# Analiza matematyczna 1
# Zajęcia nr 2

## Wykres punktowy

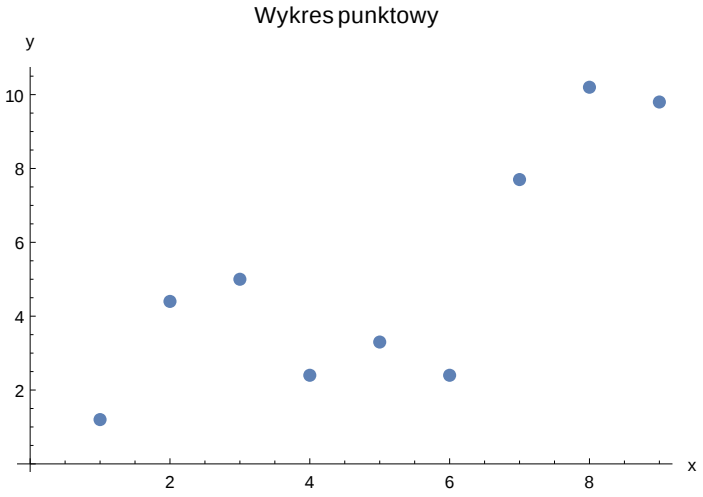
```
? ListPlot
```

ListPlot[{$y_1$, $y_2$, …}] plots points corresponding
to a list of values, assumed to correspond to $x$ coordinates 1, 2, … .
ListPlot[{{$x_1$, $y_1$}, {$x_2$, $y_2$}, …}] plots a list of points with specified $x$ and $y$ coordinates.
ListPlot[{$list_1$, $list_2$, …}] plots several lists of points. »

Lista liczb:

```
punkty := {1.2, 4.4, 5., 2.4, 3.3, 2.4, 7.7, 10.2, 9.8}
```

```
ListPlot[punkty, PlotStyle -> PointSize[0.02],
 PlotLabel → "Wykres punktowy", AxesLabel → {"x", "y"}]
```

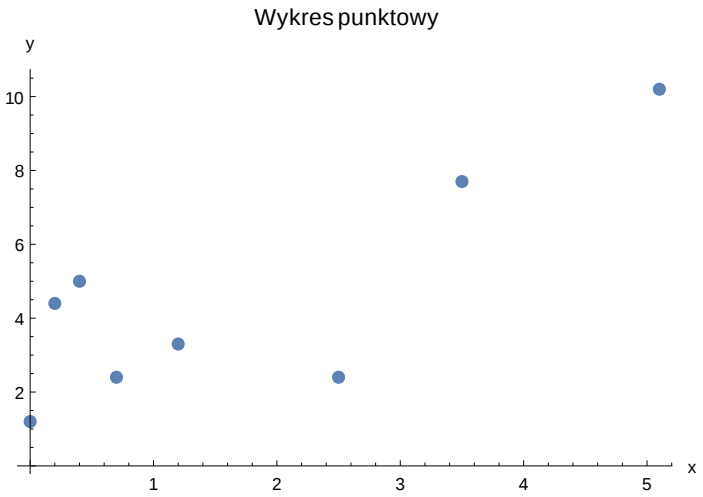



Argumenty to kolejne cyfry 1, 2, 3, ...

Aby podać wartości x-ów i y-ów trzeba stworzyc tablicę par liczb:

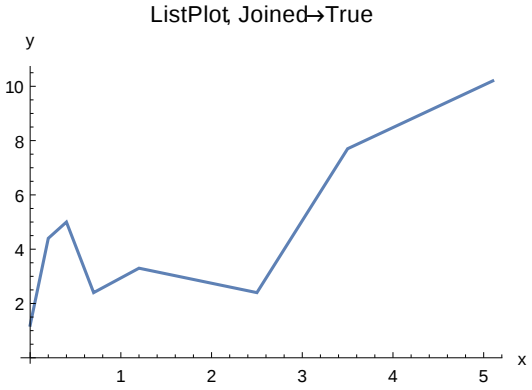
```
wspolrzedne := {{0, 1.2}, {0.2, 4.4}, {0.4, 5.},
  {0.7, 2.4}, {1.2, 3.3}, {2.5, 2.4}, {3.5, 7.7}, {5.1, 10.2}}
```

```
ListPlot[wspolrzedne, PlotStyle -> PointSize[0.02],
 PlotLabel → "Wykres punktowy", AxesLabel → {"x", "y"}]
```

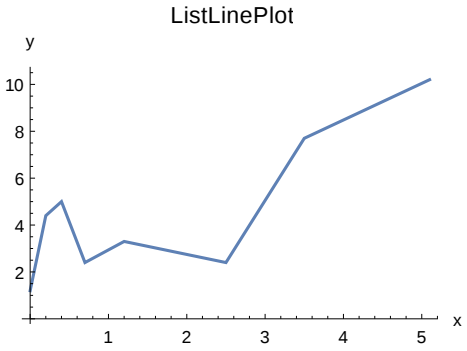



Punkty połączone prostymi

```
ListPlot[wspolrzedne, PlotStyle -> PointSize[0.02],
 PlotLabel → "ListPlot, Joined→True", AxesLabel → {"x", "y"}, Joined → True
]
```



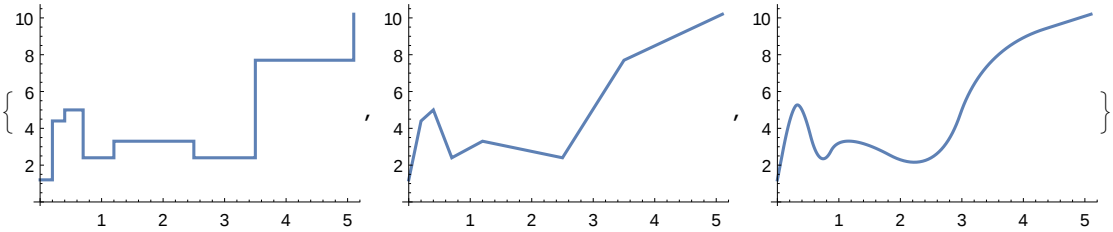

```
ListLinePlot[wspolrzedne,
 PlotLabel → "ListLinePlot", AxesLabel → {"x", "y"}]
```



```
? InterpolationOrder
```

InterpolationOrder is an option for Interpolation, as well as ListLinePlot, ListPlot3D, ListContourPlot, and related functions, that specifies what order of interpolation to use. >>

```
{ListLinePlot[wspolrzedne, InterpolationOrder → 0],
 ListLinePlot[wspolrzedne, InterpolationOrder → 1],
 ListLinePlot[wspolrzedne, InterpolationOrder → 2]}
```



## Krzywa regresji

```
? LinearModelFit
```

LinearModelFit[{$y_1$, $y_2$, ...}, {$f_1$, $f_2$, ...}, $x$] constructs a linear model of the form $\beta_0 + \beta_1 f_1 + \beta_2 f_2 + \ldots$ that fits the $y_i$ for successive $x$ values 1, 2, ....
LinearModelFit[{{$x_{11}$, $x_{12}$, ..., $y_1$}, {$x_{21}$, $x_{22}$, ..., $y_2$}, ...}, {$f_1$, $f_2$, ...}, {$x_1$, $x_2$, ...}] constructs a linear model of the form $\beta_0 + \beta_1 f_1 + \beta_2 f_2 + \ldots$ where the $f_i$ depend on the variables $x_k$.
LinearModelFit[{$m$, $v$}] constructs a linear model from the design matrix $m$ and response vector $v$. >>

LinearModelFit[{$y_1$,$y_2$,...},{$f_1$,$f_2$,...},x] znajduje parametry dopasowania funkcji $\beta_o + \beta_1 f_1 + \beta_2 f_2 + \ldots$ do danych $y_i$

Chcemy znaleźć parametry prostej regresji postaci $\beta_o + \beta_1$ x

```
LMF = LinearModelFit[wspolrzedne, x , x]
```

FittedModel[ 2.31484 + 1.32951 x ]

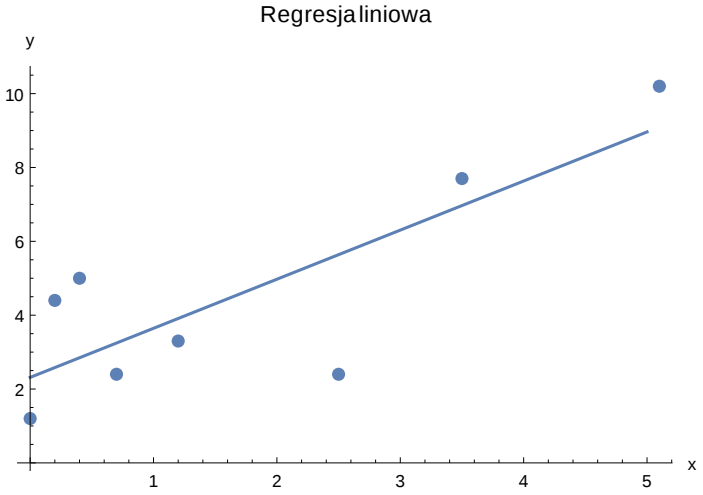
```
LMF[x]
```

2.31484 + 1.32951 x

```
LMF["ParameterTable"]
```

| | Estimate | Standard Error | t-Statistic | P-Value |
|---|---|---|---|---|
| 1 | 2.31484 | 0.964577 | 2.39985 | 0.0533059 |
| x | 1.32951 | 0.399487 | 3.32804 | 0.0158453 |

```
Show[
 ListPlot[wspolrzedne, PlotStyle -> PointSize[0.02]],
 Plot[LMF[x], {x, 0, 5}],
 PlotLabel → "Regresja liniowa", AxesLabel → {"x", "y"}
]
```




```
? NonlinearModelFit
```

NonlinearModelFit[{$y_1$, $y_2$, …}, *form* , {$\beta_1$, …}, $x$] constructs a nonlinear model with structure *form* that fits the $y_i$ for successive $x$ values 1, 2, … using the parameters $\beta_1$, … .
NonlinearModelFit[{{$x_{11}$, $x_{12}$, …, $y_1$}, {$x_{21}$, $x_{22}$, …, $y_2$}, …}, *form* , {$\beta_1$, …}, {$x_1$, …}] constructs a nonlinear model where *form* depends on the variables $x_k$.
NonlinearModelFit[*data* , {*form* , *cons* }, {$\beta_1$, …}, {$x_1$, …}] constructs a nonlinear model subject to the parameter constraints *cons* . >>

NonlinearModelFit[{$y_1$,$y_2$,...},*form*,{$\beta_1$,...},x] znajduje parametry dopasowania funkcji *form* zaleznej od parametrow { $\beta_1$ ,$\beta_2$ ,...} do danych $y_i$

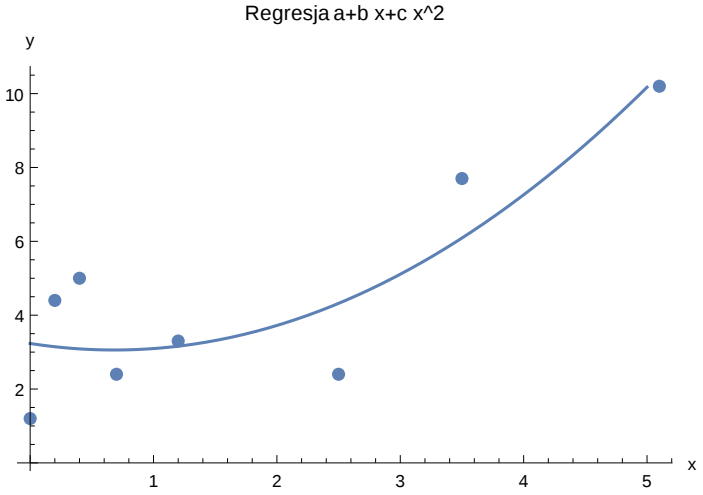
```
NMF = NonlinearModelFit[wspolrzedne, a + b x + c x^2, {a, b, c}, x]
```

FittedModel[ $3.23429 - 0.519599\,x + 0.38148\,x^2$ ]

```
NMF[x]
```

$3.23429 - 0.519599\,x + 0.38148\,x^2$

```
Show[
 ListPlot[wspolrzedne, PlotStyle -> PointSize[0.02]],
 Plot[NMF[x], {x, 0, 5}],
 PlotLabel → "Regresja a+b x+c x^2", AxesLabel → {"x", "y"}
]
```




```
NMF["ParameterTable"]
```

| | Estimate | Standard Error | t- Statistic | P- Value |
|---|---|---|---|---|
| a | 3.23429 | 1.11168 | 2.90938 | 0.0334239 |
| b | −0.519599 | 1.37428 | −0.378087 | 0.720886 |
| c | 0.38148 | 0.272989 | 1.39742 | 0.221132 |

# Słupkibłędów

Trzeba zaladowac pakiet ErrorBarPlots`

```
Needs["ErrorBarPlots`"]
```

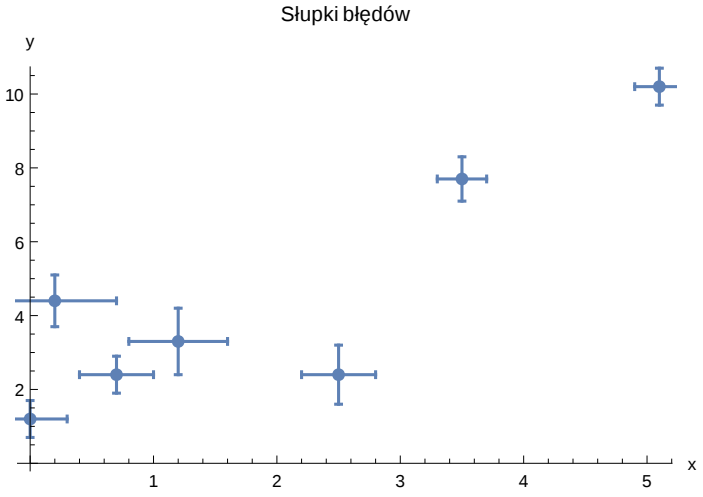
```
? ErrorBar
```

ErrorBar[{ *negerror* , *poserror* }] represents error in the positive and negative directions.
ErrorBar[*yerr*] represents error *yerr* in both the positive and negative directions.
ErrorBar[*xerr* , *yerr*] represents errors specified for both the *x* and the *y* coordinates. >>

Wspolrzedne z slupkami bledow

```
wspolrzedneError := {
  {{0, 1.2}, ErrorBar[0.3, 0.5]},
  {{0.2, 4.4}, ErrorBar[0.5, 0.7]},
  {{0.7, 2.4}, ErrorBar[0.3, 0.5]},
  {{1.2, 3.3}, ErrorBar[0.4, 0.9]},
  {{2.5, 2.4}, ErrorBar[0.3, 0.8]},
  {{3.5, 7.7}, ErrorBar[0.2, 0.6]},
  {{5.1, 10.2}, ErrorBar[0.2, 0.5]}}
```

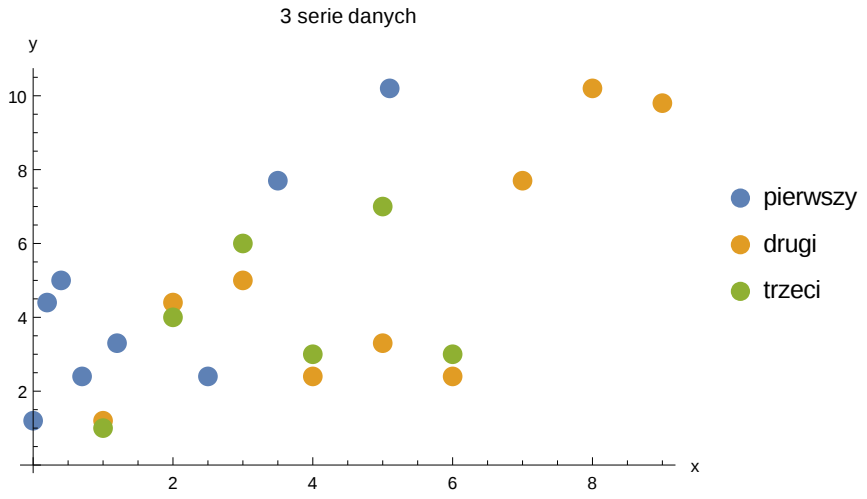
```
ErrorListPlot[wspolrzedneError,
 PlotLabel → "Słupki błędów", AxesLabel → {"x", "y"}]
```



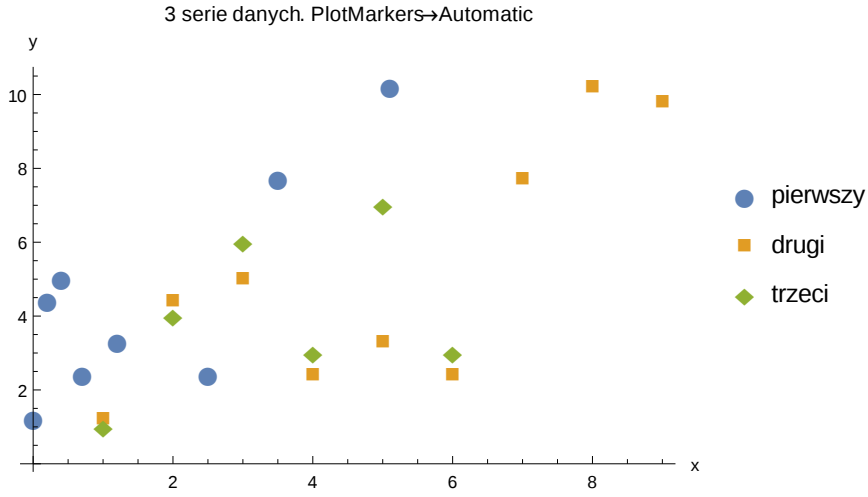

## PlotMarkers

Znaczniki danych

```
ListPlot[{wspolrzedne, punkty, {1, 4, 6, 3, 7, 3}},
 PlotLabel → "3 serie danych", AxesLabel → {"x", "y"},
 PlotLegends → {"pierwszy", "drugi", "trzeci"},
 PlotStyle -> PointSize[0.03]]
```




```
ListPlot[{wspolrzedne, punkty, {1, 4, 6, 3, 7, 3}},
 PlotLabel → "3 serie danych. PlotMarkers→Automatic",
 AxesLabel → {"x", "y"}, PlotLegends → {"pierwszy", "drugi", "trzeci"},
 PlotMarkers → {Automatic, 15}]
```




Własny wybór znaczniów danych

▲ \[ FilledUpTriangle ] (bez spacji)
■ \[ FilledSquare ]
● \[ FilledCircle ]

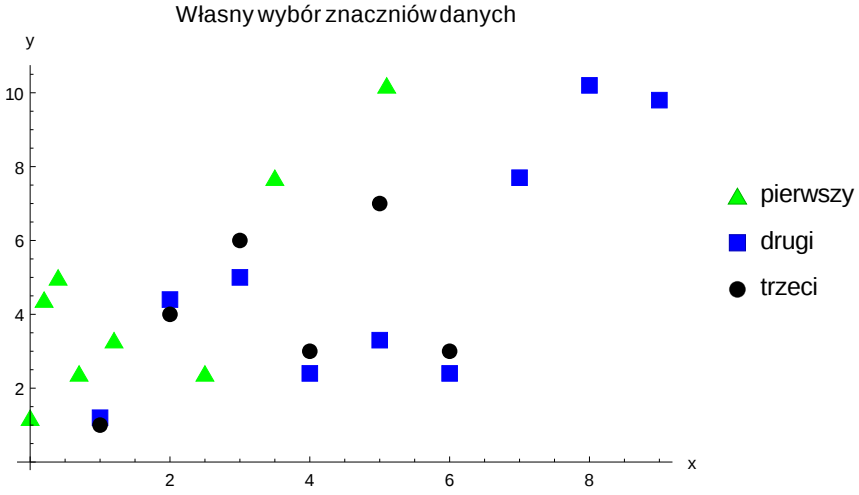
```
ListPlot[{wspolrzedne, punkty, {1, 4, 6, 3, 7, 3}},
 PlotLegends → {"pierwszy", "drugi", "trzeci"},
 PlotLabel → "Własny wybór znaczniów danych", AxesLabel → {"x", "y"},
 PlotMarkers → {"▲", "■", "●"},
 PlotStyle → {Green, Blue, Black}
]
```


Własny wybór znaczniów danych - Graphics

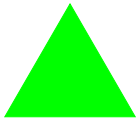
```
? Graphics
```

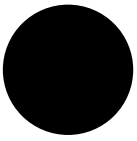
Graphics[*primitives* , *options* ] represents a two-dimensional graphical image. >>

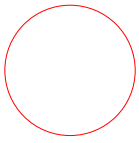
```
ListPlot[{wspolrzedne, punkty, {1, 4, 6, 3, 7, 3}},
 PlotLegends → {"pierwszy", "drugi", "trzeci"},
 PlotLabel → "Własny wybór znaczniów danych", AxesLabel → {"x", "y"},
 PlotMarkers → {
   {Graphics[{Green, Polygon[{{1, 0}, {0, Sqrt[3]}, {-1, 0}}]}], 0.04},
   {Graphics[{Blue, Rectangle[]}], 0.04},
   {Graphics[{Black, Disk[]}], 0.04}
  }
]
```



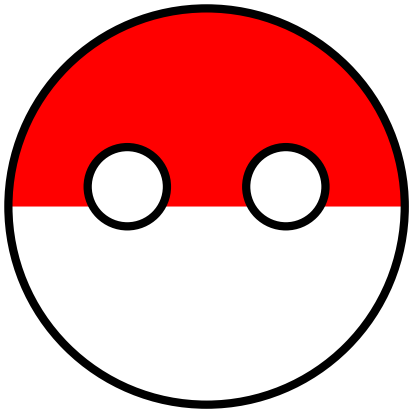

```
trojkat = Graphics[{Green, Polygon[{{1, 0}, {0, Sqrt[3]}, {-1, 0}}]}]
kwadrat = Graphics[{Blue, Rectangle[]}]
kolo = Graphics[{Black, Disk[]}]
okrag = Graphics[{Red, Circle[]}]
```







```
pb = Graphics[{Thickness[0.02],
    Red, Disk[{0, 0}, 1, {0, Pi}],
    Black, Circle[{0, 0}, 1],
    White, Disk[{-0.4, 0.1}, 0.2],
    White, Disk[{0.4, 0.1}, 0.2],
    Black, Circle[{-0.4, 0.1}, 0.2],
    Black, Circle[{0.4, 0.1}, 0.2]
   }]
```



## Generowanie list

```
? Table
```

Table[*expr* , *n*] generates a list of *n* copies of *expr* .
Table[*expr* , {*i*, $i_{max}$ }] generates a list of the values of *expr* when *i* runs from 1 to $i_{max}$ .
Table[*expr* , {*i*, $i_{min}$ , $i_{max}$ }] starts with *i* = $i_{min}$ .
Table[*expr* , {*i*, $i_{min}$ , $i_{max}$ , *di* }] uses steps *di* .
Table[*expr* , {*i*, {$i_1$, $i_2$, …}}] uses the successive values $i_1$, $i_2$, ….
Table[*expr* , {*i*, $i_{min}$ , $i_{max}$ }, {*j*, $j_{min}$ , $j_{max}$ }, …] gives a nested list. The list associated with *i* is outermost. >>

```
Table[n^2, {n, 6}]
```

```
{1, 4, 9, 16, 25, 36}
```

```
Table[n^2, {n, 0, 6}]
```

```
{0, 1, 4, 9, 16, 25, 36}
```

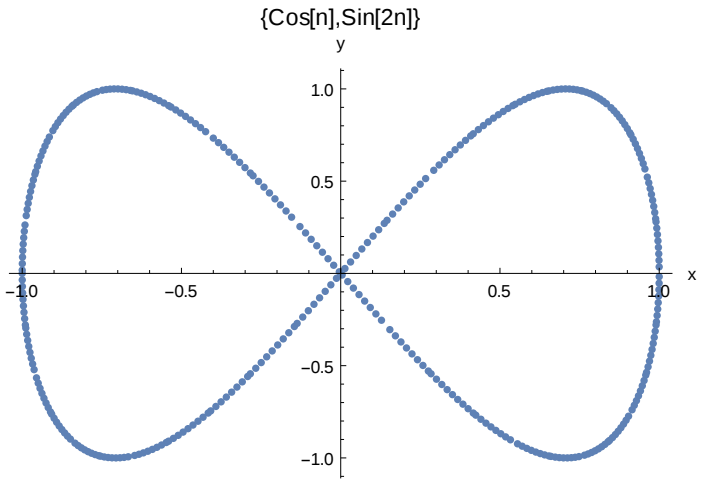
```
Table[n^2, {n, 0, 6, 2}]
```

```
{0, 4, 16, 36}
```

```
Table[{2 n, Sin[n]}, {n, 0, 4}]
```

```
{{0, 0}, {2, Sin[1]}, {4, Sin[2]}, {6, Sin[3]}, {8, Sin[4]}}
```

```
ListPlot[Table[{Cos[n], Sin[2 n]}, {n, 1, 360}],
 PlotLabel → "{Cos[n],Sin[2n]}", AxesLabel → {"x", "y"}]
```




## Zadanie

1. Narysować wykres punktowy dla listy par liczb $\{x_i, y_i\}$ takich że

$x_i$ to cyfry od 0 do 30 co 2

$y_i$ to liczby postaci $\log(3\, x_i + 7) + \text{rand}_i$

gdzie $\text{rand}_i$ to liczba losowa z przedzialu od 0 do 0.5 ( RandomReal[{$y_{\min}$, $y_{\max}$}] )

2. Niech znaczniki to jakiś stworzony przez siebie znaczek przy użyciu Graphics
3. Dodać funkcję regresji logarytmicznej log(a+b x)
4. Dodać tytul i nazwy osi

```
? RandomReal
```

RandomReal[] gives a pseudorandom real number in the range 0 to 1.
RandomReal[{$x_{min}$ , $x_{max}$ }] gives a pseudorandom real number in the range $x_{min}$ to $x_{max}$ .
RandomReal[$x_{max}$ ] gives a pseudorandom real number in the range 0 to $x_{max}$ .
RandomReal[*range* , $n$] gives a list of $n$ pseudorandom reals.
RandomReal[*range* , {$n_1$, $n_2$, …}] gives an $n_1 \times n_2 \times \ldots$ array of pseudorandom reals. »

```
RandomReal[{0, 0.5}]
```

```
0.103774
```