Zajęcia II : Rodzina kwadratowa

Zad. 1

$$x_{n+1} = \frac{y_{n+1}}{M} = \lambda \frac{y_n}{M} \left(1 - \frac{y_n}{M}\right) = \lambda \frac{y_n}{M} - \lambda \left(\frac{y_n}{M}\right)^2$$

$$y_{n+1} = \lambda\, y_n - \lambda y_n \left(\frac{y_n}{M}\right)$$

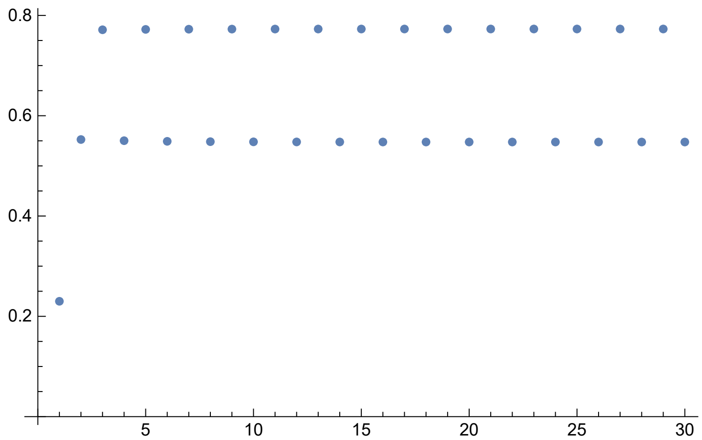
Opisuje populację o licznosci y_n w n – tym kroku,
której przyrost w kazdym kroku jest liniowy ze
stala wzrostu λ oraz uwzględniona jest pojemność środowiska
przez co $\left(\frac{y_n}{M}\right)$ z powstałych bakterii w każdym kroku wymiera
(jest to współczynnik zależny od aktualnego obciażenia środowiska przez bakterie
tzn. im liczniejsza jest populacja bakterii tym bliższy on jest
1 i tym samym więcej bakterii wymiera w danym kroku)

Zad. II

```
In[4]:= f[λ_, x_] := N[λ * x * (1 - x)]

In[2]:= orbita[λ_, x_, n_] := NestList[f[λ, #] &, x, n - 1]

In[3]:= orbita[3.12, 0.23, 30]

Out[3]= {0.23, 0.552552, 0.771383, 0.550215, 0.772133, 0.548945, 0.772526,
  0.548277, 0.772728, 0.547932, 0.772832, 0.547756, 0.772884, 0.547666, 0.772911,
  0.547621, 0.772925, 0.547598, 0.772931, 0.547586, 0.772935, 0.54758, 0.772937,
  0.547577, 0.772938, 0.547576, 0.772938, 0.547575, 0.772938, 0.547575}
```

Zad. III

In[4]:=
```
ListPlot[orbita[3.12, 0.23, 30]]
```

Out[4]=


Zauważamy, że trzeba poprawić bo startujemy z $(1, x_0)$ a powinniśmy z $(0, x_0)$

```
pom = orbita[3.12, 0.23, 30];
Table[{i - 1, pom[[i]]}, {i, 1, Length[pom]}]
```

```
{{0, 0.23}, {1, 0.552552}, {2, 0.771383}, {3, 0.550215}, {4, 0.772133},
 {5, 0.548945}, {6, 0.772526}, {7, 0.548277}, {8, 0.772728}, {9, 0.547932},
 {10, 0.772832}, {11, 0.547756}, {12, 0.772884}, {13, 0.547666}, {14, 0.772911},
 {15, 0.547621}, {16, 0.772925}, {17, 0.547598}, {18, 0.772931}, {19, 0.547586},
 {20, 0.772935}, {21, 0.54758}, {22, 0.772937}, {23, 0.547577}, {24, 0.772938},
 {25, 0.547576}, {26, 0.772938}, {27, 0.547575}, {28, 0.772938}, {29, 0.547575}}
```

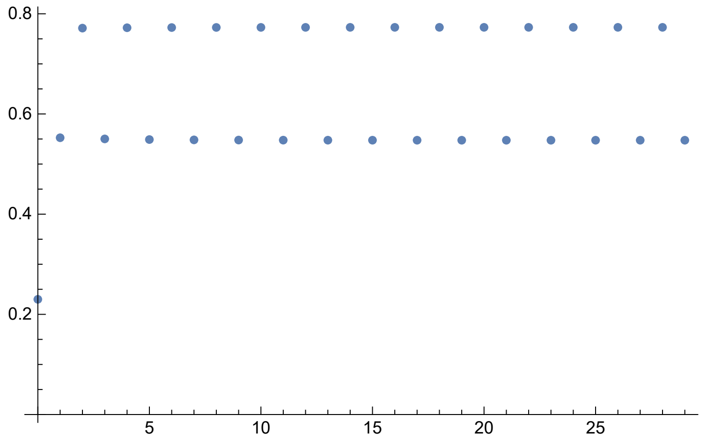
Aby pokazać, że można definiować różne funkcje definiujemy funkcję tworzącą odpowiedniego ListPlota

[[1]] - jest potrzebne aby uzyskać rozwiązanie bez { }. Proszę spróbować wywołać bez tego.

```
wykresorbity[λ_, x_, n_] :=
 {pom = orbita[λ, x, n];
   pom1 = Table[{i - 1, pom[[i]]}, {i, 1, Length[pom]}];
   ListPlot[pom1]
  }[[1]]
```

wywołujemy testowo

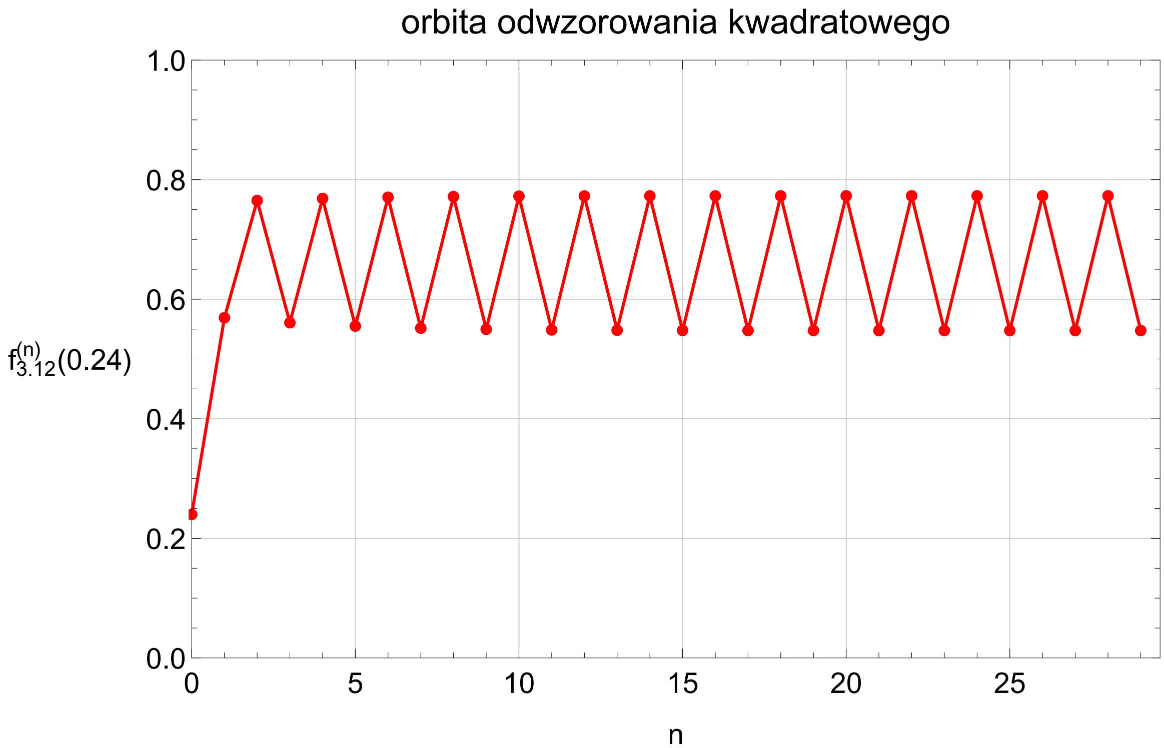
```
wykresorbity[3.12, 0.23, 30]
```



"Upiększmy" nasz wynik

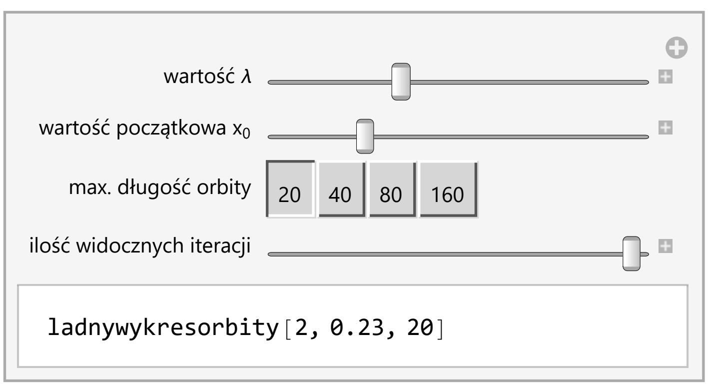
In[5]:=
```
ladnywykresorbity[λ_, x_, n_] :=
 {pom = orbita[λ, x, n];
   pom1 = Table[{i - 1, pom[[i]]}, {i, 1, Length[pom]}];
   ListPlot[pom1,
    PlotRange → {{0, All}, {0, 1}},
    Joined → True,
    Mesh → Full,
    PlotStyle → {Red, PointSize[Medium]},
    LabelStyle → Directive[15, Black],
    Frame → True,
    RotateLabel → False,
    FrameLabel → {"n",
      ToString[Subsuperscript["f", λ, "(n)"], StandardForm] <> "(" <> ToString[x] <> ")"},
    PlotLabel → "orbita odwzorowania kwadratowego",
    GridLines -> Automatic,
    ImageSize → 600
   ]
  }[[1]]
```

```
ladnywykresorbity[3.12, 0.24, 30]
```



Zad. IV

```
Manipulate[ladnywykresorbity[λ, x, Min[n, k]],
 {{λ, 2, "wartość λ"}, 1, 4},
 {{x, 0.23, "wartość początkowa x₀"}, 0, 1},
 {{n, 20, "max. długość orbity"}, {20, 40, 80, 160}},
 {{k, 20, "ilość widocznych iteracji"}, 1, n, 1}
]
```



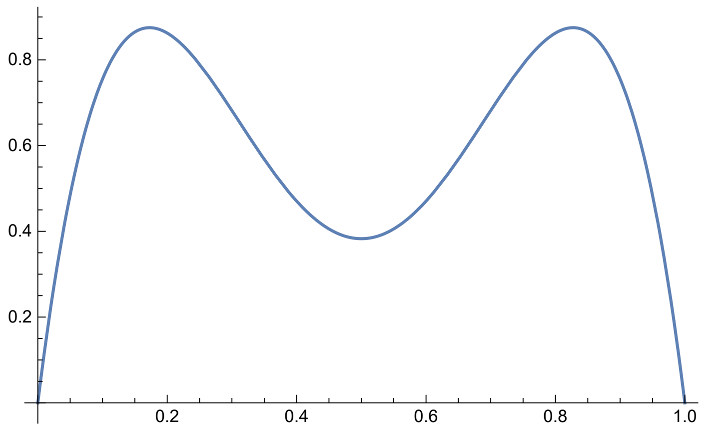
Zad. V

Definiujemy funkcję zwracającą n - ta iteracja odwzorowania logistycznego

In[1]:=
```
ntaiteracja[λ_, x_, n_] := Nest[f[λ, #] &, x, n]
```

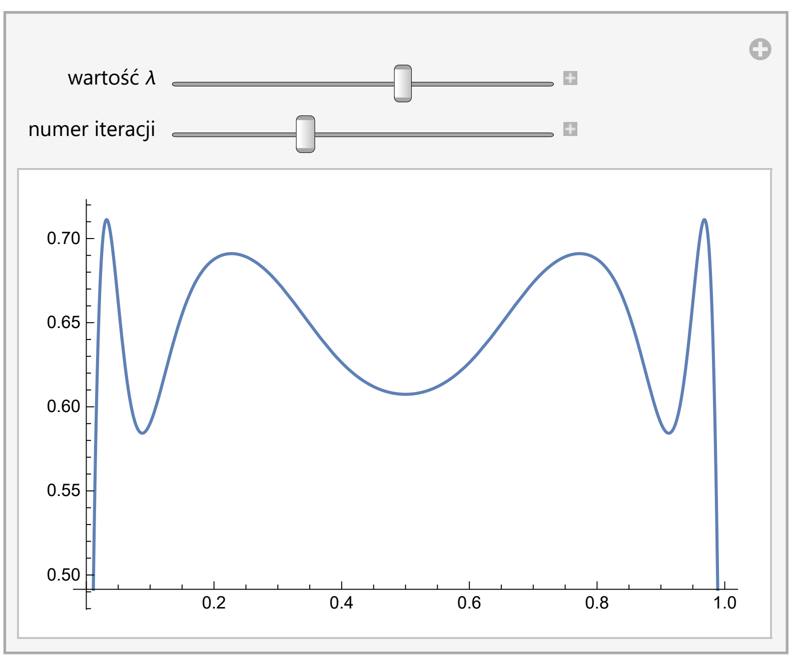
Wywołujemy wynik :

```
Plot[ntaiteracja[3.5, x, 2], {x, 0, 1}]
```



In[5]:=
```
Manipulate[Plot[ntaiteracja[λ, x, n], {x, 0, 1}],
 {{λ, 2, "wartość λ"}, 1, 4},
 {{n, 1, "numer iteracji"}, 1, 10, 1}
]
```

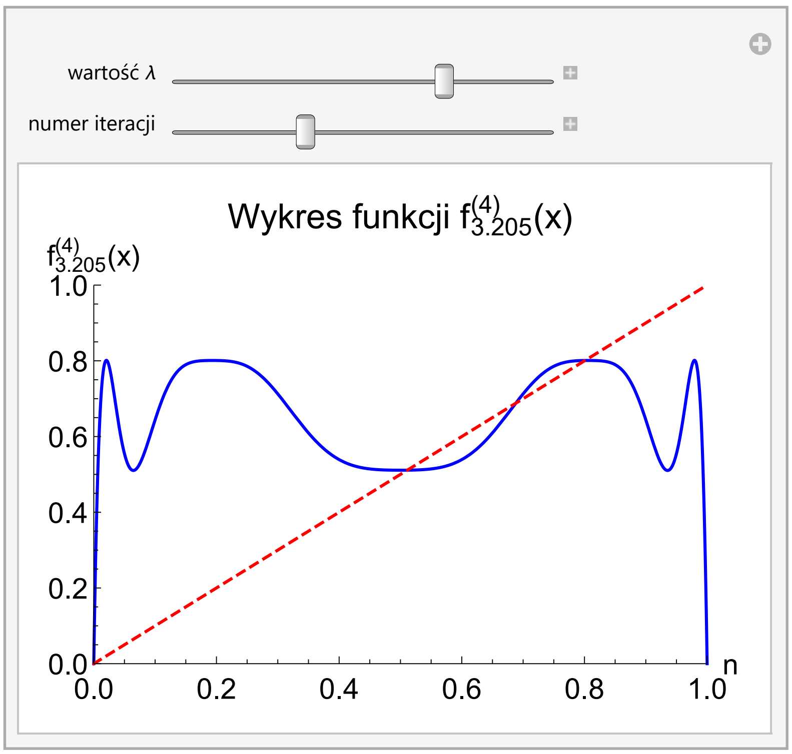
Out[5]=



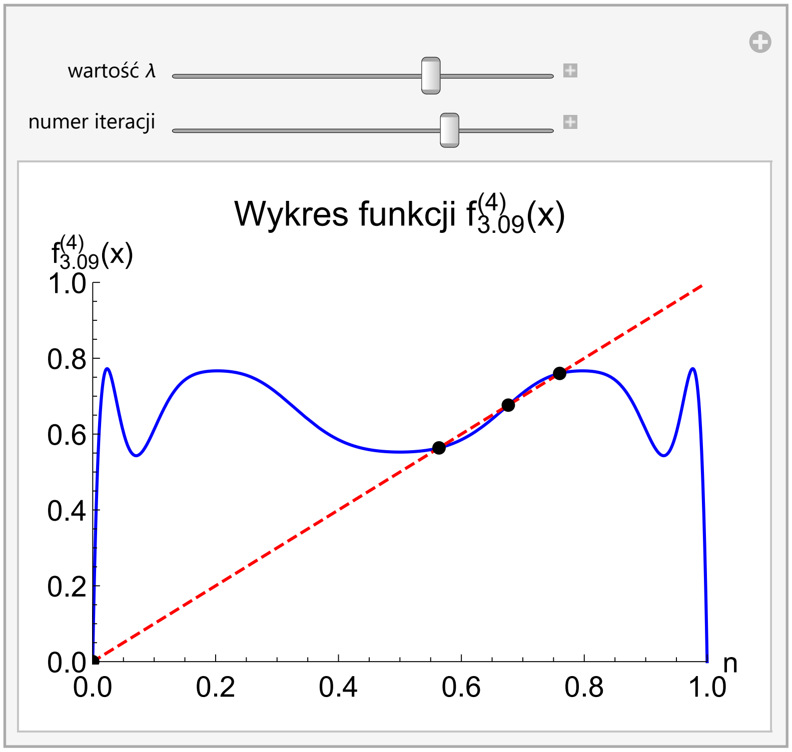
Należy "upiększyć" ten wynik aby nadać mu sens

In[6]:=
```
Manipulate[Plot[{ntaiteracja[λ, x, n], x}, {x, 0, 1},
  PlotStyle → {Blue, {Red, Dashed}},
  PlotRange → {{0, 1}, {0, 1}},
  AxesLabel → {"n",
    ToString[Subsuperscript["f", λ, "(" <> ToString[n] <> ")"], StandardForm] <> "(x)"},
  LabelStyle → Directive[15, Black],
  PlotLabel → "Wykres funkcji " <>
    ToString[Subsuperscript["f", λ, "(" <> ToString[n] <> ")"], StandardForm] <> "(x)"
 ],
 {{λ, 2, "wartość λ"}, 1, 4},
 {{n, 1, "numer iteracji"}, 1, 10, 1}
]
```

Out[6]=



Jako ciekawostkę dodajmy jeszcze jeden bajer



Zad. VI

Warto pokazać jaka jest idea:

Wybieramy sobie punkt startowy

```
x0 = RandomReal[];
```

Jak generować orbitę :

```
asymptot1[λ_, n_] := NestList[f[λ, #] &, x0, n - 1]
```

```
asymptot1[1, 10]
```

```
{0.0187159, 0.0183656, 0.0180283, 0.0177033,
 0.0173899, 0.0170875, 0.0167955, 0.0165134, 0.0162407, 0.0159769}
```

```
? Riffle
```

Riffle[$\{e_1, e_2, \ldots\}, x$] gives $\{e_1, x, e_2, x, \ldots\}$.

Riffle[$\{e_1, e_2, \ldots\}, \{x_1, x_2, \ldots\}$] gives $\{e_1, x_1, e_2, x_2, \ldots\}$.

Riffle[$list, x, n$] yields a list in which every n^{th} element is x.

Riffle[$list, x, \{i_{min}, i_{max}, n\}$] yields a list in which x appears if possible at positions $i_{min}, i_{min}+n, i_{min}+2n, \ldots, i_{max}$. >>

Wybieramy tylko końcowe punkty orbity :

```
asymptot2[λ_, n_, n0_] := Take[NestList[f[λ, #] &, x0, n - 1], - (n - n0)]
```

```
asymptot2[1, 10, 8]
```

```
{0.0378772, 0.0364425}
```

Zamieniamy listę na taką któa pasuje do ListPlota[]

```
asymptot3[λ_, n_, n0_] := Riffle[
  ConstantArray[λ, n - n0],
  Take[NestList[f[λ, #] &, x0, n - 1], - (n - n0)]
 ]
```

```
? Riffle
```

Riffle[{e_1, e_2, …}, x] gives {e_1, x, e_2, x, …}.

Riffle[{e_1, e_2, …}, {x_1, x_2, …}] gives {e_1, x_1, e_2, x_2, …}.

Riffle[*list*, x, n] yields a list in which every n^{th} element is x.

Riffle[*list*, x, {i_{min}, i_{max}, n}] yields a list in which x appears if possible at positions i_{min}, $i_{min}+n$, $i_{min}+2n$, … , i_{max}. ≫

```
asymptot3[1, 10, 8]
```

```
{1, 0.0378772, 1, 0.0364425}
```

Jak będzie wyglądał argument ListPlot[]?

```
Table[asymptot3[λ, 10, 8], {λ, 1, 4, 1}]
```

```
{{1, 0.0378772, 1, 0.0364425}, {2, 0.5, 2, 0.5},
 {3, 0.613129, 3, 0.711606}, {4, 0.873329, 4, 0.442501}}
```

Nie tak to powinno wyglądać!! Stąd

```
? Partition
```

Partition[*list*, n] partitions *list* into nonoverlapping sublists of length n.

Partition[*list*, n, d] generates sublists with offset d.

Partition[*list*, {n_1, n_2, …}] partitions a nested list into blocks of size $n_1 \times n_2 \times \ldots$.

Partition[*list*, {n_1, n_2, …}, {d_1, d_2, …}] uses offset d_i at level i in *list*.

Partition[*list*, n, d, {k_L, k_R}] specifies that the first element of *list* should appear at position k_L in the first sublist, and the last element of *list* should appear at or after position k_R in the last sublist. If additional elements are needed, Partition fills them in by treating *list* as cyclic.

Partition[*list*, n, d, {k_L, k_R}, x] pads if necessary by repeating the element x.

Partition[*list*, n, d, {k_L, k_R}, {x_1, x_2, …}] pads if necessary by cyclically repeating the elements x_i.

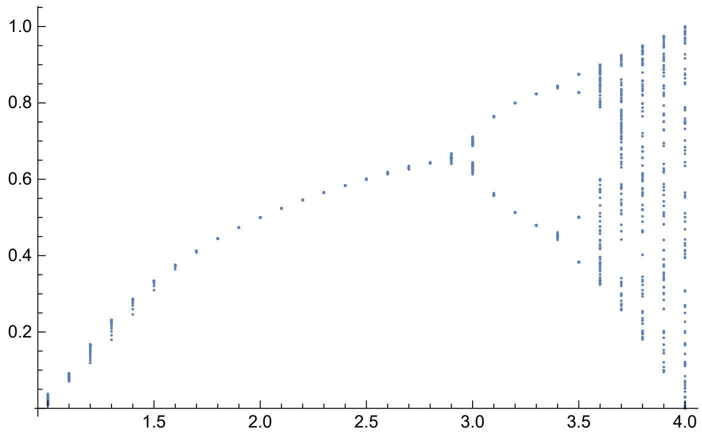
Partition[*list*, n, d, {k_L, k_R}, {}] uses no padding, and so can yield sublists of different lengths.

Partition[*list*, *nlist*, *dlist*, {$klist_L$, $klist_R$}, *padlist*] specifies alignments and padding in a nested list. ≫

```
Partition[Flatten[Table[asymptot3[λ, 10, 8], {λ, 1, 4, 1}]], 2]
```

```
{{1, 0.0378772}, {1, 0.0364425}, {2, 0.5}, {2, 0.5},
 {3, 0.613129}, {3, 0.711606}, {4, 0.873329}, {4, 0.442501}}
```

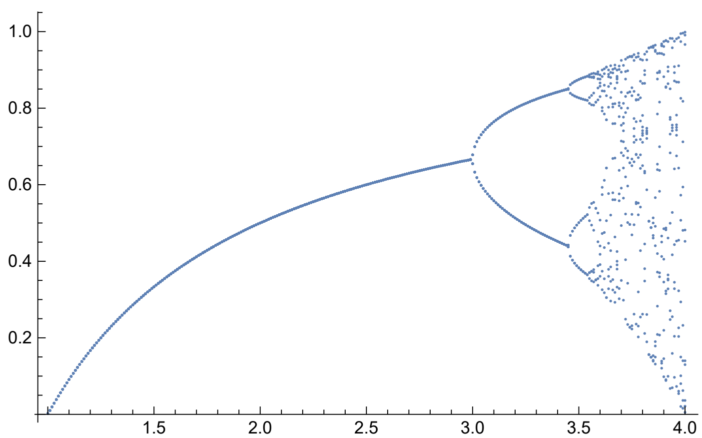
```
ListPlot[Partition[Flatten[Table[asymptot3[λ, 100, 8], {λ, 1, 4, 0.1}]], 2]]
```



Piszemy funkcję właściwą :

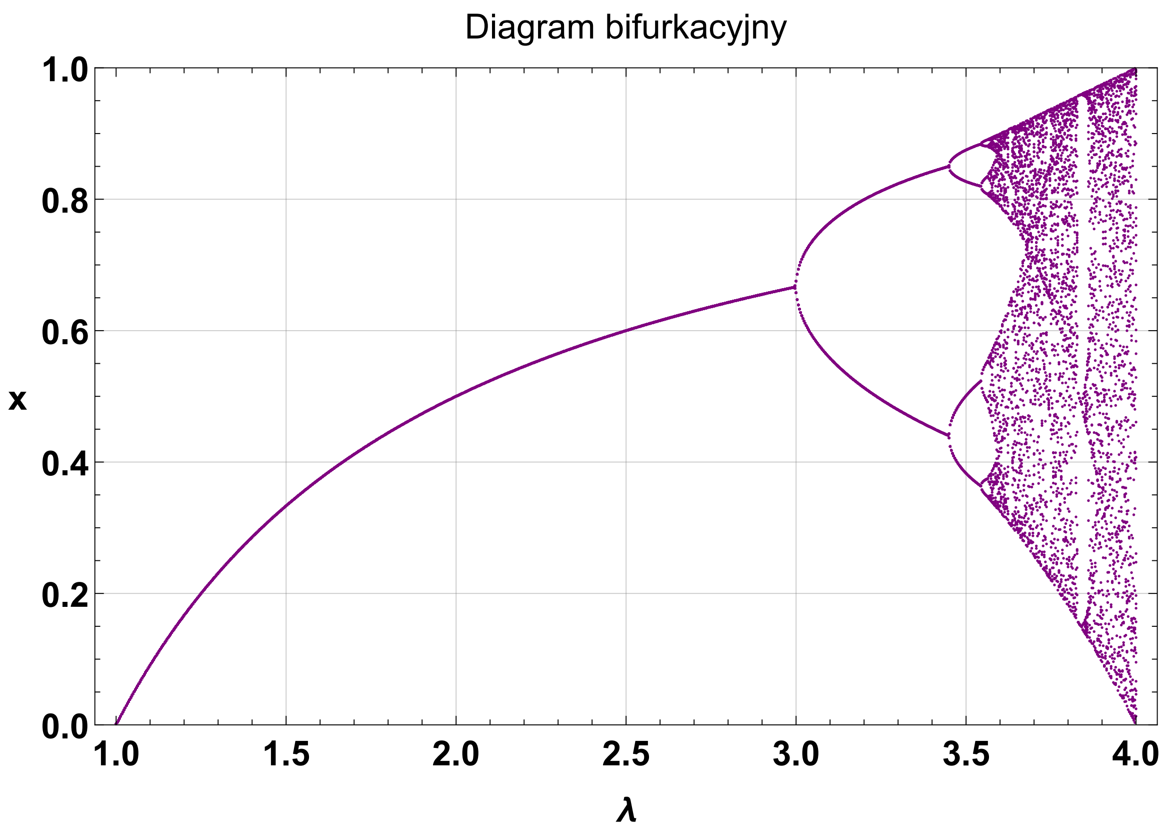
```
diagrambif[xo_, n_, n0_, krok_] :=
 ListPlot[Partition[Flatten[Table[asymptot3[λ, n, n0], {λ, 1, 4, krok}]], 2]]
```

```
diagrambif[x0, 400, 390, 1/100]
```




```
ladnydiagrambif[xo_, n_, n0_, krok_] := ListPlot[
  Partition[Flatten[Table[asymptot3[λ, n, n0], {λ, 1, 4, krok}]], 2], PlotRange -> {0, 1},
  PlotStyle → Directive[PointSize[Tiny], Purple],
  GridLines -> Automatic,
  Frame → True,
  FrameLabel → {"λ", "x"},
  RotateLabel → False,
  PlotLabel → "Diagram bifurkacyjny",
  LabelStyle → Directive[15, Black],
  FrameStyle → Directive[FontWeight → Bold, FontSize → 18],
  ImageSize → 600]
```

```
ladnydiagrambif[x0, 700, 660, 1/300]
```



Zad. VII

Losujemy wektor m liczb pseudolosowych z [0, 1]. Tu dla przykładu m=4.

```
RandomReal[1, 4]
```

```
{0.557639, 0.748447, 0.623007, 0.80672}
```

Działamy na każdej współrzędnej tego wektora n razy nasza funkcją

```
mielenie[λ_, m_, n_] := Nest[Table[f[λ, #[[i]]], {i, 1, m}] &, RandomReal[1, m], n]
```

Sprawdzamy czy dobrze działa :

```
mielenie[3.3, 5, 20]
```

```
{0.479428, 0.479427, 0.479704, 0.823603, 0.479427}
```

Tworzymy histogram :

```
lista[lista_, bin_] := HistogramList[lista, bin, "PDF"]
```

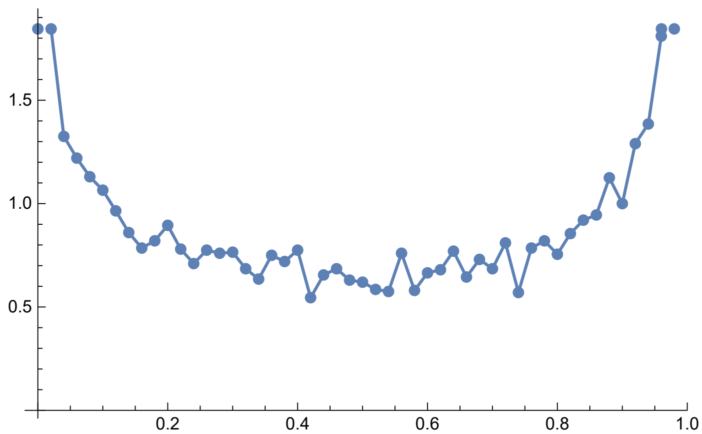
np :

```
lista[mielenie[3.3, 5, 20], 8]
```

$\left\{\left\{\frac{9}{20}, \frac{1}{2}, \frac{11}{20}, \frac{3}{5}, \frac{13}{20}, \frac{7}{10}, \frac{3}{4}, \frac{4}{5}, \frac{17}{20}\right\}, \{4, 0, 0, 0, 0, 0, 0, 16\}\right\}$

Rysowanie otrzymanej listy za pomocą ListPlot[]

```
zp = mielenie[4, 10000, 500];

miara[lis_, bin_] := {zpx = lista[lis, bin];
   ListPlot[Partition[Riffle[zpx[[1]], zpx[[2]]], 2], Joined → True, Mesh → All]}[[1]]

miara[zp, 30]
```



Gęstość niezmiennicza dla rodziny logistycznej : $\frac{1}{\pi\sqrt{x\ (1-x)}}$

```
miara2[lis_, bin_] := {zpx = lista[lis, bin]; Show[
    Plot[1/(Pi * Sqrt[x * (1 - x)]), {x, 0, 1}, PlotStyle → Red, AxesOrigin → {0, 0}],
    ListPlot[Partition[Riffle[zpx[[1]], zpx[[2]]], 2], Joined → True, Mesh → All]
   ]
  }[[1]]

miara2[zp, 60]
```