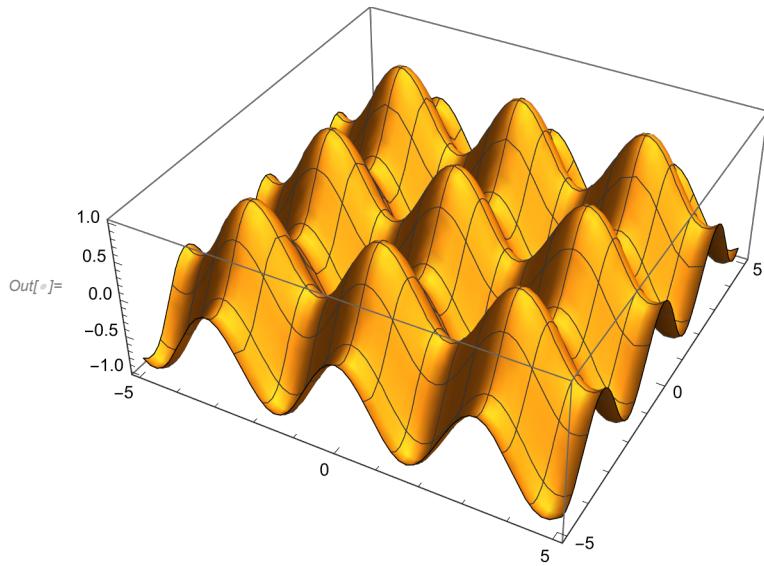


## 1. Визуализация явно заданной функции двух переменных

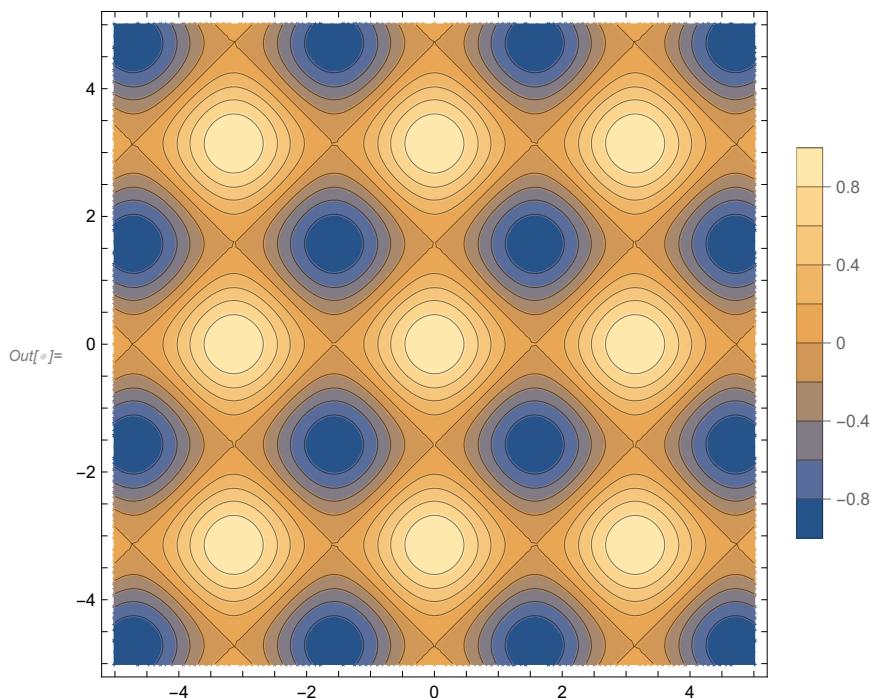
In[ $\#$ ]:=  $f[x\_, y\_] := \cos[x + y] \cos[x - y]$   
| косинус | косинус

In[ $\#$ ]:= Plot3D[f[x, y], {x, -5, 5}, {y, -5, 5}]  
| график функции 2-х переменных



Линии уровня (изолиния)  $z(x,y) = c$  (“горизонтальные срезы поверхности  $z=z(x,y)$  на различных высотах”):

In[ $\#$ ]:= h1 = ContourPlot[f[x, y], {x, -5, 5}, {y, -5, 5}, PlotLegends -> Automatic]  
| контурный график | легенды графика | автоматически



## 2. Локальные экстремумы функций двух переменных

Все аналогично функции одной переменной - только указывается 2 точки :

```
In[1]:= FindMaximum[f[x, y], {{x, 3}, {y, 0}}]
найти максимум
```

```
Out[1]= {1., {x → 3.14159, y → 0.}}
```

## 3. Градиент

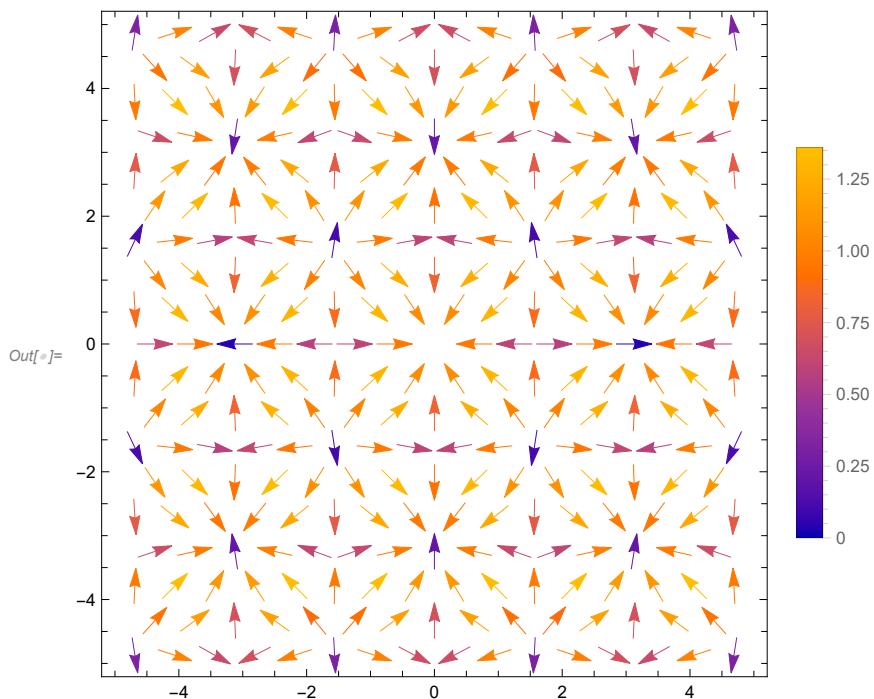
Вычисление градиента функции  $f(x, y)$ :

```
In[2]:= g = Grad[f[x, y], {x, y}]
градиент функции
```

```
Out[2]= {-Cos[x + y] Sin[x - y] - Cos[x - y] Sin[x + y], Cos[x + y] Sin[x - y] - Cos[x - y] Sin[x + y]}
```

Визуализация векторного поля:

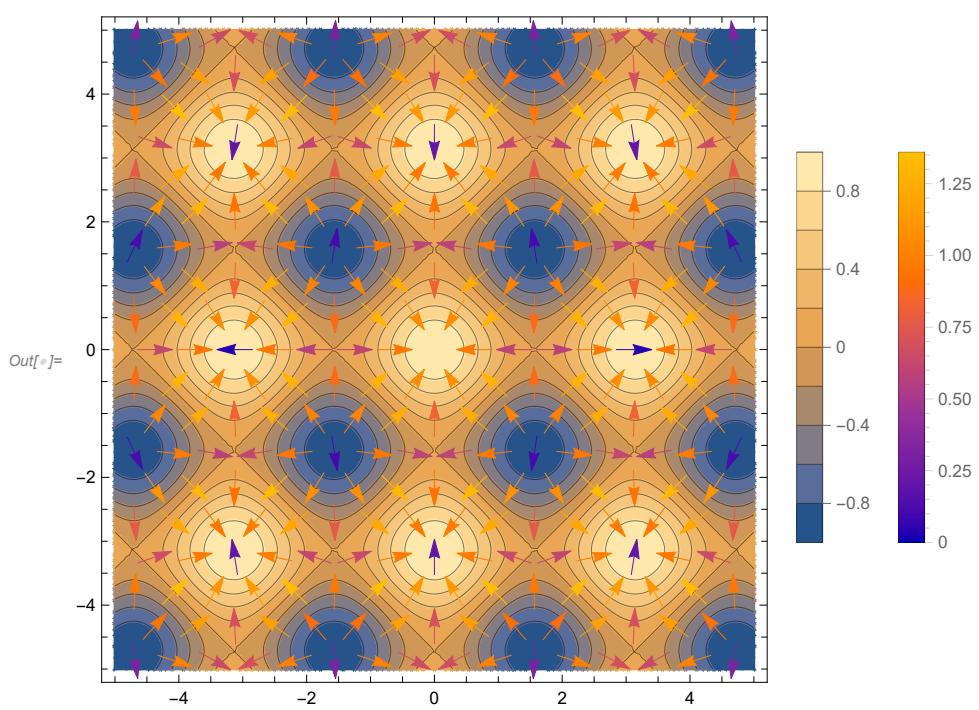
```
In[3]:= h2 = VectorPlot[g, {x, -5, 5}, {y, -5, 5}, PlotLegends → Automatic]
векторная диаграмма
легенды графика автоматически
```



Свойство: Вектора градиента ортогональны линиям уровня :

In[6]:= **Show[h1, h2]**

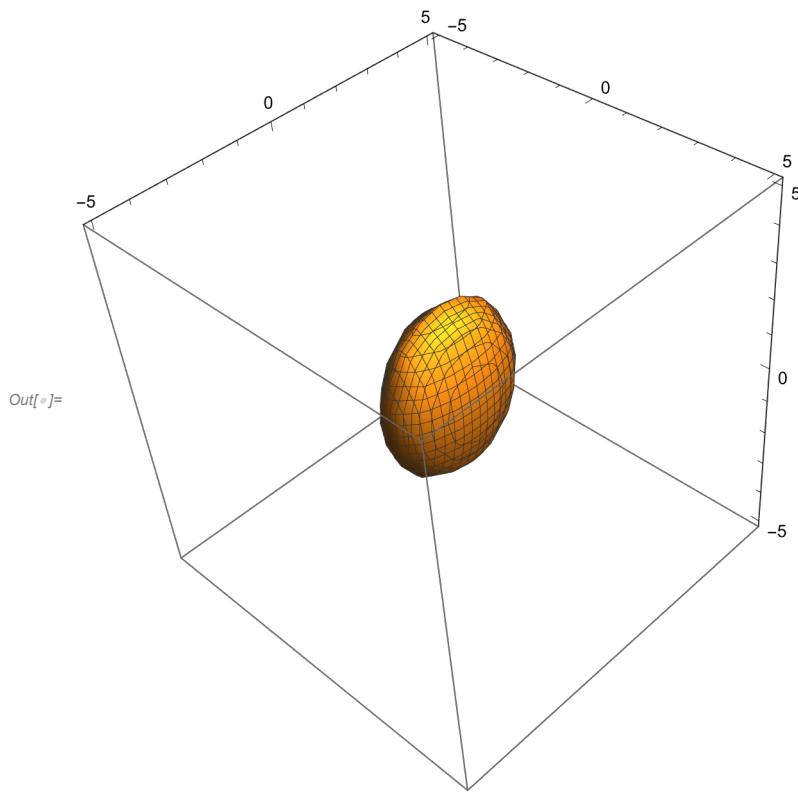
| показать



#### 4. Визуализация неявно заданной функции двух переменных

$$\left(x^2 + \frac{y^2}{4} + \frac{z^2}{6} = 1\right)$$

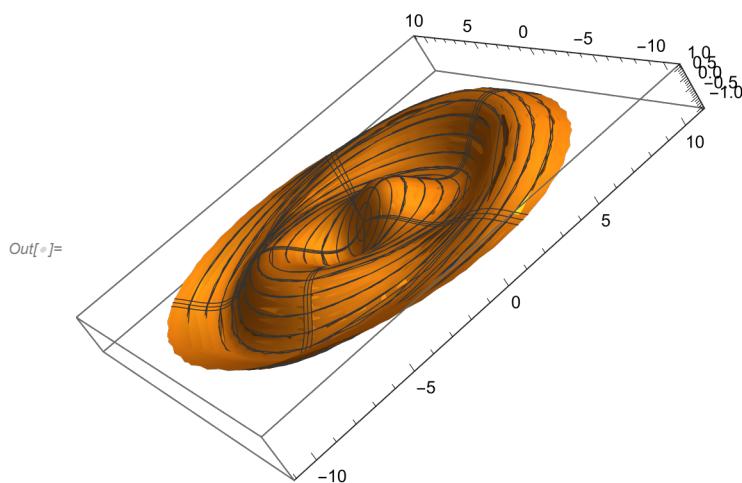
In[4]:= **ContourPlot3D[x^2 + y^2 / 4 + z^2 / 6 == 1, {x, -5, 5}, {y, -5, 5}, {z, -5, 5}]**  
 контурный график в пространстве



#### 5. Визуализация параметрически заданной функции двух переменных

$$(x(t) = \sin(t_1) \cos(t_2), y(t) = \sin(t_2) \cos(t_2) t_1, z(t) = \sin(t_2) t_1)$$

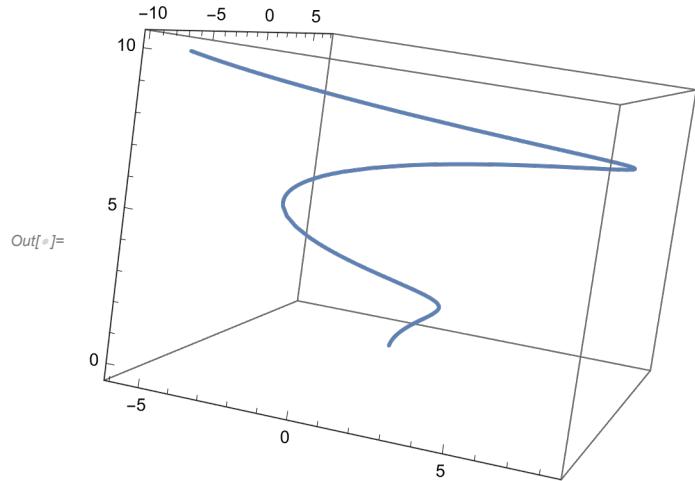
In[5]:= **ParametricPlot3D[{Sin[t1] Cos[t2], Cos[t2] t1, Sin[t2] t1}, {t1, 0, 10}, {t2, 0, 20}]**  
 график параметрического [синус] [косинус] [косинус] [синус]



#### 6. Визуализация параметрически заданной кривой в $\mathbb{R}^3$ ( $x(t) = \sin(t)$ , $y(t)$ )

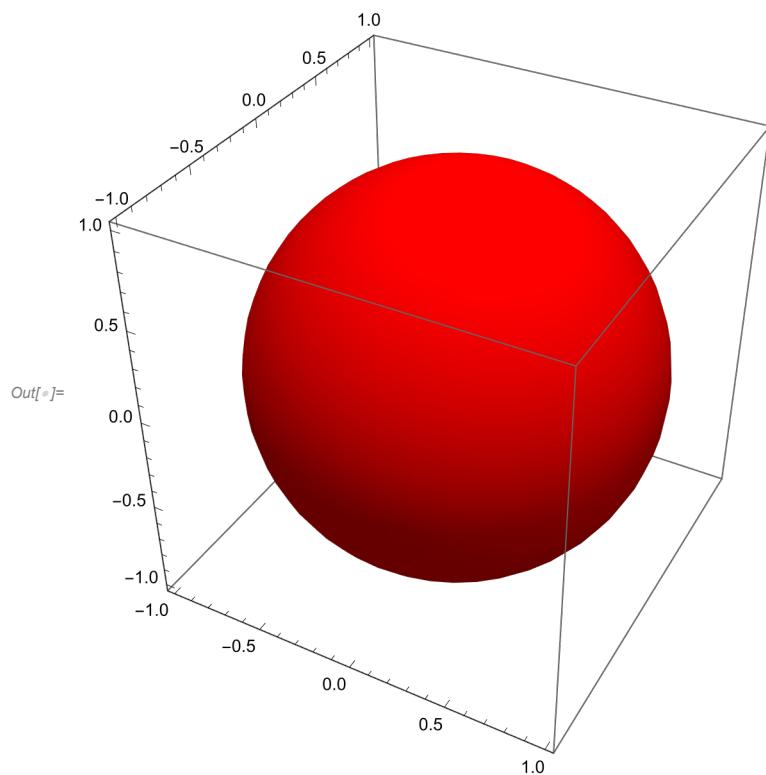
$=\cos(t) t, z(t)=t$ :

```
ParametricPlot3D[{Sin[t] t, Cos[t] t, t}, {t, 0, 10}]
График параметрического
```



### 7. 3 D -графика из примитивов:

```
In[]:= Graphics3D[{Red, Ball[{0, 0, 0}, 1]}, Axes -> True]
3-мерная группа примитивов
```



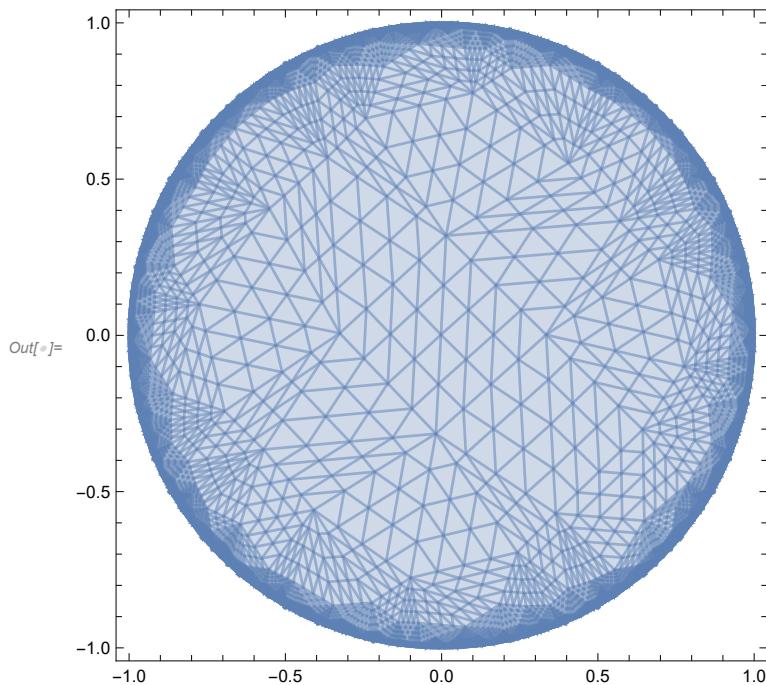
### 8. Создание областей

Задание неявно заданной области в  $R^2$ :

```
In[6]:= r1 = ImplicitRegion[x^2 + y^2 ≤ 1, {x, y}]
          |неявно заданная область
Out[6]= ImplicitRegion[x^2 + y^2 ≤ 1, {x, y}]
```

Визуализация:

```
In[7]:= RegionPlot[r1]
          |визуализация геометрической области на плоскости
```



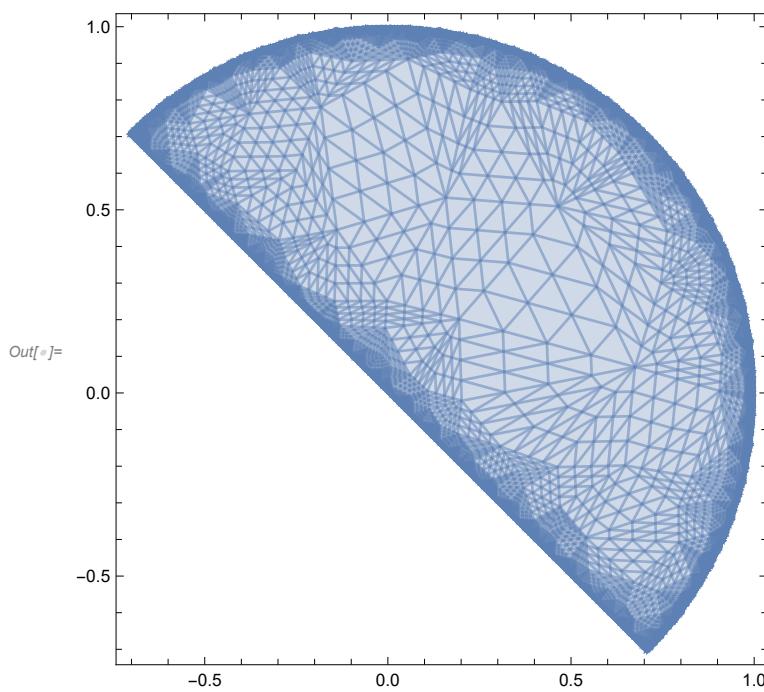
```
In[8]:= r2 = ImplicitRegion[y ≥ -x, {x, y}];
          |неявно заданная область
```

Пересечение областей (как пересечение множеств):

```
In[9]:= r3 = RegionIntersection[r1, r2];
          |пересечения геометрических регионов
```

In[4]:= **RegionPlot[r3]**

| визуализация геометрической области на плоскости

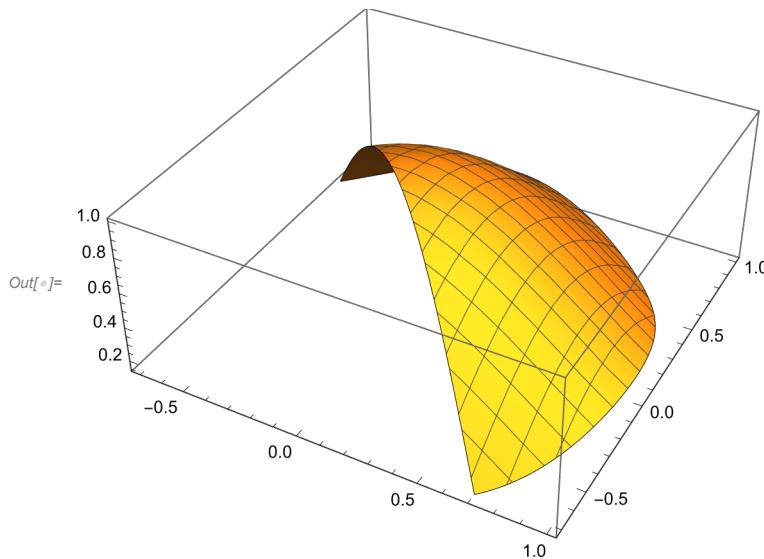


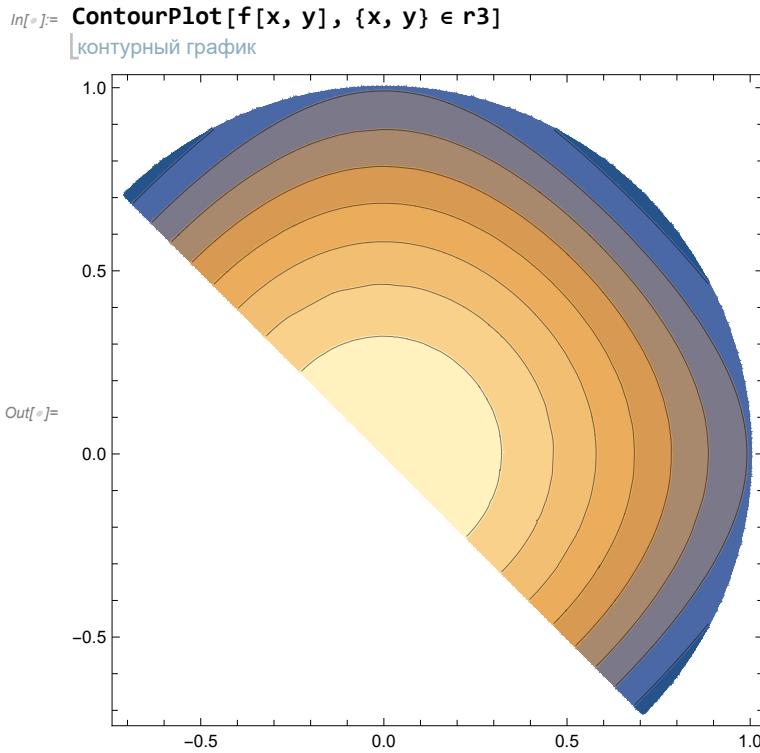
Также есть объединение и разность : RegionUnion, RegionDifference

Визуализация функции над областью:

In[5]:= **Plot3D[f[x, y], {x, y} ∈ r3]**

| график функции 2-х переменных





Наибольшее и наименьшее значение функции в области :

*In[<sub>#</sub>]:=* **NMaximize**[{ $f[x, y]$ , { $x, y$ }  $\in$  r3}, { $x, y$ }]  
 | численная максимизация

*Out[<sub>#</sub>]=*  $\{1., \{x \rightarrow 4.31859 \times 10^{-9}, y \rightarrow 4.31859 \times 10^{-9}\}\}$

*In[<sub>#</sub>]:=* **NMinimize**[{ $f[x, y]$ , { $x, y$ }  $\in$  r3}, { $x, y$ }]  
 | численная минимизация

*Out[<sub>#</sub>]=*  $\{0.155944, \{x \rightarrow 0.707107, y \rightarrow 0.707107\}\}$

(Аналогично есть точные функции **Maximize**, **Minimize**)

## 9.Работа со списками (массивами)

Список (массив) может состоять из элементов любого типа (каждый элемент может быть разного типа)

*In[<sub>#</sub>]:=* **l** = {**a**, **1**, **b**, **2**}  
*Out[<sub>#</sub>]=* {**a**, **1**, **b**, **2**}

Обращение к элементу через **Part** или **[[ ]]**, нумерация начинается с 1:

*In[<sub>#</sub>]:=* **l**[**2**]

*Out[<sub>#</sub>]=* **1**

*In[<sub>#</sub>]:=* **Part**[**l**, **2**]  
 | часть

*Out[<sub>#</sub>]=* **1**

Элементами списка могут быть списки:

In[1]:= **11** = {{1, 2}, {2, 3}}

Out[1]= {{1, 2}, {2, 3}}

Двухуровневая адресация:

In[2]:= **11[[2, 2]]**

Out[2]= 3

Любой многоуровневый список можно всегда сделать одноуровневым :

In[3]:= **12 = 11 // Flatten**

Уплотнить

Out[3]= {1, 2, 2, 3}

**Важная особенность : все функции со списком кроме нескольких не изменяют свой аргумент, т . е . результат выполнения - новый список!**

Для списков перегружены стандартные операторы (они действуют поэлементно) :

In[4]:= **1 + 12**

Out[4]= {1 + a, 3, 2 + b, 5}

In[5]:= **1 + 2**

Out[5]= {2 + a, 3, 2 + b, 4}

In[6]:= **1 \* 12**

Out[6]= {a, 2, 2 b, 6}

Также многие встроенные функции действуют поэлементно к списку :

In[7]:= **Sqrt[1]**

квадратный корень

Out[7]= { $\sqrt{a}$ , 1,  $\sqrt{b}$ ,  $\sqrt{2}$ }

In[8]:= **Sin[1]**

синус

Out[8]= {Sin[a], Sin[1], Sin[b], Sin[2]}

In[9]:= **1^2**

Out[9]= {a<sup>2</sup>, 1, b<sup>2</sup>, 4}

Чтобы применить свою функцию поэлементно, нужно воспользоваться Map или короткой его формой /@:

In[10]:= **Clear[g]**

очистить

In[11]:= **Map[g, 1]**

преобразовать

Out[11]= {g[a], g[1], g[b], g[2]}

In[12]:= **g /@ 1**

Out[12]= {g[a], g[1], g[b], g[2]}

Чтобы применить функцию сразу ко всем элементам есть Apply или короткая его форма @@ :

```
In[]:= Apply[Plus, l]
└ прим... └ сложить
```

```
Out[]= 3 + a + b
```

```
In[]:= Plus @@ l
└ сложить
```

```
Out[]= 3 + a + b
```

```
In[]:= Plus[l, 12]
└ сложить
```

```
Out[]= {1 + a, 3, 2 + b, 5}
```

Функция Plus - полная форма оператора +

Длина списка:

```
In[]:= Length[l]
└ длина
```

```
Out[]= 6
```

```
In[]:= l
```

```
Out[]= {-11, a, 1, b, 2, 11}
```

```
In[]:= l1
```

```
Out[]= {{1, 2}, {2, 3}}
```

Размерность многоуровневого списка:

```
In[]:= Dimensions[l1]
└ размеры массива
```

```
Out[]= {2, 2}
```

## 10. Преобразование списков:

Функций преобразований списков довольно много, вот некоторые из них

```
In[]:= l
```

```
Out[]= {a, 1, b, 2}
```

Вставить элемент на указанную позицию:

```
In[]:= Insert[l, q, 3]
└ вписать
```

```
Out[]= {a, 1, q, b, 2}
```

**Важная особенность : все функции со списком кроме нескольких не изменяют свой аргумент, т . е . результат выполнения - новый список!**

Удалить элемент с указанной позиции:

```
In[]:= Delete[1, 3]
└удалить элемент
Out[]= {a, 1, 2}
```

Вставить элемент на указанные позиции:

```
In[]:= Insert[1, q, {{3}, {5}}]
└вписать
Out[]= {a, 1, q, b, 2, q}
```

Добавить элемент в конец списка:

```
In[]:= Append[1, 0]
└добавить в конец
Out[]= {a, 1, b, 2, 0}
```

Добавить элемент в начало списка:

```
In[]:= Prepend[1, 9]
└добавить в начало
Out[]= {9, a, 1, b, 2}
```

**Есть две функции аналогичного действия, но изменяющие свой аргумент!**

```
In[]:= AppendTo[1, 11]
└добавить в конец к
Out[]= {a, 1, b, 2, 11}
```

```
In[]:= 1
Out[]= {a, 1, b, 2, 11}
```

```
In[]:= PrependTo[1, -11]
└добавить в начало к
Out[]= {-11, a, 1, b, 2, 11}
```

```
In[]:= 1
Out[]= {-11, a, 1, b, 2, 11}
```

Получить первый элемент списка:

```
In[]:= First[1]
└первый
Out[]= -11
```

Получить последний элемент списка:

```
In[]:= Last[1]
└последний
Out[]= 11
```

Получить список без первого элемента:

```
In[]:= Rest[1]
└остаток
Out[]= {a, 1, b, 2, 11}
```

Получить список без последнего элемента:

*In[<sub>#</sub>]:= **Most**[1]  
└ большинство*

*Out[<sub>#</sub>]= { -11, a, 1, b, 2 }*

Выбрать элементы списка с ... по ... :

*In[<sub>#</sub>]:= **Take**[1, {2, 4}]  
└ извлечь*

*Out[<sub>#</sub>]= {a, 1, b}*

Короткая форма Take :

*In[<sub>#</sub>]:= **1[[2;;4]]**  
Out[<sub>#</sub>]= {a, 1, b}*

Отбросить элементы списка с ... по ... :

*In[<sub>#</sub>]:= **Drop**[1, {2, 4}]  
└ отбросить*

*Out[<sub>#</sub>]= { -11, 2, 11 }*

Сделать два списка : первый - выбрать элементы с ... по ..., во втором - список без них :

*In[<sub>#</sub>]:= **TakeDrop**[1, {2, 4}]  
└ извлечь и отбросить*

*Out[<sub>#</sub>]= {{a, 1, b}, {-11, 2, 11}}*

Несколько раз применить функцию к списку :

*In[<sub>#</sub>]:= **Nest**[g, 1, 3]  
└ итерировать*

*Out[<sub>#</sub>]= g[g[g[{ -11, a, 1, b, 2, 11}]]]*

Пример :

*In[<sub>#</sub>]:= **Nest**[First, 11, 2]  
└ ите... └ первый*

*Out[<sub>#</sub>]= 1*

Циклические перестановки элемента списка :

На один влево :

*In[<sub>#</sub>]:= **RotateLeft**[1, 1]  
└ циклически сдвинуть влево*

*Out[<sub>#</sub>]= {a, 1, b, 2, 11, -11}*

На один вправо :

*In[<sub>#</sub>]:= **RotateRight**[1, 1]  
└ циклически сдвинуть вправо*

*Out[<sub>#</sub>]= {11, -11, a, 1, b, 2}*

В обратном порядке

```
In[1]:= Reverse@1
          | расположить в обратном порядке
Out[1]= {11, 2, b, 1, a, -11}
```

Вставить элемент в список через один :

```
In[2]:= Riffle[l, t]
          |перемежать
Out[2]= {-11, t, a, t, 1, t, b, t, 2, t, 11}
```

## Выбор элементов списка по заданному правилу:

Функция - индикатор отрицательного аргумента:

```
In[3]:= f[x_]:= If[x < 0, True, False]
          |условный... |ист... |ложь
```

```
In[4]:= f[1]
Out[4]= False
```

```
In[5]:= f[-1]
Out[5]= True
```

```
In[6]:= l
Out[6]= {-11, a, 1, b, 2, 11}
```

Выбор отрицательных элементов списка:

```
In[7]:= Select[l, f]
          |выбрать
Out[7]= {-11}
```

Второй аргумент Select - любая булева функция! Отбираются элементы, на которых она true.

Можно не определять свою функцию, а воспользоваться чистыми функциями - функциями без имени :

```
In[8]:= 2 # + 5 &[1]
Out[8]= 7
```

& означает что слева стоит **чистая функция** - т.е. что решетка - это аргумент, в этом примере вместо # подставляется 1.

Тогда в Select для отбора отрицательных аргументов можно записать такую чистую функцию :

```
In[9]:= Select[l, # < 0 &]
          |выбрать
Out[9]= {-11}
```

У чистой функции может быть несколько аргументов :

```
In[10]:= (#1^2 + 2 #2) &[a, b]
Out[10]= a^2 + 2 b
```

```
In[1]:= (#1^2 + 2 #2) & [2, 2]
```

```
Out[1]= 8
```

```
In[2]:= 10^&[3]
```

```
Out[2]= 1000
```

Её также можно применять к списку:

```
In[3]:= 10^& /@ Range[5]
          |диапазон
```

```
Out[3]= {10, 100, 1000, 10000, 100000}
```

Создание списка от 1 до 5:

```
In[4]:= Range[5]
          |диапазон
```

```
Out[4]= {1, 2, 3, 4, 5}
```

Смысл чистых функций - не занимать имя (т. е. она чистая от имени).

Также можно использовать шаблонные проверки. MatchQ - булева функция, выдающая true если Аргумент сходится с аргументом-шаблоном:

```
In[5]:= MatchQ[1, _Integer]
          |сходиться :: |целое число
```

```
Out[5]= True
```

```
In[6]:= 1 // MatchQ[_Integer]
          |сходи... |целое число
```

```
Out[6]= True
```

В Select:

```
In[7]:= Select[1, MatchQ[___?Negative]]
          |выбрать |сходиться с... |отрицательны
```

```
Out[7]= {-11}
```

$\_\_$  - означает что количество аргументов может быть любым:

```
In[8]:= y[x_Symbol, z__Integer] := Apply[Times, x^{z}]
          |символ |целое число |прим... |умножить
```

```
In[9]:= y[2, 2]
```

```
Out[9]= y[2, 2]
```

```
In[10]:= y[x, 2]
```

```
Out[10]= x^2
```

```
In[11]:= y[x_Symbol, z__Integer] := Apply[Times, x^{z}]
```

```
          |символ |целое число |прим... |умножить
```

In[12]:= **y1[x, 2]**

Out[12]=  $x^2$

In[13]:= **y1[x, 2, 3]**

Out[13]=  $y1[x, 2, 3]$

? - оператор теста шаблона

Отложенные правила (формируется в момент обращения, также как и отложенное присваивание):

In[14]:= **1**

Out[14]=  $\{-11, a, 1, b, 2, 11\}$

Заменить отрицательные элементы на 0

In[15]:= **1 /. \_?Negative :> 0**  
          └отрицательный

Out[15]=  $\{0, a, 1, b, 2, 11\}$

Оператор в правиле :>

В шаблон можно подставлять свои функции:

**1 /. \_?f :> o**

Out[16]=  $\{o, a, 1, b, 2, 11\}$

Минимальный элемент списка:

In[17]:= **Min[1]**  
          └минимум

Out[17]=  $\text{Min}[-11, a, b]$

Максимальный элемент списка :

In[18]:= **Max[1]**  
          └максимум

Out[18]=  $\text{Max}[11, a, b]$

Нет ответа, т . к . не заданы a и b

Если все задано, то будет ответ :

In[19]:= **Max@Range[5]**  
          └ма... диапазон

Out[19]=  $5$

In[20]:= **Min[Range[5]]**  
          └м... диапазон

Out[20]=  $1$

In[21]:= **Range[5]**  
          └диапазон

Out[21]=  $\{1, 2, 3, 4, 5\}$

Сортировка:

```
In[]:= Sort[Range[5], Greater]
 $\lfloor \text{сор...} \rfloor \text{диапазон} \quad \lfloor \text{больше чем}$ 
Out[]= {5, 4, 3, 2, 1}
```

Генерация случайных списков

Случайный целочисленный список длины 15, элементы которого числа из [-5, 5]:

```
In[]:= RandomInteger[{-5, 5}, 15]
 $\lfloor \text{случайное целое число}$ 
Out[]= {-4, 5, 2, -1, 1, -4, -3, -3, -1, -1, 3, 0, -4, 3, -5}
```

Случайный вещественный список длины 15, элементы которого числа из [-5, 5] :

```
In[]:= RandomReal[{-5, 5}, 15]
 $\lfloor \text{случайное действительное число}$ 
Out[]= {-0.845133, 3.98779, -4.75078, -4.84664, 4.474, -1.94393, -1.00236,
3.4272, -4.80166, -3.3421, 0.847727, 3.04724, -2.39571, 2.99834, -1.84234}
```

Случайный вещественный двухуровневый список из 15 списков длины 3, элементы которого числа из [-5, 5] :

```
In[]:= RandomReal[{-5, 5}, {15, 3}]
 $\lfloor \text{случайное действительное число}$ 
Out[=] {{2.45555, -2.1314, 2.68937}, {2.87132, 0.627141, 2.47854}, {2.05108, 1.80451, 0.66238},
{-2.58587, 3.67536, 4.21875}, {-0.129723, 2.30309, -2.08072},
{1.57078, -0.198541, 4.99043}, {3.47008, -0.264765, 4.99856},
{-0.80651, 4.438, -0.0194638}, {-4.19961, -3.40247, -3.44468},
{0.581757, 2.62817, -1.63377}, {2.13278, 3.48602, 4.48554},
{4.36705, -0.910173, 3.90761}, {1.86652, -4.35954, -0.311645},
{4.0117, 3.81683, 1.01794}, {3.65207, -4.91947, -1.35034}}
```

```
In[]:= 1
Out[=] {-11, a, 1, b, 2, 11}
```

Поиск индекса заданного элемента в списке:

```
In[]:= Position[1, a]
 $\lfloor \text{позиция по образцу}$ 
Out[=] {{2}}
```