Московский государственный технический университет им. Н.Э. Баумана

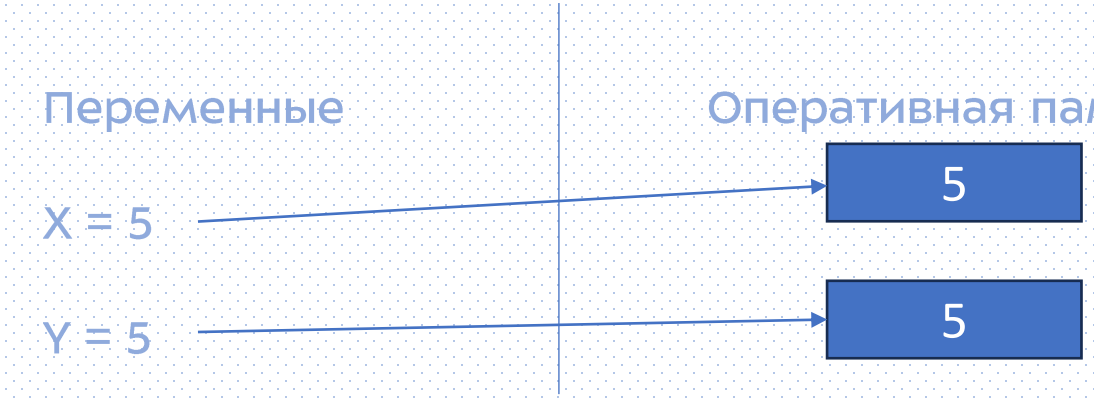
Введение в Python

сентябрь 2023 г.

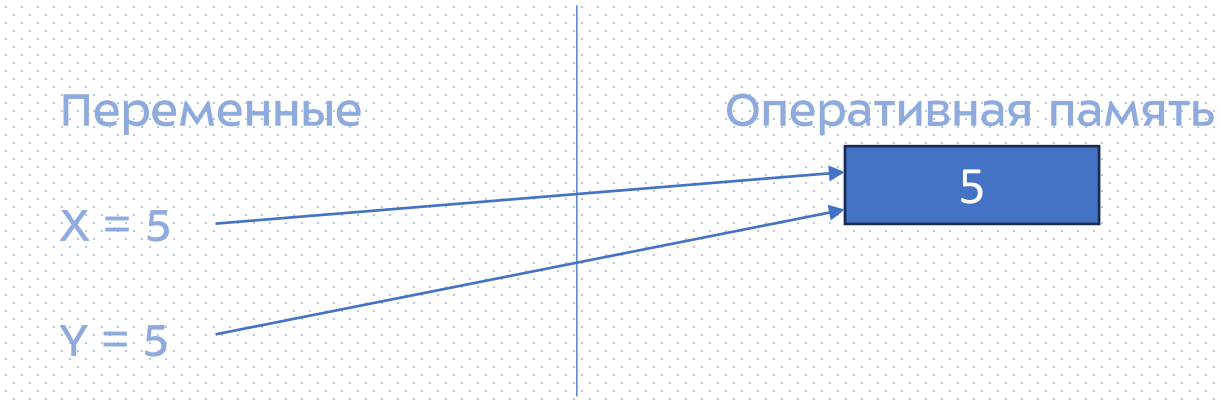
Переменные в Python

В Python используется неявная типизация

Алгоритмические языки с явной типизацией



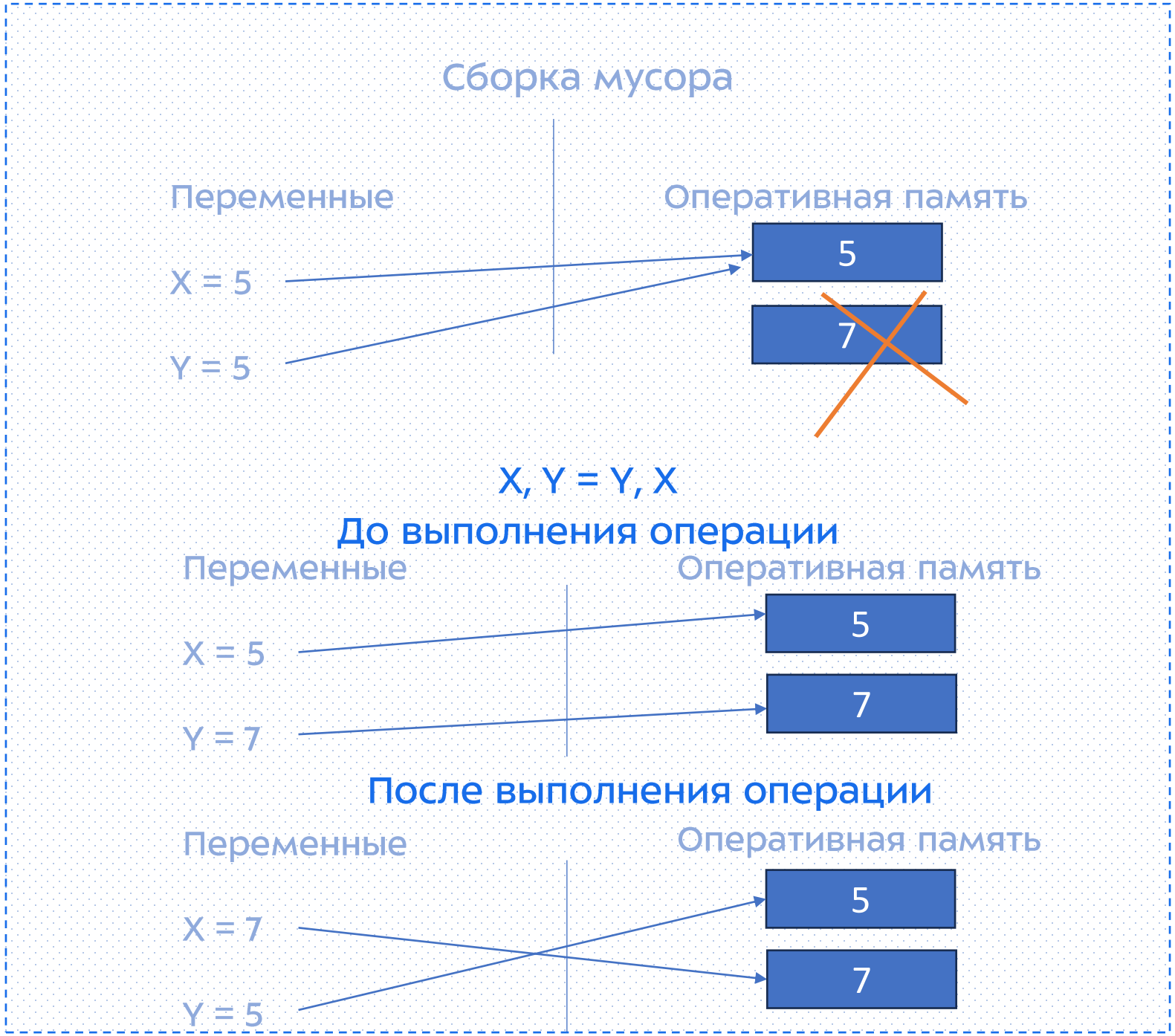
Алгоритмические языки с неявной типизацией



Переменные в Python

Объект в оперативной памяти существует, пока на него есть по крайней мере одна ссылка.

В Python ссылки на объекты возможно перекрёстно поменять за одно действие.



Типы переменных в Python

Объект в оперативной памяти существует, пока на него есть по крайней мере одна ссылка.

В Python ссылки на объекты возможно перекрёстно поменять за одно действие.

Целые числа

Пример: 5

Числа с плавающей точкой

Пример: 23.9

Строки

Пример: ‘строка’

“string”

Логические значения

Пример: True

False

Ввод значений с клавиатуры

При вводе в компьютер поступает последовательность двоичных чисел. Длина одного числа – 1 байт.

```
S = input()
```

Тип переменной S – строка.

```
X = int(S)
```

Тип переменной X – целое число.

```
X = float(S)
```

Тип переменной X – число с плавающей точкой.

Арифметические операции

+ - сложение

- - вычитание

***** - умножение

/ - деление

// - целочисленное деление

% - вычисление остатка

****** - возведение в степень

Старшинство операций:

Уровень 1:

умножение, деление, целочисленное деление, вычисление остатка, возведение в степень

Уровень 2:

Сложение, вычитание

Арифметические операции

В Python доступна Длинная арифметика.

Старшинство операций:

Уровень 1:

умножение, деление, целочисленное деление, вычисление остатка, возведение в степень

Уровень 2:

Сложение, вычитание

Операции одного уровня выполняются слева направо.

Примеры целочисленного деления в Python:

7 // 2 = 3; 7%3 = 1

-7 // 2 = -4; -7 % 2 = 1

Операции сравнения

Результат операции сравнения – True или False

> - больше

>= - больше или равно

< - меньше

<= - меньше или равно

== - сравнение на эквивалентность

!= - сравнение на неравенство

% - вычисление остатка

****** - возведение в степень

Допускаются двухсторонние операции.

Пример: 4 < x < 9

Логические операции

Операндами логических операций являются True и False.

and – логическое И (конъюнкция)

or – логическое ИЛИ (дизъюнкция)

not – отрицание

Старшинство операций:

Уровень 1:
отрицание (not)

Уровень 2:
конъюнкция (and)

Уровень 3:
дизъюнкция (or)

Операции одного уровня выполняются слева направо.

Логические операции

Операндами логических операций являются True и False.

Последовательные операции **and** выполняются до тех пор, пока не встретится значение False.

Последовательные операции **or** выполняются до тез пор, пока не встретится значение True.

Деления на ноль не будет.

```
if (x != 0) and (z / x > 1):
        ……
```

Попытка деления на ноль возможна.

```
if (z / x > 1) and (x != 0):
        …
```

Оператор вывода print

Оператор print преобразует числа в строки.

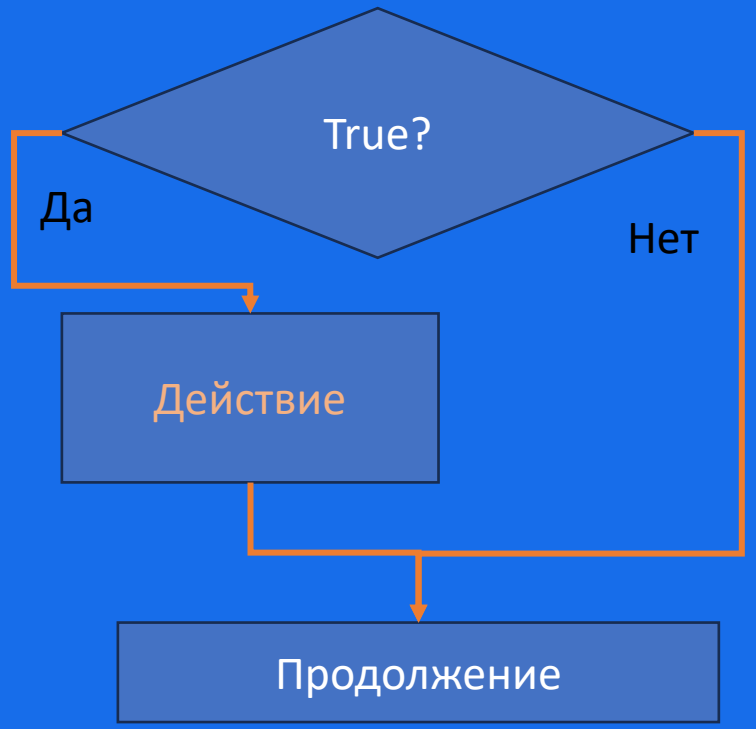
```
x = 5
y = 7
print(x, y)
>>5 7

print(x, y, sep=‘;’)
>>5;7

print(x)
print(y)
>>5
>>7

print(x, end=‘,’)
print(y)
>>5,7
```

Оператор ветвления if без else



if выражение, имеющее логическое значение:

Действия, если значение выражения True

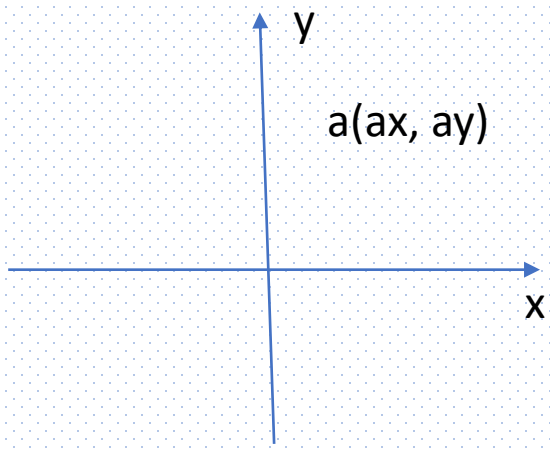
Табуляция (клавиша Tab)

РЕШЕНИЕ

```
if ax > 0:
        if ay > 0:
                print('I')
        if ay <= 0:
                print('IV')
if ax <= 0:
        if ay > 0:
                print('II')
        if ay <= 0:
                print('III')
```

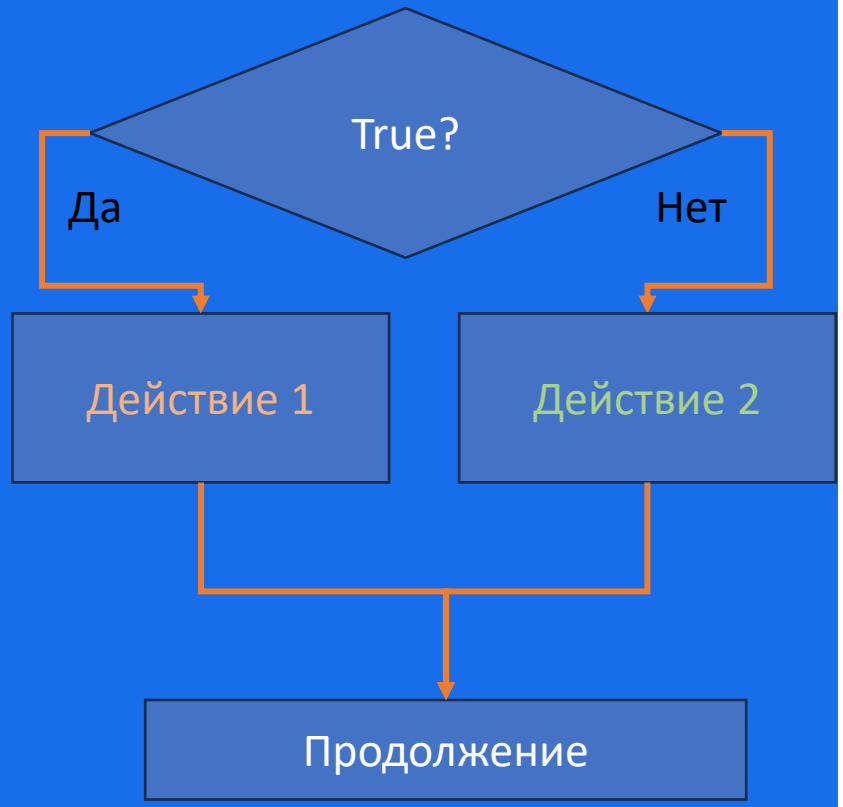
Решение содержит логические неточности.

ПРИМЕР



Определить, в какой четверти лежит точка a.

Оператор ветвления if с else



if выражение, имеющее логическое значение:

Действия, если значение выражения True

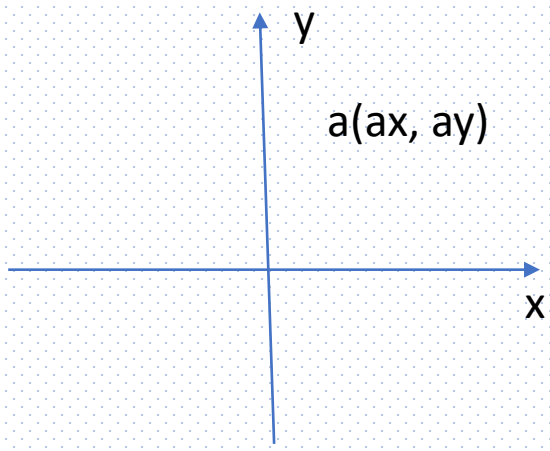
else:

Действия, если значение выражения False

РЕШЕНИЕ

```
if ax > 0:
        if ay > 0:
                print('I')
        else:
                print('IV')
else:
        if ay > 0:
                print('II')
        else:
                print('III')
```

ПРИМЕР



Определить, в какой четверти лежит точка a.

В решении нет лишних операций.

Оператор ветвления if с elif

if выражение 1, принимающее логическое значение:

    Действия, если значение выражения 1 True

elif выражение 2, принимающее логическое значение:

    Действия, если значение выражения 2 True

…

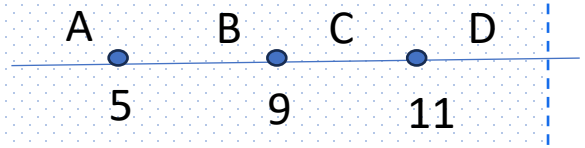
else:

    Действия, если все лог. выражения False

РЕШЕНИЕ

```
if x < 5:
        print('A')
else:
        if 5<= x < 9
                print('B')
        else:
                if 9 <= x < 11:
                        print('C')
                else:
                        if x >= 11:
                                print(;'D')
```

ПРИМЕР



Определить, в каком интервале лежит значение **x**

Запись решения «растягивается» вправо.

Оператор ветвления if с elif

if выражение 1, принимающее логическое значение:

    Действия, если значение выражения 1 True

elif выражение 2, принимающее логическое значение:

    Действия, если значение выражения 2 True

…

else:

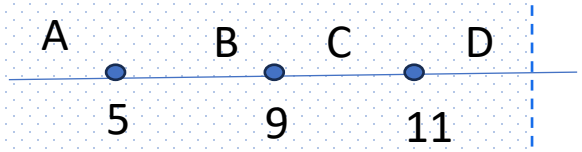
    Действия, если все лог. выражения False

РЕШЕНИЕ

```
if x < 5:
        print('A')
elif 5 <= x < 9:
        print('B')
elif 9 <= x < 11:
        print('C')
else:
        print(;'D')
```

Понятна логика решения, читать легко.

ПРИМЕР



Определить, в каком интервале лежит значение **x**

Оператор цикла for

Генератор последовательности чисел range

```
for x in 1, 2, 3, 4, 5:
        print(x ** 2)
>>1
>>4
>>9
>>16
>>25
```

range(n, m, k) – **создаёт итерируемый объект, заполненный числами в диапазоне от n до m (не включая m), с шагом k.**

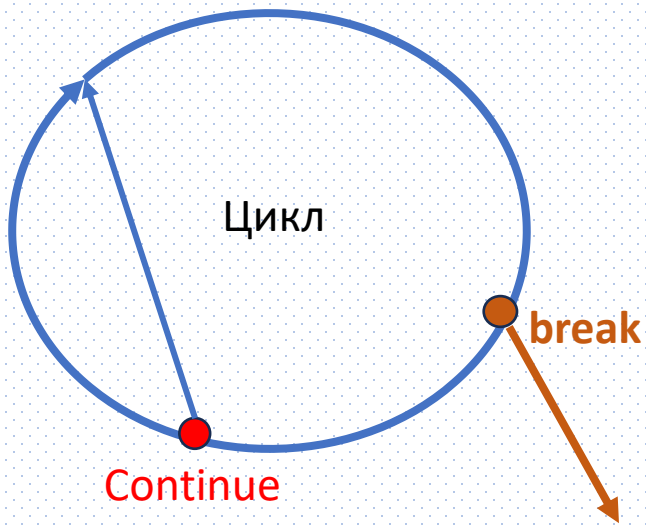
```
for x in range(1, 6, 1):
        print(x ** 2)
>>1
>>4
>>9
>>16
>>25
```

Если n=0, m=1, их значения можно не указывать.

```
for x in range(1, 6):
        print(x ** 2)
>>1
>>4
>>9
>>16
>>25
```

Оператор завершения текущей итерации **continue**

Оператор принудительного выхода из цикла **break**




```
for x in range(1, 6):
        y = int(input())
        if y <= 5:
                print(y ** 2)
        else:
                break
else:
        print('Все введённые числа не больше 5)
print('Продолжение программы')
```

Оператор цикла while

while выражение, принимающее логическое значение:

Тело цикла

```
i = 1
while I < 6:
        print(I ** 2)
        i += 1
>>1
>>4
>>9
>>16
>>25

i = 1
while True:
        if i ** 2 > 100:
                res = i
                break
        i += 1
print(res)
>> 11
```