Программирование на языке Python 9 класс

1. Повторение
2. Обработка потока данных
3. Обработка потока данныхОбработка потока данных (Обработка потока данных (цикл Обработка потока данных (цикл for)
4. Обработка массивов
5. Как разрабатывают программы
6. Процедуры

Программирование на языке Python

1. Повторение

Вывод на экран

Текст:

```
print ( "a", "b" )
```

Значения переменных из памяти:

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

Арифметические выражения:

```
print ( a + 2*b )
```

Все вместе:

```
print ( a, "+", b, "=",  a+b )
```

Подключение русского языка:

```
# coding: utf-8
```

Вывод на экран

С пробелами:

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

Без пробелов:

```
print ( a, b, sep = "" )
```

Без перехода на новую строку:

```
print ( a, b, end = "" )
```

Ввод данных с клавиатуры

Символьная строка:

```
print( 'Введите имя:' )
s = input()
```

или так:

```
s = input('Введите имя:')
```

Целое число:

```
print( 'Введите целое число:' )
n = int(input())
```

или так:

```
n = int(input('Введите целое число:'))
```

Ввод данных с клавиатуры

Вещественное число:

```
print( 'Введите число:' )
x = float(input())
```

или так:

```
x = float(input('Введите число:'))
```

Ввод данных с клавиатуры

Два целых числа (каждое в отдельной строке):

```
print( 'Введите два числа:' )
a = int (input())
b = int (input())
```

в одной строке:

```
print( 'Введите два числа:' )
a, b = map(int, input().split())
```

```
input()              # "21 35"
input().split()      # ["21", "35"]
a = int("21")
b = int("35")
```

символьные строки

Присваивание

```
a = 6
b = 4
a = 2*a + 3*b
b = a / 2 * b
```

Сокращённая запись операций:

```
a += 1
b += a
a *= 2 + 3*b
b /= 2 * a
```

Остаток от деления – %

```
a = 1234
d = a % 10; print( d )
a = a // 10
d = a % 10; print( d )
a = a // 10
d = a % 10; print( d )
a = a // 10
d = a % 10; print( d )
a = a // 10 :
```

4

3

2

1

Задания

«3»: Ввести три числа: цену пирожка (два числа: рубли, потом – копейки) и количество пирожков. Найти сумму, которую нужно заплатить (рубли и копейки)

Пример:

```
Стоимость пирожка:
12 50
Сколько пирожков:
5
К оплате: 62 руб. 50 коп.
```

«4»: Ввести число, обозначающее количество секунд. Вывести то же самое время в часах, минутах и секундах.

Пример:

```
Число секунд:
8325
2 ч. 18 мин. 45 с
```

Задания

«5»: Занятия в школе начинаются в 8-30. Урок длится 45 минут, перерывы между уроками – 10 минут. Ввести номер урока и вывести время его окончания.

Пример:

```
Введите номер урока:
6
13-50
```

Условный оператор

```
if a > b:
    # что делать, если a > b
else:
    # что делать, если a <= b
```

отступы!

```
a = 12
if a > 20:
   a = 15
print ( a )
```

```
a = 12
if a > 2:
   a = 15
else:
   a = 8
print ( a )
```

Цепочка условий

```
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!

Задачи

«3»: Напишите программу, которая получает три числа - рост трёх спортсменов, и выводит сообщение «По росту.», если они стоят по возрастанию роста, или сообщение «Не по росту!», если они стоят не по росту.

Пример:

```
Введите рост трёх спортсменов:
165 170 172
По росту.
```

Пример:

```
Введите рост трёх спортсменов:
175 170 172
Не по росту!
```

Задачи

«4»: Напишите программу, которая получает номер месяца и выводит соответствующее ему время года или сообщение об ошибке.

Пример:

```
Введите номер месяца:
5
Весна.
```

Пример:

```
Введите номер месяца:
15
Неверный номер месяца.
```

Задачи

«5»: Напишите программу, которая получает возраст человека (целое число, не превышающее 120) и выводит этот возраст со словом «год», «года» или «лет». Например, «21 год», «22 года», «25 лет».

Пример:

```
Введите возраст: 18
Вам 18 лет.
```

Пример:

```
Введите возраст: 21
Вам 21 год.
```

Пример:

```
Введите возраст: 22
Вам 22 года.
```

Цикл с условием

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

При каком условии заканчивает работу?

`k >= 10`

```
k = 10
while k > 0:
    print ( "Привет" )
    k -= 1
```

При каком условии заканчивает работу?

`k <= 0`

Цикл по переменной

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

сделай N раз

range(N) = [0, 1, 2, …, N-2, N-1]

[0,1,2,3]

N раз

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

0
1
2
3

Что выведет?

[0,1,2,3,4]

```
s = 0
for i in range(5):
    s += i
print(s)
```

10

Цикл по переменной

```
s = 0
for i in range(2,5):
   s += i
print(s)
```

от

до (не включая!)

[2,3,4]

s=2+3+4=9

→ 9

Кумир:

```
s := 0
нц для i от 2 до 4
   s := s + i
кц
вывод s
```

Паскаль:

```
s := 0;
for i:=2 to 4 do
   s := s + i;
writeln(s);
```

Цикл по переменной

```
s = 8
for i in range(2,15):
    s += 5
print(s)
```

[2,3,…,14]

? Сколько раз?

N=15-2=13

s=8+5*13=73

→ 73

Кумир:

N=14-2+1

```
s := 8
нц для i от 2 до 14
   s := s + 5
кц
вывод s
```

Паскаль:

```
s := 8;
for i:=2 to 14 do
   s = s + 5;
writeln(s);
```

Что выведет программа?

```
s = 3
for i in range(5,25):
   s += 10
print(s)
```

203

Кумир:

```
s := 3
нц для i от 5 до 24
   s := s + 10
кц
вывод s
```

Паскаль:

```
s := 3;
for i:=5 to 24 do
   s = s + 10;
writeln(s);
```

Что выведет программа?

```
s = 1
for i in range(3,8):
   s *= 2
print(s)
```

32

Кумир:

```
s := 1
нц для i от 3 до 7
   s := s * 2
кц
вывод s
```

Паскаль:

```
s := 1;
for i:=3 to 7 do
   s = s * 2;
writeln(s);
```

Что выведет программа?

```
k = 3
for i in range(4,8):
   k = 2*k + i
print(k)
```

```
k = 3
    2*3+4=10
    2*10+5=25
    2*25+6=56
    2*56+7= 119
```

Кумир:

```
k := 3
нц для i от 4 до 7
   k := 2*k + i
кц
вывод k
```

Паскаль:

```
k := 3;
for i:=4 to 7 do
   k = 2*k + i;
writeln(k);
```

Задачи

«3»: Ввести число N и вывести на экран все степени числа 2 от 2^1 до 2^N.

Пример:

```
Введите N:
3
2 4 8
```

«4»: Найдите все пятизначные числа, которые при делении на 133 дают в остатке 125, а при делении на 134 дают в остатке 111.

Задачи

«5»: Натуральное число называется **числом Армстронга**, если сумма цифр числа, возведенных в N-ную степень (где N – количество цифр в числе) равна самому числу. Например, $153 = 1^3 + 5^3 + 3^3$. Найдите все трёхзначные числа Армстронга.

Задачи

«6»: Простое число – это число, которое делится только само на себя и на 1. Ввести натуральное число N и вывести все простые числа в диапазоне от 2 до N.

Программирование на языке Python

2. Обработка потока данных

Обработка потока данных (подсчёт)

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 (чис неизвестно)

? Когда увеличивать счётчик?

? Какой цикл?

```
счётчик = 0
пока не введён 0:
    если введено число > 0:
        счётчик += 1
```

Обработка потока данных (подсчёт)

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

откуда взять x?

? Что плохо?

Найди ошибку!

```
k = 0
```

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

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

Найди ошибку!

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

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

Задачи

«3»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить, сколько было введено положительных и сколько отрицательных чисел.

Пример:

```
5
3
-1
0
Положительных:  2
Отрицательных:  1
```

Задачи

«4»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить, сколько было введено двузначных натуральных чисел, и сколько других.

Пример:

```
15
7
13
-12
1
0
Двузначных: 2
Других: 3
```

Задачи

«5»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить, сколько было введено двузначных натуральных чисел, которые оканчиваются на «5», и сколько других.

Пример:

```
15
7
13
-12
0
Двузначные, оканчиваются на 5: 1
Другие: 3
```

Задачи

«6»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить, сколько было введено простых натуральных чисел (которые делятся только сами на себя и на 1), и сколько составных.

Пример:

```
15
7
13
-12
6
0
Простых: 2
Составных: 3
```

Обработка потока данных (сумма)

Задача: с клавиатуры вводятся числа, ввод завершается числом 0. Найти сумму введённых чисел, оканчивающихся на "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 x оканчивается на "5":
      сумма += x
```

Обработка потока данных (сумма)

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

? Что плохо?

"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:

Как иначе?

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

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

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:
    print("Ответ: нет")
else:
    print("Ответ:", s)
```

k = 0

s += x

x = int(input())

Задачи

«3»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить сумму тех введённых чисел, которые делятся на 5.

Пример:

```
5
3
34
15
0
Ответ: 20
```

Задачи

«4»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить сумму тех введённых чисел, которые делятся на 3 и заканчиваются на 1. Вывести "нет", если таких чисел нет.

Пример:

```
5
31
18
21
15
0
Ответ: 21
```

Пример:

```
5
31
18
41
15
0
Ответ: нет
```

Задачи

«5»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить, среднее арифметическое тех введённых двузначных чисел, которые делятся на 5. Вывести "нет", если таких чисел нет.

Пример:

```
5
3
35
185
34
15
0
Ответ: 25
```

Пример:

```
5
3
315
185
34
17
0
Ответ: нет
```

Задачи

«6»: с клавиатуры вводятся числа, ввод завершается числом 0. Определить, среднее арифметическое тех введённых чисел, которые являются степенями числа 2. Вывести "нет", если таких чисел нет.

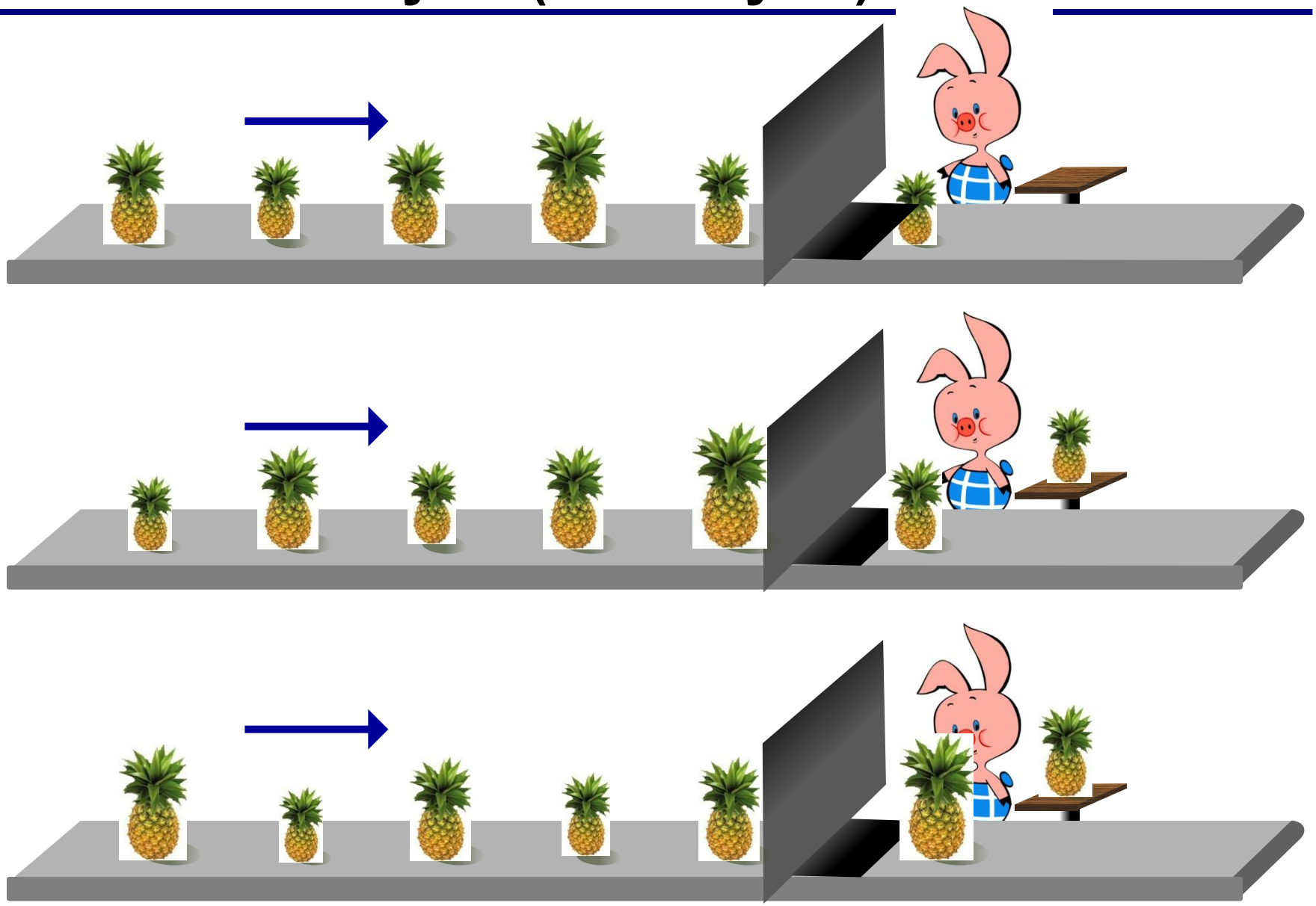
Пример:

```
5
8
185
4
16
0
Ответ: 9.333
```

Пример:

```
5
18
185
44
116
0
Ответ: нет
```

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

первое

новый максимум

Задачи

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

Пример:

```
5
3
34
15
0
Минимум: 5
Максимум: 34
```

Задачи

«4»: 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

«5»: 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

«6»: 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

Программирование на языке Python

3. Обработка потока данных (цикл for)

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)
```

0
1
2
3

Что выведет?

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

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

«3»: 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

«4»: 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

«5»: 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

«6»: 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
198
163
39
Минимум: 5
Максимум: 163
```

Пример:

```
5
12
25
132
6
39
Ответ: нет
```

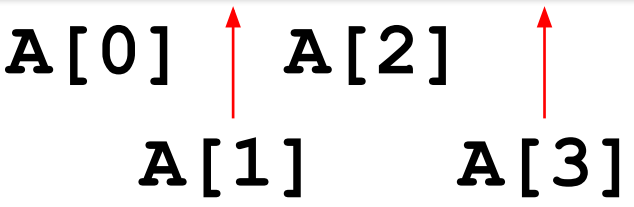
Программирование на языке Python

4. Обработка массивов

Массивы (списки) в Python

Создание массива:

```
A = [1, 5, 0, -1, 12]
```




```
print(A[1])
```

```
5
```

```
print(2*A[0]+A[3])
```

```
1
```

```
A = 5*[0]
```

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

Вывод массива на экран

Как список:

```
print ( A )
```

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

В строчку через пробел:

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

```
1 2 3 4 5
```

пробел после вывода очередного числа

или так:

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

```
1 2 3 4 5
```

или так:

```
print ( *A )
```

↔

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

разбить список на элементы

Заполнение случайными числами

```
from random import randint
A = []
for i in range(5):
  A.append(randint(1,6))
print(A)
```

наращиваем с каждым шагом

? В чём отличие?

Или так:

```
from random import randint
A = 5*[0]
for i in range(5):
  A[i] = randint(1,6)
print(A)
```

сначала выделили память, потом меняем

Подсчёт элементов

```
A = [1, 2, 3, 4, 5, 6, 7]
k = 0
for i in range(7):
   if A[i] > 3: k += 1
print(k)
```

? Что выведет?

4

Кумир:

```
k := 0
нц для i от 1 до 7
   если A[i] > 3 то
      k := k + 1
   все
кц
вывод k
```

Паскаль:

```
k := 0;
for i:=1 to 7 do
   if A[i] > 3 then
      k = k + 1;
writeln(k);
```

! Элементы массива нумеруются с 1!

Подсчёт элементов

```
A = [1, 21, 3, 46, 53, 6, 17]
k = 0
for i in range(7):
    if A[i] % 3 == 0: k += 1
print(k)
```

Что выведет?

3

Варианты условий:

```
    if A[i] % 10 == 6: k += 1
```

2

```
    if(A[i] % 10 == 6 and
       A[i] % 3 == 0): k += 1
```

1

```
    if(A[i] >= 10 and
       A[i] < 100): k += 1
```

4

Суммирование элементов

```
A = [1, 21, 3, 46, 53, 6, 117]
s = 0
for i in range(7):
    if A[i] % 3 == 0: s += A[i]
print(s)
```

Что выведет?

30

Варианты условий:

```
    if A[i] % 10 == 6: s += A[i]
```

52

```
    if(A[i] % 10 == 6 and
       A[i] % 3 == 0): s += A[i]
```

6

```
    if(A[i] >= 10 and
       A[i] < 100): s += A[i]
```

120

Задачи

«3»: Напишите программу, которая находит в массиве количество элементов, делящихся на 5.

«4»: Напишите программу, которая находит среднее арифметическое всех элементов массива, которые делятся на 3 и заканчиваются на 1.

«5»: Напишите программу, которая находит среднее арифметическое всех элементов массива, двоичная запись которых содержит ровно 4 цифры.

«6»: Напишите программу, которая находит элемент массива, двоичная запись которого содержит больше всего единиц.

Максимум

```
A = [1, 21, 3, 46, 53, 6, 117]
m = 0
for i in range(7):
    if A[i] > m: m = A[i]
print(m)
```

меньше всех

? Что выведет?

117

Кумир:

```
m := 0
нц для i от 1 до 7
  если A[i] > m то
    m := A[i]
  все
кц
вывод m
```

Паскаль:

```
m := 0;
for i:=1 to 7 do
  if A[i] > m then
    m = A[i];
writeln(m);
```

Минимум

```
A = [1, 21, 3, 46, 53, 6, 117]
m = 999
for i in range(7):
    if A[i] < m: m = A[i]
print(m)
```

больше всех

? Что выведет?

1

Кумир:

```
m := 999
нц для i от 1 до 7
  если A[i] < m то
    m := A[i]
  все
кц
вывод m
```

Паскаль:

```
m := 999;
for i:=1 to 7 do
  if A[i] < m then
    m = A[i];
writeln(m);
```

Если значения в массиве неизвестны...

```
A = [...как-то получили...]
N = len(A) # длина массива
m = A[0]
for i in range(N):
    if A[i] < m: m = A[i]
print(m)
```

? Что записать в m?

? Как сэкономить один шаг цикла?

```
for i in range (1,N):
    ...
```

пропустить A[0]

Python: `m = min(A)`

Задачи

«3»: Напишите программу, которая находит минимальный и максимальный из чётных элементов массива. Гарантируется, что все элементы массива находятся в диапазоне [-100;100] и среди них есть хотя бы один чётный элемент.

«4»: Напишите программу, которая находит минимальный и максимальный из элементов массива, заканчивающихся на "**5**". Если в массиве нет таких элементов, нужно вывести слово "нет".

Задачи

«5»: Напишите программу, которая находит минимальный из чётных элементов массива и его номер. Если в массиве нет таких элементов, нужно вывести слово "нет".

Пример:

```
Массив: [1, 12, 3, 4, 5, 18, 24]
Минимум: A[3] = 4
```

Пример:

```
Массив: [1, 13, 3, 19, 5, 71, 241]
Минимум: нет
```

Сортировка

Сортировка – это расстановка элементов массива в заданном порядке (возрастания, убывания, …).

Было:

```
9 6 2 7 3 1 5 4 8 0
```

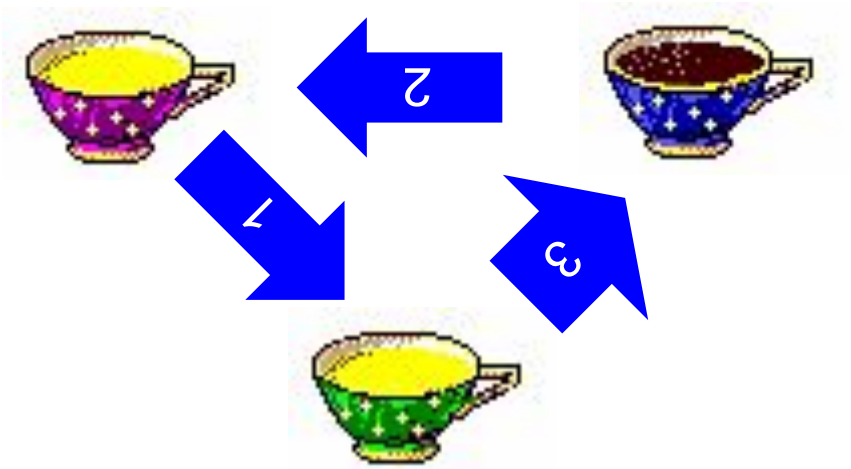
Стало:

```
0 1 2 3 4 5 6 7 8 9
```

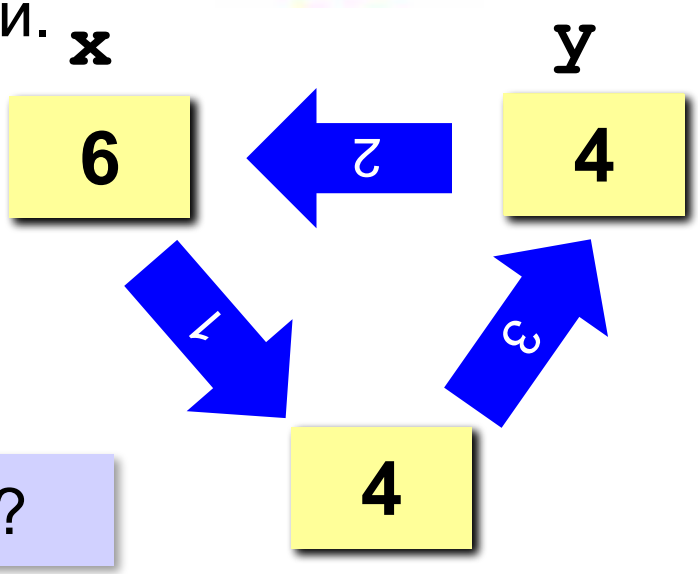
Основная операция – перестановка элементов!

Перестановка элементов

Задача: поменять местами содержимое двух чашек.



Задача: поменять местами содержимое двух ячеек памяти.




```
x = y
y = x
```

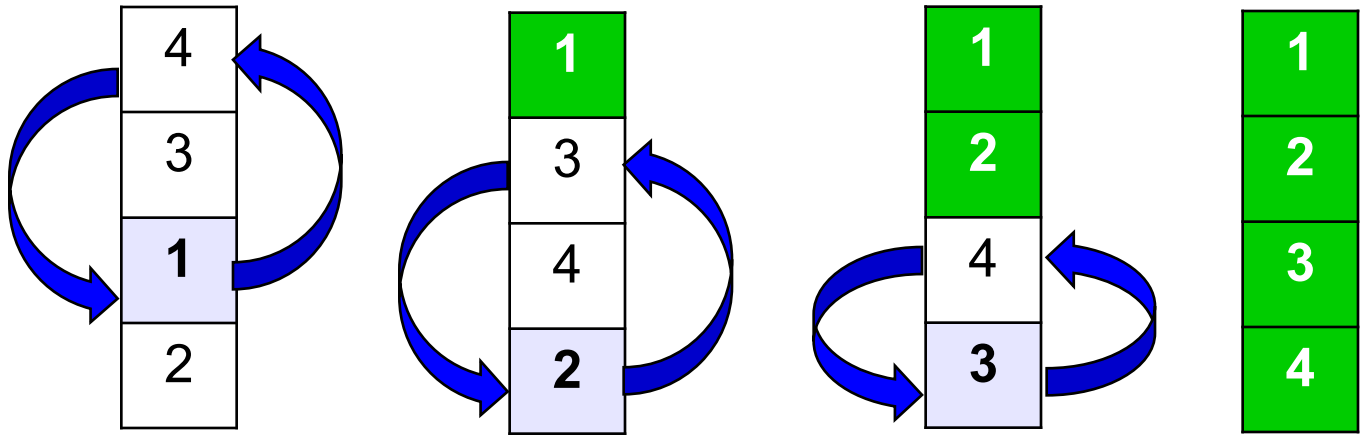
```
c = x
x = y
y = c
```

? Можно ли обойтись без **c**?

Python:

```
x, y = y, x
```

Метод выбора (минимального элемента)



Идея:

- найти минимальный элемент и поставить на первое место (поменять местами с **`A[0]`**)
- из оставшихся найти минимальный элемент и поставить на второе место (поменять местами с **`A[1]`**), и т.д.

`N-1`

? Сколько раз сделать цикл?

Как найти номер минимального элемента?

```
A = [1, 21, 3, -46, 53, -6, 117]
N = len(A) # длина массива
m = A[0]   # считаем A[0] = min
nM = 0     # номер минимального
for i in range(N):
    if A[i] < m:
        m = A[i]
        nM = i  # новый номер
print(nM)
```

? Нельзя ли обойтись без переменной m?

```
m = A[nM]
```

Как найти номер минимального элемента?

```
A = [1, 21, 3, -46, 53, -6, 117]
N = len(A) # длина массива
nM = 0     # номер минимального
for i in range(N):
    if A[i] < A[nM]:
        nM = i  # новый номер
print(nM)
```

часть алгоритма сортировки

Python:

```
m = min(A)
nM = A.index(m)
```

Сортировка выбором

```
A = [1, 21, 3, -46, 53, -6, 117]
N = len(A) # длина массива
for k in range(N-1):
    nM = k
    for i in range (k,N):
        if A[i] < A[nM]:
            nM = i
    A[k],A[nM] = A[nM],A[k]
print(A)
```

поиск минимального

перестановка

? Почему эта программа не работает?

искать минимальный, начиная с номера **k**!

Python:

```
A.sort()
```

Задания

«3»: Заполнить массив из 10 элементов случайными числами в интервале [0..99] и отсортировать его по убыванию последней цифры.

Пример:

```
Исходный массив:
14  25  13  12  76  58  21  87  10  98
Результат:
98  58  87  76  25  14  13  12  21  10
```

Задания

«4»: Заполнить массив из 10 элементов случайными числами в интервале [0..99] и отсортировать его по возрастанию суммы цифр (*подсказка: их всего две*).

Пример:

```
Исходный массив:
14  25  13  12  76  58  21  87  10  98
Результат:
10  21  12  13  14  25  76  58  87  98
```

Задания

«5»: Заполнить массив из 10 элементов случайными числами в интервале [0..100] и отсортировать первую половину по возрастанию, а вторую – по убыванию.

Пример:

```
Исходный массив:
14  25  13  30  76 | 58  32  11  41  97
Результат:
13  14  25  30  76 | 97  58  41  32  11
```

Программирование (Python)

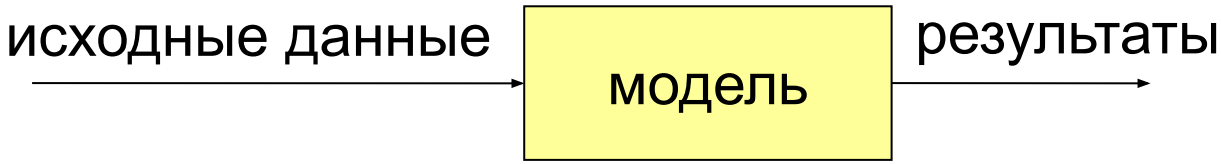
Как разрабатывают программы?

Этапы разработки программ

I. Постановка задачи

Документ: *техническое задание*.

II. Построение модели



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

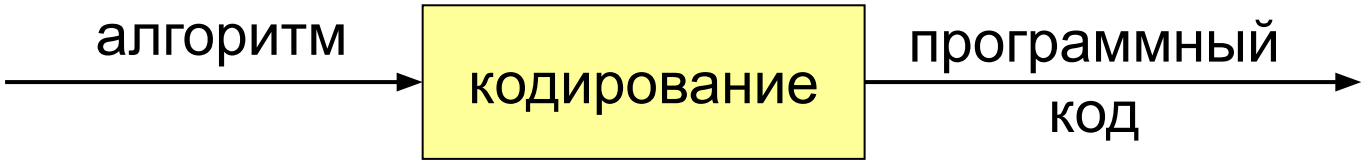
III. Разработка алгоритма и способа хранения данных

«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

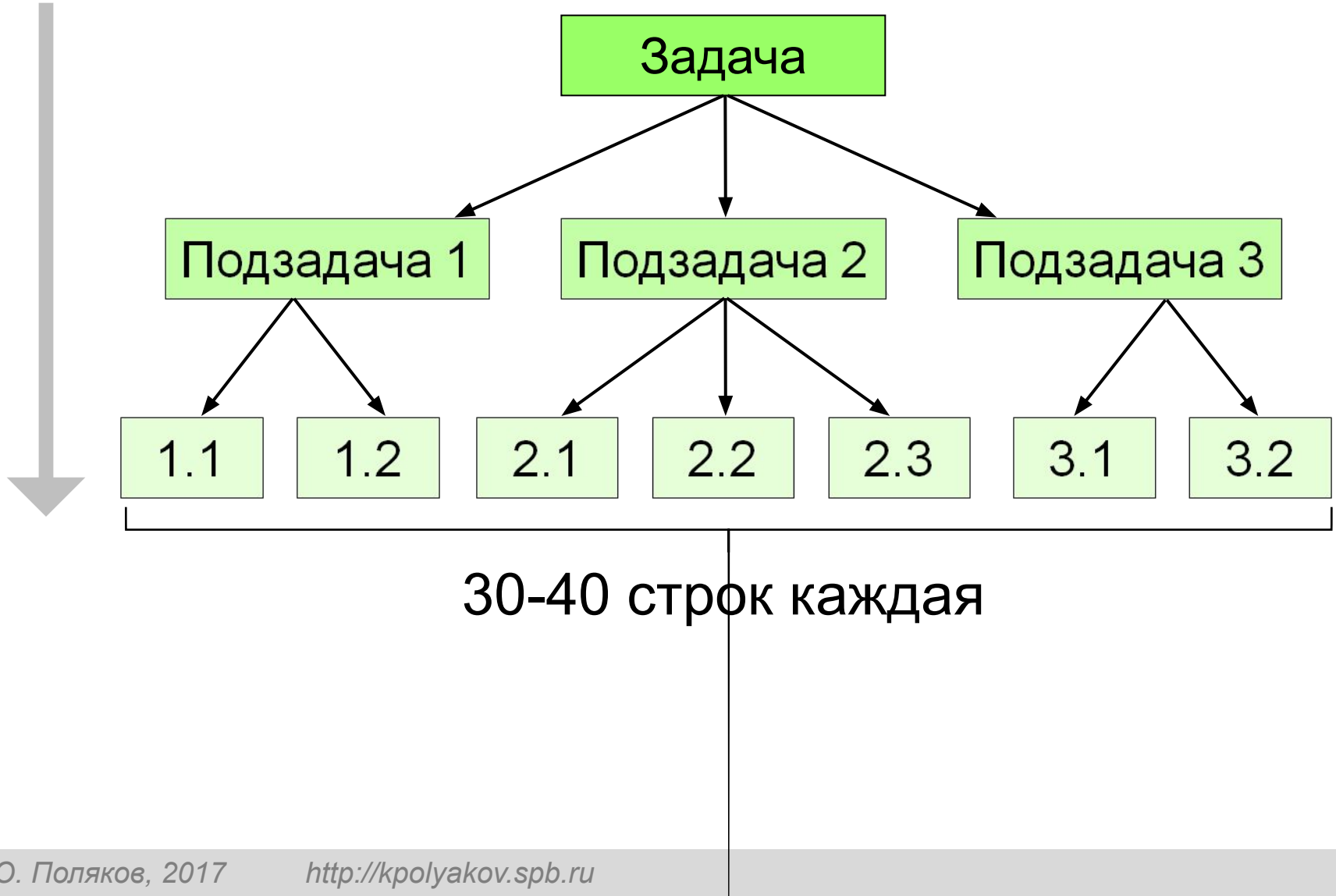
Технические писатели

VIII. Внедрение и сопровождение

- обучение пользователей
- исправление найденных ошибок
- техподдержка

Методы проектирования программ

«Сверху вниз» (последовательное уточнение)



Методы проектирования программ

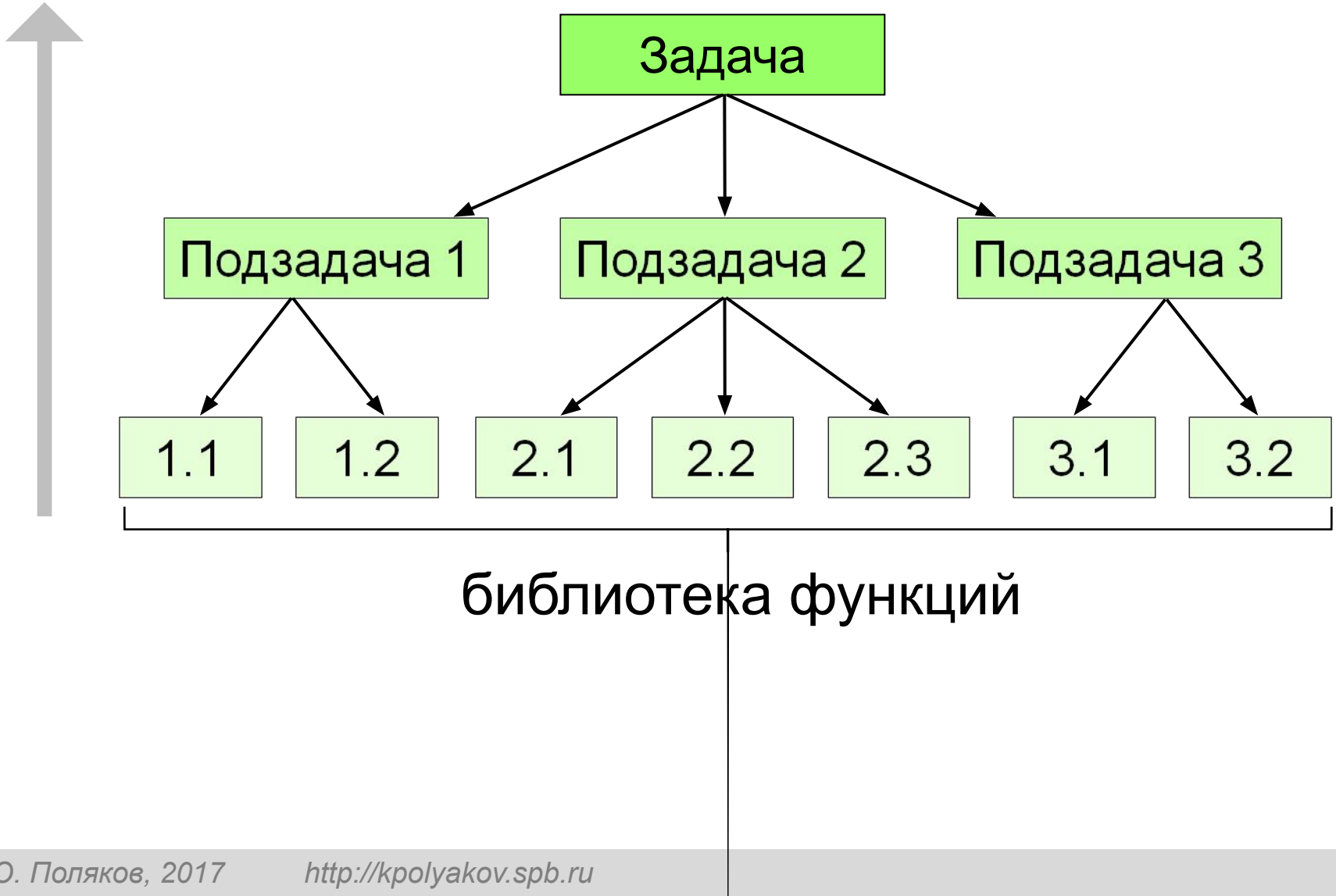
«Сверху вниз» (последовательное уточнение)

- 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 быстродействие

- 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=4.791288 x2=0.208712
```

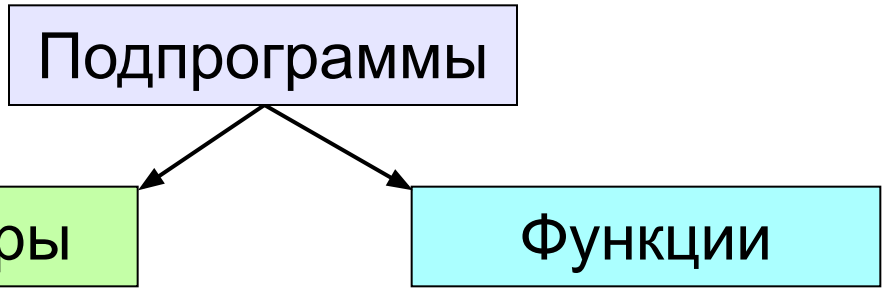
2. Решение уравнения $x^2 + x + 6 = 0$

```
Введите a, b, c: 1 1 6
Нет.
```

Программирование (Python)

Процедуры

Два типа подпрограмм



выполняют действия

+ возвращают некоторый результат

? 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");
}
```

Задачи

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

Пример:

```
Длина цепочки: 7
-------
-------
```

«4»: 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

«5»: 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

«6»: 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
```

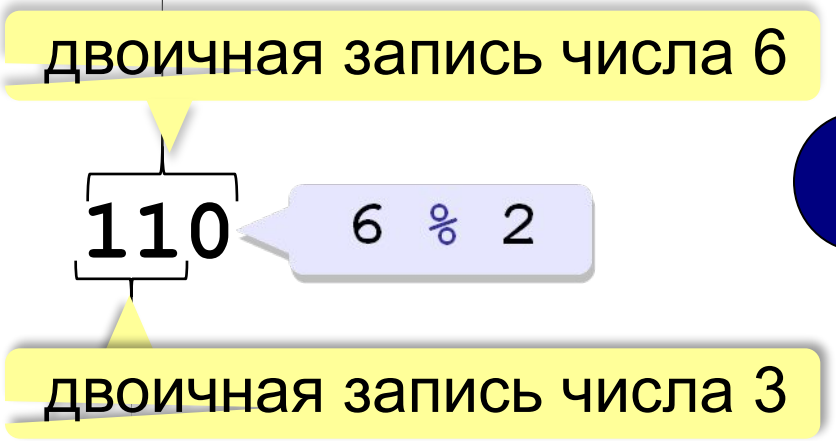
Что получится при **n = 6**?

011 ✗

в обратном порядке!

Рекурсия

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



! 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.

? Что получится? `printBin(6)`

```
printBin(6)
    printBin(3)
        printBin(1)
            printBin(0)
                printBin(0)
```

бесконечные вызовы

? Как исправить?

Рекурсивная процедура

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

? 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

«C»: 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
```

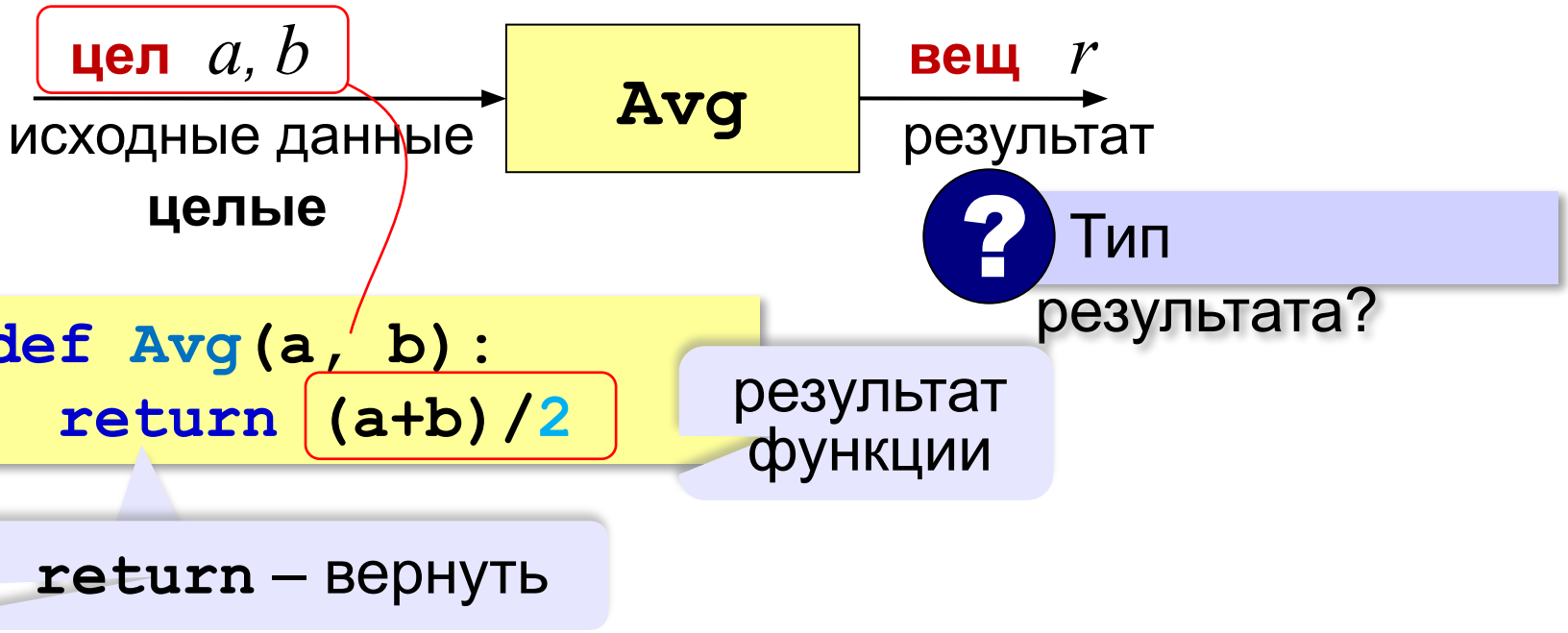
Программирование (Python)

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)
print( Avg(12,7) )
print( sr + Avg(x,12) )
```

5

9.5

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.
```

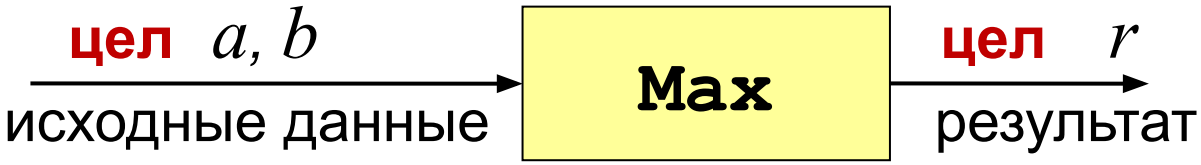
специальная переменная для записи результата функции

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
```

Вызов процедуры:

```
m = int(input())
s = sumDigits( m )
print( s )
```

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 нулей в двоичной записи числа.

Пример:

```
Введите число: 75
Количество нулей: 3
```

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

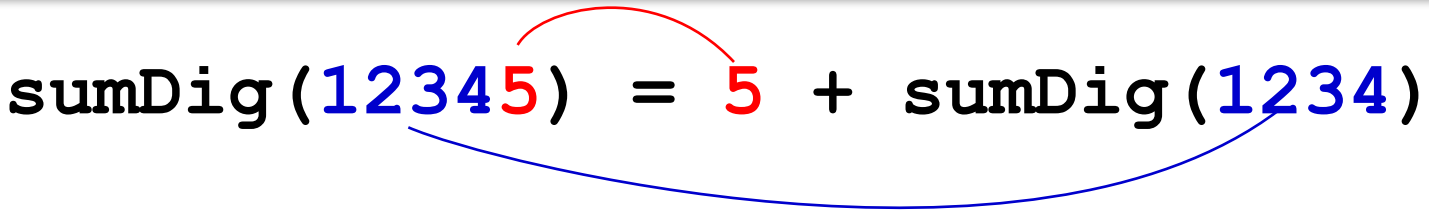
Рекурсивная функция — это функция, которая вызывает сама себя.

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.



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

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

Пример:

```
Введите число: 1230
Ответ: Да
```

«B»: 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