Программирование на языке Python

Вывод на экран

```
print ("2+2=?")
print ("Ответ: 4")
```

автоматический переход на новую строку

Вывод:

```
2+2=?
Ответ: 4
```

```
print ('2+2=?')
print ('Ответ: 4')
```

Имена переменных

МОЖНО использовать

- латинские буквы (A-Z, a-z)

заглавные и строчные буквы **различаются**

- русские буквы (не рекомендуется!)
- цифры

имя не может начинаться с цифры

- знак подчеркивания _

НЕЛЬЗЯ использовать

- ~~скобки~~
- ~~знаки +, =, !, ? и др.~~

Типы переменных

```
a = 4
```

целое число (*integer*)

```
a = 4.5
```

вещественное число

```
a = "Вася"
```

символьная строка

```
a = True
```

логическая

Ввод значения с клавиатуры

```
a = input()
```

ввести строку с клавиатуры и связать с переменной **a**

```
b = input()
```

```
c = a + b
```

```
print ( c )
```

Протокол:

```
21
33
2133
```

Почему?

Результат функции **input** – строка символов!

преобразовать в целое число

```
a = int( input() )
```

преобразовать в дробное число

```
b = float( input() )
```

Вывод данных

```
print ( a )
```

значение переменной

```
print ( "Ответ: ", a )
```

значение и текст

перечисление через запятую

```
print ( "Ответ: ", a+b )
```

вычисление выражения

Арифметическое операции

1) сложение (3+5)
2) вычитание (8.7-4)
3) умножение (4*8)
4) деление (45/9)
5) возведение в степень (3**4 = 81)

Деление

Классическое деление:

```
a = 9; b = 6
x = 3 / 4     # = 0.75
x = a / b     # = 1.5
x = -3 / 4    # = -0.75
x = -a / b    # = -1.5
```

Целочисленное деление (округление «вниз»!):

```
a = 9; b = 6
x = 3 // 4    # = 0
x = a // b    # = 1
x = -3 // 4   # = -1
x = -a // b   # = -2
```

Остаток от деления

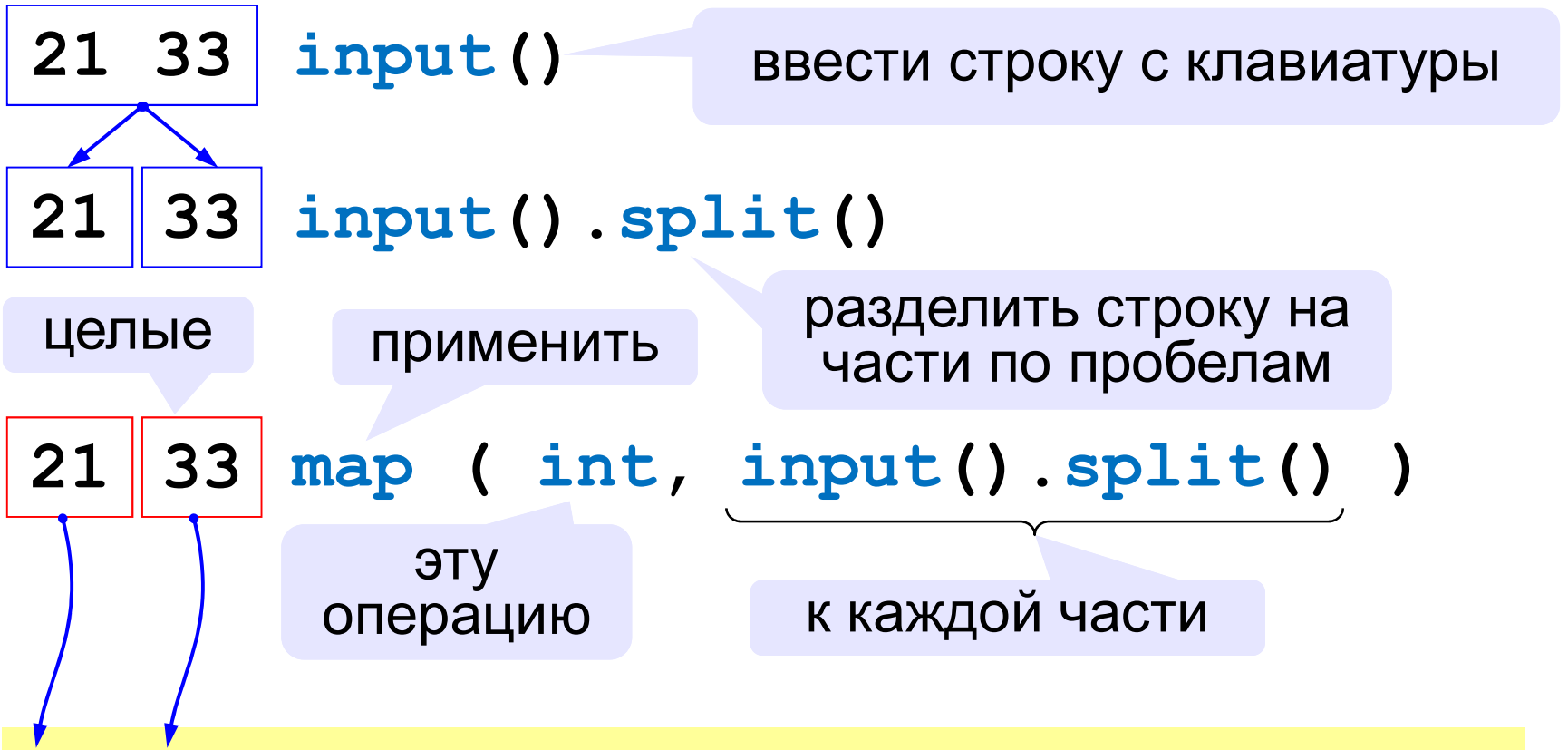
% – остаток от деления

```
d = 85
b = d // 10
a = d % 10
d = a % b
d = b % a
```

```
a = 15
b = 19
d = a // b
a = a % b
```

Ввод двух значений в одной строке

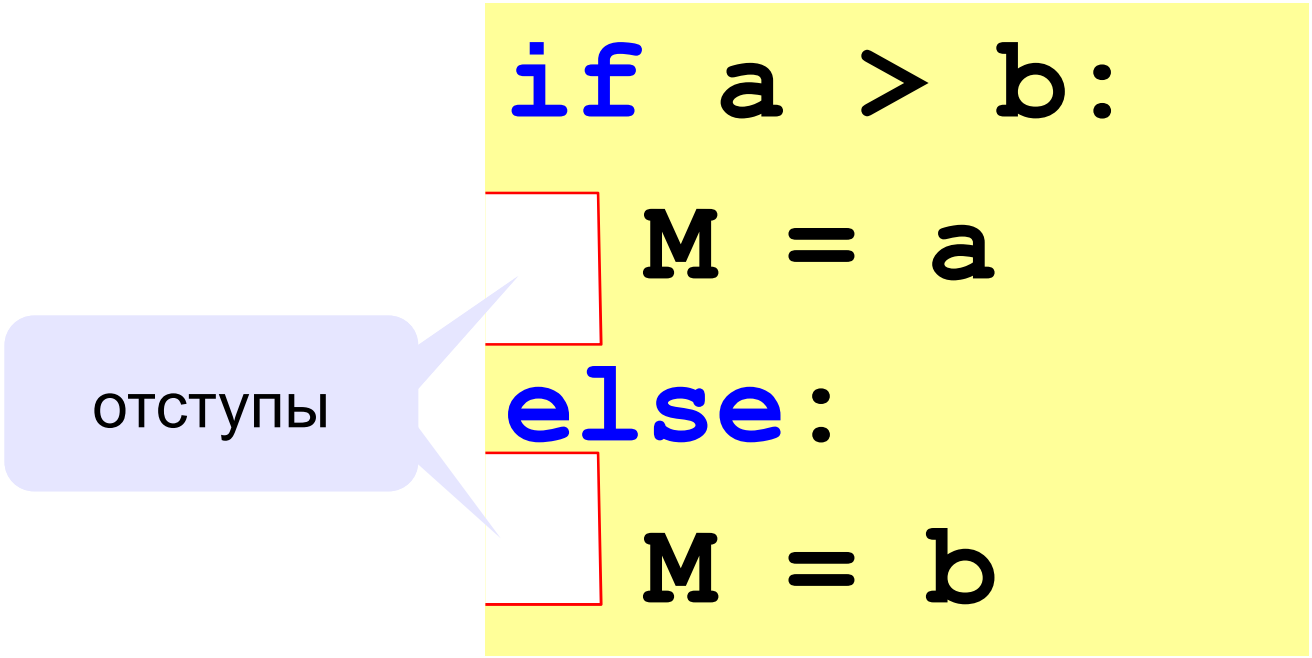
```
a, b = map ( int, input().split() )
```




```
a, b = map ( int, input().split() )
```

Условный оператор

Задача: **изменить порядок действий** в зависимости от выполнения некоторого условия.



Решение в стиле Python:

```
M=a if a>b else b
```

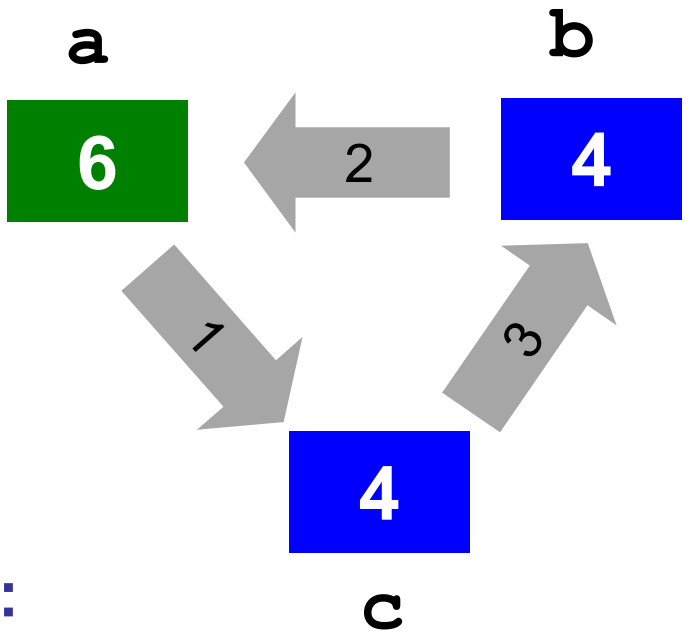
Условный оператор: неполная форма

```
M = a
if b > a:
    M = b
```

Условный оператор

```
if a > b:
   c = a
   a = b
   b = c
```

Что делает?



Можно ли обойтись без переменной **c**?

Решение в стиле Python:

```
a, b = b, a
```

Знаки отношений

`>` `<`	больше, меньше
`>=`	больше или равно
`<=`	меньше или равно
`==`	равно
`!=`	не равно

Каскадное ветвление

```
if a > b:
  print("Андрей старше")
elif a == b:
  print("Одного возраста")
else:
  print("Борис старше")
```

`elif = else if`

Каскадное ветвление

```
cost = 1500
if cost<1000:
  print( "Скидок нет." )
elif cost<2000:
  print( "Скидка 2%." )
elif cost<5000:
  print( "Скидка 5%." )
else:
  print( "Скидка 10%." )
```

первое сработавшее условие

? Что выведет?

```
Скидка 2%.
```

Сложные условия

Задача: набор сотрудников в возрасте **25-40 лет** (включительно).

сложное условие

```
if v >= 25 and v <= 40 :
  print("подходит")
else:
  print("не подходит")
```

and «И»: **одновременное** выполнение всех условий!

Сложные условия

Задача: набор сотрудников в возрасте **25-40 лет** (включительно).

сложное условие

```
if v < 25 or v > 40 :
  print("не подходит")
else:
  print("подходит")
```

or «ИЛИ»: выполнение **хотя бы одного** из двух условий!

Сложные условия

```
if not(a < b):
  print("Старт!")
```

Как без «НЕ»?

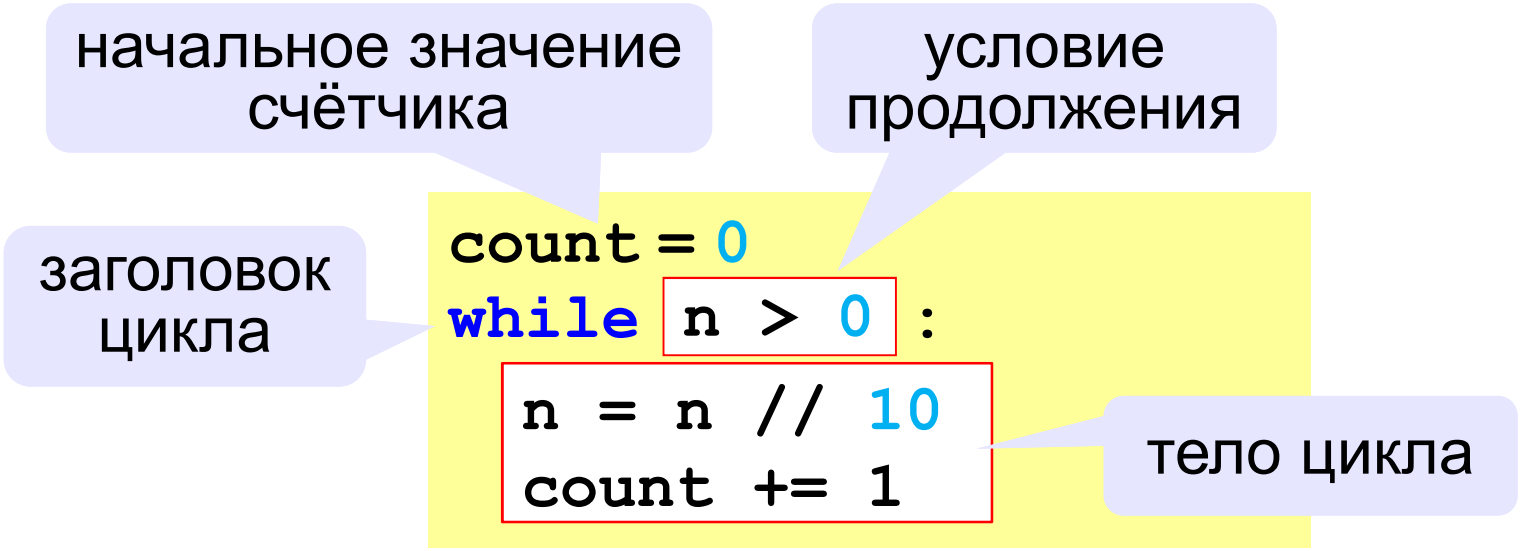
not «НЕ»: если выполняется обратное условие

```
if a >= b:
  print("Старт!")
```

Приоритет :

1) отношения (<, >, <=, >=, ==, !=)
2) not («НЕ»)
3) and («И»)
4) or («ИЛИ»)

Цикл с условием



Цикл с предусловием – проверка на входе в цикл!

Цикл с переменной

Задача. Вывести 10 раз слово «Привет!».

? Можно ли сделать с циклом «пока»?

```
i = 0
while i < 10:
    print("Привет!")
    i += 1
```

Цикл с переменной:

```
for i in range(10):
    print("Привет!")
```

в диапазоне [0,**10**)

! Не включая **10**!

```
range(10) → 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

Цикл с переменной

Задача. Вывести все степени двойки от 2^1 до 2^{10}.

Как сделать с циклом «пока»?

```
k = 1
while k <= 10
    print ( 2**k )
    k += 1
```

возведение в степень

Цикл с переменной:

```
for k in range(1,11) :
    print ( 2**k )
```

в диапазоне [1,**11**)

Не включая **11**!

```
range(1,11) → 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

Цикл с переменной: другой шаг

10,9,8,7,6,5,4,3,2,1

шаг

```
for k in range(10,0,-1):
    print ( k**2 )
```

```
100
81
64
49
36
25
16
9
4
1
```

Что получится?

1,3,5,7,9

```
for k in range(1,11,2):
    print ( k**2 )
```

```
1
9
25
49
81
```