Задание 32.2.3

**def** **get\_less**(list\_in, num):

**for** i **in** list\_in:

**if** i < num:

**return** i

**return** None

Задание 32.2.5

**def** **split\_date**(date):

day, month, year = date[:**2**], date[**2**:**4**], date[**4**:]

**return** int(day), int(month), int(year)

Задание 32.3.1

**def** **get\_time**(distance, speed):

**if** distance < **0** **or** speed < **0**:

**raise** **ValueError**("Distance or speed cannot be below 0!")

**if** speed == **0**:

**raise** **ValueError**("Speed cannot be equal to 0!")

result = distance / speed

**return** result

Альтернативный вариант решения:

**def** **get\_time**(distance, speed):

**if** speed == **0**:

**raise** **ValueError**("Speed cannot be equal to 0!")

**if** distance < **0** **or** speed < **0**:

**raise** **ValueError**("Distance or speed cannot be below 0!")

result = distance / speed

**return** result

Задание 32.3.2

# Необходимо добавить только n=2,

# чтобы задать параметру n значение по умолчанию

**def** **root**(value, n=**2**):

result = value \*\* (**1**/n)

**return** result

Задание 32.3.5

**def** **add\_mark**(name, mark, journal=None):

**if** mark **not** **in** [**2**,**3**,**4**,**5**]:

**raise** **ValueError**("Invalid Mark!")

**if** journal **is** None:

journal = {}

journal[name] = mark

**return** journal

Задание 32.4.1

**print**("Shopping list:", end=' ')

# Достаточно добавить параметр sep=', '

**print**("bread", "butter", "eggs", sep=', ')

# Будет напечатано:

# Shopping list: bread, butter, eggs

Задание 32.4.2

**def** **mult**(\*args):

result = **1**

**for** num **in** args:

result \*= num

**return** result

Задание 32.4.3

**print**(\*langs, sep=', ')

# Будет напечатано:

# Python, SQL, Machine Learning, Statistics

Задание 32.4.4

mean\_mark('Kuznetsov', \*marks)

# Должно быть напечатано:

# Kuznetsov: 4.454545454545454

Задание 32.4.7

**def** **print\_lists**(\*lists, \*\*how):

**for** l **in** lists:

**print**(\*l, \*\*how)

Задание 32.5.3

**def** **saver**():

money = **0**

**def** **adder**(val):

nonlocal money

money += val

**return** money

**return** adder

Задание 32.6.3

**def** **fib**(n):

**if** n == **0**: **return** **0**

**if** n == **1**: **return** **1**

**return** fib(n-**1**) + fib(n-**2**)

Задание 32.7.3

**def** **inf\_iter**(l\_in):

**while** True:

**for** elem **in** l\_in:

**yield** elem

Задание 32.8.1

**def** **get\_length**(line):

**return** len(line)

Задание 32.8.3

hyp = **lambda** a, b: (a\*\***2** + b\*\***2**) \*\* (**1**/**2**)

Задание 32.8.4

**def** **sort\_sides**(l\_in):

l\_in.sort(key=**lambda** x: (x[**0**]\*\***2** + x[**1**]\*\***2**) \*\* (**1**/**2**))

**return** l\_in

Задание 32.10.3

**def** **logger**(name):

**def** **decorator**(func):

**def** **decorated\_func**(\*args, \*\*kwargs):

**print**(name + ':', "Function", func.\_\_name\_\_, 'started')

res = func(\*args, \*\*kwargs)

**print**(name + ':', "Function", func.\_\_name\_\_, 'finished')

**return** res

**return** decorated\_func

**return** decorator

Задание 32.11.14

**def** **power**(val, n):

**if** n==**0**: **return** **1**

**if** n==**1**: **return** val

**return** val\*power(val, n-**1**)

Задание 32.11.15

**def** **is\_leap**(year):

**if** year % **400** == **0**: **return** True

**if** year % **100** == **0**: **return** False

**if** year % **4** == **0**: **return** True

**return** False

Задание 32.11.16

**def** **check\_date**(day, month, year):

**def** **is\_leap**(year):

**if** year % **400** == **0**: **return** True

**if** year % **100** == **0**: **return** False

**if** year % **4** == **0**: **return** True

**return** False

#Проверяем день, месяц и год на целочисленность

**if** (type(day) **is** **not** int) **or** (type(month) **is** **not** int) **or** (type(year) **is** **not** int):

**return** False

#Проверяем год на заданный диапазон

**if** (year <= **1900**) **or** (year >= **2022**):

**return** False

#Проверяем месяц на заданный диапазон

**if** (month > **12**) **or** (month < **1**):

**return** False

#Проверяем день на заданный диапазон

**if** (day < **1**) **or** (day > **31**):

**return** False

#Проверяем апрель, июнь, сентябрь и ноябрь на количество дней

**if** (month **in** [**4**,**6**,**9**,**11**]) **and** (day > **30**):

**return** False

#Проверяем количество дней в феврале в зависимости от того, високосный ли год

**if** month == **2**:

**if** is\_leap(year):

**if** day > **29**: **return** False

**else**:

**if** day > **28**: **return** False

**return** True

Задание 32.11.17

**def** **check\_date**(day, month, year):

**def** **is\_leap**(year):

**if** year % **400** == **0**: **return** True

**if** year % **100** == **0**: **return** False

**if** year % **4** == **0**: **return** True

**return** False

**if** type(day) **is** **not** int:

**return** False

**elif** type(month) **is** **not** int:

**return** False

**elif** type(year) **is** **not** int:

**return** False

**if** month > **12** **or** month < **1**:

**return** False

**if** day < **1**: **return** False

**if** day > **31**: **return** False

**if** month **in** [**2**,**4**,**6**,**9**,**11**] **and** day > **30**:

**return** False

**if** month == **2**:

**if** is\_leap(year):

**if** day > **29**: **return** False

**else**:

**if** day > **28**: **return** False

**return** True

**def** **register**(surname, name, date, middle\_name=None, registry=None):

**if** registry **is** None:

registry = list()

**if** **not** check\_date(\*date):

**raise** **ValueError**("Invalid Date!")

registry.append((surname, name, middle\_name, date[**0**], date[**1**], date[**2**]))

**return** registry

Задание 32.11.18

**def** **sort\_registry**(registry):

registry.sort(key=**lambda** x:

(x[-**1**], x[-**2**], x[-**3**], x[**0**], x[**1**], x[**2**]))

**return** registry

Задание 32.11.19

**def** **get\_strings**(registry):

**def** **get\_line**(record):

m, d, y = record[-**3**:]

m = str(m).zfill(**2**)

d = str(d).zfill(**2**)

y = str(y).zfill(**4**)

**if** record[**2**] **is** None:

result = record[**0**]+" "+record[**1**][**0**]+"., "

result += m+'.'+d+'.'+y

**else**:

result = record[**0**]+" "+record[**1**][**0**]+"."+record[**2**][**0**]+"., "

result += m+'.'+d+'.'+y

**return** result

reg = filter(**lambda** x: x[-**1**] >= **2000**, registry)

reg = map(get\_line, reg)

**return** list(reg)

Задание 32.11.20

**def** **group\_gen**(n):

**while** True:

**for** i **in** range(**1**, n+**1**):

**yield** i

Задание 32.11.21

**def** **print\_students**(students, groups):

**def** **group\_gen**(n):

**while** True:

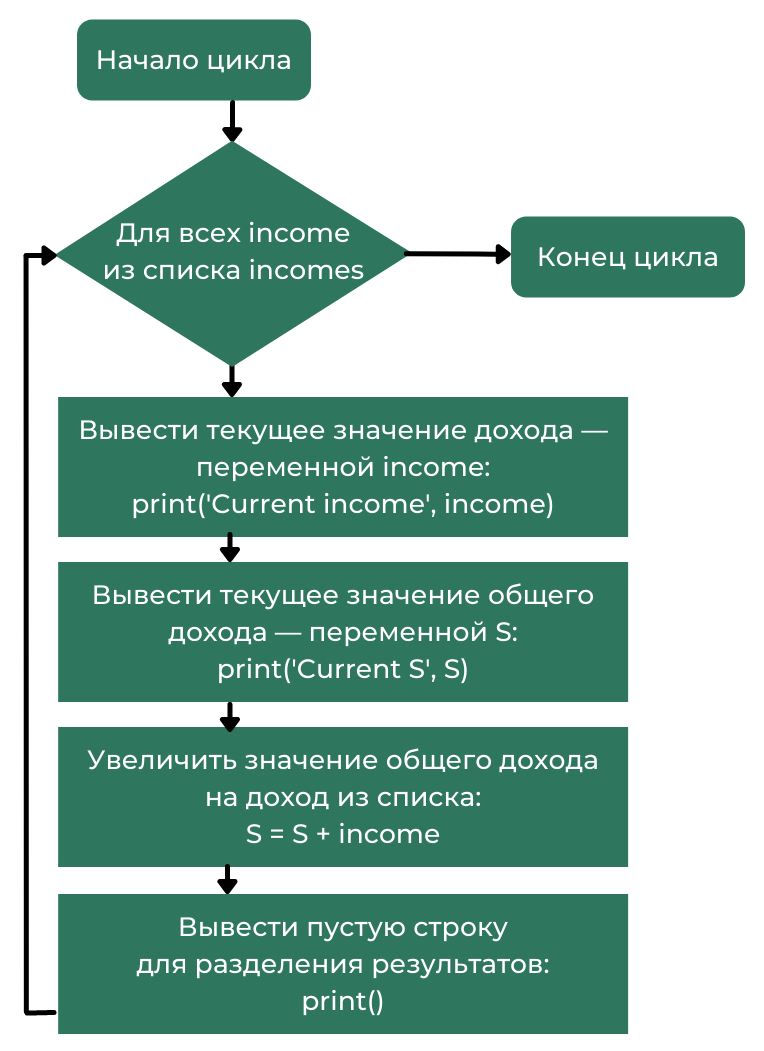
**for** i **in** range(**1**, n+**1**):

**yield** i

**for** student, group **in** zip(students, group\_gen(groups)):

**print**(student, 'studies in group', group)

Задание 31.1.1



Задание 31.1.2

"""обратите внимание, что P = 1, потому что если бы

мы умножали на 0, то всё произведение было бы равно 0"""

lst = [**98**, **24**, **23**, **12**, **3**]

P = **1** #создаём переменную, в которой будем считать произведение

#создаём цикл по элементам списка

**for** element **in** lst:

P = P \* element #умножаем текущее значение P на текущий element

**print**(P) #выводим результат

Задание 31.1.4

"""обратите внимание, что P = 1, потому что если бы

мы умножали на 0, то всё произведение было бы равно 0"""

P = **1** # создаём переменную-счётчик, в которой мы будем считать произведение

N = **5**

# запишите цикл for для подсчёта произведения

**for** i **in** range(**1**, N+**1**):

P \*= i

**print**(P)

Задание 31.1.5

N = **5**

**for** i **in** range(**1**, N + **1**):

**print**("\*" \* i)

Задание 31.1.9

word\_list = ['My', 'name', 'is', 'Ivan'] #список слов

sentence = '' #создаём накопительную переменную, в которую будем добавлять слова

#создаём цикл по словам из списка word\_list

**for** word **in** word\_list:

sentence += word + ' ' #добавляем к текущему предложению слово и пробел

**print**(sentence) #выводим предложение на экран

Задание 31.1.10

num\_list = [**1**, **10**, **3**, -**5**] #заданный список

num\_list.sort() #сортируем список по возрастанию

#создаём цикл по элементам отсортированного списка

**for** num **in** num\_list:

**print**(num) #выводим каждый элемент на экран

Задание 31.1.11

my\_list = list(range(**0**, **100**, **3**)) #заданный список

count = **0** #создаём переменную, в которой будем считать количество чётных элементов

#создаём цикл по элементам списка

**for** elem **in** my\_list:

#число является чётным, если оно делится на 2 без остатка

**if** elem % **2** == **0**: #сравниваем остаток от деления на 2 с 0

count += **1** #если условие выполняется, увеличиваем количество

**print**(count) #выводим количество чётных элементов на экран

Задание 31.1.12

my\_list = [True, **1**, -**10**, "hello", False, "string\_1", **123**, **2.5**, [**1**, **2**], "another"] #заданный список

str\_count = **0** #вводим переменную, в которой будем считать количество строк

#создаём цикл по элементам исходного списка

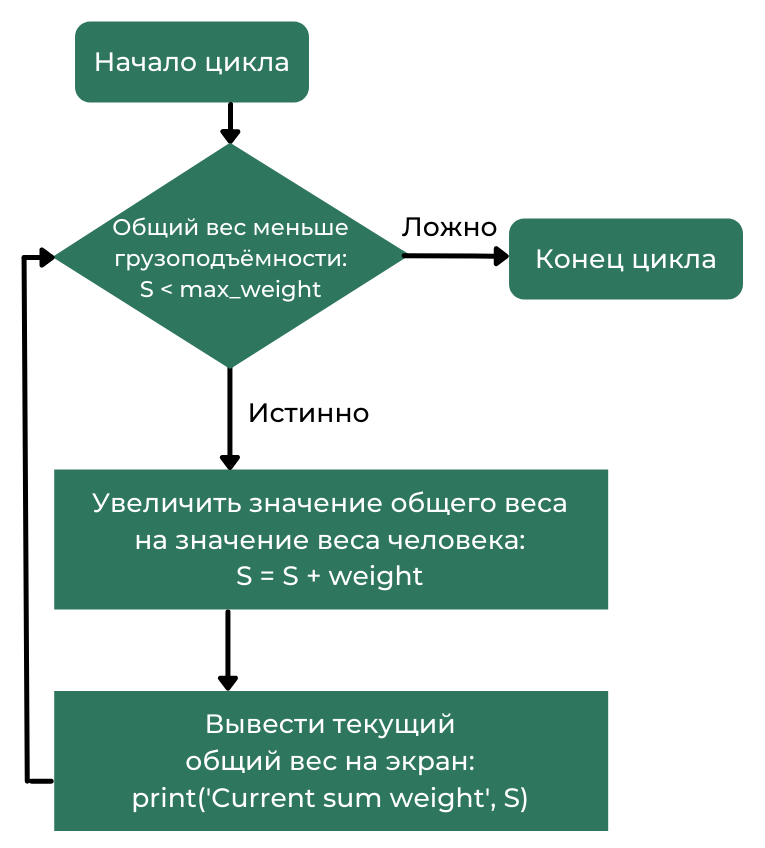
**for** el **in** my\_list:

**if** type(el) **is** str: #проверяем тип данных элемента

str\_count += **1** #если условие выполняется, увеличиваем количество на 1

**print**(str\_count) #выводим результат

Задание 31.2.1



Задание 31.2.2

volume = **10** #задаём объём сосуда в литрах

v = **0.0033** \* **1000** #задаём количество заливаемой жидкости в литрах

S = **0** #задаём суммарный объём залитой жидкости

#создаём цикл, который будет выполняться, пока объём залитой жидкости меньше объёма сосуда

**while** S < volume: #делай, пока...

S += v #увеличиваем суммарный объём

**print**(S - volume) #выводим объём вылитой жидкости

Задание 31.2.3

n = **1**

**while** n\*\***2** < **1000**:

n += **1**

**print**("Required number", n - **1**) # искомое число

Задание 31.2.4

n = **1**

**while** True:

**if** n\*\***2** >= **1000**:

**print**("Required number", n - **1**) # искомое число

**break**

n += **1**

Задание 31.2.10

"""обратите внимание, что P = 1, потому что если бы

мы умножали на 0, то всё произведение было бы равно 0"""

p = **1** #задаём начальное значение произведения

n = **1** #задаём первое натуральное число

#создаём цикл, который будет выполняться, пока произведение не превысит 1 000

**while** p < **1000**: #делай, пока...

p \*= n #умножаем произведение на текущее натуральное число

n += **1** #увеличиваем натуральное число на 1

**print**(p) #выводим результирующее значение произведения

Задание 31.2.11

n = **1** #задаём начальное значение степени

count = **0** #задаём количество итераций

#создаём цикл, который будет выполняться, пока 3 в степени n меньше 1000

**while** **3** \*\* n < **1000**: #делай, пока...

count += **1** #увеличиваем количество итераций

n += **1** #увеличиваем степень на 1

**print**(count) #выводим результирующее количество итераций

Задание 31.2.12

money = **1000** #задаём начальное значение суммы на счёте

year\_count = **0** #задаём начальное значение срока вклада

#создаём цикл, который будет выполняться, пока сумма на счёте меньше 3000

**while** money < **3000**: #делай, пока...

money = money \* **1.08** #увеличиваем сумму на счёте на 8 %

year\_count += **1** #увеличиваем срок хранения денег на счёте

**print**(year\_count) #выводим результирующий срок вклада

Задание 31.3.1

**print**(temperature[**0**][**2**]) #первая строка, третий столбец

**print**(temperature[**1**][**1**]) #вторая строка, второй столбец

**print**(temperature[**2**][**0**]) #третья строка, первый столбец

Задание 31.3.2

hours = list(range(**9**, **24**, **2**)) #создаём список часов

minutes = list(range(**0**, **60**, **15**)) #создаём список минут

#создаём цикл по элементам списка часов

**for** hour **in** hours: #hour — текущее значение часа (9, 11, 13, ...)

#создаём цикл по элементам списка минут

**for** minute **in** minutes: #second — текущее значение минуты

**print**('Alarm is set {}:{}'.format(hour, minute))

Задание 31.3.3

text\_list = [

'afbaad',

'faaf',

'afaga',

'agag'

] #заданный список

count = **0** #задаём начальное значение количества

#создаём цикл по элементам списка text\_list

**for** row **in** text\_list: #row — текущая строка из списка text\_list

#создаём цикл по символам в строке row

**for** symbol **in** row: #symbol — текущий символ в строке row

#проверяем условие, что текущий символ — это символ 'а'

**if** symbol == 'a': #если условие истинно,

count+=**1**

**print**(count) #выводим результат на экран

Задание 31.3.4

random\_matrix = [

[**9**, **2**, **1**],

[**2**, **5**, **3**],

[**4**, **8**, **5**]

] #заданная матрица

max\_value\_rows = [] #задаём пустой список с максимальными значениями строк

#создаём цикл по строкам матрицы random\_matrix

**for** row **in** random\_matrix: #row — текущая строка таблицы

max\_value = row[**0**] #начальное значение кандидата на максимум

#создаём цикл по элементам списка row

**for** elem **in** row: #elem — текущий элемент из списка row

#проверяем условие, что текущий элемент больше кандидата на максимум

**if** elem > max\_value: #если условие выполняется,

max\_value = elem #заменяем кандидата на максимум

max\_value\_rows.append(max\_value) #добавляем полученный максимум строки в список

**print**("Maximum elements:", max\_value\_rows) #выводим максимальные элементы

Задание 31.3.8

test\_matrix1 = [

[**1**, **2**, **3**],

[**7**, -**1**, **2**],

[**123**, **2**, -**1**]

] #заданная тестовая матрица

num\_lines = len(test\_matrix1) #вычисляем число строк в матрице

count = **0** #задаём начальное количество строк, длина которых совпадает с числом строк в матрице

#создаём цикл по строкам матрицы

**for** line **in** test\_matrix1: #line — текущая строка матрицы

#проверяем, что длина текущей строки равна количеству строк

**if** len(line) == num\_lines: #если условие выполняется,

count += **1** #увеличиваем количество строк, длина которых совпадает с числом строк в матрице

#сравниваем полученное число с числом строк в матрице и выводим результат

**print**(num\_lines == count)

Задание 31.4.3

user\_dynamics = [-**5**, **2**, **4**, **8**, **12**, -**7**, **5**] #заданный список динамики пользователей

number\_negative = None # объявляем переменную, в которой будем хранить номер последнего дня оттока, изначально она пустая (None)

#создаём цикл по индексам и элементам списка

**for** i, dynamic **in** enumerate(user\_dynamics): #index — индекс текущего элемента, dymanic — текущее значение из списка

**if** dynamic < **0**:

number\_negative = i+**1** #перезаписываем значение номера дня

**print**("Churn value: ", user\_dynamics[i]) # выводим количество ушедших в этот день пользователей

**print**("Number day: ", number\_negative) # выводим номер дня

Задание 31.4.6

n = **19** #задаём число

#создаём бесконечный цикл

**while** True:

#проверяем условие, что число чётное

**if** n % **2** == **0**: #если число чётное,

n = n // **2** #делим число на 2

**else**: #если число нечётное,

n = (n\***3** + **1**) // **2** #умножаем число на 3, прибавляем к нему 1 и делим на 2

**if** n == **1**: #если результат равен 1,

**print**('Syracuse hypothesis holds') #выводим утвердительное сообщение

**break** #выходим из цикла

Задание 31.4.9

my\_dict = {'a': **15**, 'b': **10.5**, 'c': '15', 'd': **50**, 'e': **15**, 'f': '15'} #заданный словарь

count = **0** #задаём начальное количество

#создаём цикл по ключам словаря my\_dict

**for** key **in** my\_dict: #key — текущий ключ из словаря

#проверяем условие, что тип данных значения — строка

**if** type(my\_dict[key]) **is** str: #если условие выполняется,

**continue** #переходим на следующую итерацию

**else**: #в противном случае

count += **1** #увеличиваем количество

**print**(count) #выводим результат

Задание 31.4.10

text = """

She sells sea shells on the sea shore;

The shells that she sells are sea shells I am sure.

So if she sells sea shells on the sea shore,

I am sure that the shells are sea shore shells.

"""

text = text.lower() #приводим текст к нижнему регистру

text = text.replace(" ", "") #заменяем пробелы на пустые строки

text = text.replace("**\n**", "") #заменяем символы переноса строки на пустые строки

count\_dict = {} #создаём пустой словарь для подсчёта количества символов

#создаём цикл по символам в строке text

**for** symbol **in** text: #symbol — текущий символ в тексте

#проверяем условие, что символа ещё нет среди ключей словаря

**if** symbol **not** **in** count\_dict: #если условие выполняется,

count\_dict[symbol] = **1** #заносим символ в словарь со значением 1

**else**: #в противном случае

count\_dict[symbol] += **1** #увеличиваем частоту символа

**print**(count\_dict) #выводим результирующий словарь

Задание 31.5.1

**import** **random**

my\_list = []

**for** i **in** range(**0**, **10**):

my\_list.append(random.randint(**0**, **10**))

#создаём цикл по элементам полученного списка

**for** elem **in** my\_list: #elem — текущий элемент списка

#проверяем, что элемент встречается в списке не один раз

**if** my\_list.count(elem) > **1**: #если условие выполняется,

**print**('yes') #выводим фразу на экран

**break** #прерываем цикл

Задание 31.5.2

**import** **random**

my\_list = []

**for** i **in** range(**0**, **10**):

my\_list.append(random.randint(**0**, **10**))

S = **0** #создаём накопительную переменную суммы

i = **0** #создаём переменную, которая будет хранить индекс текущего элемента списка

#создаём цикл, условие выхода из которого — сумма >= 10

**while** S < **10**:

S += my\_list[i] #добавляем к сумме текущий элемент списка

i += **1** #увеличиваем индекс элемента

**print**('Result', S)

Задание 31.5.3

current\_health = **500** #заданный показатель здоровья

attack = **80** #заданная атака босса

seconds\_num = **0** #задаём начальное время схватки

#создаём цикл, который будет выполняться, пока показатель здоровья >= 0

**while** current\_health >= **0**:

current\_health -= attack #уменьшаем текущий показатель здоровья

seconds\_num += **1** #увеличиваем время схватки

Задание 31.5.4

str\_list = ["Hello", "my", "name", "is", "Ezeikel", "I", "like", "knitting"] #заданный список

cut\_str\_list = [] #новый список

#создаём цикл по индексам и элементам списка str\_list

**for** index, word **in** enumerate(str\_list): #index — текущий индекс (номер) элемента, word — текущая строка

cut\_str\_list.append([index, word[:**3**]]) #добавляем в список cut\_str\_list список из номера строки и первых трёх символов из строки

Задание 31.5.5

sentence = "A roboT MAY Not injure a humAn BEING or, tHROugh INACtion, allow a human BEING to come to harm"

sentence = sentence.lower() #приводим строку к нижнему регистру

sentence = sentence.replace(",", "") #удаляем знаки препинания из предложения

word\_list = sentence.split(' ') #разделяем строку по пробелам

word\_dict = {} #создаём словарь для подсчёта количества слов

#создаём цикл по словам в списке word\_list

**for** word **in** word\_list: #word — текущее слово из списка

#проверяем условие, что текущего слова ещё нет в словаре

**if** word **not** **in** word\_dict: #если условие выполняется,

word\_dict[word] = **1** #заносим новое слово в словарь со значением 1

**else**: #в противном случае

word\_dict[word] += **1** #увеличиваем количество слов

Задание 31.5.6

str\_list = ["text", "morning", "notepad", "television", "ornament"]

symbol\_to\_check = "t" #задаём символ, который будем искать

words\_dict = {} #задаём пустой словарь для подсчёта

#создаём внешний цикл по элементам списка

**for** word **in** str\_list:

#заносим текущее слово в словарь со значением 0

words\_dict[word] = **0**

#создаём внутренний цикл по символам в строке

**for** letter **in** word:

#проверяем условие, что текущий символ в строке — это искомый символ

**if** letter == symbol\_to\_check: #если условие выполняется,

words\_dict[word] += **1** #увеличиваем количество символов в слове

**print**(words\_dict) #выводим результирующий словарь