
КОЛЛЕКЦИИ
Занятие №5

Проверка пройденного на занятии №4.2

1. Из каких двух частей состоит цикл while?
2. В каком случае используются циклы?
3. Как прописывать условие цикла while?
4. Как выполняется цикл while?
5. Что вы знаете о бесконечном цикле?
6. Расскажите о конструкции for-else, while-else.
7. Выполняется ли условие else, если завершить цикл принудительно?

Проверка домашнего задания

Задача №1:

Казино. Компьютер генерирует числа от 1 до 10 и от 1 до 2-х соответственно. Цифры от одного до 10 отвечают за номера, а от 1 до 2 за цвета(1-красный,2 черный).

Пользователю дается 5 попыток угадать номер и цвет(Прим. введения с клавиатуры: 3 красный).В случае неудачи вывести на экран правильную комбинацию.

Решение и результат :

```python
import random
num = random.randint(1, 10)
col = random.randint(1, 2)
print(num)
print(col)
i = 0
while i < 5:
    i += 1
    guess = int(input("Ваше число: "))
    color = (int(input("Введите цвет:1-красный или 2-чёрный: ")))
    if guess > num:
        print("Число меньше твоего.")
        if color == col:
            print("Но!Вы угадали цвет")
        else:
            print("Вы не угадали цвет")
    elif guess < num:
        print("Число больше твоего.")
        if color == col:
            print("Но!Вы угадали цвет")
        else:
            print("Вы не угадали цвет")
    elif guess == num and col != color:
        print("Вы не угадали цвет, но угадали число!")
    elif guess == num and col == color:
        print("Поздравляю! ", num, col)
        break
```

```python
if i < 5:
    print("Попробуй снова")
else:
    print("Это было число ", num, col, ".Game over!")
```

План занятия

1) Что такое коллекции.
2) Списки. Особенности, реализация, встроенные функции для работы со списками.

Что такое коллекции?

Коллекция в Python — программный объект (переменная-контейнер), хранящая набор значений одного или различных типов, позволяющий обращаться к этим значениям, а также применять специальные функции и методы, зависящие от типа коллекции.

Стандартные встроенные коллекционные типы данных в Python: список (list), кортеж (tuple), строку (string), множества (set, frozenset), словарь (dict).

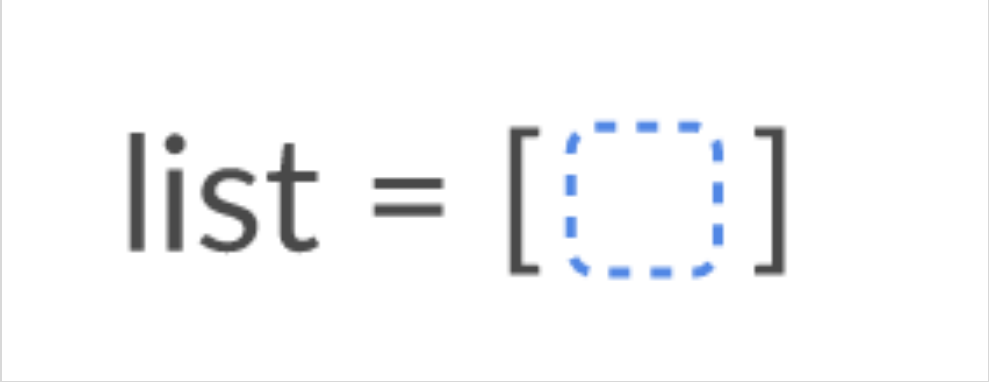
Обобщение свойств встроенных коллекций в сводной таблице:

Тип коллекции	Изменяемость	Индексированность	Уникальность	Как создаём
Список (list)	+	+	-	*[ ]* *list( )*
Кортеж (tuple)	-	+	-	*( ) ,* *tuple( )*
Строка (string)	-	+	-	*''* *""*
Множество (set)	+	-	+	*{elm1, elm2}* *set( )*
Неизменное множество (frozenset)	-	-	+	*frozenset( )*
Словарь (dict)	+ элементы - ключи + значения	-	+ элементы + ключи - значения	*{ }* *{key: value,}* *dict()*

Списки

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

Объявление списка – самый первый и главный этап его создания. Для объявления списка в Python существует несколько способов.



Вариант №1: Через литерал (выражение, создающее объект):

```
elements = [1, 3, 'a', 6, 'b']


print(elements)
```

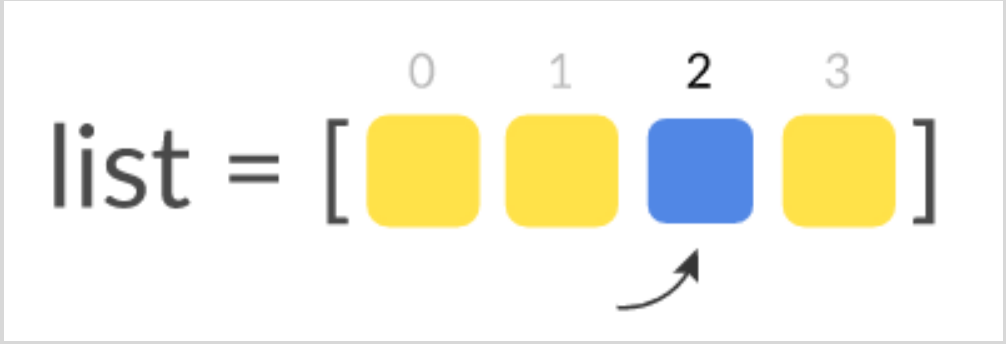
В данном примере мы создали список с заранее известными данными. Если нужен пустой список, в квадратных скобках ничего не указывается – elements = []

Вариант №2: Через функцию list():

```
elements = list()


print(elements)
```

В этом примере создается пустой список.

Обращение к элементу списка в Python



Некоторые операции, имеют два варианта выбора элемента: либо выбор непосредственно его по имени, либо обращение по индексу. Индексом называют его порядковый номер, начиная с нуля.

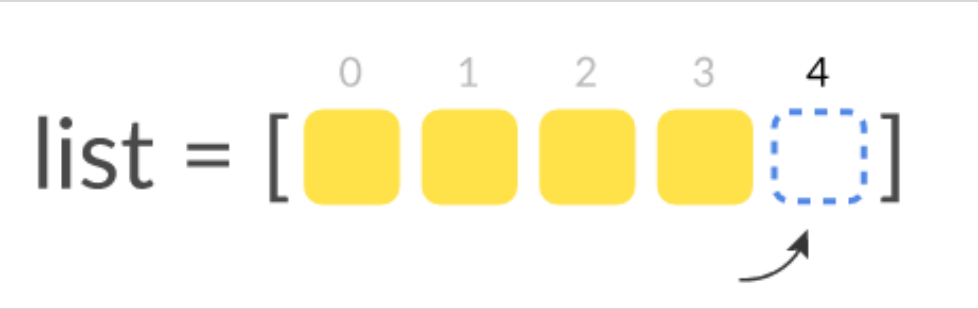
Существует также **отрицательный** индекс. Отрицательными индексами называют расположение элементов в списке справа налево

Отрицательным индексом удобно пользоваться, когда необходимо обратиться к последнему в списке элементу, не высчитывая его номер. Любой конечный элемент будет с индексом, равным -1

Пример программы

```
import random


# Объвим список c и заполним его 10 элементами от 0 до 100
c = [random.randint(0, 100) for i in range(10)]
print(c)


print(c[0])
print(c[-1])
print(c[5])
print(c[-4])
```

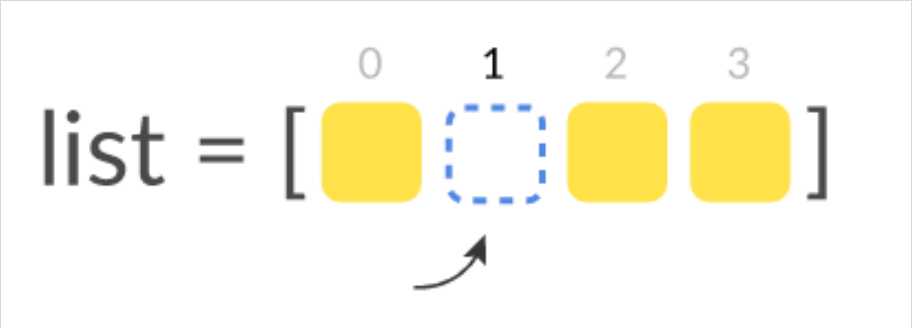
Добавление в список



Для того, чтобы добавить новый элемент в список, используется list.append(x) , где list – список, x – нужное значение.

```
elements = []


elements.append('a')
elements.append(1)


print(elements)
```

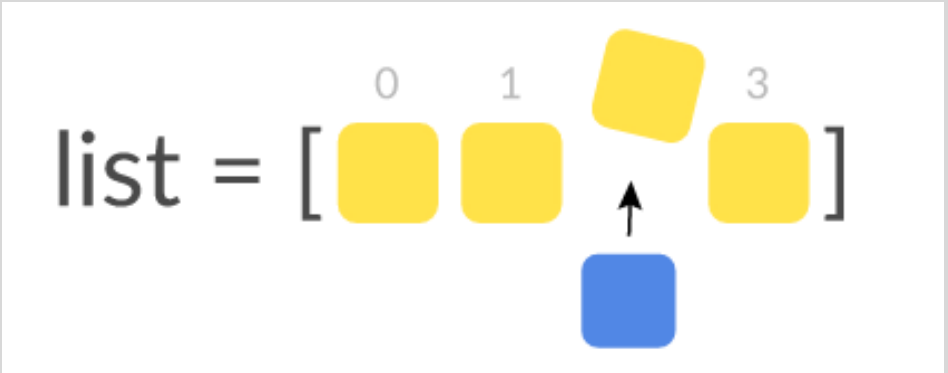
Добавление в список на указанную позицию



Немаловажно обратить внимание на метод **list.insert(i, x)** , где list – список, i – позиция, x – нужное значение.

```
elements = [1, 3, 'a', 6, 'b']
elements.insert(1, 2)
print(elements)
```

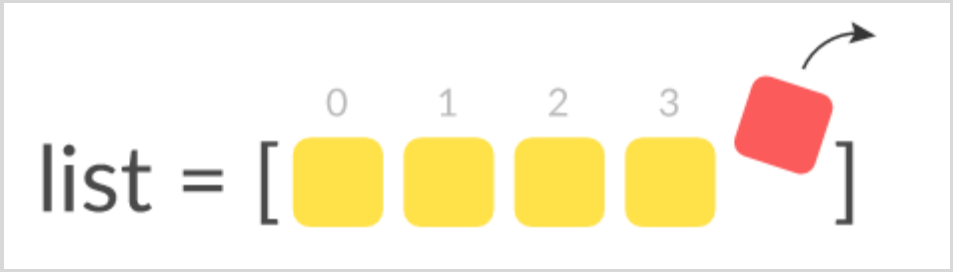
Изменение элементов списка



Изменение элементов списка происходит следующим образом: нужно выбрать элемент по индексу (порядковому номеру элемента) и присвоить новое значение.

```
elements = [1, 3, 'a', 6, 'b']
elements[1] = 2
print(elements)
```

Удаление элемента из списка



Для удаление из списка используют инструкцию del list[i] , где list – список, i – индекс (позиция) элемента в списке.

Еще один способ удаления из списка – list.remove(x) , где list – список, x – значение, которое нужно удалить.

```
elements = [1, 3, 'a', 6, 'b']
del elements[1]
print(elements)
```

```
elements = [1, 3, 'a', 6, 'a', 'b']
elements.remove('a')
print(elements)
```

Удалять можно как из текущего списка, так и из вложенных списков:

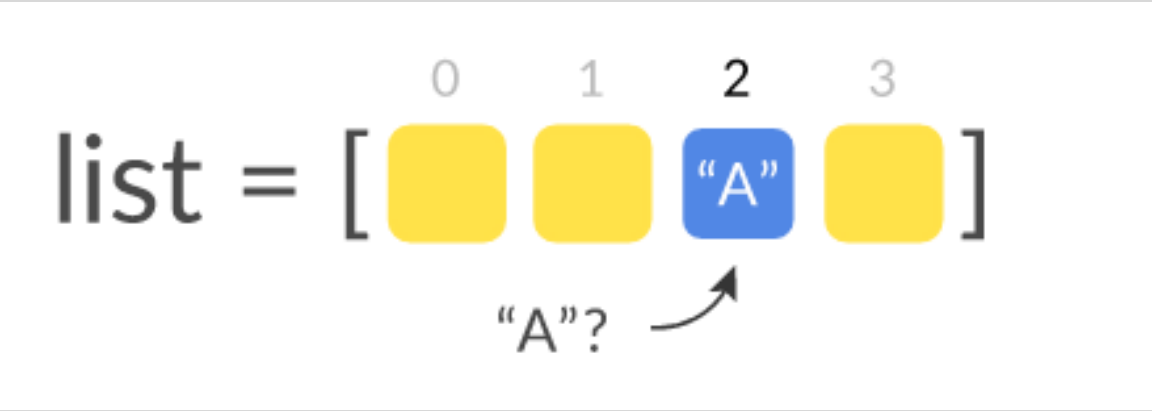
```
my_list = ['hello', 'world']
elements = [1, 3, my_list, 6, 'a', 'b']

del elements[2][1]
```

Можно удалять целыми диапазонами:

```
elements = [1, 3, 6, 'a', 'b', 'abc', 123, 435]

del elements[4:]  # удаляем все элементы после 4-го элемента (включительно)
print(elements)
del elements[:2]  # удаляем все элементы до 2-го элемента
print(elements)
del elements[1:3]  # удаляем от 1-го элемента до 3-го элемента
print(elements)
```

```
[1, 3, 6, 'a']
[6, 'a']
[6]
```

Как проверить наличие элемента в списке



Для того, чтобы проверить существование какого-либо элемента в списке, нужно воспользоваться оператором **in** . Рассмотрим на примере:

```
elements = [1, 3, 6, 'a', 'b', 'abc', 123, 435]

if 'abc' in elements:
    print('Yes.')
```

Объединение списков

Списки в Python можно объединят с помощью оператора + или метода extend . Выглядит это так:

```
a = [1, 3, 5]
b = [1, 5, 8, 9]
print(a + b)

d = ['h', 'e', 'l', 'l', 'o']
e = ['w', 'o', 'r', 'l', 'd']

d.extend(e)  # extend не возвращает новый список, а дополняет текущий
print(d)
```

Копирование списка в Python

Если вы захотите скопировать список оператором = , вы скопируете не сам список, а только его ссылку.

```
a = [1, 2, 3, 4]
b = a  # переменной b присваивается не значение списка a, а его адрес

a.append(5)
b.append(6)
print('a', a, 'b', b)
```

```
a [1, 2, 3, 4, 5, 6] b [1, 2, 3, 4, 5, 6]

Process finished with exit code 0
```

Для копирования списков можно использовать несколько вариантов:

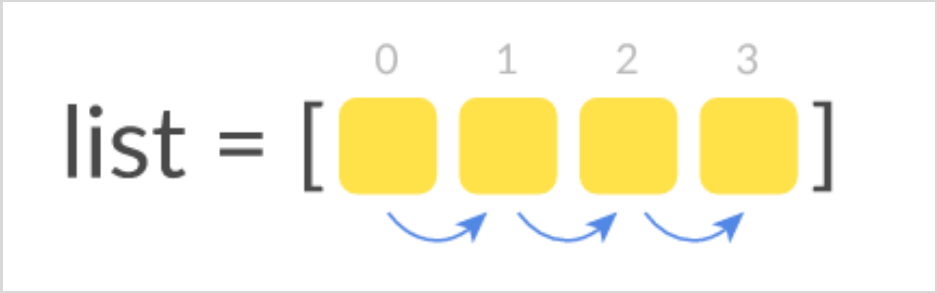
1. elements.copy() – встроенный метод copy (доступен с Python 3.3);
2. list(elements) – через встроенную функцию list() ;
3. copy.copy(elements) – функция copy() из пакета copy;
4. elements[:] – через создание среза (устаревший синтаксис);

```
a = ["кот", "слон", "змея"]

b = a.copy()
print(a, b)

d = list(a)
print(a, d)

import copy

e = copy.copy(a)  #
print(a, e)

f = copy.deepcopy(a)
print(a, f)

c = a[:]  # устаревший синтаксис
print(a, c)
```

Скопировать часть списка можно с помощью срезов. Есть несколько вариантов использования:

```
a = ["кот", "слон", "змея"]

b = a[2:]  # с 2-го элемента (включительно) до конца списка
print(b)

c = a[:2]  # с начала списка по 2-й элемент
print(c)

d = a[1:2]  # с 1-го элемента (включительно) по 2-й элемент
print(d)

a = [1, 2, 3, 4, 5, 6, 7, 8]
e = a[0:8:2]  # с 0-го элемента по 8-й элемент с шагом 2
print(e)
```

Цикл по списку



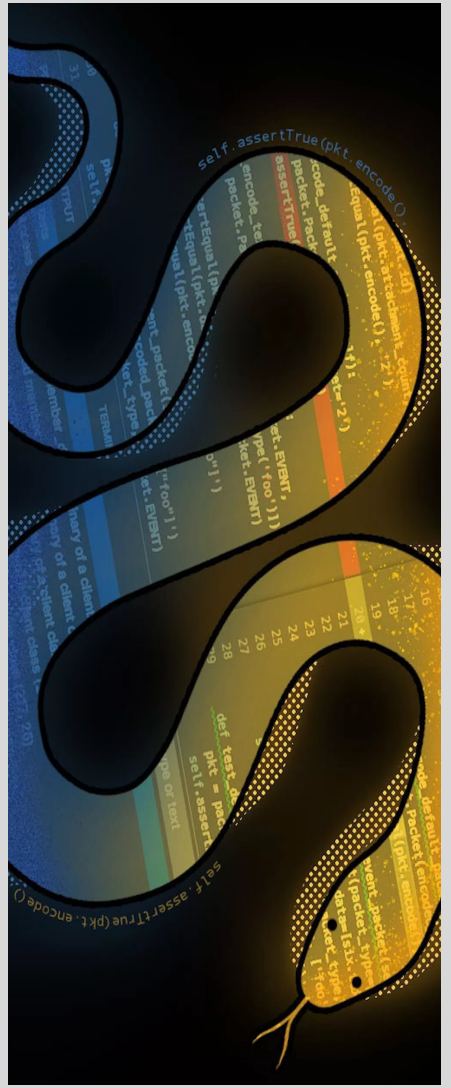
Для перебора списков в Python есть два цикла: for и while.
Из примеров ниже можем сделать вывод, что конструкция for выглядит заметно компактнее, чем while

```
elements = [1, 2, 3, "meow"]
for i in elements:
    print(i)
```

```
elements = [1, 2, 3, "meow"]
elements_len = len(elements)
i = 0
while i < elements_len:
    print(elements[i])
    i += 1
```

Методы списков

list.append(x) – позволяет добавлять элемент в конец списка;
list1.extend(list2) – предназначен для сложения списков;
list.insert(i, x) – служит для добавления элемента на указанную позицию(i – позиция, x – элемент);
list.remove(x) – удаляет элемент из списка (только первое вхождение);
list.clear() – предназначен для удаления всех элементов (после этой операции список становится пустым []);
list.copy() – служит для копирования списков.
list.count(x) – посчитает количество элементов x в списке;
list.index(x) – вернет позицию первого найденного элемента x в списке;
list.pop(i) - удалит элемент из позиции i ;
list.reverse() – меняет порядок элементов в списке на противоположный;
list.sort() – сортирует список;



```python
# clear
a = [1, 2, 3]
a.clear()
print(a)

# copy
a = [1, 2, 3]
b = a.copy()
print(id(a), id(b), a, b)

# count
elements = ["one", "two", "three", "one", "two", "one"]
print(elements.count("one"))

# index
elements = ["one", "two", "three", "one", "two", "one"]
print(elements.index("three"))
```

```python
# pop
elements = [1, "meow", 3, "meow"]
elements.pop(1)  # удаляем элемент с индексом 1
'meow'  # pop возвращает удаленный элемент списка
print(elements)


elements.pop()  # удаляем первый элемент списка

print(elements)

elements.pop(-1)  # удаляем последний элемент списка

print(elements)

# reverse
a = [1, 2, 3]
a.reverse()
print(a)
```

```python
# sort (по возрастанию)
elements = [3, 19, 0, 3, 102, 3, 1]
elements.sort()
print(elements)

# sort (по убыванию)
elements = [3, 19, 0, 3, 102, 3, 1]
elements.sort(reverse=True)
print(elements)
```

Вложенные списки

Список может содержать объекты разных типов: числовые, буквенные, а также списки. Список списков выглядит следующим образом:

```
elements = [1, 3, 4, [5, 7, 8], 'abc', 4]
```

Для обращения к элементу вложенного списка нужно использовать два индекса: первый указывает на индекс главного списка, второй — индекс элемента во вложенном списке. Вот пример:

```
elements = [["яблоки", 50], ["апельсины", 190], ["груши", 100]]
print(elements[0])



print(elements[1][0])
```

Срезы

Срезы (slices) – это подмножества элементов списка. Срезу нужны, когда необходимо извлечь часть списка из полного списка.
У них есть свой собственный синтаксис. Записывается срез так же, как обращение к элементу, используя индекс.

Пример:

```
elements[START:STOP:STEP]
```

В этом случае берётся срез от номера start (включительно) до stop (не включая его), а step – это шаг. По умолчанию start и stop равны 0, step равен 1.

```
elements = [0.1, 0.2, 1, 2, 3, 4, 0.3, 0.4]
int_elements = elements[2:6]  # с 2-го элемента включительно по 6-й элемент

print(id(elements), id(int_elements))  # elements и int_elements - 2 разных списка



print(elements)
# срез не модифицирует исходный список

print(int_elements)
```

Задание №1

Дан произвольный список. Представьте его в обратном порядке.

Решение

```python
# Дан произвольный список.
# Представьте его в обратном порядке.

# 1 вариант
my_list = [2, 4, 8]
print(my_list[::-1])

# 2 вариант
my_list = [2, 4, 8]
my_list.reverse()
print(my_list)
```

Задание №2

Дан список некоторых целых чисел, найдите значение 20 в нем и, если оно присутствует, замените его на 200. Обновите список только при первом вхождении числа 20.

Решение

Мы можем использовать метод index(), который позволит получить индекс первого вхождения некоторого объекта (в нашем случае числа 20). Затем просто изменим элемент списка с этим индексом до нужного нам значения (то есть 200)

```
list1 = [5, 10, 15, 20, 25, 50, 20]

print("Список: \n", list1)

ind = list1.index(20)
list1[ind] = 200

print("Изменённый список: \n", list1)
```

Задание №3

Список из 7 цифр. Если четных цифр в нем больше чем нечетных, то найти сумму всех его цифр, если нечетных больше, то найти произведение 1 3 и 6 элемента.

Решение и результат :

```python
# Практическое 3

mas = [1, 3, 4, 5, 6, 7, 8]
count = 0
count_2 = 0
for i in mas:
    if i % 2 == 0:
        count += 1
    else:
        count_2 += 1
print("Чётных: ", count, "Нечётных:", count_2)
sum = 0
pr = 1
if count > count_2:
    for i in mas:
        sum += i
    print("Сумма: ", sum)
else:
    pr = mas[0] * mas[2] * mas[5]
    print("Произведение: ", pr)
```

```
Чётных:  3 Нечётных: 4
Произведение:  28

Process finished with exit code 0
```

Задание №4

Найти совпадающие элементы двух списков.
a = [5,[1,2],2,'r',4,'ee']
b = [4,'we','ee',3,[1,2]]
Эти значения записать в новый список

Решение

```python
a = [5, [1, 2], 2, 'r', 4, 'ee']
b = [4, 'we', 'ee', 3, [1, 2]]
c = []

for i in a:
    if i in b:
        c.append(i)
print(c)
```

```
[[1, 2], 4, 'ee']

Process finished with exit code 0
```

Задание №5

Даны 2 списка:

a = [4,6,'py','tell',78]

b = [44,'hello’,56,'exept’,3]

Выполнить следующие операции:

1. Сложить два списка
2. Добавьте элемент 6 на 3 позицию.
3. Удалите все текстовые переменные
4. Посчитайте количество элементов списка

Решение

```python
a = [4, 6, 'py', 'tell', 78]
b = [44, 'hello', 56, 'exept', 3]

a.extend(b)
a.insert(3, 6)
print(a)
for i in a:
    if type(i) is str:
        a.remove(i)
a.sort()
print(a)
```

Домашнее задание

Прочитать про заполнение списков с помощью random.

Задача №1

Дан список list=[15,48,'hello',6,19,'world’].
Все числа этого списка проверить на чётность. Если число чётное, то посчитать сумму его цифр.
Если нечётное, то заменить его на 1 в списке.
Все слова: посчитать количество гласных и согласных.
Вывести всё на экран.

Подготовиться к экзамену

Повторить темы:

1. Введение в программирование
2. Условные операторы
3. Строки
4. Циклы