

# Программирование (Python)

§ 19. Символьные строки

§ 20. Обработка массивов

§ 21. Матрицы (двумерные массивы)

§ 22. Сложность алгоритмов

§ 23. Как разрабатывают программы?§

23. Как разрабатывают программы?

§ 24. Процедуры

§ 25. Функции

# Программирование (Python)

## § 19. Символьные строки

# Что такое символьная строка?

---

**Символьная строка** – это последовательность СИМВОЛОВ.

- строка – единый объект
- длина строки может меняться во время работы программы

# Символьные строки

---

## Присваивание:

```
s = "Вася пошёл гулять"
```

## Ввод с клавиатуры:

```
s = input()
```

## Вывод на экран:

```
print(s)
```

## Длина строки:

```
n = len(s)
```

*length* – длина

# Сравнение строк

```
print("Введите пароль: ")
s = input()
if s == "sEzAm":
    print("Слушаюсь и повинуюсь!")
else:
    print("Пароль неправильный")
```



Какой правильный пароль?



Как одна строка может быть меньше другой?

стоит раньше в отсортированном списке

# Сравнение строк

```
s1 = "паровоз"  
s2 = "пароход"  
if s1 < s2:  
    print(s1, "<", s2)  
elif s1 == s2:  
    print(s1, "=", s2)  
else:  
    print(s1, ">", s2)
```



Что выведет?

паровоз < пароход

первые отличающиеся  
буквы

Сравниваем с начала: паровоз  
пароход

«В»: код 1074

«Х»: код 1093



В < Х!

# Обращение к символу по номеру

```
print ( s[5] )
```

```
print ( s[-2] )
```

0	1	2	3	4	5	6	$s[\text{len}(s)-2]$
П	р	и	в	е	т	!	
$s[0]$	$s[1]$	$s[2]$	$s[3]$	$s[4]$	$s[5]$	$s[6]$	



Символы нумеруются с нуля!

СОСТАВИТЬ «КОТ»

```
s = "информатика"  
kot = s[-2]+s[3]+s[-4]
```

# Посимвольная обработка строк

`s[4] ≠ "a"`



Строка неизменна!

Задача. Ввести строку и заменить в ней все буквы «э» на буквы «е».

строим новую строку!

```
sNew = ""  
for i in range(len(s)) :  
    if s[i] == "э":  
        sNew += "е"  
    else:  
        sNew += s[i]
```

для каждого символа строки

`len(s) - 1`

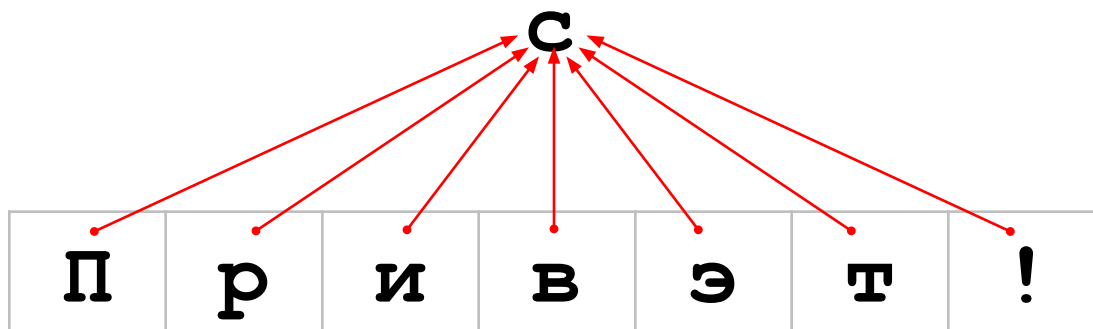
0	1	2	3	4	5	6
П	р	и	в	э	т	!



# Цикл перебора символов

```
sNew = ""  
for c in s:  
    if c == "э":  
        sNew += "е"  
    else:  
        sNew += c
```

перебрать  
все СИМВОЛЫ  
строки



# Задачи

---

«А»: Напишите программу, которая вводит строку, состоящую только из точек и букв X, и заменяет в ней все точки на нули и все буквы X на единицы.

**Пример:**

Введите строку: **..X.XX.**

Двоичный код: 0010110

«В»: Напишите программу, которая в символьной строке заменяет все нули на единицы и наоборот. Остальные символы не должны измениться.

**Пример:**

Введите строку: **10a01Vx1010c**

Инверсия: 01a10Vx0101c

# Задачи

---

«С»: Введите битовую строку и дополните её последним битом, который должен быть равен 0, если в исходной строке чётное число единиц, и равен 1, если нечётное (в получившейся строке должно всегда быть чётное число единиц).

## Пример:

Введите битовую строку: **01101010110**

Результат: **011010101100**

# Операции со строками

**Объединение (конкатенация) :**

```
s1 = "Привет"  
s2 = "Вася"  
s  = s1 + ", " + s2 + "!"
```

"Привет, Вася!"

**Умножение:**

```
s = "Ау"  
s5 = s*5
```

s5 = s + s + s + s + s

АУАУАУАУАУ



Что получим?

# Срезы строк (выделение части строки)

```
s = "0123456789"  
s1 = s[3:8]      # "34567"
```

с какого  
символа

до какого  
(не включая 8)

```
s = "0123456789"  
s1 = s[:8]      # "01234567"
```

от начала строки

```
s = "0123456789"  
s1 = s[3:]      # "3456789"
```

до конца строки

# Срезы строк

---

## Срезы с отрицательными индексами:

```
s = "0123456789"  
s1 = s[:-2] # "01234567"
```

`len(s) - 2`

```
s = "0123456789"  
s1 = s[-6:-2] # "4567"
```

`len(s) - 6`

`len(s) - 2`

# Операции со строками

---

## Удаление:

```
s = "0123456789"
```

```
s1 = s[:3] + s[9:]
```

"012"

"9"

"0129"

## Вставка:

```
s = "0123456789"
```

```
s1 = s[:3] + "ABC" + s[3:]
```

"012"

"3456789"

"012ABC3456789"

# Поиск в строках

```
s = "Здесь был Вася."  
n = s.find ( "с" )      # n = 3  
if n >= 0:  
    print ( "Номер символа", n )  
else:  
    print ( "Символ не найден." )
```



Находит первое слева вхождение подстроки!

Поиск с конца строки:

```
s = "Здесь был Вася."  
n = s.rfind ( "с" )     # n = 12
```



# Задачи

---

**«А»:** Ввести с клавиатуры в одну строку фамилию и имя, разделив их пробелом. Вывести первую букву имени с точкой и потом фамилию.

**Пример:**

**Введите фамилию и имя:**

**Иванов Петр**

**П. Иванов**

**«В»:** Ввести с клавиатуры в одну строку фамилию, имя и отчество, разделив их пробелом. Вывести фамилию и инициалы.

**Пример:**

**Введите фамилию, имя и отчество:**

**Иванов Петр Семёнович**

**П.С. Иванов**

# Задачи

---

«С»: Ввести адрес файла и «разобрать» его на части, разделенные знаком " / ". Каждую часть вывести в отдельной строке.

## Пример:

Введите адрес файла:

**C: /фото/2015/Байкал/shaman.jpg**

C:

фото

2015

Байкал

shaman.jpg

# Преобразования «строка» → «число»

## Из строки в число:

```
s = "123"
N = int ( s )      # N = 123
s = "123.456"
X = float ( s )   # X = 123.456
```

## Из числа в строку:

```
N = 123
s = str ( N )      # s = "123"
s = "{:5d}".format(N) # s = "  123"

X = 123.456
s = str ( X )      # s = "123.456"
s = "{:7.2f}".format(X) # s = " 123.46"
s = "{:10.2e}".format(X) # s = " 1.23e+02"
```

# Задачи

---

**«А»:** Напишите программу, которая вычисляет сумму двух чисел, введенную в форме символьной строки. Все числа целые.

**Пример:**

**Введите выражение :**

**12+3**

**Ответ: 15**

**«В»:** Напишите программу, которая вычисляет сумму трёх чисел, введенную в форме символьной строки. Все числа целые.

**Пример:**

**Введите выражение :**

**12+3+45**

**Ответ: 60**

# Задачи

---

«С»: Напишите программу, которая вычисляет сумму произвольного количества чисел, введенную в форме символьной строки. Все числа целые.

**Пример:**

**Введите выражение :**

**12+3+45+10**

**Ответ: 70**

«D»: Напишите программу, которая вычисляет выражение, содержащее целые числа и знаки сложения и вычитания.

**Пример:**

**Введите выражение :**

**12+134-45-17**

**Ответ: 84**

# Программирование (Python)

## § 20. Обработка массивов. Поток данных

# Обработка потока данных

**Задача.** С клавиатуры вводятся числа, ввод завершается числом 0. Определить, сколько было введено положительных чисел.

- 1) нужен счётчик
- 2) счётчик увеличивается
- 3) нужен цикл
- 4) это цикл с условием (число шагов неизвестно)

 Когда увеличивать счётчик? Какой цикл?

**счётчик = 0**

**пока не введён 0:**

**если введено число > 0 то**

**счётчик := счётчик + 1**

# Обработка потока данных

---

```
count = 0
x = int(input())
while x != 0:
    if x > 0:
        count += 1
    x = int(input())
print( count )
```

откуда взять x?



Что плохо?



# Найди ошибку!

---

```
count = 0
x = int(input())
while x != 0:
    if x > 0:
        count += 1
pr x = int(input())
```

# Найди ошибку!

---

```
count = 0      ut ( ) )
while x == 0:
    if x > !=:
        count += 1
    x = int(input ( ) )
print ( count )
```

# Задачи

---

- «А»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сколько получено чисел, которые делятся на 3.
- «В»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сколько получено двузначных чисел, которые заканчиваются на 3.

# Задачи

---

- «С»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти среднее арифметическое всех двузначных чисел, которые делятся на 7.
- «D»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти максимальное из введённых чётных чисел.

# Обработка потока данных: сумма

**Задача.** С клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на цифру "5".

- 1) нужна переменная для суммы
- 2) число добавляется к сумме, если оно заканчивается на "5"
- 3) нужен цикл с условием

```
сумма = 0
```

```
пока не введён 0:
```

```
    если число оканчивается на "5" то
```

```
        сумма := сумма + число
```



Как это записать?

```
if x % 10 == 5:
```

## Обработка потока данных: сумма

*Задача.* С клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на цифру "5".

```
Sum = 0
x = int(input())
while x != 0:
    if x % 10 == 5:
        Sum += x
    x = int(input())
print( Sum )
```



Чего не хватает?

# Найди ошибку!

---

```
Sum = 0
x = int(input())
    if x % 10 == 5:
        Sum += x
    x = int(input())
print( Sum )
```

# Задачи

---

- «А»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сумму чисел, которые делятся на 3.
- «В»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Определить, сумму двузначных чисел, которые заканчиваются на 3.



# Задачи

---

- «C»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти сумму цифр всех введённых чисел.
- «D»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Найти число, у которого максимальная сумма цифр. Если таких чисел несколько, нужно вывести последнее их них.

# "Бесконечный" цикл

```
s = 0
while True:
    x = int(input())
    if x == 0: break
    if x % 10 == 5:
        s += x
print( s )
```

ВЫЙТИ ИЗ  
ЦИКЛА



Что плохо?



Выход из цикла `while True` возможен только через оператор `break`!

# Условия отбора

---

Положительные числа:

```
if x > 0: ...
```

Числа, делящиеся на 3:

```
if x % 3 == 0: ...
```

Числа, оканчивающиеся на 6:

```
if x % 10 == 6: ...
```

Числа, делящиеся на 3 и оканчивающиеся на 6:

```
if x % 3 == 0 and x % 10 == 6: ...
```

Двузначные числа:

```
if 10 <= x and x <= 99: ...
```

```
if 9 < x and x < 100: ...
```



Как иначе?

## Если ни одного числа не нашли...

**Задача:** с клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на "5". Вывести "нет", если таких чисел нет.



Как определить, что таких чисел нет?

сумма = 0

счётчик = 0

пока не введён 0:

если x оканчивается на "5":

сумма += x

счётчик += 1



Как вывести результат?

## Если ни одного числа не нашли...

---

```
сумма = 0
```

```
счётчик = 0
```

```
пока не введён 0:
```

```
    если x оканчивается на "5":
```

```
        сумма += x
```

```
        счётчик += 1
```

```
if счётчик == 0:
```

```
    print("Ответ: нет")
```

```
else:
```

```
    print("Ответ: ", s)
```

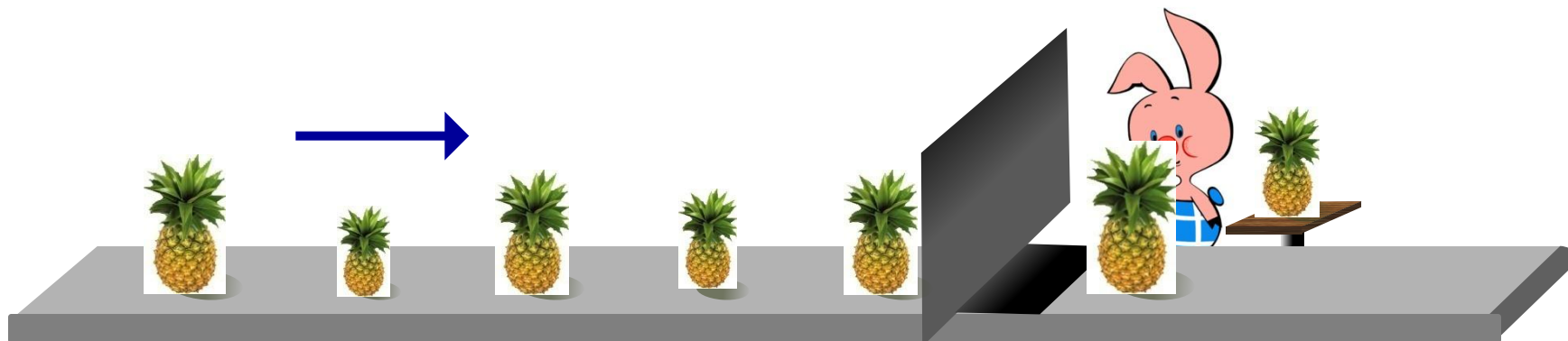
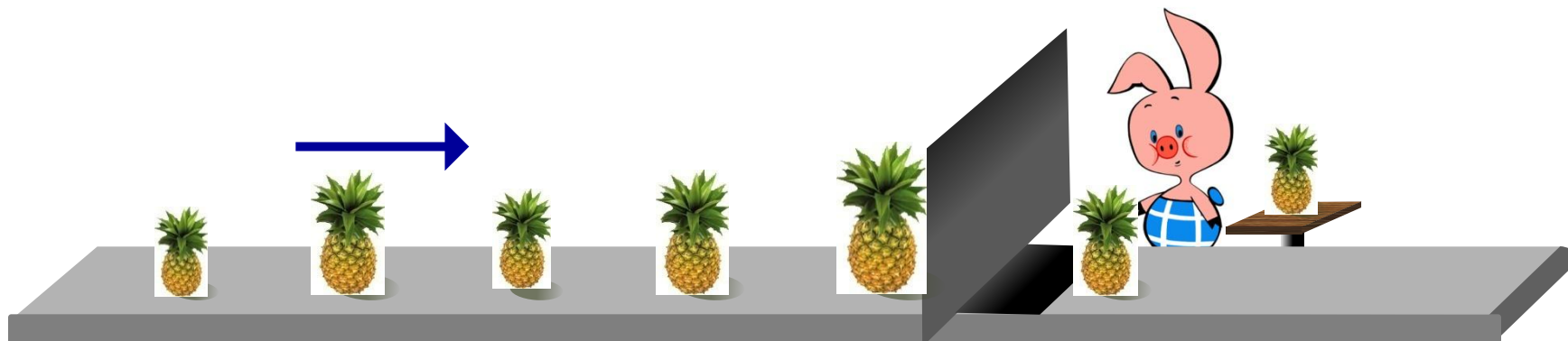
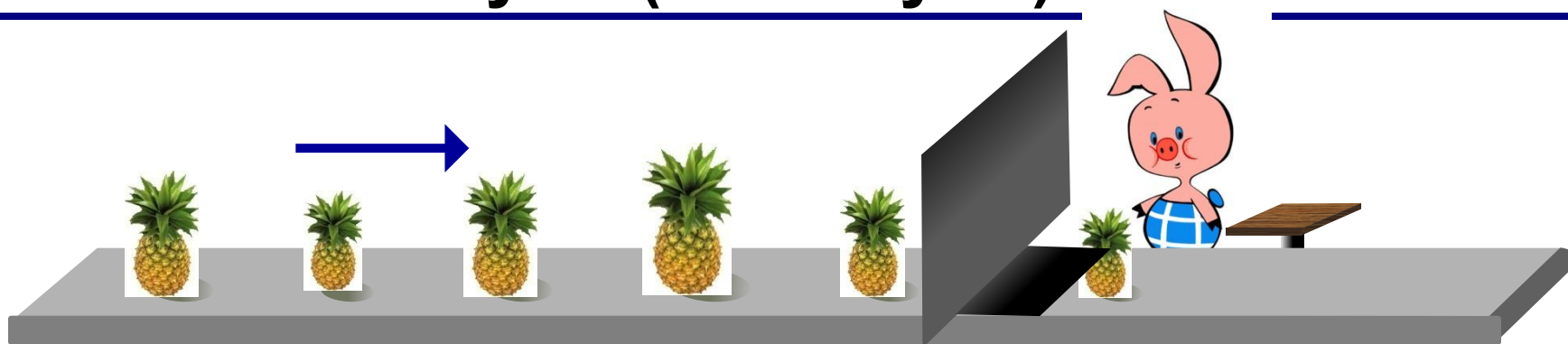
# Найди ошибку!

```
s = 0
x = int(input())
while x != 0:
    if x % 10 == 5:
        k += 1
    if k == 0:
        x = int(input())
    print("Ответ: нет")
else:
    print("Ответ: ", s)
```

Annotations in the image:

- A callout box containing `k = 0` is positioned above the `while` loop.
- A callout box containing `s += x` is positioned above the `if x % 10 == 5:` block.
- A callout box containing `x = int(input())` is positioned above the `if k == 0:` block.
- The value `0` in `if k == 0:` is enclosed in a red box.
- The variable `s` in `print("Ответ: ", s)` is enclosed in a red box.

# Поиск максимума (минимума)



## Поиск максимума (минимума)

---

- 1) нужна переменная для хранения максимума
- 2) как только прочитали первое число, сохранили максимум («из одного»):

```
x = int(input())  
M = x
```

- 3) читаем следующее значение:

```
x = int(input())
```

- 4) цикл: если новое число больше максимума, заменяем M:

```
while x != 0:  
    if x > M: M = x
```

- 5) выводим результат M



# Поиск максимума (минимума)

```
x = int(input())
M = x
while x != 0:
    if x > M: M = x
    x = int(input())
print(M)
```



Что плохо?

## Поиск максимума (минимума) – II

---

```
x = int(input())
M = x
while True:
    x = int(input())
    if x == 0: break
    if x > M: M = x
print(M)
```



Что плохо?

## Максимум не из всех

**Задача:** с клавиатуры вводятся числа в диапазоне **[-100;100]**, ввод завершается числом 0. Найти наибольшее **среди чётных чисел**. Вывести "нет", если таких чисел нет.

```
x = int(input())
M = x
while x != 0:
    if       ???      :
        M = x
    x = int(input())
print(M)
```

Может быть, что **x** нечётное!

**?** Что плохо?

# Минимум не из всех

По условию:  $x \in [-100; 100]$

```
x = int(input())
M = -1000
while x != 0:
    if x % 2 == 0 and x > M:
        M = x
    x = int(input())
if M == -1000:
    print("Нет таких чисел")
else:
    print(M)
```

Любое  $x$  больше  
этого числа!



Как определить, что  
ни одного числа не  
нашли?



Как искать минимум?

## Если диапазон неизвестен...

```
x = int(input())
M = 1 # любое нечётное
while x != 0:
    if x % 2 == 0:
        if M % 2 == 1 or x > M:
            M = x
    x = int(input())
if M == 1:
    print("Нет таких чисел")
else:
    print(M)
```

НОВЫЙ  
максимум



Когда не нашли?

# Задачи

---

«А»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить минимальное и максимальное из введённых чисел.

**Пример:**

5

13

34

15

0

**Минимум: 5**

**Максимум: 34**

# Задачи

---

«В»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить минимальное и максимальное из тех введённых чисел, которые делятся на 3. Вывести "нет", если таких чисел нет.

**Пример:**

5  
31  
18  
21  
15  
0

**Минимум: 15**

**Максимум: 21**

**Пример:**

5  
34  
17  
41  
11  
0

**Ответ: нет**

# Задачи

---

«С»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить минимальное и максимальное из тех введённых двузначных натуральных чисел, которые оканчиваются на 6. Вывести "нет", если таких чисел нет.

**Пример:**

6  
36  
18  
26  
15  
0

**Минимум: 26**

**Максимум: 36**

**Пример:**

6  
32  
176  
41  
11  
0

**Ответ: нет**



# Задачи

---

«D»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить минимальное из введённых чисел Фибоначчи. Вывести "нет", если чисел Фибоначчи в последовательности нет.

Числа Фибоначчи – это последовательность чисел, которая начинается с двух единиц и каждое следующее число равно сумме двух предыдущих: 1, 1, 2, 3, 5, 8, 13, ...

**Пример:**

5

36

12

26

13

0

**Ответ:** 5

**Пример:**

6

32

176

41

11

0

**Ответ:** нет

## Обработка потока данных ( $N$ чисел)

**Задача:** с клавиатуры вводится число  $N$ , а затем –  $N$  целых чисел. Определить, сколько было введено положительных чисел.



В чём отличие?

задано количество!

ввести  $N$

счётчик = 0

сделай  $N$  раз:

ввести число

если введено число  $> 0$ :

счётчик += 1

вывести счётчик

в Python нет такого!



Есть цикл `for`!

# Обработка потока данных ( $N$ чисел)

сделай  $N$  раз:

...

`for i in range(N):`

...



`range(N) = [0, 1, 2, ..., N-2, N-1]`

$N$  раз

```
for i in range(4):
    print(i)
```

```
s = 0
for i in range(4):
    s += i
print(s)
```

0  
1  
2  
3



Что выведет?

6

# Обработка потока данных ( $N$ чисел)

```
N = int(input())
k = 0
for i in range(N):
    x = int(input())
    if x > 0: k += 1
print(k)
```

сделай  $N$   
раз!

Числа, делящиеся на 3:

```
if x % 3 == 0: k += 1
```

Числа, оканчивающиеся на 6:

```
if x % 10 == 6: k += 1
```

Числа, делящиеся на 3 и оканчивающиеся на 6:

```
if x % 3 == 0 and x % 10 == 6: k += 1
```

# Задачи

---

«А»: с клавиатуры вводится число  $N$ , а затем –  $N$  целых чисел. Определить, сколько было введено положительных и сколько отрицательных чисел (нули не считать!).

**Пример:**

5

1

3

-34

15

0

**Положительных : 3**

**Отрицательных : 1**

# Задачи

---

«В»: с клавиатуры вводится число  $N$ , а затем –  $N$  целых чисел. Определить сумму двузначных чисел (как положительных, так и отрицательных). Если двузначных чисел не было, вывести "нет".

**Пример:**

5  
1  
13  
-34  
5  
31

**Ответ:** 10

**Пример:**

5  
1  
213  
-134  
5  
3

**Ответ:** нет

# Задачи

---

«С»: с клавиатуры вводится число  $N$ , а затем –  $N$  целых чисел. Определить минимальное и максимальное среди двузначных чисел, которые делятся на 3. Если таких чисел не было, вывести "нет".

**Пример:**

5  
18  
33  
98  
513  
31

Минимум: 18  
Максимум: 33

**Пример:**

5  
1  
-18  
-6  
-21  
32

Минимум: -21  
Максимум: -18

**Пример:**

5  
1  
23  
132  
6  
32

Ответ: нет

# Задачи

---

«D»: с клавиатуры вводится число  $N$ , а затем –  $N$  натуральных чисел. Определить минимальное и максимальное среди простых чисел (которые делятся на сами не себя и на 1). Если таких чисел не было, вывести "нет".

**Пример:**

5

41

15

11

163

39

**Минимум: 11**

**Максимум: 163**

**Пример:**

5

12

25

132

6

39

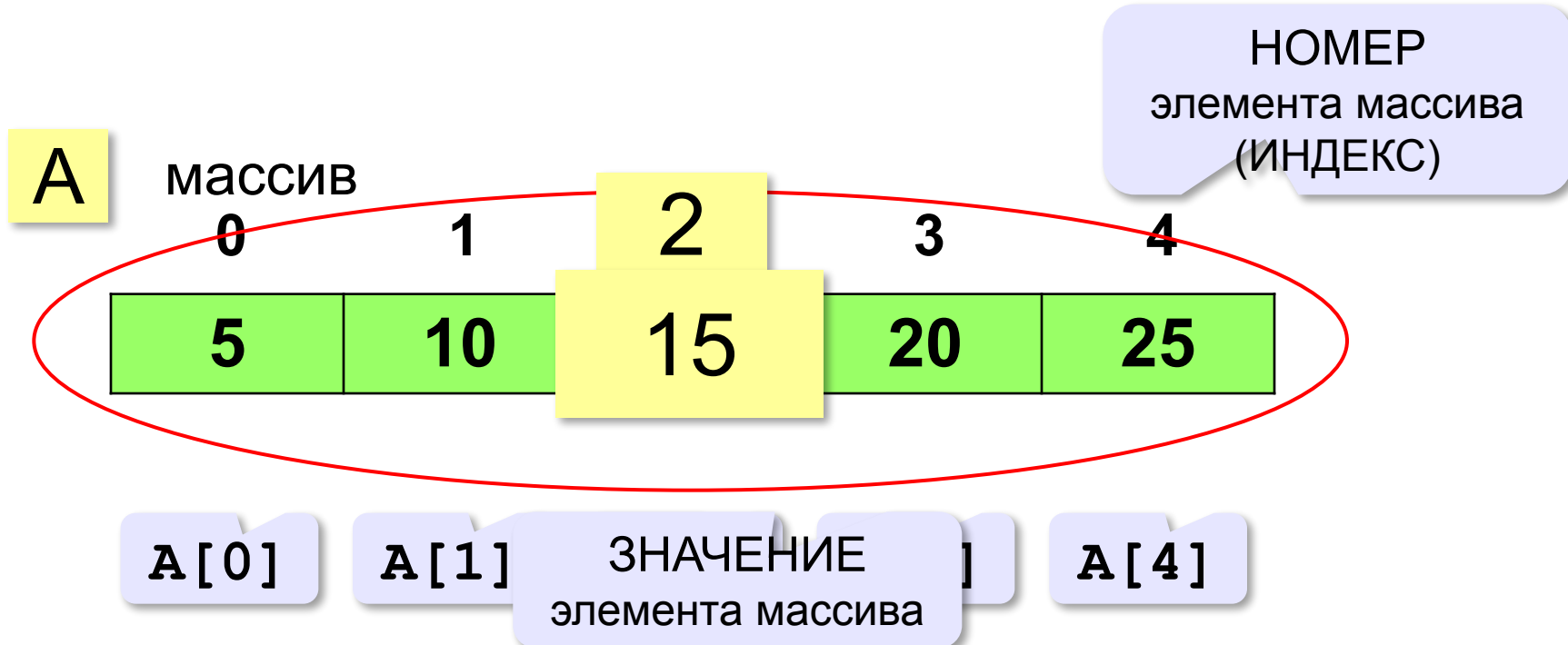
**Ответ: нет**



# Программирование (Python)

## **Массивы (повторение)**

# Обращение к элементу массива



**Индекс элемента** — это значение, которое указывает на конкретный элемент массива.

**!** Нумерация с нуля!

# Создание массива

---

11	22	35	41	53
----	----	----	----	----

```
A = [11, 22, 35, 41, 53]
```

```
A = [11, 22] + [35, 41] + [53]
```

```
A = [11] * 5
```

```
A = [11] + [11] + [11] + [11] + [11]
```

11	11	11	11	11
----	----	----	----	----

# Обращение к элементу массива

**A[2]**

ИНДЕКС элемента массива: 2

ЗНАЧЕНИЕ элемента массива

0	1	2	3	4
23	12	7	43	51

```
i = 1
A[2] = A[i] + 2*A[i-1] + A[2*i+1]
print( A[2]+A[4] )
```

**?** Что получится?

```
A[2] = A[1] + 2*A[0] + A[3]    101
print( A[2]+A[4] )           152
```

# Что неверно?

0	1	2	3	4
1	2	3	4	5

```
A = [1, 2, 3, 4, 5]
x = 1
print( A[x-8] )
A[x+4] = A[x-1] + A[2*x]
```



Что плохо?



```
print( A[-1] )
A[-1] = A[0] + A[2]
```

**Выход за границы массива** — это обращение к элементу с индексом, который не существует в массиве.

# Перебор элементов массива

**Перебор элементов:** просматриваем все элементы массива и, если нужно, выполняем с каждым из них некоторую операцию.

```
N = 10
```

```
A = [0]*N # память уже выделена
```

```
0, 1, 2, 3, ..., N-1
```

```
for i in range(N):
```

```
    # здесь работаем с A[i]
```

# Заполнение массива

[0, 1, 2, 3, ..., N-1]

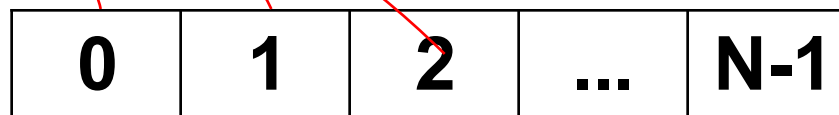
```
for i in range(N):  
    A[i] = i
```



Что произойдёт?

В развёрнутом виде

```
A[0] = 0  
A[1] = 1  
A[2] = 2  
...  
A[N-1] = N-1
```



В стиле Python:

```
A = [ i for i in range(N) ]
```

# Заполнение массива в обратном порядке

N	...	3	2	1
---	-----	---	---	---

```
A[0] = N
A[1] = N-1
A[2] = N-2
...
A[N-1] = 1
```

```
X = N
for i in range(N):
    A[i] = X
    X = X - 1
```

**?** Как меняется  $x$ ?

$x = N, N-1, \dots, 2, 1$

начальное  
значение

уменьшение  
на 1



# Заполнение массива в обратном порядке

N	...	3	2	1
---	-----	---	---	---

$$A[i] = X$$

**?** Как связаны  $i$  и  $X$ ?

$i$	$X$
0	N
1	N-1
2	N-2
...	...
N-1	1

+1      -1

```
for i in range(N):
    A[i] = N - i
```

В стиле Python:

```
A = [ N-i
      for i in range(N) ]
```

**!** Сумма  $i$  и  $X$  не меняется!

$$i + X = N$$

$$X = N - i$$

# Вывод массива на экран

Весь массив сразу:

```
print( A )
```

```
[1, 2, 3, 4, 5]
```

По одному элементу:

```
for i in range(N):  
    print( A[i] )
```

В столбик

или так:

```
for x in A:  
    print( x )
```

для всех элементов в массиве A



Как вывести в строчку?

```
for x in A:  
    print( x, end=" " )
```

пробел между элементами

# Вывод массива на экран (Python)

---

[1, 2, 3, 4, 5]

```
print ( *A )
```



```
print ( 1 , 2 , 3 , 4 , 5 )
```

разбить список  
на элементы

```
1 2 3 4 5
```

# Задачи

---

- «А»: а) Заполните все элементы массива из 10 элементов значением  $X$ , введённым с клавиатуры.
- б) Заполните массив из 10 элементов последовательными натуральными числами, начиная с  $X$  (значение  $X$  введите с клавиатуры).
- «В»: а) Заполните массив из 10 элементов натуральными числами в порядке убывания. С клавиатуры вводится значение  $X$ . Последний элемент должен быть равен  $X$ , предпоследний равен  $X+1$  и т.д.
- б) Заполните массив из 10 элементов степенями числа 2 (от  $2^1$  до  $2^N$ ), так чтобы первый элемент был равен 2, следующий – 4, следующий – 8 и т.д.

# Задачи

---

- «С»: а) Заполните массив из 10 элементов степенями числа 2, начиная с конца, так чтобы последний элемент массива был равен 1, а каждый предыдущий был в 2 раза больше следующего.
- б) С клавиатуры вводится целое число  $X$ . Заполните массив из 11 элементов целыми числами, так чтобы средний элемент массива был равен  $X$ , слева от него элементы стояли по возрастанию, а справа – по убыванию. Соседние элементы отличаются на единицу. Например, при  $X = 3$  массив из 5 элементов заполняется так: 1 2 3 2 1.

# Заполнение случайными числами

из библиотеки  
(модуля) random

взять функцию randint

```
from random import randint
N = 10          # размер массива
A = [0]*N      # выделить память
for i in range(N):
    A[i] = randint(20, 100)
```

В краткой форме:

```
from random import randint
N = 10
A = [ randint(20, 100)
      for i in range(N) ]
```

# Обработка элементов массива

```
N = 10
```

```
A = [0]*N # память уже выделена
```

```
for i in range(N):  
    # здесь работаем с A[i]
```

Вывести на экран в столбик:

```
for i in range(N):  
    print( A[i] )
```



Что вместо «???»?

Вывести на экран в строчку:

```
for i in range(N):  
    print( A[i], end = " " )
```

```
print( *A )
```

# Обработка элементов массива

---

Вывести числа, на 1 большие, чем  $A[i]$ :

```
for i in range(N):  
    print( A[i]+1 )
```



Что вместо «**???**»?

Вывести последние цифры:

```
for i in range(N):  
    print( A[i]%10 )
```



# Обработка элементов массива

---

Заполнить нулями:

```
for i in range(N) :  
    A[i] = 0
```



Что вместо «???»?

Увеличить на 1:

```
for i in range(N) :  
    A[i] += 1
```

Умножить на 2:

```
for i in range(N) :  
    A[i] *= 2
```

## Задачи-2

---

**«А»:** Напишите программу, которая заполняет массив из 10 элементов случайными числами в диапазоне  $[0, 10]$ , выводит его на экран, а затем выводит на экран квадраты всех элементов массива.

**Пример:**

**Массив:** 5 6 2 3 1 4 8 7

**Квадраты:** 25 36 4 9 1 16 64 49

**«В»:** Напишите программу, которая заполняет массив из 10 элементов случайными числами в диапазоне  $[100, 300]$  и выводит его на экран. После этого на экран выводятся средние цифры (число десятков) всех чисел, записанных в массив.

**Пример:**

**Массив:** 142 324 135 257 167 295 126 223 138 270

**Число десятков:** 4 2 3 5 6 9 2 2 3 7

## Задачи-2

---

**«С»:** Напишите программу, которая заполняет массив из 10 элементов случайными числами в диапазоне  $[100, 500]$  и выводит его на экран. После этого на экран выводятся суммы цифр всех чисел, записанных в массив.

### Пример:

**Массив:** 162 425 340 128 278 195 326 414 312 177

**Суммы цифр:** 9 11 7 11 17 15 11 9 6 15

# Сумма элементов массива

Задача. Найти сумму элементов массива из N элементов.

**?** Какие переменные нужны?

5	2	8	3	1
---	---	---	---	---

i	Sum
	0
0	5
1	7
2	15
3	18
4	19

```
Sum = 0
for i in range(N):
    Sum += A[i]
print( Sum )
```

В стиле Python:

```
print( sum(A) )
```

# Сумма элементов массива (Python)

Задача. Найти сумму элементов массива A.

```
Sum = 0
for x in A:
    Sum += x
print ( Sum )
```

для всех  
элементов из A



Не нужно знать размер!

или так:

```
print ( sum (A) )
```

встроенная  
функция

# Сумма не всех элементов массива

---

Задача. Найти сумму чётных элементов массива.

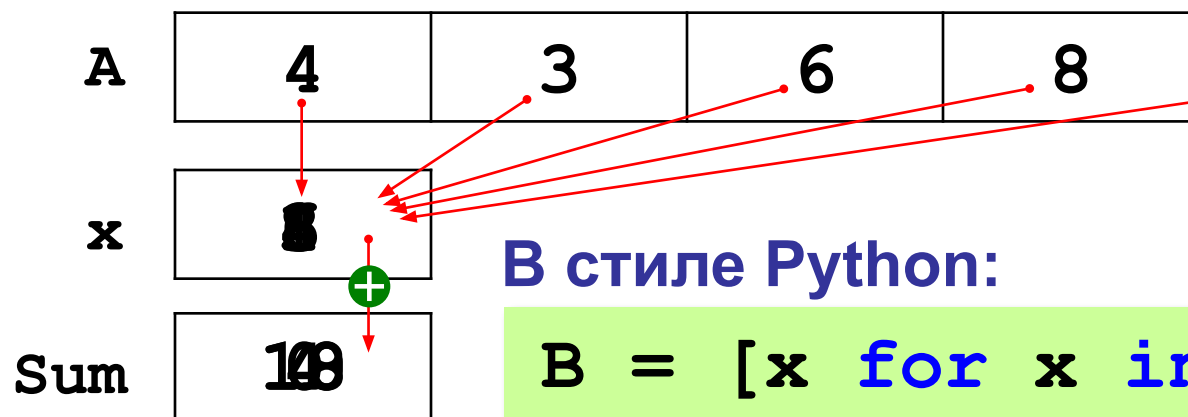
**?** Что делаем с нечётными?

```
Sum = 0
for i in range(N):
    if A[i] % 2 == 0:
        Sum += A[i]
print( Sum )
```

# Сумма не всех элементов массива

Задача. Найти сумму чётных элементов массива.

```
Sum = 0
for x in A:
    if x % 2 == 0:
        Sum += x
print( Sum )
```



отбираем в новый массив все нужные значения

В стиле Python:

```
B = [x for x in A
      if x % 2 == 0]
print( sum(B) )
```

# Задачи

---

- «А»: Напишите программу, которая заполняет массив из 10 элементов случайными числами на отрезке  $[-5; 5]$  и находит сумму положительных элементов.
- «В»: Напишите программу, которая заполняет массив из 10 элементов случайными числами на отрезке  $[-2; 2]$  и находит произведение ненулевых элементов.
- «С»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[100; 1000]$  и находит отдельно сумму элементов в первой и во второй половинах массива.



# Подсчёт элементов по условию

Задача. Найти количество чётных элементов массива.

**?** Какие переменные нужны?

переменная-  
счётчик

```
count = 0
for i in range(N):
    if A[i] % 2 == 0:
        count += 1
print(count)
```

**?** Что тут делаем?

# Подсчёт элементов по условию (Python)

Задача. Найти количество чётных элементов массива.

```
count = 0
for x in A:
    if x % 2 == 0:
        count += 1
print( count )
```

В стиле Python:

```
B = [x for x in A
      if x % 2 == 0]
print( len(B) )
```

размер массива

## Среднее арифметическое

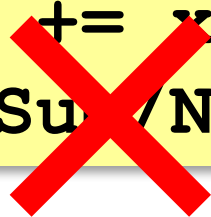
---

Задача. Найти среднее арифметическое элементов массива, которые больше 180 (рост в см).

```
Sum = 0
for x in A:
    if x > 180:
        Sum += x
print( Sum / N )
```



Что плохо?




# Среднее арифметическое

*Задача.* Найти среднее арифметическое элементов массива, которые больше 180 (рост в см).

 Какие переменные нужны?

```
Sum = 0
count = 0
for x in A:
    if x > 180:
        count += 1
        Sum += x
print( Sum/count )
```

 Что тут делаем?

# Среднее арифметическое (Python)

---

Задача. Найти среднее арифметическое элементов массива, которые больше 180 (рост в см).

```
B = [ x for x in A
      if x > 180 ]
print ( sum(B) / len(B) )
```

отбираем нужные

# Задачи

---

- «А»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[0; 200]$  и считает число элементов, которые делятся на 10.
- «В»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[0; 200]$  и считает число двузначных чисел в массиве.
- «С»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[10; 100]$  и считает число пар соседних элементов, сумма которых делится на 3.

# Перестановка элементов массива

**?** Как поменять местами значения двух переменных  $a$  и  $b$ ?

вспомогательная  
переменная

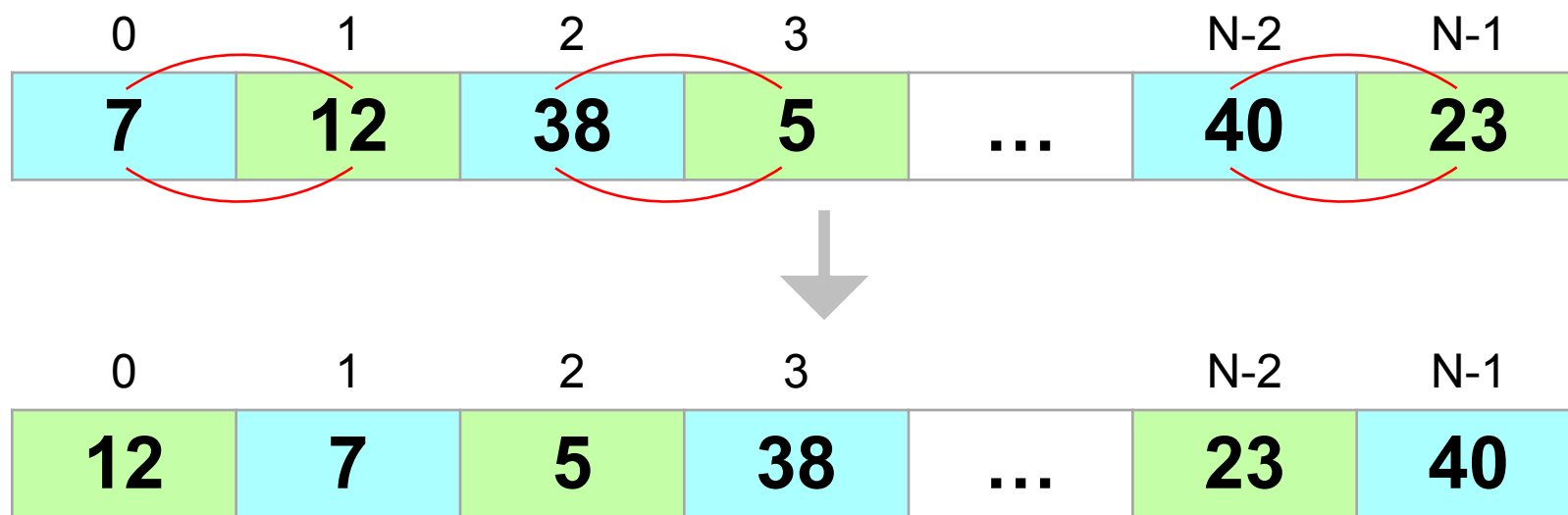
```
c = a
a = b
b = c
```

элементы массива:

```
c = A[i]
A[i] = A[k]
A[k] = c
```

# Перестановка пар соседних элементов

**Задача.** Массив  $A$  содержит чётное количество элементов  $N$ . Нужно поменять местами пары соседних элементов: 0-й с 1-м, 2-й — с 3-м и т. д.





# Перестановка пар соседних элементов

```
for i in range(N) :
```

```
    поменять местами A[i] и A[i+1]
```



Что плохо?

0	1	2	3	4	5
7	12	38	5	40	23
12	7	38	5	40	23
12	38	7	5	40	
12	38	5	7	40	23
12	38	5	40	7	23
12	38	5	40	23	7

выход за границы массива



# Перестановка пар соседних элементов

```
for i in range(0, N-1, 2):  
    # переставляем A[i] и A[i+1]  
    c = A[i]  
    A[i] = A[i+1]  
    A[i+1] = c
```

$A[0] \leftrightarrow A[1]$

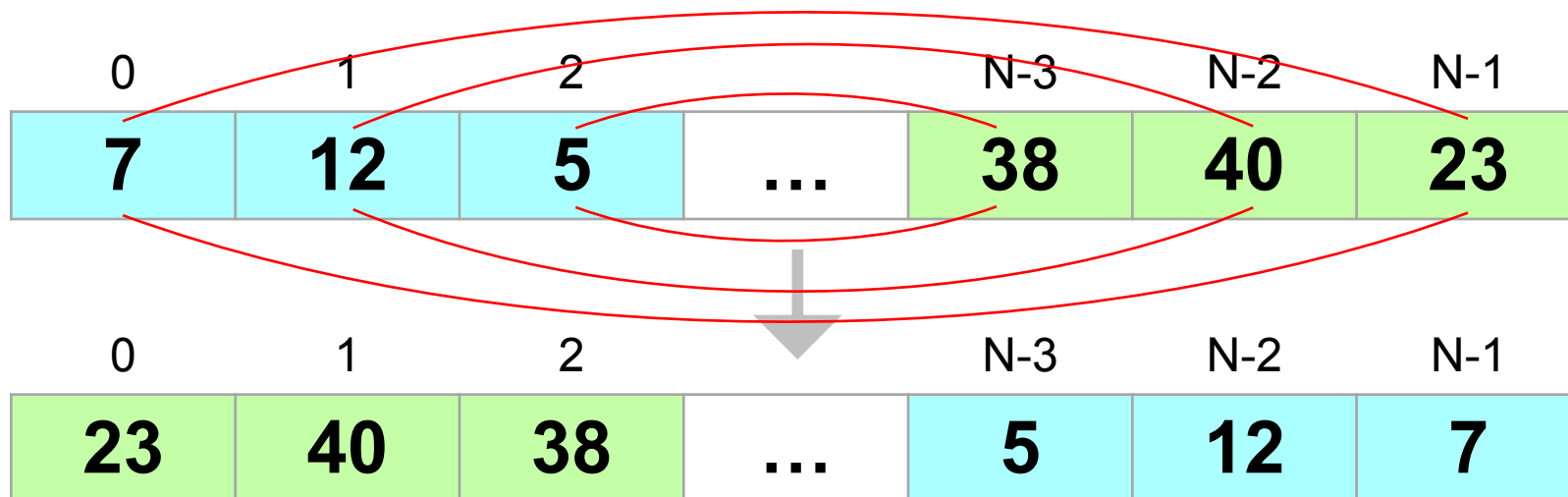
$A[2] \leftrightarrow A[3]$

...

$A[N-2] \leftrightarrow A[N-1]$

# Реверс массива

Задача. Переставить элементы массива в обратном порядке (выполнить *реверс*).



$$A[0] \leftrightarrow A[N-1]$$

$$0 + N - 1 = N - 1$$

$$A[1] \leftrightarrow A[N-2]$$

$$1 + N - 2 = N - 1$$

$$A[i] \leftrightarrow A[N-1-i]$$

$$i + ??? = N - 1$$

$$A[N-1] \leftrightarrow A[0]$$

$$N - 1 + 0 = N - 1$$

# Реверс массива

```
for i in range(N // 2):  
    поменять местами A[i] и A[N-1-i]
```

0	1	2	3
7	12	40	23
23	12	40	7
23	40	12	7
23	12	40	7
7	12	40	23

$i=0$

$i=1$

$i=2$

$i=3$



Что плохо?



Как исправить?

# Задачи

---

«А»: Заполнить массив из 10 элементов случайными числами в диапазоне [0; 100], поменять местами пары соседних элементов, не трогая первый и последний элементы.

Массив:        53 66 88 39 26 88 35 24 38 50

Результат:    53 88 66 26 39 35 88 38 24 50

«В»: Заполнить массив из 10 элементов случайными числами в диапазоне [0; 100] и выполнить циклический сдвиг влево: каждый элемент перемещается на место предыдущего, а первый – на место последнего.

Массив:        53 66 88 39 26 88 35 24 38 50

Результат:    66 88 39 26 88 35 24 38 50 53

# Задачи

---

«С»: Заполнить массив из 10 элементов случайными числами в диапазоне [0; 100], сделать реверс отдельно в первой и во второй половине массива.

**Массив:**            53  66  88  39  26  88  35  24  38  50

**Результат:**    26  39  88  66  53  50  38  24  35  88

# Линейный поиск в массиве

Задача. Найти в массиве элемент, равный  $X$ , и его номер.

$X = 5$

	5					
0	1	2	3	4	5	
7	12	38	5	40	23	

```
i = 0
while A[i] != X:
    i += 1
print("A[" + i + "] = " + X)
```



Что плохо?



Если искать 4?



Нельзя выходить за границы массива!

# Линейный поиск в массиве

не выходим за  
границу

```
i = 0
while i < N and A[i] != X:
    i += 1
if i < N:
    print( "A[" , i , "]=" , X )
else:
    print( "Не нашли!" )
```



Как проверить, нашли  
или нет?



# Досрочный выход из цикла

**Задача.** Найти в массиве элемент, равный X, и его номер.

```
nX = -1 # номер элемента
for i in range(N):
    if A[i]==X:
        nX = i # запомнить номер
        break
if nX >= 0:
    print( "A[" , nX, "]=" , X )
else:
    print( "Не нашли!" )
```

нашли!

сразу выйти из цикла

# Поиск в массиве

## Варианты в стиле Python:

```
for i in range ( N ) :
    if A[i] == X:
        print ( "A[" , i , "]" = " , X , sep = " " )
        break
else:
    print ( "Не нашли!" )
```

если не было досрочного выхода из цикла

```
if X in A:
    nX = A.index (X)
    print ( "A[" , nX , "]" = " , X , sep = " " )
else:
    print ( "Не нашли!" )
```

# Задачи

---

**«А»:** Напишите программу, которая заполняет массив из  $N = 10$  элементов случайными числами в диапазоне  $[0, 20]$ , выводит его на экран, а затем находит индекс первого элемента, равного введённому числу  $X$ . Программа должна вывести ответ «не найден», если в массиве таких элементов нет.

## Пример:

Массив: 5 16 2 13 3 14 18 13 16 9

Что ищем: 13

$A[3] = 13$

# Задачи

---

«В»: Напишите программу, которая заполняет массив из  $N = 10$  элементов случайными числами в диапазоне  $[-5,5]$ , выводит его на экран, а затем находит индекс **последнего** элемента, равного введённому числу  $X$ . Программа должна вывести ответ «не найден», если в массиве таких элементов нет.

**Пример:**

Массив: -5 -6 2 3 -3 0 8 -3 0 9

Что ищем: 0

$A[8] = 0$

# Задачи

---

**«С»:** Напишите программу, которая заполняет массив из  $N = 10$  элементов случайными числами в диапазоне  $[10, 15]$ , выводит его на экран, а затем находит индексы всех элементов, равных введённому числу  $X$ . Программа должна вывести ответ «не найден», если в массиве таких элементов нет.

## Пример:

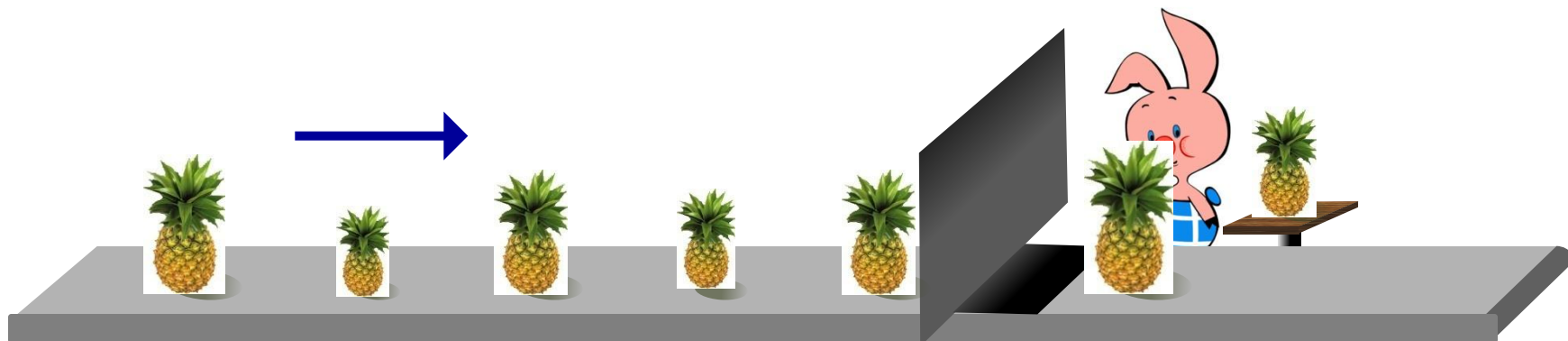
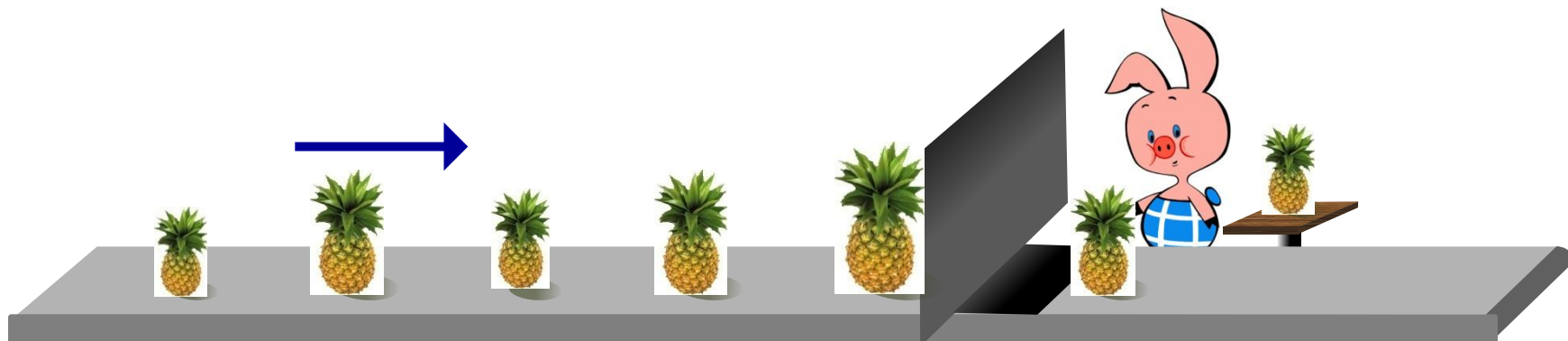
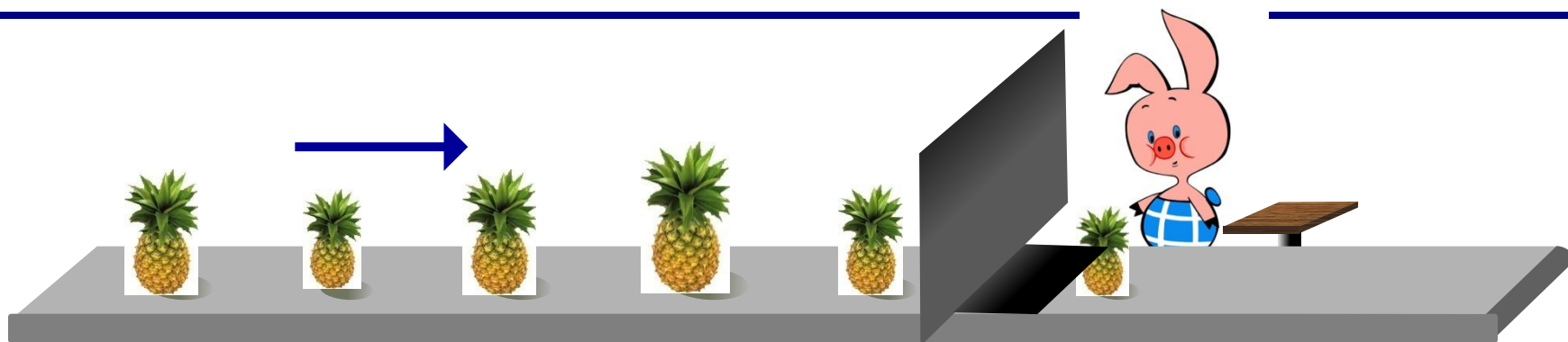
Массив: 12 15 10 13 13 15 10 14 12 13

Что ищем: 12

$A[0] = 12$

$A[8] = 12$

# Поиск максимального элемента



# Поиск максимального элемента

? Какие переменные нужны?

```
for i in range(N) :  
    if A[i] > M:  
        M = A[i]  
print( M )
```

? Чего не хватает?

? Какое начальное значение взять для M?

1) **M** – значение, которое заведомо меньше всех элементов массива

**или**

2) **M = A[0]** (или любой другой элемент)

максимальный не меньше, чем **A[0]**

# Поиск максимального элемента

---

```
M = A[0]
for i in range(1, N):
    if A[i] > M:
        M = A[i]
print( M )
```

начинаем с  $A[1]$ , так как  $A[0]$  мы уже посмотрели



Как найти минимальный?



# Поиск максимального элемента (Python)

```
M = A[0]
for x in A:
    if x > M:
        M = x
print( M )
```

перебрать все элементы  
в массиве A



Не нужно знать размер!

```
print( max(A) )
```


```
print( min(A) )
```

# Номер максимального элемента

Задача. Найти в массиве максимальный элемент и его номер.

 Какие переменные нужны?

```
M = A[0]; nMax = 0
for i in range(1, N):
    if A[i] > M:
        M = A[i]
        nMax = i
print( "A[" , nMax , "]" = " , M )
```

 Можно ли убрать одну переменную?

# Номер максимального элемента

❗ Если знаем `nMax`, то `M=A[nMax]`!

```
M = A[0]; nMax = 0
for i in range(1, N):
    if A[i] > A[nMax]:
        M = A[i]
        nMax = i
print( "A[" , nMax, "]" = " , A[nMax] )
```

# Максимальный элемент и его номер

---

## Вариант в стиле Python:

```
M = max (A)
nMax = A . index (M)
print ( "A[" , nMax , "]" = " , M )
```

номер заданного  
элемента (первого из...)

# Максимальный не из всех

Задача. Найти в массиве максимальный из отрицательных элементов.

```
M = A[0]
for i in range(1, N):
    if A[i] < 0 and A[i] > M:
        M = A[i]
print( M )
```



Что плохо?



Как исправить?

0	1	2	3	4
5	-2	8	3	-1

**M = 5**

# Максимальный не из всех

Задача. Найти в массиве максимальный из отрицательных элементов.

```
M = A[0]
for i in range(1, N):
    if A[i] < 0:
        if M >= 0 or A[i] > M:
            M = A[i]
print( M )
```

сначала записали  
неотрицательный!



Если нет отрицательных?

# Максимальный не из всех (Python)

Задача. Найти в массиве максимальный из отрицательных элементов.

```
B = [ x for x in A
      if x < 0 ]
print( max(B) )
if len(B) :
    print( max(B) )
else:
    print("Нет таких!")
```

отбираем нужные



Если нет отрицательных?

~~if len(B) != 0:~~

## Задачи (без **min** и **max**)

---

- «**A**»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [50; 150] и находит в нём минимальный и максимальный элементы и их номера.
- «**B**»: Напишите программу, которая заполняет массив из 10 элементов в виде  
 $A = [10, 2, 3, 4, 5, 10, 7, 8, 9, 10]$   
и выводит количество элементов, имеющих максимальное значение.
- «**C**»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке [100; 200] и находит в нём пару соседних элементов, сумма которых минимальна.



# Задачи

---

«D»: Напишите программу, которая заполняет массив из 20 элементов случайными числами на отрезке  $[-100; 100]$  и находит в каждой половине массива пару соседних элементов, сумма которых максимальна.

## Задачи-2 (максимум в потоке)

---

- «А»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Напишите программу, которая находит минимальное и максимальное среди полученных чисел.
- «В»: На вход программы поступает неизвестное количество целых чисел, ввод заканчивается нулём. Напишите программу, которая находит минимальное и максимальное из тех чисел, которые делятся на 3.
- «С»: На вход программы поступает неизвестное количество чисел целых, ввод заканчивается нулём. Напишите программу, которая находит максимальное двузначное число, заканчивающееся на 6, среди полученных чисел. Если такого числа нет, нужно вывести слово «Нет».

## Задачи-2 (максимум в потоке)

---

«D»: На вход программы поступает неизвестное количество целых чисел (не менее 2), ввод заканчивается нулём. Напишите программу, которая находит среди полученных чисел пару полученных друг за другом чисел, сумма которых максимальна.

# Сортировка

---

**Сортировка** — это расстановка элементов списка (массива) в заданном порядке.

*Задача.* Отсортировать элементы в порядке **возрастания** (*неубывания* – если есть одинаковые).

**Алгоритмы сортировки:**

- простые, но медленные (при больших  $N$ )
- быстрые, но сложные...

# Сортировка выбором

**?** Где должен стоять минимальный элемент?

- нашли минимальный, поставили его на первое место

```
c = A[nMin]
```

**?** Как?

```
A[0], A[nMin] = A[nMin], A[0]
```

```
A[0] = c
```

- из оставшихся нашли минимальный, поставили его на второе место и т.д.

**?** Что дальше?

5	-2	8	3	-1
-2	5	8	3	-1
-2	-1	8	3	5

# Сортировка выбором

```
for i in range(N-1):  
    # ищем минимальный среди A[i]..A[N-1]  
    nMin = i  
    for j in range(i+1, N):  
        if A[j] < A[nMin]:  
            nMin = j  
    # переставляем A[i] и A[nMin]  
    A[i], A[nMin] = A[nMin], A[i]
```

не трогаем те, которые  
уже поставлены



Почему цикл  $N-1$  раз?

Решение в стиле Python:

```
A.sort()
```

# Задачи

---

**«А»:** Напишите программу, которая заполняет массив из  $N = 10$  элементов случайными числами в диапазоне  $[0,20]$  и сортирует его в порядке убывания.

**Пример:**

**Массив:** 5 16 2 13 3 14 18 13 16 9

**Сортировка:** 18 16 16 14 13 13 9 5 3 2

**«В»:** Напишите программу, которая заполняет массив из  $N = 10$  элементов случайными числами в диапазоне  $[10,100]$  и сортирует его по возрастанию последней цифры числа (сначала идут все числа, которые заканчиваются на 0, потом все, которые заканчиваются на 1, и т.д.).

**Пример:**

**Массив:** 12 10 31 40 55 63 28 87 52 92

**Сортировка:** 10 40 31 12 52 92 63 55 87 28

# Задачи

---

**«С»:** Напишите программу, которая заполняет массив из  $N = 10$  элементов случайными числами в диапазоне  $[0,20]$  и сортирует его в порядке возрастания. На каждом шаге цикла выполняется поиск максимального (а не минимального!) элемента.

**Пример:**

**Массив:** 5 16 2 13 3 14 18 13 16 9

**Сортировка:** 2 3 5 9 13 13 14 16 16 18



# Программирование (Python)

## § 21. Матрицы (двумерные массивы)

# Что такое матрица?

	○	×
	○	×
○	×	

нет знака

НОЛИК

крестик

строка 2,  
столбец 3



Как закодировать?

**Матрица** — это прямоугольная таблица, составленная из элементов одного типа (чисел, строк и т.д.).

Каждый элемент матрицы имеет два индекса — номера строки и столбца.

# Создание матриц

**!** Матрица – это массив массивов!

```
A = [[-1, 0, 1],  
      [-1, 0, 1],  
      [0, 1, -1]]
```

перенос на другую  
строку внутри скобок

или так:

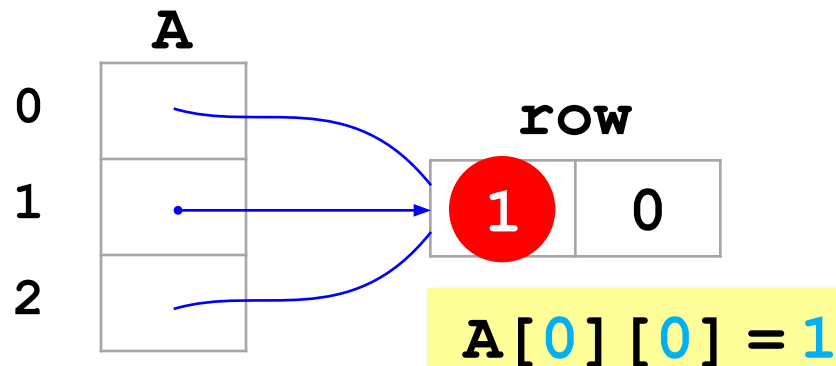
```
A = [[-1, 0, 1], [-1, 0, 1], [0, 1, -1]]
```

**!** Нумерация элементов с нуля!

# Создание матриц

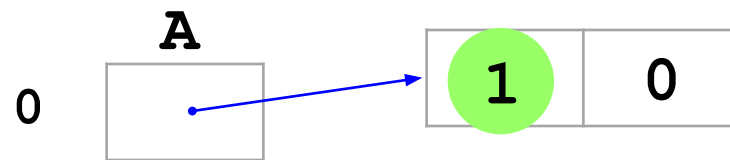
## Нулевая матрица:

```
N = 3
M = 2
row = [0] * M
A = [row] * N
```



## а правильно так:

```
A = []
for i in range(N):
    A.append( [0] * M )
```



```
A[0][0] = 1
```

# Вывод матриц

```
print ( A )
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
def printMatrix( A ):  
    for row in A:  
        for x in row:  
            print ( "{:4d}" .format(x) , end = "" )  
        print ()
```

```
1     2     3  
4     5     6  
7     8     9
```



Зачем форматный вывод?

# Простые алгоритмы

## Заполнение случайными числами:

```
from random import randint
for i in range(N):
    for j in range(M):
        A[i][j] = randint ( 20, 80 )
        print ( "{:4d}".format(A[i][j]),
                end = "" )
print()
```



Вложенный цикл!

## Суммирование:

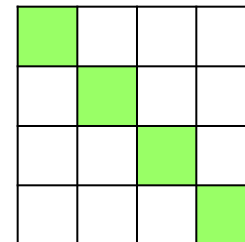
```
s = 0
for i in range(N):
    for j in range(M):
        s += A[i][j]
print ( s )
```

```
s = 0
for row in A:
    s += sum(row)
print ( s )
```

# Перебор элементов матрицы

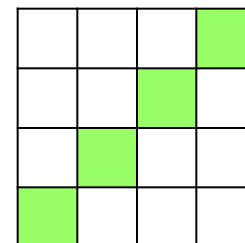
## Главная диагональ:

```
for i in range(N):  
    # работаем с A[i][i]
```



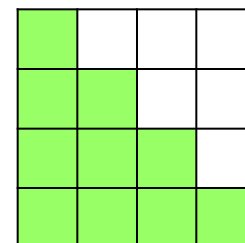
## Побочная диагональ:

```
for i in range(N):  
    # работаем с A[i][N-1-i]
```



## Главная диагональ и под ней:

```
for i in range(N):  
    for j in range(i+1):  
        # работаем с A[i][j]
```



# Перестановка строк

2-я и 4-я строки:

```
for j in range(M):
```

```
    A[2][j], A[4][j] = A[4][j], A[2][j]
```

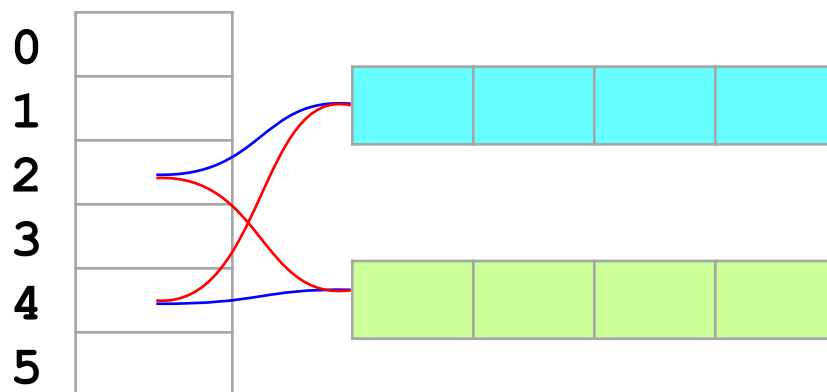
```
    A[2][j] = A[4][j]
```

```
    A[4][j] = c
```

	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						

Решение в стиле Python:

```
A[2], A[4] = A[4], A[2]
```





# Перестановка столбцов

2-й и 4-й столбцы:

```
for i in range(N):
```

```
    A[i][2], A[i][4] = A[i][4], A[i][2]
```

```
    A[i][2] = A[i][4]
```

```
    A[i][4] = c
```

	0	1	2	3	4	5
0			←→		←→	
1			←→		←→	
2			←→		←→	
3			←→		←→	
4			←→		←→	
5			←→		←→	

# Задачи

---

«А»: Напишите программу, которая заполняет матрицу случайными числами в диапазоне [10; 100] и находит максимальный элемент на главной диагонали квадратной матрицы.

**Пример:**

**Матрица А:**

12 34 14 65

71 88 23 45

87 46 53 39

76 58 24 92

**Результат:**  $A[3][3] = 92$

# Задачи

---

«В»: Напишите программу, которая заполняет матрицу случайными числами в диапазоне [10; 100] и находит максимальный элемент матрицы и его индексы (номера строки и столбца).

**Пример:**

**Матрица A:**

12 34 14 65

71 88 23 98

87 46 53 39

76 58 24 92

**Максимум:  $A[1][3] = 98$**

# Задачи

---

«С»: Напишите программу, которая заполняет матрицу случайными числами в диапазоне  $[-50; 50]$  и находит минимальный из чётных положительных элементов матрицы. Учтите, что таких элементов в матрице может и не быть.

**Пример:**

**Матрица A:**

16 34 14 65

71 88 23 45

87 12 53 39

76 58 24 92

**Результат:**  $A[2][1] = 12$

# Программирование (Python)

## § 22. Сложность алгоритмов

# Как сравнивать алгоритмы?

- быстродействие (**временная сложность**)
- объём требуемой памяти (**пространственная сложность**)
- понятность



Обычно не бывает все хорошо!

**Время работы алгоритма** – это количество элементарных операций  $T$ , выполненных исполнителем.

Функция  $T(N)$  называется

**временной сложностью алгоритма**

зависит от  
количества данных  
(размера массива  $N$ )

$$T(N) = 2N^3$$



Как увеличится время работы при увеличении  $N$  в 10 раз?

# Примеры определения сложности

Задача 1. Вычислить сумму первых трёх элементов массива (при  $N \geq 3$ ).

```
Sum = A[0] + A[1] + A[2]
```

$$T(N) = 3$$

2 сложения  
+ запись в  
память

Задача 2. Вычислить сумму всех элементов массива.

```
Sum = 0  
for i in range(N):  
    Sum += A[i]
```

$$T(N) = 2N + 1$$

$N$  сложений,  $N+1$   
операций записи

## Примеры определения сложности

---

Задача 3. Отсортировать все элементы массива по возрастанию методом выбора.

```
for i in range(N-1):  
    nMin = i  
    for j in range(i+1, N):  
        if A[i] < A[nMin]:  
            nMin = j  
    A[i], A[nMin] = A[nMin], A[i]
```

Число сравнений:

$$T_c(N) = (N-1) + (N-2) + \dots + 2 + 1 = \frac{N(N-1)}{2} = \frac{1}{2}N^2 - \frac{1}{2}N$$

Число перестановок:  $T_n(N) = N - 1$



# Примеры определения сложности

---

Задача 4. Найти сумму элементов квадратной матрицы размером  $N \times N$ .

```
Sum = 0
for i in range(N) :
    for j in range(N) :
        Sum += A[i][j]
```



Самостоятельно! 😊

# Сравнение алгоритмов по сложности

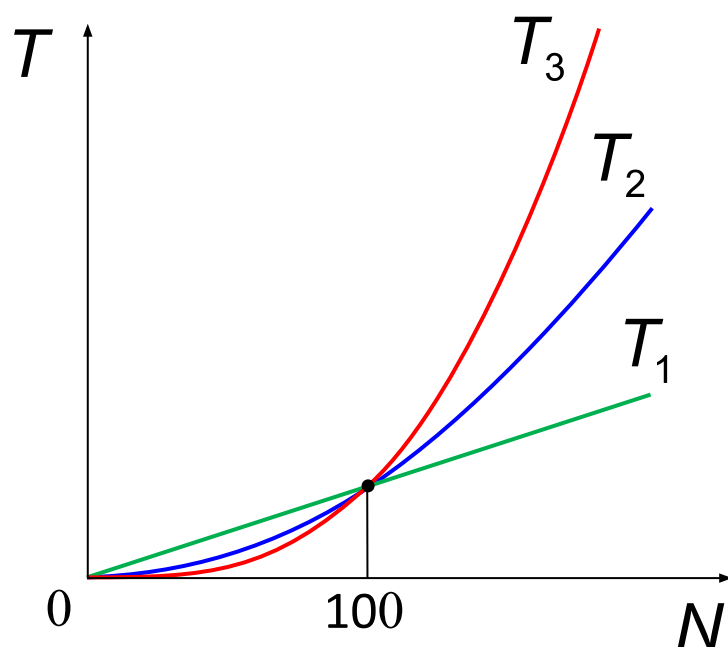
$$T_1(N) = 10000 \cdot N$$

$$T_2(N) = 100 \cdot N^2$$

$$T_3(N) = N^3$$



Какой алгоритм выбрать?



при  $N < 100$ :

$$T_3(N) < T_2(N) < T_1(N)$$

при  $N > 100$ :

$$T_3(N) > T_2(N) > T_1(N)$$



Нужно знать размер данных!

# Асимптотическая сложность

**Асимптотическая сложность** – это оценка скорости роста количества операций при больших значениях  $N$ .

не зависит от  $N$

постоянная

сложность  $O(1)$   $\Leftrightarrow T(N) \leq c$  для  $N \geq N_0$

$$T(N) = 100 \leq 101 \text{ для всех } N \geq 1 \Rightarrow O(1)$$

линейная

сложность  $O(N)$   $\Leftrightarrow T(N) \leq c \cdot N$  для  $N \geq N_0$

сумма элементов массива:

$$T(N) = 2 \cdot N - 1 \leq 2 \cdot N \text{ для } N \geq 1 \Rightarrow O(N)$$

# Асимптотическая сложность

квадратичная

сложность  $O(N^2)$   $\Leftrightarrow T(N) \leq c \cdot N^2$  для  $N \geq N_0$

сортировка методом выбора:

$$T_c(N) = \frac{1}{2}N^2 - \frac{1}{2}N \leq \frac{1}{2}N^2 \text{ для } N \geq 0 \Rightarrow O(N^2)$$

кубическая

сложность  $O(N^3)$   $\Leftrightarrow T(N) \leq c \cdot N^3$  для  $N \geq N_0$

# Асимптотическая сложность

сложность  $O(2^N)$

сложность  $O(N!)$

задачи оптимизации,  
полный перебор вариантов

**Факториал числа  $N$ :**  $N! = 1 \cdot 2 \cdot 3 \dots \cdot N$

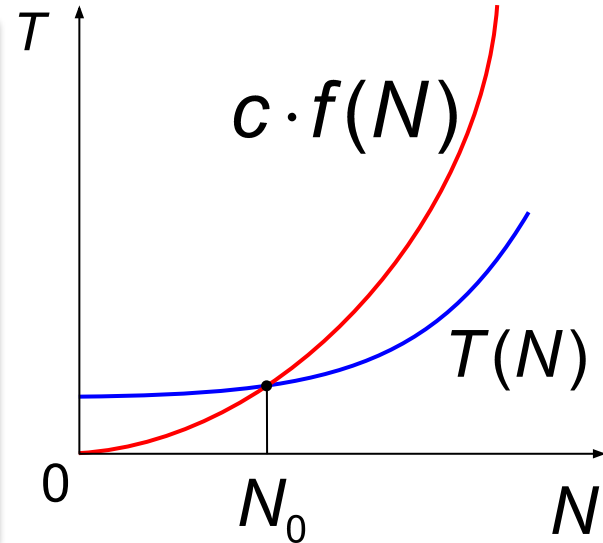
$N$	$T(N)$	время выполнения
	$N$	100 нс
	$N^2$	10 мс
	$N^3$	0,001 с
	$2^N$	$10^{13}$ лет
	$N!$	$10^{141}$ лет

$N = 100,$   
1 млрд оп/с

# Асимптотическая сложность

Алгоритм относится к классу  $O(f(N))$ , если найдется такая постоянная  $c$ , что начиная с некоторого  $N = N_0$  выполняется условие

$$T(N) \leq c \cdot f(N)$$



это верхняя оценка!

$$O(N) \Rightarrow O(N^2) \Rightarrow O(N^3) \Rightarrow O(2^N)$$

«Алгоритм имеет сложность  $O(N^2)$ ».

обычно – наиболее точная верхняя оценка!

# Программирование (Python)

## § 23. Как разрабатывают программы

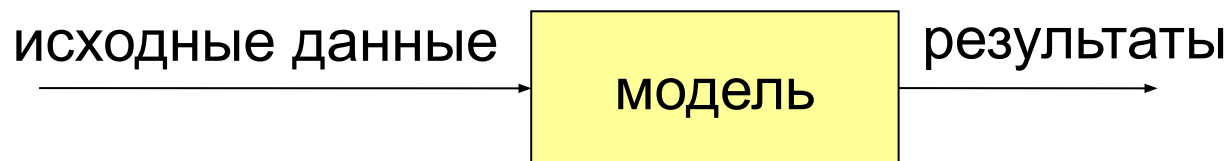
# Этапы разработки программ

---

## I. Постановка задачи

Документ: *техническое задание*.

## II. Построение модели



*Формализация*: запись модели в виде формул (на формальном языке).

## III. Разработка алгоритма и способа хранения данных

«Алгоритмы + структуры данных = программы»  
(Н. Вирт)

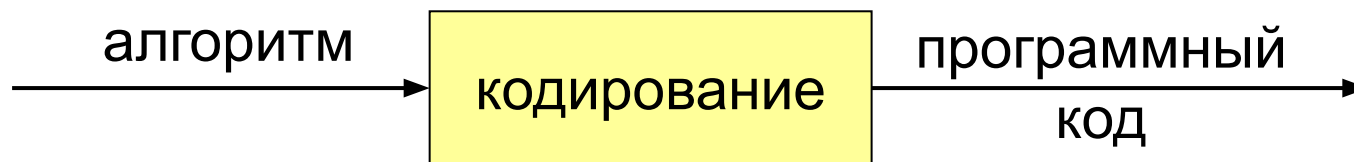


# Этапы разработки программ

---

## IV. Кодирование

Запись алгоритма на языке программирования.



## V. Отладка

Поиск и исправление ошибок в программах:

- **синтаксические** – нарушение правил языка программирования
- **логические** – ошибки в алгоритме могут приводить к **отказам** – аварийным ситуациям во время выполнения (*run-time error*)

# Этапы разработки программ

---

## VI. Тестирование

Тщательная проверка программы во всех режимах:

- **альфа-тестирование** – внутри компании (тестировщики)
- **бета-тестирование** – (доверенные) пользователи

## VII. Документирование

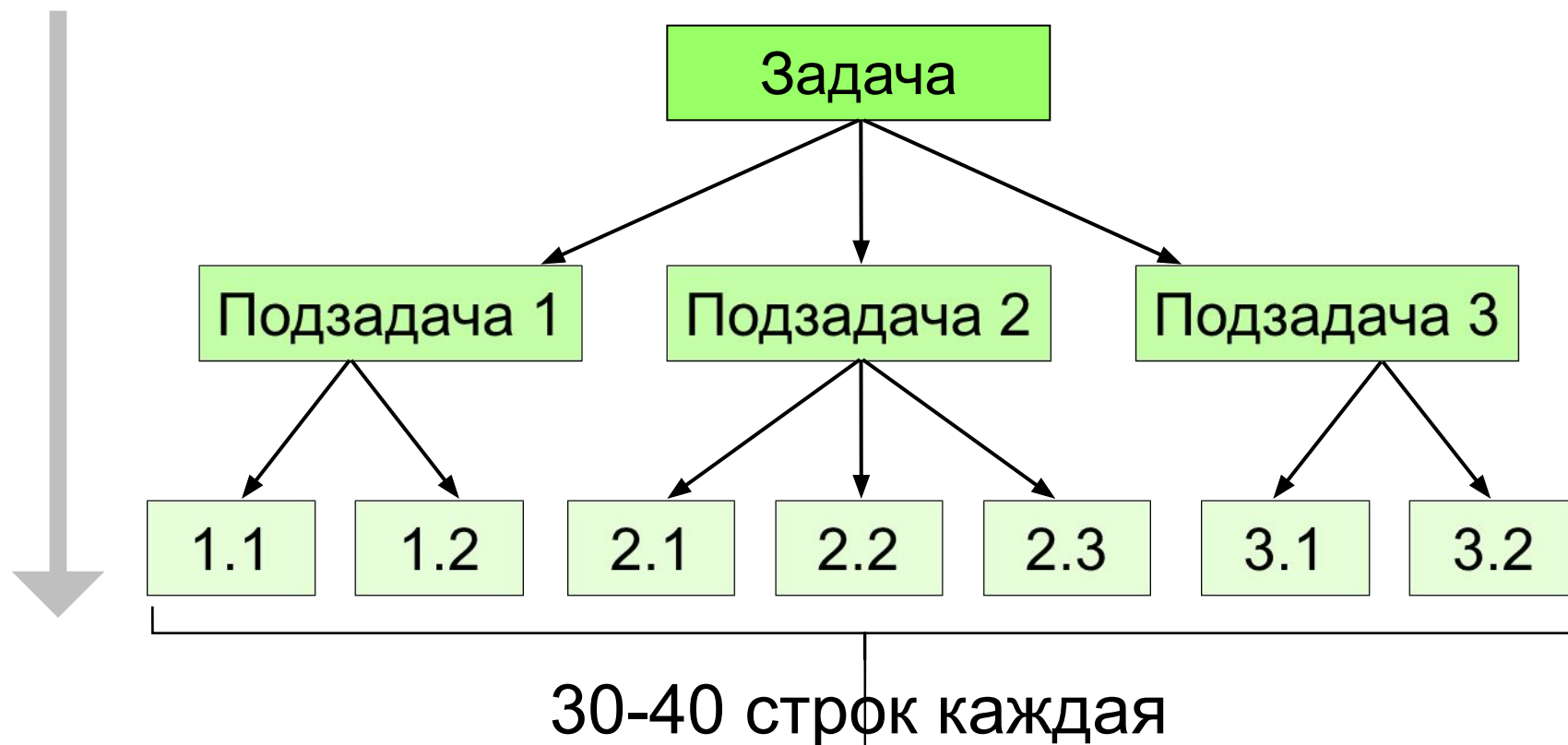
Технические писатели

## VIII. Внедрение и сопровождение

- обучение пользователей
- исправление найденных ошибок
- техподдержка

# Методы проектирования программ

## «Сверху вниз» (последовательное уточнение)



# Методы проектирования программ

---

## «Сверху вниз» (последовательное уточнение)



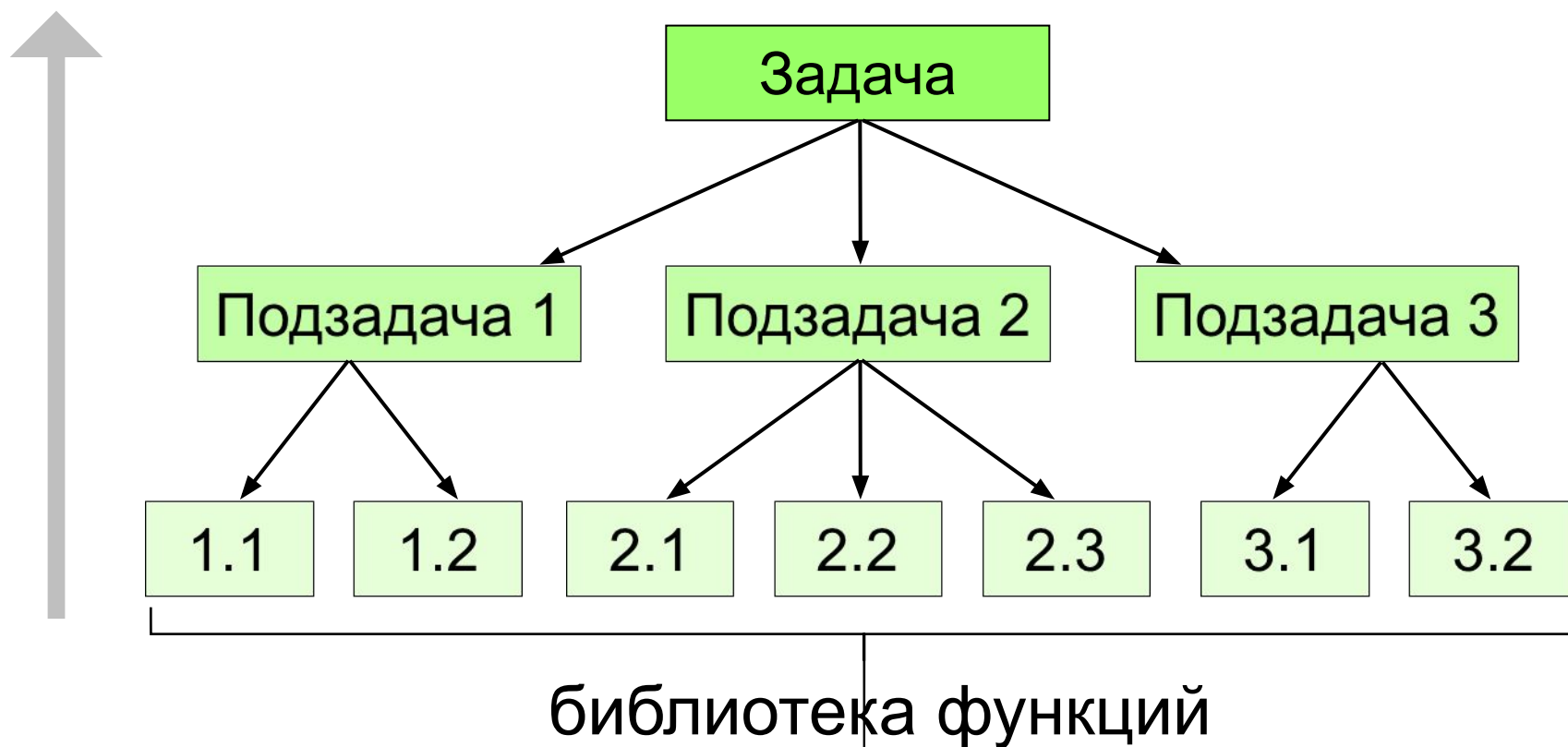
- сначала задача решается «в целом»
- легко распределить работу
- легче отлаживать программу (всегда есть полный работающий вариант)



- в нескольких подзадачах может потребоваться решение одинаковых подзадач нижнего уровня
- быстродействие не известно до последнего этапа (определяется нижним уровнем)

# Методы проектирования программ

## «Снизу вверх» (восходящее)



# Методы проектирования программ

---

## «Снизу вверх» (восходящее)



- нет дублирования
- сразу видно быстрое действие



- сложно распределять работу
- сложнее отлаживать (увеличение числа связей)
- плохо видна задача «в целом», может быть нестыковка на последнем этапе



Почти всегда используют оба подхода!

# Отладка программы

Программа решения квадратного уравнения

$$ax^2 + bx + c = 0$$

```
from math import sqrt
print("Введите a, b, c: ")
a = float(input())
b = float(input())
c = float(input())
D = b*b - 4*a*a
x1 = (-b+sqrt(D))/2*a
x2 = (-b-sqrt(D))/2*a
print("x1=", x1, " x2=", x2, sep="")
```

float – преобразовать в вещественное число

# Тестирование

Тест 1.  $a = 1, b = 2, c = 1.$

Ожидание:

`x1=-1.0 x2=-1.0`

Реальность:

`x1=-1.0 x2=-1.0`



Тест 2.  $a = 1, b = -5, c = 6.$

`x1=3.0 x2=2.0`

`x1=4.791 x2=0.209`



Найден вариант, когда программа работает неверно.  
Ошибка **воспроизводится!**

## Возможные причины:

- неверный ввод данных
- неверное вычисление дискриминанта
- неверное вычисление корней
- неверный вывод результатов

$$D = b^2 - 4ac$$

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$$



# Отладочная печать

---

Идея: выводить все промежуточные результаты.

```
a = float(input())
```

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

```
c = float(input())
```

```
print(a, b, c)
```

```
D = b*b - 4*a*a
```

```
print("D=", D)
```

```
...
```

# Отладочная печать

Идея: выводить все промежуточные результаты.

Результат:

Введите  $a$ ,  $b$ ,  $c$ :

1

-5

6

1.0 -5.0 6.0

D= 21.0

$$D = b^2 - 4ac = 25 - 4 \cdot 1 \cdot 6 = 1$$

```
D = b*b - 4*a*c ;
```



Одна ошибка найдена!

# Отладка программы

Тест 1.  $a = 1, b = 2, c = 1.$

Ожидание:

`x1=-1.0 x2=-1.0`

Реальность:

`x1=-1.0 x2=-1.0`



Тест 2.  $a = 1, b = -5, c = 6.$

`x1=3.0 x2=2.0`

`x1=3.0 x2=2.0`



Программа работает верно?

Тест 3.  $a = 8, b = -6, c = 1.$

`x1=0.5 x2=0.25`

`x1=32.0 x2=16.0`



`x1 = (-b+sqrt(D)) / (2*a)`

`x2 = (-b-sqrt(D)) / (2*a)`



Что неверно?

# Документирование программы

---

- назначение программы
- формат входных данных
- формат выходных данных
- примеры использования программы

## Назначение:

программа для решения уравнения

$$ax^2 + bx + c = 0$$

## Формат входных данных:

значения коэффициентов  $a$ ,  $b$  и  $c$  вводятся с клавиатуры через пробел в одной строке

# Документирование программы

---

## Формат выходных данных:

значения вещественных корней уравнения;  
если вещественных корней нет, выводится  
слово «нет»

## Примеры использования программы:

1. Решение уравнения  $x^2 - 5x + 6 = 0$

Введите a, b, c: **1 -5 6**

**x1=3 x2=2**

2. Решение уравнения  $x^2 + x + 6 = 0$

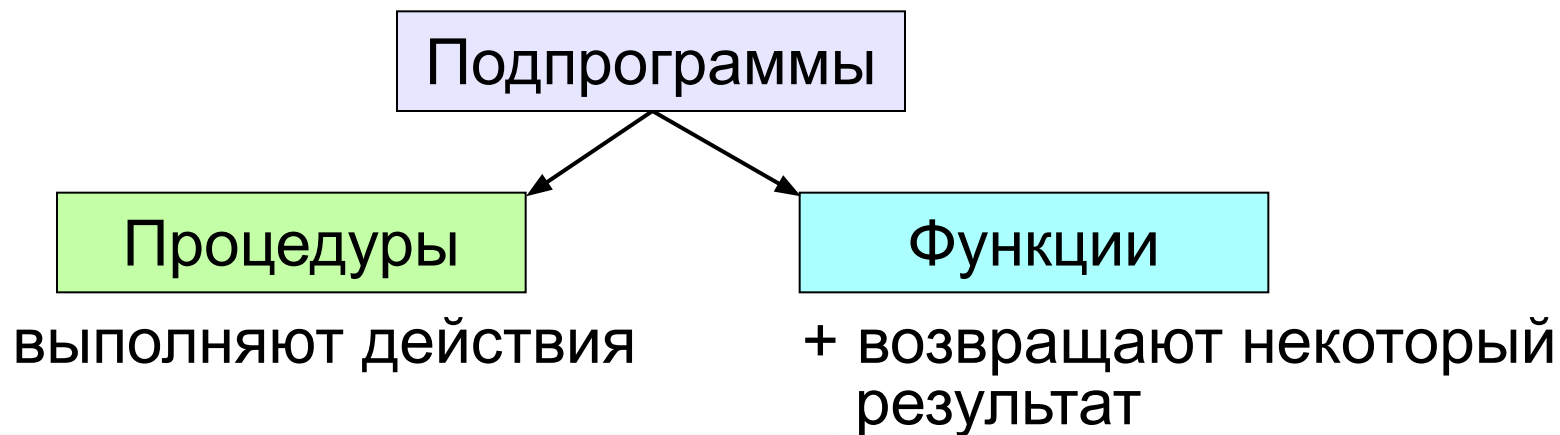
Введите a, b, c: **1 1 6**

**Нет.**

# Программирование (Python)

## § 24. Процедуры

# Два типа подпрограмм



**?** Процедура или функция?

- а) рисует окружность на экране
- б) определяет площадь круга
- в) вычисляет значение синуса угла
- г) изменяет режим работы программы
- д) возводит число  $x$  в степень  $y$
- е) включает двигатель автомобиля
- ж) проверяет оставшееся количество бензина в баке
- з) измеряет высоту полёта самолёта

# Простая процедура

*define* – определить

```
def printLine():  
    print("-----")
```

```
...  
printLine()  
...
```

ВЫЗОВ  
процедуры

какие-то  
операторы



Что делает?



- можно вызывать сколько угодно раз
- нет дублирования кода
- изменять – в одном месте



# Линии разной длины

```
def printLine5():  
    print("-----")
```

```
def printLine10():  
    print("-----")
```

```
def printLine10():  
    print("-"*10)
```

```
def printLine(n):  
    print("-"*n)
```



Как улучшить?

параметр  
процедуры

# Процедура с параметром

**Параметр** – величина, от которой зависит работа процедуры.

```
def printLine ( n ) :
```

```
...
```

```
...  
printLine (10)
```

```
...  
printLine (7)  
printLine (5)  
printLine (3)
```



Что делает?

**Аргумент** – значение параметра при конкретном вызове.

# Несколько параметров

символьная строка



Что изменилось?

```
def printLine (c, n) :  
    print (c*n)
```



Как вызывать?

✓ `printLine ( "+" , 5 )`

✓ `printLine ( "+-+" , 5 )`

✓ `printLine ( 5 , "+" )`

# В других языках программирования

---

## Паскаль:

```
procedure printLine (c: string; n: integer);  
var i: integer;  
begin  
    for i:=1 to n do  
        write(c);  
    writeln  
end;
```

# В других языках программирования

---

C:

```
void printLine(int n)
{
    int i;
    for (i=1; i<=n; i++)
        putchar("-");
    putchar("\n");
}
```

# Как не нужно писать процедуры

```
def summa():  
    print(x + y)
```

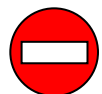
```
x = 10  
y = 5  
summa()
```



Что плохо?

```
def summa(x, y):  
    print(x + y)
```

```
x = 10  
y = 5  
summa(x, y)  
summa(2*x+y, 7)
```



- только  $x + y$
- не перенести в другую программу



Процедура принимает данные только через параметры!

# Задачи

---

«А»: Напишите процедуру, которая принимает параметр – натуральное число  $N$  – и выводит на экран две линии из  $N$  символов "–".

**Пример:**

Длина цепочки: 7

```
-----  
-----
```

«В»: Напишите процедуру, которая принимает один параметр – натуральное число  $N$ , – и выводит на экран прямоугольник длиной  $N$  и высотой 3 символа.

**Пример:**

Длина прямоугольника: 7

```
ooooooo  
o      o  
ooooooo
```

# Задачи

---

«С»: Напишите процедуру, которая выводит на экран квадрат со стороной  $N$  символов. При запуске программы  $N$  нужно ввести с клавиатуры.

**Пример:**

Сторона квадрата: 5

ooooo

o     o

o     o

o     o

ooooo



# Задачи

---

«D»: Напишите процедуру, которая выводит на экран треугольник со стороной  $N$  символов. При запуске программы  $N$  нужно ввести с клавиатуры.

**Пример:**

Сторона : 5

```
o
oo
ooo
oooo
ooooo
```

# Рекурсия

Задача. Вывести на экран двоичный код натурального числа.

```
def printBin( n ):  
    ...
```

Алгоритм перевода через остатки:

```
while n != 0:  
    print( n % 2, end="" )  
    n = n // 2
```

011 ❌

в обратном порядке!



Что получится  
при  $n = 6$ ?

# Рекурсия

Чтобы вывести двоичную запись числа  $n$ , нужно сначала вывести двоичную запись числа  $(n // 2)$ , а затем — его последнюю двоичную цифру, равную  $(n \% 2)$ .

двоичная запись числа 6

110

$6 \% 2$

двоичная запись числа 3



Чтобы решить задачу, нужно решить ту же задачу для меньшего числа!

Это и есть рекурсия!



Чтобы понять рекурсию, нужно понять рекурсию! 😊

# Рекурсивная процедура

```
def printBin( n ) :  
    printBin( n % 2 )  
    print( n % 2, end = "" )
```

вызывает сама себя!

**Рекурсивная процедура** — это процедура, которая вызывает сама себя.

```
printBin(6)
```

```
printBin(3)
```

```
printBin(1)
```

```
printBin(0)
```

```
printBin(0)
```



Что получится? `printBin(6)`

бесконечные вызовы



Как исправить?

# Рекурсивная процедура

```
def printBin( n ) :  
    if n == 0: return  
    printBin( n // 2 )  
    print( n % 2, end="" )
```



Что получится?  
`printBin(6)`

```
printBin(6)
```

```
    printBin(3)
```

```
        printBin(1)
```

```
            printBin(0)
```

```
            print(1 % 2)
```

```
        print(3 % 2)
```

```
    print(6 % 2)
```

рекурсия  
заканчивается!

1 1 0

# Задачи

---

«**A**»: Напишите рекурсивную процедуру, которая переводит число в восьмеричную систему.

**Пример:**

Введите число: **66**

В восьмеричной: 102

«**B**»: Напишите рекурсивную процедуру, которая переводит число в любую систему счисления с основанием от 2 до 9.

**Пример:**

Введите число: **75**

Основание: **6**

В системе с основанием 6: 203

# Задачи

---

**«С»:** Напишите рекурсивную процедуру, которая переводит число в шестнадцатеричную систему.

**Пример:**

Введите число: **123**

В шестнадцатеричной: **7B**

**«D»:** Напишите рекурсивную процедуру, которая переводит число в любую систему счисления с основанием от 2 до 36.

**Пример:**

Введите число: **350**

Основание: **20**

В системе с основанием 20: **HA**

# Программирование (Python)

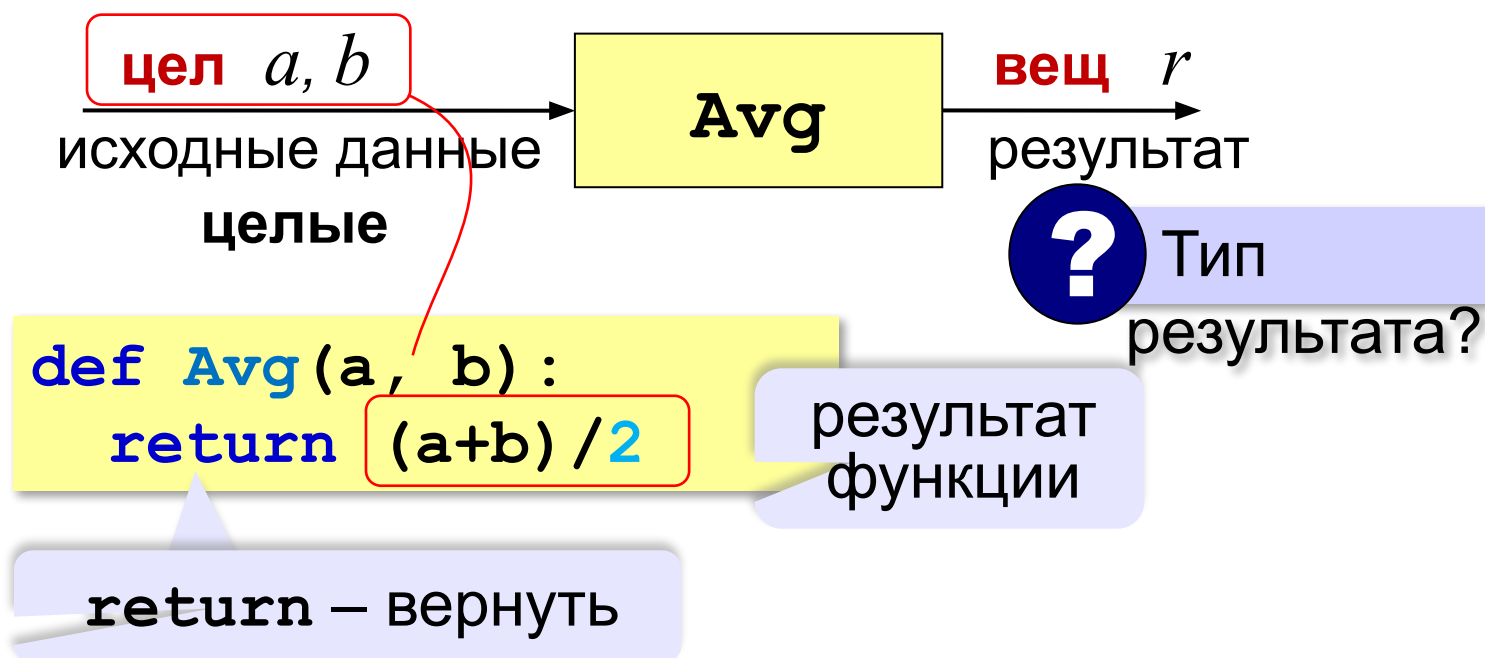
## § 25. Функции



# Что такое функция?

**Функция** — это вспомогательный алгоритм, который возвращает результат (число, строку символов и др.).

**Задача.** Написать функцию, которая вычисляет среднее арифметическое двух целых чисел.



# Как вызывать функцию?

Запись результата в переменную:

```
sr = Avg (5 , 8) 6.5
```



Чему равно?

```
x = 2; y = 5  
sr = Avg (x, 2*y+8) 10
```

Вывод на экран:

```
x = 2; y = 5  
sr = Avg (x, y+3) 5  
print ( Avg (12, 7) ) 9.5  
print ( sr + Avg (x, 12) ) 12
```

# Как вызывать функцию?

---

Использование в условных операторах:

```
a = int(input())
b = int(input())
if Avg(a,b) > 5:
    print("Да!")
else:
    print("Нет!");
```



Когда печатает «Да»?

# Как вызывать функцию?

---

Использование в циклах:

```
a = int(input())
b = int(input())
while Avg(a,b) > 0:
    print("Нет!")
    a,b = map(int, input().split())
print("Угадал!");
```

ВВОД ДВУХ ЧИСЕЛ В  
ОДНОЙ СТРОЧКЕ



Когда напечатает «Угадал»?

# В других языках программирования

## Паскаль:

```
function Avg(a, b: integer): real;  
begin  
  Avg := (a+b) / 2  
end.
```

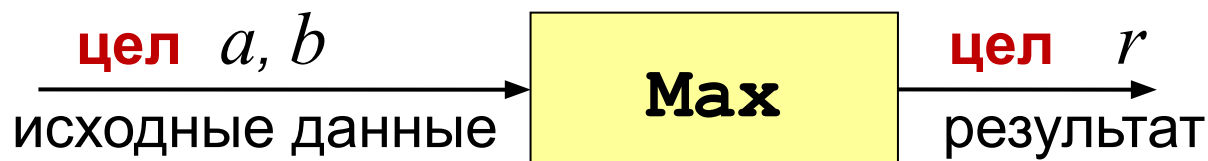
специальная переменная для записи результата функции

## C:

```
float Avg(int a, int b)  
{  
  return (a+b) / 2.0;  
}
```

# Максимум из двух (трёх) чисел

*Задача.* Составить функцию, которая определяет наибольшее из двух целых чисел.



```
def Max(a, b):  
    if a > b then  
        return a  
    else:  
        return b
```



Как с её помощью найти максимум из трёх?

```
def Max3(a, b, c):  
    return Max(Max(a, b), c)
```

# Сумма цифр числа

---

*Задача.* Составить функцию, которая вычисляет сумму значений цифр натурального числа.

```
def sumDigits ( N ) :  
    sum = 0           # накапливаем сумму с 0  
    while N != 0 :  
        d = N % 10   # выделим последнюю цифру  
        sum += d     # добавим к сумме  
        N = N // 10  # удалим последнюю цифру  
    return sum
```

# Задачи

---

«**A**»: Напишите функцию, которая вычисляет среднее арифметическое пяти целых чисел.

**Пример:**

Введите 5 чисел: **1 2 3 4 6**

Среднее: **3.2**

«**B**»: Напишите функцию, которая находит количество цифр в десятичной записи числа.

**Пример:**

Введите число: **751**

Количество цифр: **3**



# Задачи

---

**«С»:** Напишите функцию, которая находит количество единиц в двоичной записи числа.

**Пример:**

Введите число: **75**

Количество единиц: **4**

# Логические функции

---

**Логическая функция** — это функция, возвращающая логическое значения (**да** или **нет**).

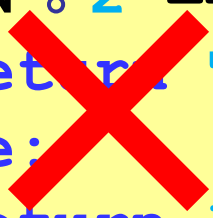
- можно ли применять операцию?
- успешно ли выполнена операция?
- обладают ли данные каким-то свойством?

# Логические функции

---

**Задача.** Составить функцию, которая возвращает «**True**», если она получила чётное число и «**False**», если нечётное.

```
def Even ( N ) :  
    if N % 2 == 0 :  
        return True  
    else :  
        return False
```



```
def Even ( N ) :  
    return (N % 2 == 0)
```

# Рекурсивные функции

**Рекурсивная функция** — это функция, которая вызывает сама себя.

*Задача.* Составить рекурсивную функцию, которая вычисляет сумму цифр числа.

**?** Как сформулировать решение рекурсивно?

Сумму цифр числа N нужно выразить через сумму цифр другого (меньшего) числа.

Сумма цифр числа N равна значению последней цифры плюс сумма цифр числа, полученного отбрасыванием последней цифры.

`sumDig(12345) = 5 + sumDig(1234)`

# Рекурсивная функция

## Сумма цифр числа N

Вход: натуральное число  $N$ .

Шаг 1:  $d = N \% 10$

Шаг 2:  $M = N // 10$

Шаг 3:  $s$  = сумма цифр числа  $M$

Шаг 4:  $sum = s + d$

Результат:  $sum$ .

последняя цифра

число без  
последней цифры



Что забыли?



Когда остановить?

# Сумма цифр числа (рекурсия)

```
def sumDigRec( N ):  
    if N == 0: return 0  
    else:  
        d = N % 10  
        sum = sumDigRec( N // 10 )  
        return sum + d
```



Зачем это?



Где рекурсивный вызов?

# Задачи

---

**«А»:** Напишите логическую функцию, которая возвращает значение «истина», если десятичная запись числа заканчивается на цифру 0 или 1.

**Пример:**

Введите число: **1230**

Ответ: Да

**«В»:** Напишите логическую функцию, которая возвращает значение «истина», если переданное ей число помещается в 8-битную ячейку памяти.

**Пример:**

Введите число: **751**

Ответ: Нет

# Задачи

---

**«С»:** Напишите логическую функцию, которая возвращает значение «истина», если переданное ей число простое (делится только на само себя и на единицу).

**Пример:**

Введите число: **17**

Число простое!

**Пример:**

Введите число: **18**

Число составное!



# Конец фильма

---

**ПОЛЯКОВ Константин Юрьевич**

д.т.н., учитель информатики

ГБОУ СОШ № 163, г. Санкт-Петербург

[kpolyakov@mail.ru](mailto:kpolyakov@mail.ru)

**ЕРЕМИН Евгений Александрович**

к.ф.-м.н., доцент кафедры мультимедийной

дидактики и ИТО ПГГПУ, г. Пермь

[eremin@pspu.ac.ru](mailto:eremin@pspu.ac.ru)

# Источники иллюстраций

---

1. иллюстрации художников издательства «Бином»
2. авторские материалы