0.

Для занятий необходим пакет Wolfram Mathematica . Бесплатная облачная версия : https : // www . wolframcloud . com/ (необходимо зарегестрироваться) .

Для просмотра файлов пакета можно воспользоваться бесплатной программой Wolfram Player : https : // www . wolfram . com/player/

1.Введение

Основным элементом интерфейса пакета Wolfram Mathemaica является блокнот (Notebook). Для выполнения команды необходимо ее напечатать, и после этого нажать Shift+Enter или в меню Evaluation выбрать Evaluate Cells.

```
In[1]:= 2 + 2
Out[1]= 4
```

После у ячейки, в которой вы написали команду, появится “In[...]” с номером, а ниже появится ячейка “Out[...]” с результатом вычисления. Отметим что ячейки в файле можно выполнять в любом порядке.
Отметим особенность пакета: умножение можно выполнять через классическую *, но также пробел между двумя цифрами воспринимается пакетом как умножение (при этом серый крестик появится автоматически):

```
In[2]:= 2 * 3
Out[2]= 6
```

```
In[3]:= 2 × 3
Out[3]= 6
```

Последний полученный результат помещается в специальную переменную - %

```
In[4]:= %
Out[4]= 6
```

Основной любого языка являются переменные. В Wolfram Mathematica переменные могут быть абсолютно любого типа (хоть “картинкой”), а типизация происходит автоматически. Для задания переменной a необходимо выполнить команду:

```
In[5]:= a = 2
Out[5]= 2
```

Для вывода значения переменной на экран нужно выполнить ячейку с именем переменной:

```
In[6]:= a
Out[6]= 2
```

Чтобы очистить переменную нужно выполнить следующую функцию:

In[7]:=
```
Clear[a]
```
очистить

Сразу отметим особенность пакета Wolfram Mathematica : все встроенные функции начинаются с заглавной буквы, а аргумент пишется в квадратных скобках.

Пакет Wolfram Mathematica построен на клиент - серверной архитектуре . Интерфейс отделен от вычислительного ядра Wolfram Kernel. Ядро запускается при выполнении первой команды и остается работать до завершения работы пакета. Номер In в ячейках - порядковый номер обращения к ядру. Все инициализированные переменные глобальны - они доступны из всех открытых файлов пакета! Если необходимо завершить ядро: Evaluation -> Quit Kernel -> Local. Удаление выполненной ячейке в блокноте не удаляет результат действия из памяти ядра!

2. Обычное и отложенное присваивание

Помимо обычного оператора присваивания, в пакете есть оператор отложенного присваивания :=, который выполняется в момент обращения к переменной. Разницу между := и = поможет понять следующий пример:

In[8]:=
```
a = 2;
```

In[9]:=
```
b = a + 2
```
Out[9]=
```
4
```

In[10]:=
```
c := a + 2
```

In[11]:=
```
a = 13
```
Out[11]=
```
13
```

In[12]:=
```
b
```
Out[12]=
```
4
```

In[13]:=
```
c
```
Out[13]=
```
15
```

При отложенном присвоении переменная динамически обновляется. (С точки зрения классического программирования отложенное присваивание - передача по ссылке). Отметим также, что тут использовался оператор подавления вывода результата на экран - ;

3. Символьное и численное вычисления

В пакетах компьютерной алгебры реализовано два типа вычислений: символьное (бесконечно точное) и численное (приближенное). По умолчанию пакет работает в символьном режиме. Выполнение следующей команды:

In[14]:=
```
Sin[5]
```
синус

Out[14]=
```
Sin[5]
```

не дает приближенного выражения sin(5) - он выдает максимально точный ответ.

Чтобы вычислить приближенно, необходимо воспользоваться функцией N[]

In[15]:=
```
N[Sin[5]]
```
синус

Out[15]= −0.958924

У нее есть второй необязательный аргумент - количество выводимых цифр на экран:

In[16]:=
```
N[Sin[5], 40]
```
синус

Out[16]= −0.9589242746631384688931544061559939733525

Также, если пакет встречает хотя бы одно вещественное число в выражении, то все вычисления автоматически становятся приближенными:

In[17]:=
```
Sin[5.0]
```
синус

Out[17]= −0.958924

Основные математические функции: tg(x) - Tan[x], ctg(x) - Cot[x], arcsin(x) - ArcSin[x],..., ln(x) - Log[x], e^x – Exp[x]

In[18]:=
```
Exp[4.]
```
показательная функция

Out[18]= 54.5982

Число e:

In[19]:=
```
N[E]
```
основание натурального логарифма

Out[19]= 2.71828

Внимание : помимо E в пакете есть другие зарезервированные переменные, и они также поэтому имена собственных переменных лучше начинать с малой буквы:

Также число e можно записать с помощью встроенного символа с помощью комбинации клавиш : Esc + e + e + Esc

In[20]:=
```
N[ⅇ]
```
численное приближение

Out[20]= 2.71828

Число π

In[21]:=
```
Pi
```
число пи

Out[21]= π

In[22]:=
```
N[Pi]
```
число пи

Out[22]= 3.14159

Любую греческую букву можно записать путем комбинации клавиш Esc + название буквы + Esc . На примере числа π: Esc + p + i + Esc

In[23]:= π

Out[23]= π

4. Постфиксная и префиксная запись функций

Помимо классического вызова функции вида имя_функции[аргумент], в пакете есть постфиксная форма вызова функции:

In[24]:= Sin[5] // N
синус численное приближение

Out[24]= -0.958924

и префиксная:

In[25]:= N@Sin[5]
··· синус

Out[25]= -0.958924

In[26]:= π // N
численное приближение

Out[26]= 3.14159

5. Определение собственных функций

Конструкция определения следующая: имя_функции[переменная_]:=выражение. Например, f(x)=2ln(2x):

In[27]:= f[x_] := 2 Log[2 x]
натуральн

Оператор x_ - это шаблон. С точки зрения программирования, тут генерируется локальная переменная с уникальным именем.

In[28]:= f[2.]

Out[28]= 2.77259

In[29]:= f[t]

Out[29]= 2 Log[2 t]

Функция двух переменных определяется аналогично:

In[30]:= f[x_, y_] := 2 y Log[2 x] // N
натуральны··· числе

In[31]:= f[2, 3]

Out[31]= 8.31777

Пакет поддерживает перегрузку функций .

6. Булевы операторы:

In[32]:=

In[33]:=
```
f[2] > f[3]
```
Out[33]=
```
False
```

In[34]:=
```
f[2] == f[3]
```
Out[34]=
```
False
```

In[35]:=
```
f[2] ≠ f[3]
```
Out[35]=
```
True
```

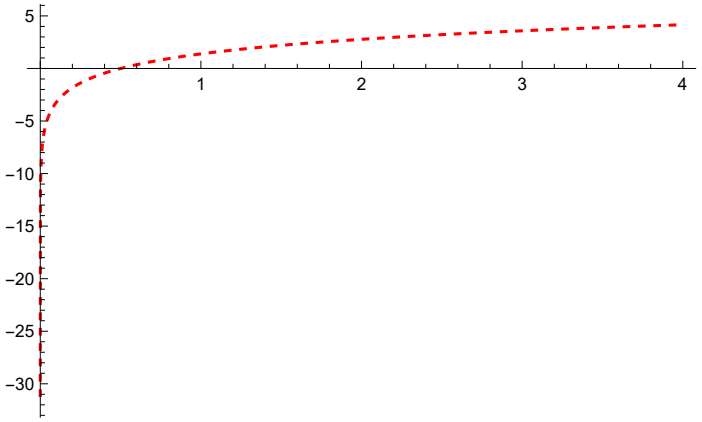
In[36]:=
```
f[2] == f[2.]
```
Out[36]=
```
True
```

7. Построение графиков функций

Для построения графика функции f(x) на отрезке x∈ [a,b] используется функция Plot[f[x],{x,a,b}]:

In[37]:=
```
h1 = Plot[f[x], {x, 0, 4}, PlotStyle → {Red, Dashed}, PlotRange → All]
```
график функции стиль графика кр··· штрихов··· отображаем··· всё

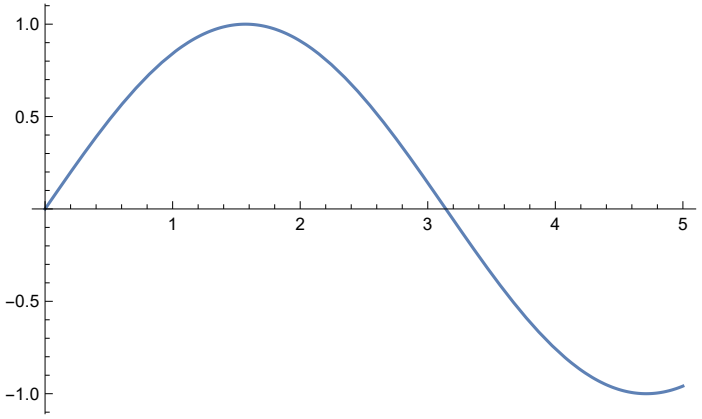
Out[37]=



В функции Plot огромное число опций - смотрите Help!

In[38]:=
```
h2 = Plot[Sin[x], {x, 0, 5}]
```
гр··· синус

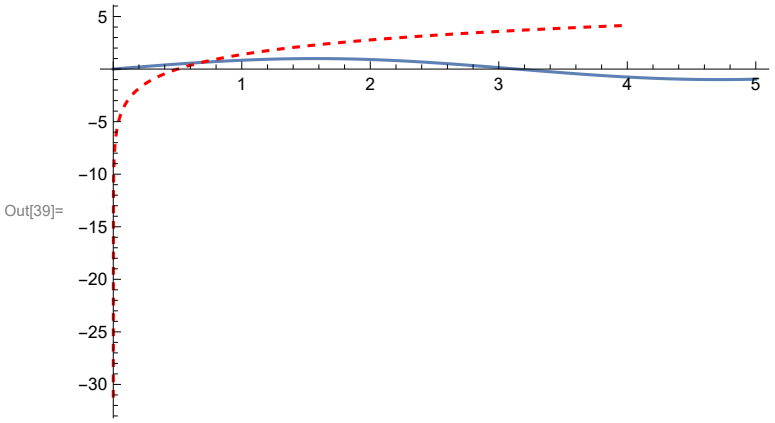
Out[38]=



Результат Plot можно присваивать в переменные и объединять на одном графике

In[39]:=
```
Show[h2, h1, PlotRange → All]
```
показать отображаем··· всё

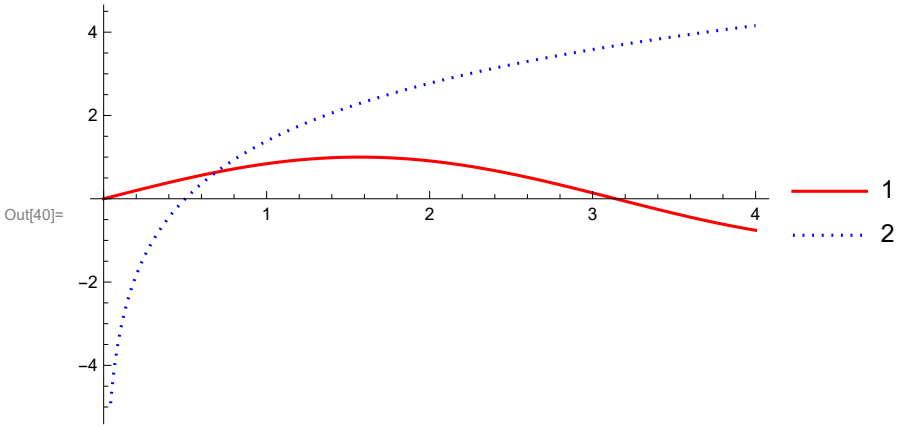
Out[39]=



Другой способ нарисовать два графика на одном - поставить список (массив) из функций в первый аргумент Plot.

In[40]:=
```
Plot[{Sin[x], f[x]}, {x, 0, 4},
 PlotLegends → {1, 2}, PlotStyle → {{Red}, {Blue, Dotted}}]
```
граф··· синус
легенды графика стиль графика красный синий точечный пун

Out[40]=



In[41]:=

8. Визуализация последовательностей

Сначала нужно построить список элементов последовательности. Для вычисления списка значений последовательности a_n при n=1,2,...,b есть функция Table[a_n, {n, 1, b}]. Рассмотрим последовательность $a_n = \frac{1}{n}$ и вычислим список ее значений при n=1,2,...,10:

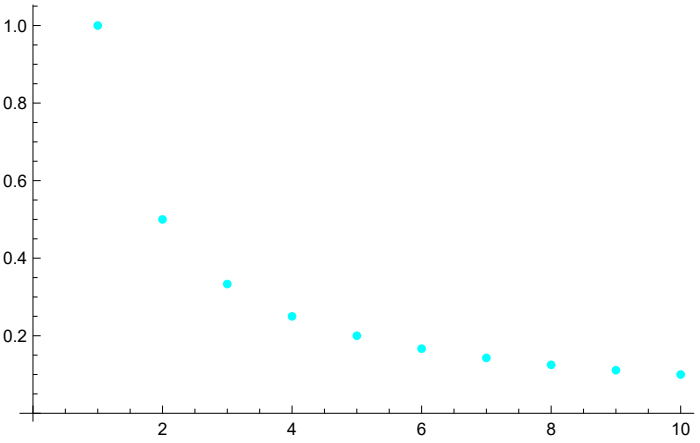
In[42]:=
```
a = Table[1 / n, {n, 1, 10}]
```
таблица значений

Out[42]=
$$\left\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}\right\}$$

Визуализируем полученный список:

In[43]:=
```
h3 = ListPlot[a, PlotStyle → Cyan]
```
диаграмма р··· стиль графика голубо

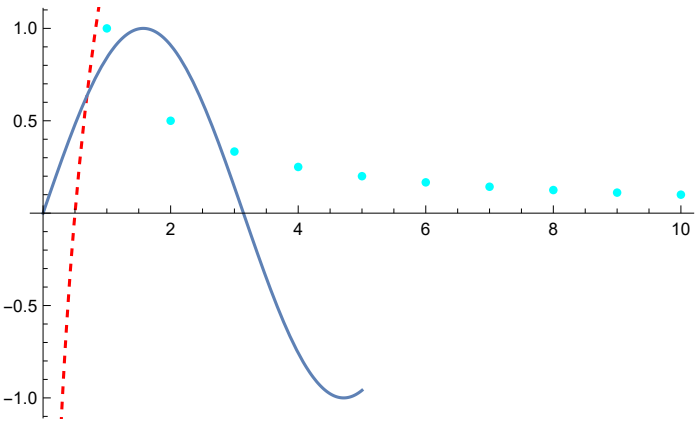
Out[43]=



Отметим, что Show позволяет комбинировать любые графики:

In[44]:=
```
Show[h1, h2, h3, PlotRange → {-1, 1}]
```
показать отображаемый диапазон гр

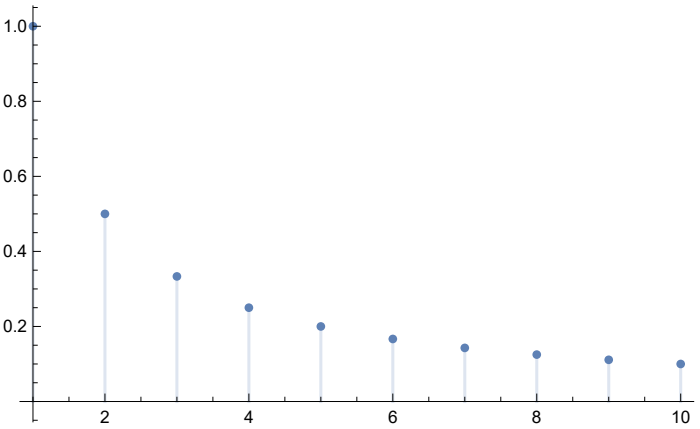
Out[44]=



Также можно визуализировать последовательность без генерации списка значений:

In[45]:=
```
DiscretePlot[1 / n, {n, 1, 10}]
```
график последовательности

Out[45]=



In[46]:=

9. Задание разрывных функций

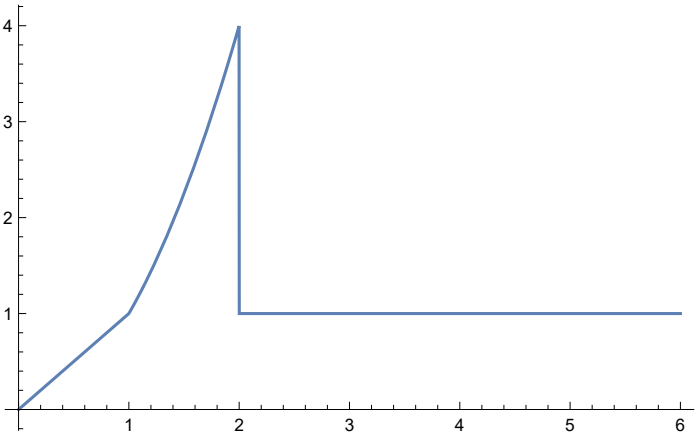
Зададим кусочно - непрерывную функцию : g(x) = $\begin{cases} x & 0 \le x \le 1 \\ x^2 & 1 < x \le 2 \\ 1 & \text{вне этого} \end{cases}$

In[47]:=
```
g[x_] := Piecewise[{{x, 0 ≤ x ≤ 1}, {x^2, 1 < x ≤ 2}}, 1]
```
кусочно–заданная функция

In[48]:=
```
g[x]
```
Out[48]= $\begin{cases} x & 0 \le x \le 1 \\ x^2 & 1 < x \le 2 \\ 1 & \text{True} \end{cases}$

In[49]:=

In[50]:=
```
Plot[g[x], {x, 0, 6}]
```
график функции

Out[50]=


10. Функции математического анализа

Предел:

In[51]:=
```
Limit[1 / x, x → 1]
```
предел

Out[51]= 1

In[52]:=
```
Limit[1 / x, x → 0]
```
предел

Out[52]= Indeterminate

Предел справа:

In[53]:=
```
Limit[1 / x, x → 0, Direction → "FromAbove"]
```
предел направление

Out[53]= ∞

Предел слева:

In[54]:=
```
Limit[1 / x, x → 0, Direction → "FromBelow"]
```
предел направление

Out[54]= −∞

Производная:

In[55]:=
```
D[x^2, x]
```
дифференциировать

Out[55]= 2 x

In[56]:=
```
h[x_] := x^2
```

In[57]:=
```
h'[x]
```

Out[57]= 2 x

Вторая производная:

In[58]:=
```
D[x^2, x, x]
```
дифференциировать

Out[58]= 2

In[59]:=
```
D[x^2, {x, 2}]
```
дифференциировать

Out[59]= 2

Неопределенный интеграл:

In[60]:=
```
Integrate[2 x, x]
```
интегрировать

Out[60]= x^2

Определенный интеграл:

In[61]:=
```
Integrate[2 x, {x, 0, 1}]
```
интегрировать

Out[61]= 1