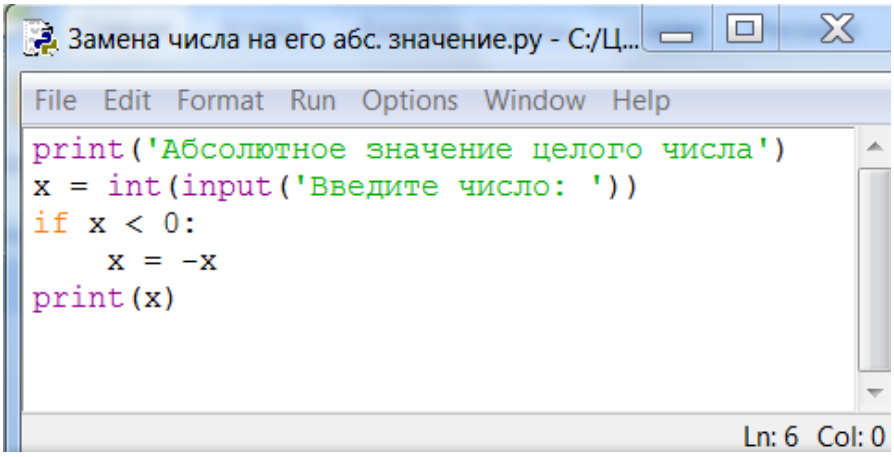


Задача 3. Написать программу, в которой введённое целое число x заменяется на его абсолютную величину.

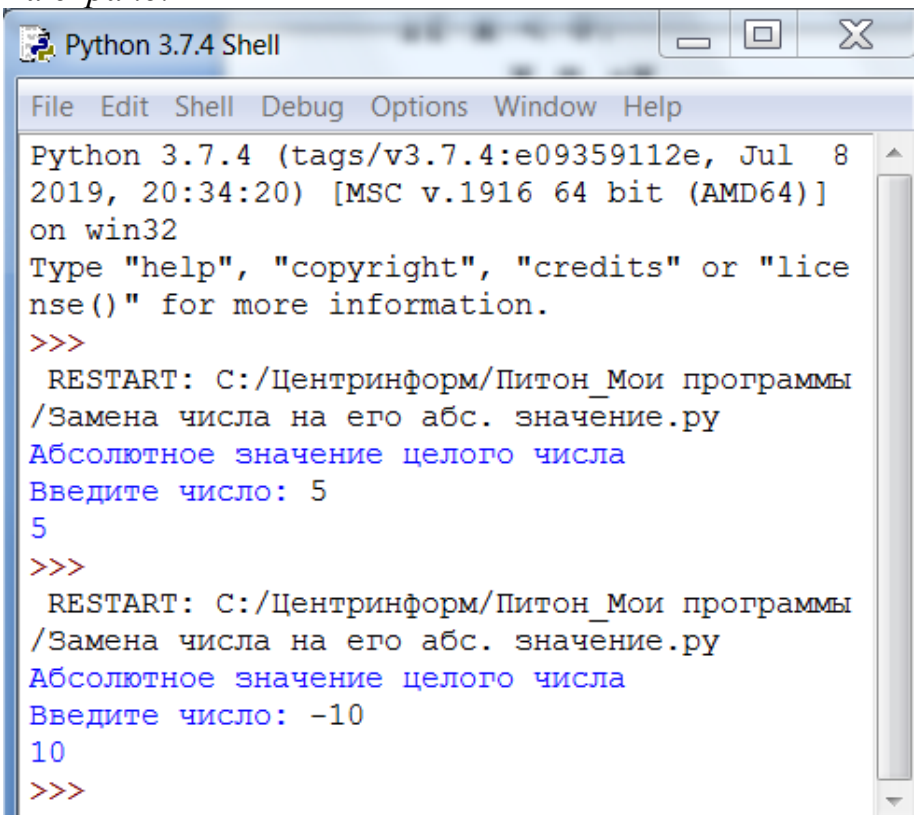
Решение.

```
print('Абсолютное значение целого числа')
x = int(input('Введите число: '))
if x < 0:
    x = -x
print(x)
```

Пояснение. В данной программе отсутствует альтернативная ветвь «Нет», поскольку в ней нет необходимости. Если условие истинно (true), то есть введённое число отрицательное, программа, с помощью команды **x = -x,** изменит его значение на противоположное и выведет на экран. Если условие ложно (falth), то есть введённое число положительное, программа просто выведет его на экран.



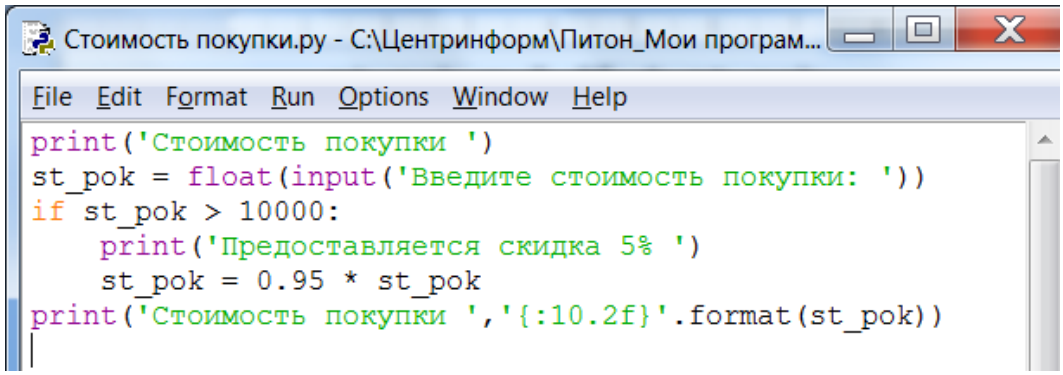
На экране:



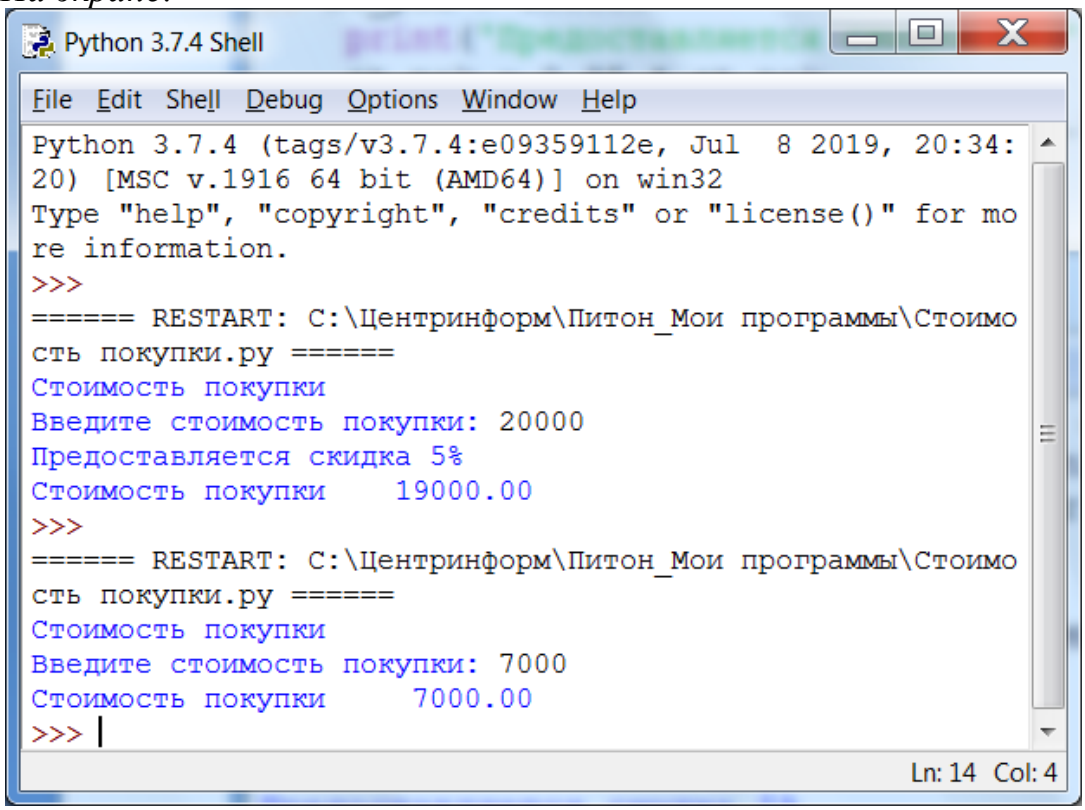
Задача 4. Написать программу вычисления стоимости покупки с учётом скидки. Скидка 5% предоставляется, если стоимость покупки больше 10000 рублей. Предусмотреть вывод сообщения о наличии скидки (если она есть).

Решение.

```
print('Стоимость покупки ')
st_pok = float(input('Введите стоимость покупки: '))
if st_pok > 10000:
    print('Предоставляется скидка 5% ')
    st_pok = 0.95 * st_pok
print('Стоимость покупки ','{:10/2f}'.format(st_pok))
```



На экране:



3. Просмотр учебного видеоролика «Ветвление в Python. Условный оператор if» (с целью расширения знаний, наглядного представления и закрепления материала):

https://rutube.ru/video/1b6c18478a21a01c56ed364e5265ad74/?r=plemwd

(длина ролика: 10:58).

4. Закрепление

ЗАДАНИЕ 1.

Что будет выведено на экран в результате работы следующих программ?

а)

```
a=10
b=15
c=3
if a+b>10:
    print('Yes')
else:
    print('No')
c=a+b
print(c)
```

б)

```
a=10
b=15
c=3
if a+b>10:
    print('Yes')
else:
    print('No')
    c=a+b
print(c)
```

ЗАДАНИЕ 2.

Написать программу вычисления квадратного корня из выражения ***1000 – a.*** Значение ***a*** ввести с клавиатуры во время выполнения программы. Результат вывести с тремя знаками после запятой.

В случае, если значение ***1000 – a*** меньше нуля, предусмотреть вывод поясняющего текста «Под корнем отрицательное число!» и завершить выполнение программы.

!!! Решение будет на следующей странице

РЕШЕНИЕ

Задание 1.

а)
```
Yes
25
```

б)
```
Yes
3
```

Как видим, результаты работы программ различаются, потому что в первом примере оператор c=a+b выполняется в любом случае, так как стоит вне оператора ветвления, а во втором примере – только если условие ложно. Поэтому во втором случае на экран выводится исходное значение **c**.

Задание 2.

Решение.

```
a = float(input('Введите число: ' ))
if 1000 - a >= 0:
      import math
      kor = math.sqrt(1000 - a)
      print('Ответ: ', '{:6.3f}'.format(kor))
else:
      print("Под корнем отрицательное число!")
```

Выч-ие квадр. корня.py - C:/Центринформ/Питон_Мои программы/Выч-ие квадр. корня.py (3.7.4)

File Edit Format Run Options Window Help

```
a = float(input('Введите число: '))
if 1000 - a >= 0:
        import math
        kor = math.sqrt(1000 - a)
        print('Ответ: ', '{:6.3f}'.format(kor))
else:
        print("Под корнем отрицательное число!")
```

На экране:

Python 3.7.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Центринформ/Питон_Мои программы/Выч-ие квадр. корня.py =====
Введите число: 1500
Под корнем отрицательное число!
>>>
===== RESTART: C:/Центринформ/Питон_Мои программы/Выч-ие квадр. корня.py =====
Введите число: 500
Ответ:  22.361
>>>
```

Ln: 11 Col: 4

Платформа Programiz

main.py Run Output

```
1  a = float(input('Введите число: '))
2  if 1000 - a >= 0:
3      import math
4      kor = math.sqrt(1000 - a)
5      print('Ответ: ', '{:6.3f}'.format(kor))
6  else:
7      print("Под корнем отрицательное число!")
8
```

```
Введите число: 500
Ответ:  22.361

=== Code Execution Successful ===
```

main.py Run Output

```
1  a = float(input('Введите число: '))
2  if 1000 - a >= 0:
3      import math
4      kor = math.sqrt(1000 - a)
5      print('Ответ: ', '{:6.3f}'.format(kor))
6  else:
7      print("Под корнем отрицательное число!")
8
9
```

```
Введите число: 1500
Под корнем отрицательное число!

=== Code Execution Successful ===
```