# Обзор языка Python. Часть 1

## Самая простая программа. Печатаем текст

Напечатаем текст "Hello"

Чтобы программа могла работать не нужно ничего лишнего. Сразу пишем команду печати **print**

print('Hello')

DEA! Строку можно писать:

* 'в одинарных кавычках'
* "в двойных кавычках"
* '''текст на несколько строк  
  можно написать в тройных кавычках'''

## Вычисления

Для вычислений нужны числа. Числа помещают в память компьютера. Каждое число помещают в свое место. Каждое место имеет имя и тип числа. Могут быть целые числа и дробные числа.

Посчитаем значение выражения 2+3 и напечатаем результат

Можно сделать так:

print(2+3)

А можно использовать переменную

x = 2+3 # создали переменную х и положили в нее 5. Тип х сейчас int

print(x) # печатаем значение, которое хранит х.

### Арифметические операторы:

print(11+2) # 13

print(11-2) # 9

print(11/2) # 5.5 деление

print(11//2) # 5 целочисленное деление

print(11%2) # 1 остаток от деления

## Типы переменных

Каждая переменная знает, какой тип данных сейчас в ней хранится.

У переменной нет типа. Тип есть только у данных, которые она хранит.

Функция **type()** возвращает тип данных.

x = 5

print(x, type(x)) # 5 int

x = 3.14

print(x, type(x)) # 3.14 float

x = "Hello"

print(x, type(x)) # Hello string

## Изменить тип

Изменить тип данных можно функциями **int()**, **float()**, **str()**

x = "3.14"

print(x, type(x)) # 3.14 string

x = float(x)

print(x, type(x)) # 3.14 float

x = int(x)

print(x, type(x)) # 3 int

x = "Hello"

print(x, type(x)) # Hello string

x = int("Hello") # НЕЛЬЗЯ!

### Печатаем значения разных типов

Если нужно напечатать значение нескольких выражений, напишите их через запятую

x = 3

y = 5.5

z = "Hello"

print(x, y, z) # 3 5.5 Hello

Если хотим печатать красиво, надо писать формат. Формат зависит от типа данных.

x = 3

y = 5.5

z = "Hello"

print("первое число %d, второе %f, потом текст %s" % (x, y, z))

# первое число 3, второе 5.5, потом текст Hello

DEA! print(**"%.2f"** % (3.1415) ) напечатает число 3.1415 как 3.14 (2 знака после .)

DEA! print(**"%02d:%02d"** % (13, 5) ) напечатает 13 и 5 как 13:05

## Чтение переменных

Функция **input()** читает 1 строку с клавиатуры.

x = input()

Чтобы прочитать с клавиатуры 2 строки, нужно 2 раза вызвать input()

x = input() # первую строку прочитали и записали в переменную х

y = input() # вторую строку прочитали и записали в переменную y

Напишем программу, которая складывает 2 целых числа.

x = input() # 3

y = input() # 5

print(x+y) # 35 ???

Почему 35?

x = input() # 3

y = input() # 5

print(x, type(x)) # 3 string

print(y, type(y)) # 5 string

z = x+y

print(z, type(z)) # 35 string (две строки написали рядом - конкатенация, concatenation)

Потому что input() вернула строку. У нас есть строка "3" и строка "5", а не числа 3 и 5. Строки оператором + соединяются вместе в строку "35". Строки пишутся одна за другой.

Как исправить?

Мы знаем, что числа будут целые. Поэтому сразу изменим тип данных на int.

x = input() # 3

y = input() # 5

x = int(x)

y = int(y)

print(x, type(x)) # 3 int

print(y, type(y)) # 5 int

z = x+y

print(z, type(z)) # 8 int (работаю правила сложения целых чисел)

Можно написать короче. Сразу делаем прочитанные данные int

x = int(input()) # 3, прочитали строку, сделали из строки int

y = int(input() # 5, прочитали строку, сделали из строки int

print(x, type(x)) # 3 int

print(y, type(y)) # 5 int

z = x+y

print(z, type(z)) # 8 int (работаю правила сложения целых чисел)

## Пример 1. Периметр и площадь

Дано: стороны прямоугольника a и b.  
Написать программу, которая по двум сторонам находит (и печатает) периметр и площадь прямоугольника.

# -\*- coding: utf-8 -\*-

# чтобы можно было писать русские буквы и иероглифыa = int(input())

a = int(input())

b = int(input())

per = (a+b)\*2

s = a\*b

print("Периметр = %d" %(per))

print("Площадь = %d" % (s))

## Задача 0. Обед

Дано: рис стоит k рублей, салат стоит n рублей. На обед студент купил 1 порцию риса и 2 порции салата. Напечатать: сколько рублей стоит обед.

Написать в тетради на каких данных будете проверять программу.

|  |  |  |  |
| --- | --- | --- | --- |
| [**Рис**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=0&table=1&up=0#sorted_table) | [**Салат**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=1&table=1&up=0#sorted_table) | [**Стоит обед (ожидаю)**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=2&table=1&up=0#sorted_table) | [**Напечатала программа**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=3&table=1&up=0#sorted_table) |
| 30 | 40 | 110 |  |
|  |  |  |  |

## Задача 1. s = v\*t

Дано: электричка проехала s километров за t часов.   
Написать программу, которая печатает скорость электрички в км/ч.

Написать в тетрадку данные, на которых вы проверяли программу. Например:

|  |  |  |  |
| --- | --- | --- | --- |
| [**s дано**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=0&table=2&up=0#sorted_table) | [**t дано**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=1&table=2&up=0#sorted_table) | [**v - ожидаемый результат**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=2&table=2&up=0#sorted_table) | [**напечатала программа**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=3&table=2&up=0#sorted_table) |
| 72 | 2 | 36 |  |
| 31 | 2 | 15.5 |  |
|  |  |  |  |

## Задача 2. Делаем стулья

Дано: n - количество ножек стула. У 1 стула 4 ножки. Напечатать: сколько стульев можно сделать из этих ножек?

Написать в тетрадку данные, на которых вы проверяли программу.

## Задача 3. Ставим стулья

Площадь комнаты - s (дробное число). Длина одной стороны ln (дробное число). Ширина стула ch (дробное число). Вычислить сколько стульев res (целое число) можно поставить у **ДРУГОЙ** стороны комнаты. Сделать таблицу для проверки результатов.

## Задача 4. Когда закончится урок?

Уроки в школе начинаются в 8:00. Каждый урок длится 40 минут. Перемена 15 минут.

Будет К уроков.

Во сколько закончится К уроков?

## Задача 5. Автомобиль и велосипедист

Между городами A и В **s** км. Из города A выехал автомобиль со скоростью **avto** км/час, Одновременно с ним из города B по той же дороге навстречу выехал велосипедист со скоростью **velo** м/сек.

Написать программу, чтобы вычислить на каком километре от города A они встретятся.

## Функции

Вы уже пользовались функциями языка python. Это print(), input(), int(), float().

Можно написать функцию самим.

### Простая функция

Напишем функцию, у которой нет аргументов и которая ничего не возвращает. Придумаем имя функции hi. Функция печатает hello.

# делаем функцию.

# def - ключевое слово

# hi - придумали (сами) имя функции

def hi():

print("hello") # код функции пишем с отступами

# закончились отступы - закончилась функция.

hi() # вызов функции hi, функция печатает hello

hi() # вызов функции hi, функция печатает hello

DEA! Не забывайте : после )

### Передаем в функцию числа

Напишем вычисление периметра и площади прямоугольника через функции. Тогда можно будет просто посчитать периметр и площадь разных прямоугольников в одной программе.

У функции могут быть аргументы. Для вычисления периметра и площади прямоугольника нужно передать функции стороны прямоугольника.

DEA! Один раз создали функцию. Много раз можем использовать функцию.

# -\*- coding: utf-8 -\*-

# чтобы можно было писать русские буквы и иероглифыa = int(input())

def perimetr(a, b): # создали первую функцию perimetr, в нее передают два числа a и b

res = (a+b)\*2

return res # возвращает число

# первая функция закончилась

def area(a, b): # создали другую функцию area, в нее передают два числа a и b

res = a\*b

return res

p = perimetr(3,5) # результат функции perimetr поместили в переменную p

print("Периметр = %d" % (p)) # напечатали p (Периметр = 16)

s = area(3,5) # результат функции area поместили в переменную s

print("Площадь = %d" % (s)) # напечатали s (Площадь = 15)

# можно сразу печатать результат функции

print("Периметр = %d" % (perimetr(3,5)))

print("Площадь = %d" % (area(3,5)))

p = perimetr(3.3, 5) # функция может считать и дробные числа

print("Периметр = %f" % (p)) # напечатали p по формату %f (Периметр = 16.6)

s = area(3.3, 5) # результат функции area поместили в переменную s

print("Площадь = %f" % (s)) # напечатали s по формату %f (Площадь = 16.5)

### Возвращаем несколько значений

Функция может возвращать несколько значений. Их пишут через запятую (,)

Функции height передаем рост в сантиметрах, а возвращает функция рост в метрах и сантиметрах

def height(h): # функция height, в нее передают одно число h

m = h // 100 # подсчитали рост в метрах

sm = h % 100 # подсчитали рост в сантиметрах

return m, sm # вернули сразу метры и сантиметры

# дальше программа. Пользуемся функцией height и проверяем ее.

# мой рост 169 см. Посчитаем его в метрах и сантиметрах

mym, mysm = height(169) # результаты функции поместили в переменные mym и mysm

print("мой рост %d метров %d сантиметров" % (mym, mysm))

you = int(input()) # прочитали ваш рост

ym, ysm = height(you) # результаты функции поместили в переменные ym и ysm

print("ваш рост %d метров %d сантиметров" % (ym, ysm))

### Функция вызывает функцию

Напишем программу, которая по координатам 2 точек на плоскости считает расстояние между ними.

# -\*- coding: utf-8 -\*-

from math import sqrt

def length(x1, y1, x2, y2): # создали функцию length

dx = x1 - x2

dy = y1 - y2

res = sqrt(dx\*dx + dy\*dy)

return res

x1, y1, x2, y2 = map(int, input().split()) # прочитали сразу много чисел из 1 строки

dist = length(x1, y1, x2, y2) # результат работы функции length записали в dist

print(dist)

Функция length(x1, y1, x2, y2) считает расстояние между 2 точками на плоскости.

Теперь напишем другую программу. Которая по координатам 3 точек на плоскости считает площадь треугольника по формуле Герона.

Нужно писать мало кода. Возьмем старую функцию length и используем ее.

# -\*- coding: utf-8 -\*-

from math import sqrt

def length(x1, y1, x2, y2): # функция length уже написана и проверена

dx = x1 - x2

dy = y1 - y2

res = sqrt(dx\*dx + dy\*dy) # из функции length вызываем функцию sqrt

return res

def area3(x1, y1, x2, y2, x3, y3): # новая функция area3

a = length(x1, y1, x2, y2) # из функции area3 вызываем функцию length

b = length(x1, y1, x3, y3) # из функции area3 вызываем функцию length

c = length(x3, y3, x2, y2) # из функции area3 вызываем функцию length

p = (a+b+c)/2 # записываем формулы

res = sqrt(p\*(p-a)\*(p-b)\*(p-c))

return res

x1, y1, x2, y2, x3, y3 = map(int, input().split())

s = area3(x1, y1, x2, y2, x3, y3)

print(s)

### Не обязательные аргументы

Задача: написать функцию, которая считает расстояние до точки (x,y) на плоскости от начала координат (0,0)

Вариант 1. Самый плохой, потому что нужно писать много кода и отлаживать его. Можем ошибиться при написании формулы.

def length0(x, y): # создали функцию length0

res = sqrt(x\*x + y\*y)

return res

Вариант 2. Лучше. Пишем еще одну функцию, которая использует функцию length

def length0(x, y): # создали функцию length0

return length(x, y, 0, 0)

Вариант 3. Хорошо. Не нужно писать новый код.

Когда пишем функцию length(x1, y1, x2, y2) записываем в x2 и y2 значения по умолчанию 0. Аргументы x2 и y2 стали не обязательными. Можно вызвать функцию без этих аргументов, а она будет работать так, будто их значение 0.

DEA! Значение по умолчанию можно сделать любое. Не обязательно 0.

def length(x1, y1, x2=0, y2=0): # создали функцию length

dx = x1 - x2

dy = y1 - y2

res = sqrt(dx\*dx + dy\*dy)

return res

Когда вызываем функцию length, можем передавать все параметры, а можем не передавать x2 и y2. Тогда их значение будет по умолчанию 0.

d = length(3, 4, 3, -4) # расстояние между точками (3, 4) и (3, -4)

d = length(3, 4, 3) # расстояние между точками (3, 4) и (3, 0)

d = length(3, 4) # расстояние между точками (3, 4) и (0, 0)

### Именованные аргументы

Мы передавали аргументы в функцию по их позиции.

d = length(3, 4, 3, -4) # расстояние между точками (3, 4) и (3, -4)

И понимали, что -4 - это значение аргумента y2.

Можно передавать аргументы в функцию по имени аргумента.

d = length(3, 4, x2=5, y2=-4) # расстояние между точками (3, 4) и (5, -4)

d = length(3, 4, y2=5, x2=-4) # расстояние между точками (3, 4) и (-4, 5)

# порядок вызова аргументов по имени НЕ важен

d = length(x1=3, x2=4, y1=5, y2=-4) # расстояние между точками (3, 5) и (4, -4)

# это тоже работает, потому что аргументы вызваны по имени и порядок не важен

d = length(3, y1=4, x2=5, y2=-4) # расстояние между точками (3, 4) и (5, -4)

# любой аргумент можно вызвать по имени

d = length(x1=3, 4, x2=3, y2=-4) # ОШИБКА! сначала аргумент по имени, потом - нет.

DEA! Если вызван аргумент по имени, все аргументы после него должны вызваться по имени

### Проверить функцию

Напишем функцию вычисления периметра и проверим, что она правильная.

# -\*- coding: utf-8 -\*-

def perimetr(a, b): # создали первую функцию perimetr, в нее передают два числа a и b

res = (a+b) # Ошибка! Забыли \*2

return res # возвращает число

# Проверим функцию perimetr

print(perimetr(3,5)) # периметр должен быть равен 16

Программа напечатает 8. Мы посмотрим на perimetr(3,5) и посчитаем, что периметр должен равняться 16.

Нашли, что функция perimetr работает неправильно. Надо исправить.

И проверить еще раз.

Легче проверять, если печатать что посчитали и какое число должно быть.

print(perimetr(3,5), 16) # должно напечатать 16 и 16

print(perimetr(7,2), 18) # должно напечатать 18 и 18

print(perimetr(5,5), 25) # должно напечатать 25 и 25

Печатаются числа. Надо посмотреть, что числа одинаковые. Если числа разные - ошибка.

Можно заставить проверять компьютер. **assert**(выражение) - проверяет, правильное выражение или нет. Если правильное, то ничего не делает. Если неправильное, печатает где ошибка.

assert(perimetr(3,5)==16) # проверить, что perimetr(3,5) вернул 16

assert(perimetr(7,2)==18) # проверить, что perimetr(7,2) вернул 18

assert(perimetr(5,5)==25) # проверить, что perimetr(5,5) вернул 25

Если функция возвращает несколько значений, то их пишем в ( ) через ,

Функция msm из роста в сантиметрах (157) вычисляет рост в метрах (1) и сантиметрах (57). Возвращает метры и сантиметры (1, 57)

Напишем функцию и проверим ее.

def msm(h):

m = h//100

sm = h % 100

return m, sm

print(msm(157), 1, 57) # напечатает (1 57) 1 57 - можно проверить глазами

assert(msm(157))==(1, 57)) # программа сама проверит, что msm(157) вернет 1 и 57

## Задача 4a. time2min(h,m)

На часах h часов m минут. Напишите функцию time2min(h,m), которая переводит часы и минуты в минуты с 0:00.

Проверьте функцию.

Cделать таблицу для проверки результатов.

## Задача 4b. h, m = min2time(m)

Написать фукнцию h, m = min2time(m), которая из минут с начала суток (0:00) делает часы и минуты.

Проверьте функцию.

Cделать таблицу для проверки результатов.

## Задача 4с. Приехал поезд

Поезд вышел в h1 часов m1 минут. Поезд ехал h2 часа m2 минут. Во сколько часов h и минут m прибыл поезд.

**Использовать написанные функции time2min и min2time**

Cделать таблицу для проверки результатов.

## Задача 5. Время в пути

Поезд вышел в h1 часов m1 минут. Поезд приехал в h2 часа m2 минут. Сколько часов h и минут m ехал поезд, если ехал он не больше 24 часов?

**Использовать написанные функции time2min и min2time**

Cделать таблицу для проверки результатов.

## if, else, elif

if условие :

команды\_ДА

if условие :

команды\_ДА

else :

команды\_НЕТ

if условие1 :

команды1\_ДА

elif условие2 :

команды2\_ДА

elif условие3 :

команды3\_ДА

...

else :

команды\_НЕТ

### Пример if. Сколько лодок нужно?

На берегу стоит n человек. В лодку помещается k человек. Написать программу, которая печатает сколько лодок нужно для перевозки всех людей.

Cделать таблицу для проверки результатов.

n = int(input()))

k = int(input()))

boat = n // k # полных лодок

if n%k > 0 : # остались люди

boat = boat + 1 # нужна еще 1 неполная лодка

print(boat)

### Пример if else - четное, нечетное

Дано число. Напечатать четное оно или нечетное.

x = int(input())

if x%2 == 0 :

print("четное")

else :

print("нечетное")

### Пример if elif else - положительное, отрицательное, ноль

Дано число. Напечатать, оно < 0, > 0 или ноль.

x = int(input())

if x == 0 :

print("0")

elif x > 0:

print("< 0")

else :

print("> 0")

## Задача 6. min

Даны 2 числа. Напечатать меньшее из них.

Cделать таблицу для проверки результатов.

## Задача 7. бег

В соревновании по бегу разные люди бегут разные дистанции.  
Дети до 10 лет бегут 1 км.   
Подростки до 16 лет бегут 3 км.  
Старики старше 50 лет бегут 2 км.  
Все остальные бегут 5 км.

Дан возраст человека. Напечатать сколько километров он должен бежать.

Cделать таблицу для проверки результатов.

|  |  |  |
| --- | --- | --- |
| [**Возраст**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=0&table=3&up=0#sorted_table) | [**км (ожидаю)**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=1&table=3&up=0#sorted_table) | [**напечатало**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=2&table=3&up=0#sorted_table) |
| 7 | 1 |  |
| 10 | 3 |  |
| 17 | 5 |  |
| 15 | 5 |  |
| 50 | 5 |  |
| 70 | 2 |  |
|  |  |  |

## Логические операторы and, or, not

### and

Проверить, что х принадлежит отрезку [5, 20].

if 5 <= x and x <= 20 :

print("5 <= x <= 20")

|  |  |  |
| --- | --- | --- |
| [**and**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=0&table=4&up=0#sorted_table) | [**Да**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=1&table=4&up=0#sorted_table) | [**Нет**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=2&table=4&up=0#sorted_table) |
| **Да** | ДА | нет |
| **Нет** | нет | нет |

Сразу, одновременно.  
Чтобы выпить чай, мне нужно сразу и вода, и пакетик чая.

DEA! Пересечение множеств. Пересечение признаков.

### or

Проверить, что х принадлежит (-∞, 5) U (20; ∞).

if x < 5 or 20 < x :

print("до 5 или после 20")

|  |  |  |
| --- | --- | --- |
| [**or**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=0&table=5&up=0#sorted_table) | [**Да**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=1&table=5&up=0#sorted_table) | [**Нет**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=2&table=5&up=0#sorted_table) |
| **Да** | ДА | ДА |
| **Нет** | ДА | нет |

Хоть что-нибудь.  
С чаем я хочу съесть конфету или печенье. Если есть конфета, то ем конфету. Если только печенье, то ем печенье. Если есть конфеты и печенье, то я ем и конфету, и печенье.

### not

Не.

Один и тот же результат

if x < 5 or 20 < x :

print("до 5 или после 20")

if not (5 <= x and x <= 20) :

print("до 5 или после 20")

|  |  |
| --- | --- |
| [**not**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=0&table=6&up=0#sorted_table) |  |
| **Да** | Нет |
| **Нет** | ДА |

### Type boolean, True, False

Логические выражения, например x < 10, имеют тип **boolean**.

Тип boolean имеет два значения. **True** (да) и **False** (нет).

x = 5

print(x, type(x)) # 5 int

print(x<10, type(x<10)) # True boolean

## Задача 8. Делится на 3 или 5, но не на 15

Дано число, напечатать YES, если оно делится на 3 или 5, но не на 15. Иначе напечатайте NO.

Cделать таблицу для проверки результатов.

## Цикл while

while условие:

команды

### Пример while. Сколько пробежал за 4 дня?

Человек бегает каждый день.

В первый день он пробежал L метров. Каждый день он пробегает на k метров больше, чем в предыдущий. Сколько метров человек пробежал за 4 дня?

Сделаем таблицу в какой день day, сколько человек пробегал в этот день sday, и сколько он пробежал всего total.

# -\*- coding: utf-8 -\*-

L = int(input()) # пробежал в первый день

k = int(input()) # завтра пробежит на k метров больше

# Готовимся бегать

day = 0 # всего пробежал 0 дней

total = 0 # еще не бегал

sday = L # побежит в первый день L метров

# побежали

while day < 4: # пока прошло меньше 4 дней

total = total + sday # к общему результату добавь сколько бегал сегодня

day = day + 1 # день закончился

print("за day=%d дней пробежал total = %d" % (day, total))

# готовимся к завтрашнему дню

sday = sday + k # завтра пробежит на k больше

print("sday = %d" % (sday))

# цикл закончился

print(total) # после цикла (1 раз) напечатать сколько всего пробежал

## Задача 9. Сколько дней бегал?

Человек бегает каждый день.

В первый день он пробежал L метров. Каждый день он пробегает на k метров больше, чем в предыдущий. Напечатать, за сколько дней он пробежит total метров.

## Задача 10. Сколько пробежал?

Человек бегает каждый день. Он заболел. И каждый день пробегает на k метров **меньше**. Если в день он сможет пробежать <0, то он не бежит (то есть бежит 0 метров).

В первый день он пробежал L метров. Сколько дней будет бегать человек? Сколько за эти дни пробежит?

## Задача 11. Сколько пробежал?

Человек бегает каждый день. Он заболел. И каждый день пробегает на k метров **меньше**.

В первый день он пробежал L метров. Сколько метров он пробежит за n дней?.

## Циклы на примере подсчета яблок

## Шаг 1. Количество яблок в ряду

В ряд растет N яблонь. На каждой яблоне ni яблок.   
Сколько всего яблок на всех яблоках?   
Формат входных данных: сначала число N, потом ряд чисел через пробел ni.   
Пример: 3 10 2 5

## Шаг 2. Яблок в нескольких рядах

Изменим постановку задачи: Яблони растут рядами. Дано количество рядов яблонь n. Далее в начале строки пишется сколько яблонь в ряду и количество яблок на каждой яблоне через пробел. Сколько яблок в саду?

Пример входных данных:

3

2 15 24

3 7 82 15

1 54

Нужно просуммировать числа (15+ 24) + (7 82 15) + (54). Заметим, что код, считающий сколько яблок в одном ряду у нас уже есть.   
Нужно выполнить его столько раз, сколько будет заявлено рядов.

### Шаг 3. Есть ворона в ряду

В ряд растет N яблонь. На каждой яблоне ni яблок.  
В саду могут быть вороны, которые отнимают яблоки (с яблони собирается отрицательное число яблок).  
Напечатать один раз YES если ворона в саду. Иначе ничего не печатать.

Несколько рядов чисел (ni) через пробел.

* Пример1: 10 -2 5
* Пример2: 10 -2 5 1 -7

### Пропускаем яблоню с вороной (не считаем яблоки на этой яблоне)

### Пропускаем весь ряд от вороны до конца

### Встретив ворону, убегаем из сада

### break, continue

### break, continue во вложенных циклах

### for .. else

## Дополнительные задачи:

Дан файл [bet.log](http://acm.mipt.ru/twiki/pub/Cintro/PythonListTask/bet.log)

В нем записаны данные работы интернет-магазина.

* секунд с 1 января 1970
* номер клиента (1, 2, 3, ....)
* номер чека

Найдите:

* Сколько всего покупок было сделано в магазине?
* сколько времени в часах, минутах и секундах работал магазин.
* время между первой и последней покупкой клиента N. Номер клиента = номер варианта.

Постройте таблицы данных:

* количество покупок в секунду от начала теста
* количество покупок в минуту от начала теста

Некоторое количество времени не было покупок. В эти времена нужно писать, что количество покупок 0.

# Округление в python

|  |  |
| --- | --- |
| [**Функция**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=0&table=7&up=0#sorted_table) | [**Описание**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=1&table=7&up=0#sorted_table) |
| int(x) | Округляет число в сторону нуля. Это стандартная функция, для ее использования не нужно подключать модуль math. |
| round(x) | Округляет число до ближайшего целого. Если дробная часть числа равна 0.5, то число округляется до ближайшего четного числа. |
| round(x, n) | Округляет число x до n знаков после точки. Это стандартная функция, для ее использования не нужно подключать модуль math. |
| floor(x) | Округляет число вниз («пол»), при этом floor(1.5) = 1, floor(-1.5) = -2 |
| ceil(x) | Округляет число вверх («потолок»), при этом ceil(1.5) = 2, ceil(-1.5) = -1 |

Примеры:

|  |  |
| --- | --- |
| [**Выражение**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=0&table=8&up=0#sorted_table) | [**Результат**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonQuickstart1?sortcol=1&table=8&up=0#sorted_table) |
| int(1.7) | 1 |
| int(-1.7) | -1 |
| ceil(4.2) | 5 |
| ceil(4.8) | 5 |
| ceil(-4.2) | -4 |
| round(1.3) | 1 |
| round(1.7) | 2 |
| round(1.5) | 2 |
| round(2.5) | 2 |
| round(2.65, 1) | 2.6 |
| round(2.75, 1) | 2.8 |

На практике могут быть сюрпризы:

>>> round(2.85, 1)

2.9

Рассмотрим подробнее:

>>> from fractions import Fraction

>>> a = Fraction(2.85)

>>> b = Fraction('2.85')

>>> a == b

False

>>> a > b

True

Что-то не так, правда? На самом деле, всё именно так, как и задумывалось. Просто из-за проблем с точностью чисел с плавающей точкой это число чуть больше, чем 2.85, а потому округляется до 2.9.

# Дополнительные материалы

* [PEP-8](https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html) ([Eng](https://www.python.org/dev/peps/pep-0008/))
* [pylint](https://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd=2&cad=rja&uact=8&ved=0ahUKEwj-m5PU-JjWAhVCJJoKHecbAogQFggtMAE&url=https%3A%2F%2Fgithub.com%2FPyCQA%2Fpylint&usg=AFQjCNFh1fky8oRZyS_rTy_HIv0sZ3j3tA) - python code analizer