Основы Python: типы данных, операторы

Технологии и языки программирования

Юдинцев В. В.

Кафедра теоретической механики
Самарский университет

Версия 09.2017

Содержание

История

https://commons.wikimedia.org

- Язык Python (Питон или Пайтон) разработан нидерландским программистом Гвидо ван Россумом (Guido van Rossum)
- Начало разработки: конец 80-х годов
  - 1994 v. 1.0 – 2000 v. 1.6
  - 2000 v. 2.0 – 2010 v. 2.7
  - 2008 v. 3.0 – 2015 v. 3.5

`https://www.python.org`

Особенности языка

- Интерпретируемый, кроссплатформенный язык
- Python поддерживает различные парадигмы программирования:
  - модульное программирование
  - объектно-ориентированное программирование
  - функциональное программирование
- Динамическая сильная типизация

  Типы переменных определяются во время выполнения программы
- Поддержка структур данных высокого уровня

  set, dict
- Автоматизированная система контроля памяти

Компании, использующие Питон

www.quora.com

Возможности Python

Библиотека numpy

Базовый библиотека для численных методов

- определяет типы данных: векторы, матрицы, многомерные массивы
- функции для эффективной работы с матрицами
- функции линейной алгебры
- генераторы случайных чисел различных типов

Пример использование библиотеки numpy

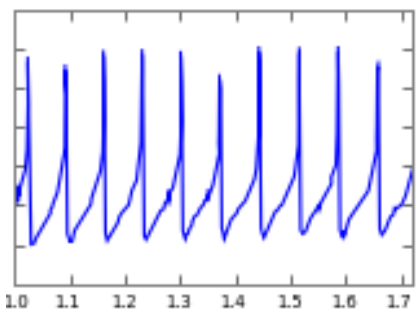
Умножение матриц:

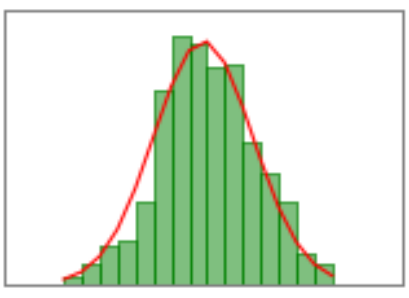
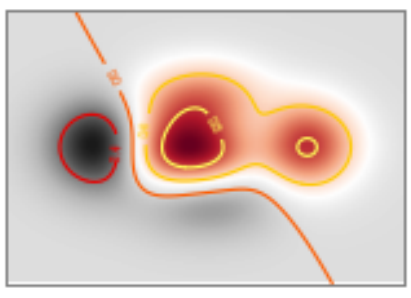
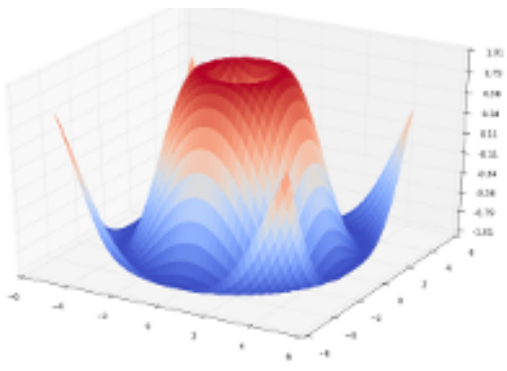
```
1  import numpy as np
2
3  m1 = np.matrix([[1, 2, 3],
4                  [5, 5, 6]])
5  m2 = np.matrix([[3, 2],
6                  [4, 2],
7                  [1, 1]])
8
9  m12 = m1*m2
10
11 print(m12)
```

```
[[14  9]
 [41 26]]
```

Библиотека matplotlib

- **matplotlib** — библиотека для построения графиков по массивам (таблицам) данных с возможностью экспорта их в различных форматы файлов
- Вместе с NumPy, SciPy и IPython предоставляет возможности, подобные MATLAB

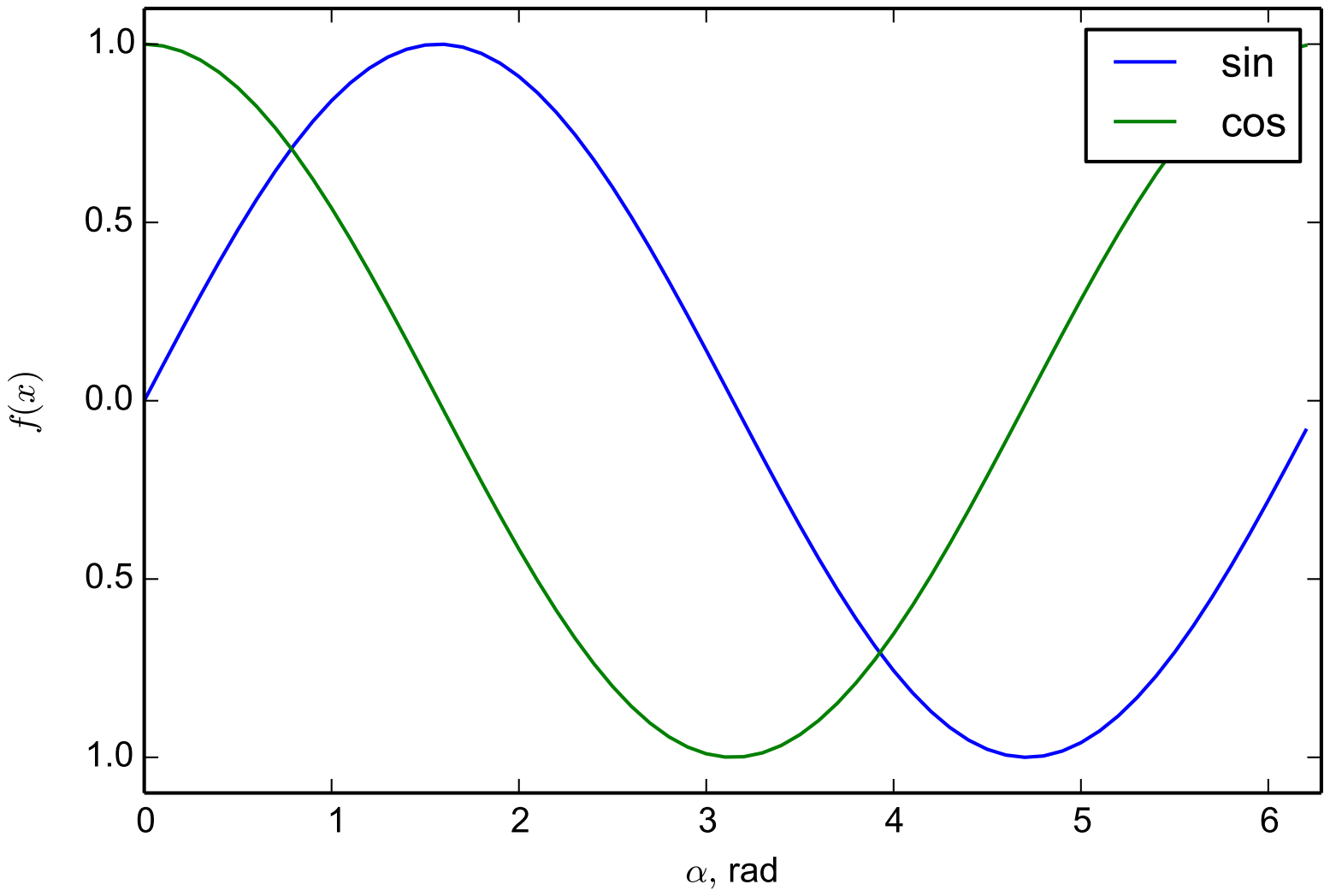


Библиотека matplotlib

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4  x = np.arange(0.0, np.pi*2.0, 0.1)
5  y1 = np.sin(x)
6  y2 = np.cos(x)
7
8  plt.plot(x, y1, x, y2)
9
10 plt.axis([0, 2*np.pi, -1.1, 1.1])
11 plt.xlabel('$\\alpha$, rad')
12 plt.ylabel('f(x)')
13 plt.legend(["sin","cos"])
```

Библиотека matplotlib



Библиотека scipy

- специальные функции,
- численное интегрирование,
- оптимизация,
- решение нелинейных уравнений,
- интерполирование,
- линейная алгебра,
- статистика

Пример использования scipy

Численное решение нелинейного уравнения:

$$x + 2\cos x = 0$$

```
1 import numpy as np
2 from scipy.optimize import root
3
4 def func(x):
5     return x + 2 * np.cos(x)
6
7 sol = root(func, 0.3)
8 print(sol.x)
```

```
>> array([-1.02986653])
```

Библиотека pandas

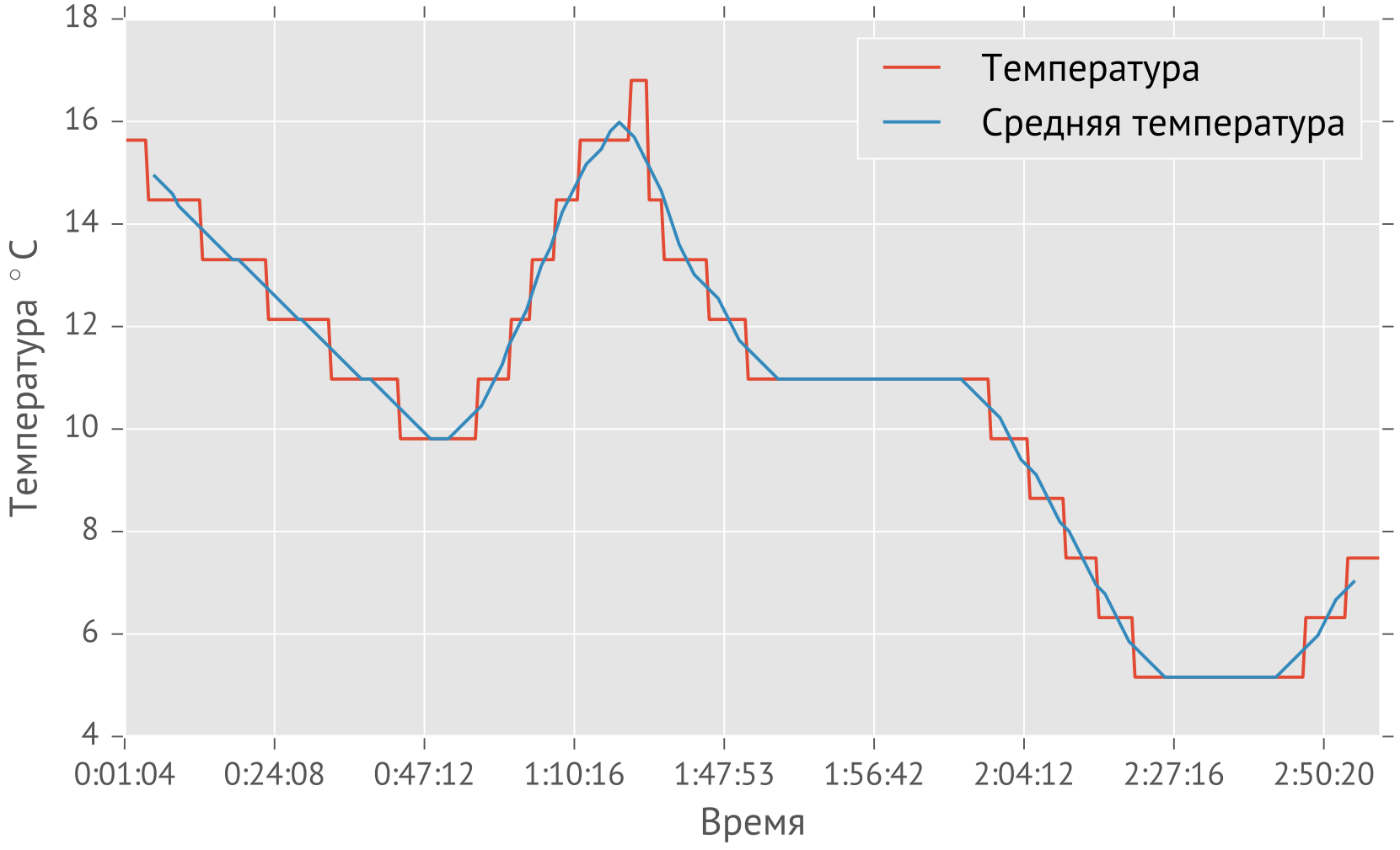
Библиотека для эффективного анализа данных, первоначально разработанная для обработки финансовой информации (2008). Возможности pandas:

- загрузка данных их текстовых файлов, таблиц XLS, баз данных;
- обработка таблиц и временных рядов, группировка данных, переформатирование данных, создание сводных таблиц;
- объединение наборов данных;
- работа с данными большой размерности;
- построение графиков.

Библиотека pandas

```
1  import pandas as pd
2  import matplotlib
3
4  data = pd.read_csv("temp.dat")
5
6  rolling = pd.rolling_mean(data, 20, center = True)
7  rolling.columns=["Время","Средняя температура"]
8  ax_data.set_ylabel('Температура $^\circ$C')
9
10 ax_data = data.plot()
11 rolling.plot(ax = ax_data)
```

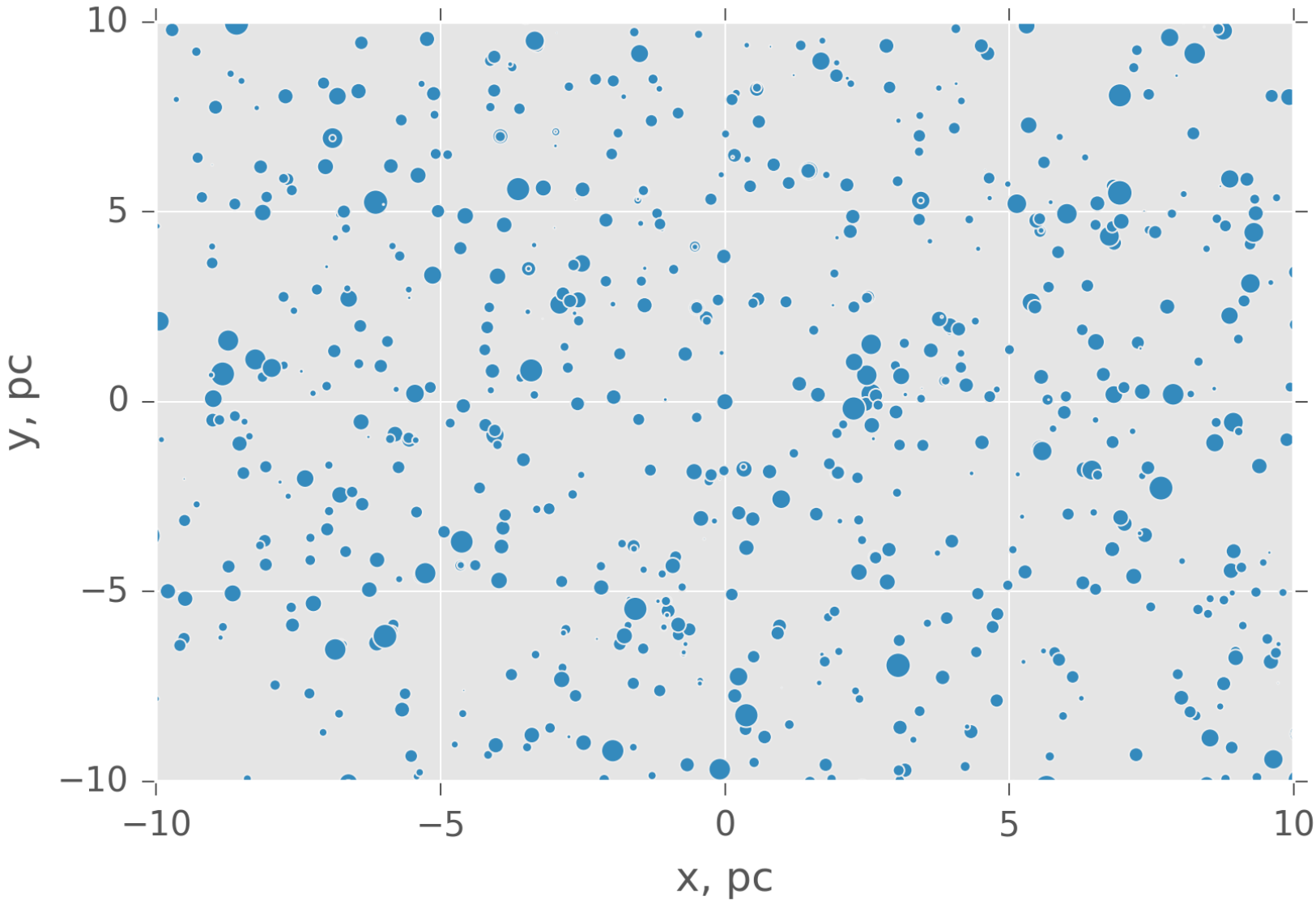
Библиотека pandas



Загрузка данных из Сети

```
import matplotlib.pylab as plt
import pandas as pd
import matplotlib
matplotlib.style.use('ggplot')

data = pd.read_csv(
"http://www.astronexus.com/files/downloads/hygdata_v3.csv.gz",
compression="gzip")

data[data["dist"]<20].plot.scatter(x="x", y="y",
s=5*data["absmag"],xlim=(-10,10),ylim=(-10,10))

plt.xlabel("x, pc")
plt.ylabel("y, pc")
plt.savefig('stars.png', dpi=300)
```

Карта ближайших звёзд



sympy

`http://www.sympy.org`

Пакет для аналитических преобразований:

- Решение уравнений
- Дифференцирование функций
- Интегрирование функций

Решение уравнения

Решение квадратного уравнения

$$x^2 + 10x + 3 = 0$$

```
1 import sympy as sp
2
3 x = sp.symbols('x')
4
5 f = x**2 + 10*x + 3
6
7 print(sp.solve(f))
```

```
>> [-5 - sqrt(22), -5 + sqrt(22)]
```

Дифференцирование функции

Производная функции:

$$f = \cos(x)\ \sin^2(x)$$

```
1 import sympy as sp
2
3 x = sp.symbols('x')
4
5 f = sp.cos(x)*sp.sin(x)**2
6
7 print(sp.diff(f, x))
```

```
>> -sin(x)**3 + 2*sin(x)*cos(x)**2
```

```python
# -*- coding: utf-8 -*-
import requests
import pandas as pd

address= 'http://ssau.ru/ratings/bakalavr.php'
html= requests.get(address).content

df_list = pd.read_html(html,encoding='utf8', header=0)

df= df_list[0]

df= (df.assign(Рейтинг = df['Количество заявлений']/df['План']).\
                sort_values('Рейтинг',ascending = False))

df= df[ (df['Тип']=='Общий конкурс') & (df['Форма обучения']=='Очная') ]

print( df.head(5)[['Направление','Рейтинг']] )
```

Направление Рейтинг
97 38.03.01 Экономика 87.25
100 38.03.02 Менеджмент 74.25
106 38.03.04 Государственное и муниципальное управ... 51.25
103 38.03.03 Управление персоналом 41.25
109 38.03.05 Бизнес-информатика 39.00

Среда разработки

- **IDLE**
  `https://docs.python.org/3/library/idle.html`
  Простейшая интегрированная среда разработки.
  Поставляется вместе с Pyhon.
- **PyCharm**
  `https://www.jetbrains.com/pycharm/download`
- Anaconda
  `https://www.continuum.io/downloads`
  Python + IDE Spyder с набором библиотек (numpy, scipy, ...)
  для научных вычислений

Spyder (Python 3.5)

/home/kidinnu/Programs/anaconda3/bin

Editor - /home/kidinnu/.spyder2-py3/temp.py

temp.py | animation.py | mot.py | interpolation.py

```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""

import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
from pylab import *

from matplotlib import rc


font = {'family': 'Times New Roman','weight': 'normal', 'siz
rc('font', **font)


def figsize(wcm,hcm):
    result = figure(figsize=(wcm/2.54,hcm/2.54))
    return result

def create_plot() :
    figure = figsize(16,10)
    ax = figure.add_subplot(111)
    figure.subplots_adjust(top=0.95, bottom=0.12, left=0.09,
    return figure, ax

def relative(vv,vr,w) :


    return []

fig, ax = create_plot()

ax.plot([0,1,2,3],[0,1,2,3],linewidth=2)
xlabel("1"), ylabel("Расстояние $x_2$")
# grid(True)
savefig('Rx1Ry1_fairing_tgk.png', dpi = 300)
```

Variable explorer

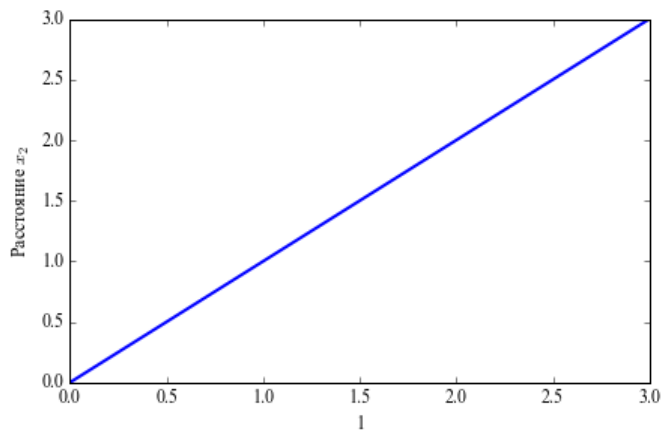
Name	Type	Size	Value
cell	list	2	[2, 1]
cells	list	5	[[1, 1], [2, 1], [3, 1], [3, 2], [2, 3]]
e	float	1	2.718281828459045
euler_gamma	float	1	0.5772156649015329
font	dict	3	{'family': 'Times New Roman', 'size': 12, 'weig…
isOccuped	bool	1	True
pi	float	1	3.141592653589793
x	int	1	2
y	int	1	1

Object inspector | Variable explorer | File explorer

IPython console

Console 1/A

```
In [4]: runfile('/home/kidinnu/.spyder2-py3/temp.py', wdir='/home/kidinnu/.spyder2-
py3')
```




```
In [5]:
```

Console | History log | IPython console

Permissions: RW | End-of-lines: LF | Encoding: UTF-8 | Line: 11 | Column: 13 | Memory: 20 %

Синтаксис

```
# -*- coding: utf-8 -*-
"""
Created on Fri Sep 22 21:44:39 2017
@author:
"""

data = [7, 3, 2, 9]

# Максимальное значение списка data
max_element = data[0]

for element in data:
    if element > max_element:
        max_element = element

print(max_element)
```

Файл программы, функции

В первой или второй строке файла программы, модуля должна быть указана кодировка файла, например, UTF-8

```
1 # -*- coding: utf-8 -*-
```

или 1251 – кодировка MS Windows

```
1 # -*- coding: cp1251 -*-
```

Логические и физические строки

- Программа состоит из логических строк
- Логическая строка состоит из одной или нескольких физических строк
- Физические строки объединяются в одну логическую при помощи символа \

```
1 if 1900 < year < 2100 and 1 <= month <= 12 \
2    and 1 <= day <= 31 and 0 <= hour < 24 :
3         return 1
```

- Выражения в скобках ( ) [ ] { } могут быть записаны в несколько строк без использования символа \

```
1 stars = ["V645 Cen", "Alpha Cen A",
2          "Alpha Cen B", "Barnard's Star",
3          "Wolf 359", "BD+362147"]
```

Имена переменных

- Буквы в нижнем или верхнем регистре
- Цифры
- Нижнее подчёркивание
- Имена переменных не могут начинаться с цифры

```
1 a1   = 10
2 a_3  = 20
3 A3   = 7
4 _ab  = 9
```

Ключевые слова

```
False      class      finally    is         return
None       continue   for        lambda     try
True       def        from       nonlocal   while
and        del        global     not        with
as         elif       if         or         yield
assert     else       import     pass       print
break      except     in         raise
```

Эти ключевые слова не могут быть использованы в качестве имен переменных (идентификаторов)

Комментарии

Комментарии начинаются с символа #, если этот символ не является частью строковой константы

```
a="Здесь символ # не начинает комментарий"
```

Комментарий заканчивается в конце физической стоки

```
# Six nearest stars
stars=["V645 Cen", "Alpha Cen A", "Alpha Cen B",
       "Barnard's Star", "Wolf 359", "BD+362147"]
```

Многострочные комментарии

Многострочные комментарии могут записаны в виде безымянной строковой константы

```
'''
Triple quotes are treated as regular strings
with the exception that
they can span multiple lines.
'''
```

Основные типы данных

Классификация

- Атомарные
  bool, int, float, complex
- Ссылочные
  list, tuple, str, dict, set
  - Изменяемые
    list, dict, set
  - Неизменяемые
    tuple, str

Булевы переменные

```
1 a = True
2 b = False
3
4 c = 7>3
5 print(c)
6 True
```

Целые числа

```
a=15687554
```

Явное преобразование в тип int с отбрасыванием дробной части:

```
b=int(10/3)
3
```

Размер int ограничен только памятью ЭВМ

```
a=2**1000
107150860718626732094842504906000181056140481170553360744375038837035105112493612249319837881569585812759467291755314682518714528569231404359845775746985748039345677748242309854210746050623711418779541821530464749835819412673987675591655439460770629145711964776865421676604298316526243868372056680693769
```

Математические операторы

+	сложение	2 + 2	= 4
-	вычитание	5 - 3	= 2
*	умножение	2 * 2	= 4
/	деление	7 / 2	= 3.5
//	целочисленное деление	7 // 2	= 3
%	остаток от деления	7 % 2	= 1
**	возведение в степень	7 ** 2	= 49

Вещественное число (64-bit double precision)

В отличии от языков Си или Паскаль в языке Питон определён только один тип вещественного числа **float** – **64** битное число **double precision** в соответствии со стандартом IEEE 754

```
1 a = 123.1445
```

Явное преобразование в тип float

```
1 a = float(3)
```

Размер ограничен:

```
1 a = 3**1000     # 3 в степени 1000 - целое число
```

Преобразование в float невозможно

```
1 b = float(a)
```

OverflowError: int too large to convert to float

Последовательности

Списки (list)

Списки значений создаются при помощи оператора [ ] с перечислением объектов, разделённых запятыми:

```
1 a = [1, 'word', 2, complex(1,2)]
```

Длина списка

```
1 len(a)
2 4
```

Доступ к элементу списка

```
a = [1, 'word', 2, complex(1,2)]
```

Доступ к элементу списка при помощи [ ] скобок.
Первый элемент списка имеет индекс 0:

```
a[0]

1
```

Последний элемент списка из четырёх элементов:

```
a[3]

(1+2j)
```

Функция range

Функция используется для генерации последовательностей **range(a,step,b)** начиная с **a** до **b** (не включая b) с шагом **step**

```
1 list(range(1,10,2))
2
3 [1, 3, 5, 7, 9]
```

Можно указать только верхнюю границу, не входящую в последовательность (шаг равен 1):

```
1 list(range(10))
2 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Аргументы функции range целые числа (положительные и отрицательные)

```
1 list(range(-2,-7,-2))
2 [-2, -4, -6]
```

Список символов из строки

Создание списка символов из текстовой строки:

```
1 symbols = list('cat')
2
3 print(symbols)
4
5 ['c', 'a', 't']
```

Срезы

```
1 a = ['Меркурий', 'Венера', 'Земля', 'Марс']
```

Первые три элемента (с 0 по 2)

```
1 a[0:3]
2 ['Меркурий', 'Венера', 'Земля']
```

Каждый второй (или с шагом 2), начиная с первого

```
1 a[::2]
2 ['Меркурий', 'Земля']
```

Каждый второй, начиная со второго

```
1 a[1::2]
2 ['Венера', 'Марс']
```

Срезы

```
a = ['Меркурий', 'Венера', 'Земля', 'Марс']
```

Обращение последовательности

```
a[::-1]
['Марс', 'Земля', 'Венера', 'Меркурий']
```

Каждый второй, начиная с предпоследнего

```
a[-2::-2]
['Земля', 'Меркурий']
```

Каждый второй, начиная с последнего

```
a[::-2]
['Марс', 'Венера']
```

Вложенные списки

Элемент списка может быть списком

```
a = [ [1,2], [3,4,5] ]
a[0]
[1,2]

a[1]
[3,4,5]

a[1][2]
5
```

Добавление элемента

```
1 a = ['Меркурий', 'Венера', 'Земля', 'Марс']
```

Добавление элемента в конец списка append

```
1 a.append('Юпитер')
2 a
3 ['Меркурий', 'Венера', 'Земля', 'Марс', 'Юпитер']
```

Добавление элемента

```
a = ['Меркурий', 'Венера', 'Земля', 'Марс']
```

Добавление элемента в конец списка append

```
a.append('Юпитер')
a
['Меркурий', 'Венера', 'Земля', 'Марс', 'Юпитер']
```

Вставка элемента в список (insert)

```
a = ['Меркурий', 'Венера', 'Марс']
a.insert(2,'Земля')
a
['Меркурий', 'Венера', 'Земля', 'Марс']
```

Объединение списков. Метод extend

```
1 p1 = ['Меркурий', 'Венера']
2 p2 = ['Земля', 'Марс']
3
4 p1.extend(p2)
5
6 p1
7 ['Меркурий', 'Венера', 'Земля', 'Марс']
```

Объединение списков. Метод extend

```
p1 = ['Меркурий', 'Венера']
p2 = ['Земля', 'Марс']

p1.extend(p2)

p1
['Меркурий', 'Венера', 'Земля', 'Марс']
```

Короткий вариант с использованием оператора +=

```
p1 = ['Меркурий', 'Венера']
p2 = ['Земля', 'Марс']

p1+= p2

p1
['Меркурий', 'Венера', 'Земля', 'Марс']
```

Удаление элемента

С использованием оператора **del** по **индексу** элемента

```
1 planets = ['Меркурий', 'Венера', 'Земля', 'Марс']
2
3 del planets[2]
4
5 planets
6 ['Меркурий', 'Венера', 'Марс']
```

Удаление элемента

С использованием оператора **del** по **индексу** элемента

```
1 planets = ['Меркурий', 'Венера', 'Земля', 'Марс']
2
3 del planets[2]
4
5 planets
6 ['Меркурий', 'Венера', 'Марс']
```

С использованием метода **remove** по **значению** элемента

```
1 planets = ['Меркурий', 'Венера', 'Земля', 'Марс']
2
3 planets.remove('Земля')
4
5 planets
6 ['Меркурий', 'Венера', 'Марс']
```

Проверка наличия элемента в списке

Оператор in

```
planets = ['Меркурий', 'Венера', 'Земля', 'Марс']

'Марс' in planets
True
```

Количество заданных элементов в списке

```
phone = [2, 3, 22, 3, 22]

phone.count(22)
2
```

Управляющие конструкции

IF ... ELSE

```
a=getA(c)
if a>0 :
  aDesc='Positive value'
else
  aDesc='Non-positive value'
print(aDesc)
```

Операторы сравнения

равно	==
не равно	!=
меньше	<
меньше или равно	<=
больше	>
больше или равно	>=
включение	in

Составные логические выражения строятся при помощи операторов

and, or, not

Логические выражения

```
1 a = 5
2 a < 0
```

False

```
3 a < 10
```

True

```
4 a > 0 and a < 10
```

True

```
5 a > 0 and not a > 10
```

True

IF ... ELIF ... ELSE

```
1 a=getA(c)
2 if a>0 :
3   aDesc='Positive value'
4 elif a<0 :
5   aDesc='Non-positive value'
6 else
7   aDesc='Zero value'
8 print(aDesc)
```

Цикл FOR

Используется для итераций – движения по элементам последовательностей:

```
1 for i in [3,2,1,'start!']:
2     print(i)
3
4 3
5 2
6 1
7 start!
```

Тело цикла обозначается отступом от положения оператора for

continue и break

Оператор **continue** начинает следующий проход цикла, минуя оставшееся тело цикла

```
1 for i in 'hello world':
2     if i == 'o':
3         continue
4     print(i,end='')
```

hell wrld

Оператор **break** прерывает выполнение цикла

```
1 for i in 'hello world':
2     if i == 'o':
3         break
4     print(i,end='')
```

hell

Цикл WHILE

Выполнение тела цикла пока выполняется заданное условие

```
a = 0
b = 1
while a<50:
    print(a)
    b = a + b
    a = b – a
```

```
0
1
1
2
3
5
8
13
21
34
```

или

```
a,b = 0,1
while a<50:
    print(a)
    a,b = b,a+b
```

Цикл WHILE

```
1 a,b = 0,1
2 while a<50:
3     print(a)
4     a,b = b,a+b
5 else:
6     print('Следующее число фибоначчи больше 50 и равно',
          a)
```

Секция **else** исполняется, если условие цикла перестало выполняться или не выполнилось ни разу

```
⋮
8
13
21
34
Следующее число фибоначчи больше 50 и равно 55
```

Цикл WHILE

```
1 a,b = 0,1
2 while a<50:
3     print(a)
4     a,b = b,a+b
5 else:
6     print('Следующее число фибоначчи больше 50 и равно',
          a)
```

Секция **else** исполняется, если условие цикла перестало выполняться или не выполнилось ни разу

⋮
8
13
21
34
Следующее число фибоначчи больше 50 и равно 55

Использование функции range в циклах FOR

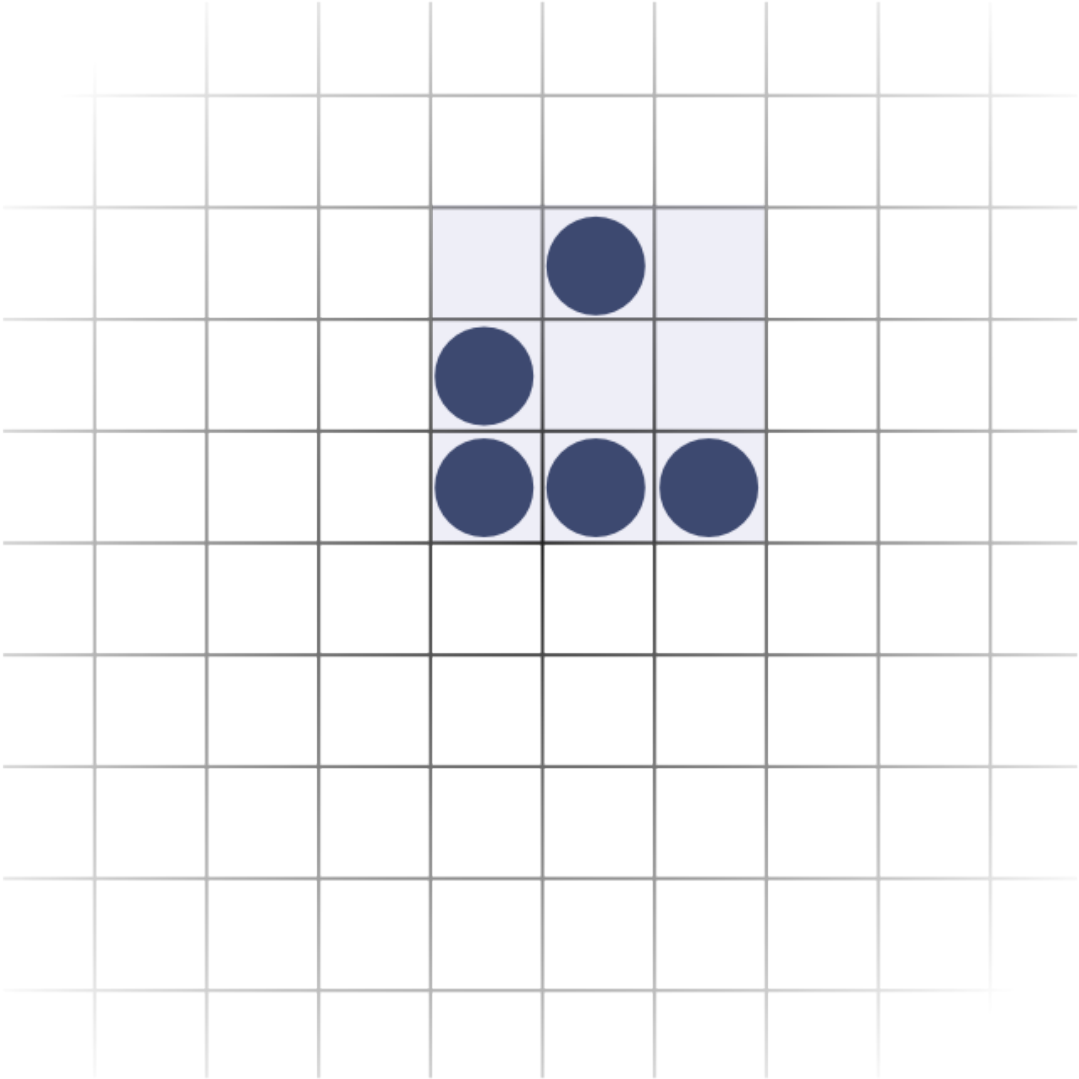
```
week = ['Пн', 'Вт', 'Ср', 'Чт', 'Пт', 'Сб', 'Вс']

n = len(week)

for i in range(n) :
    print('{} день недели — {}'.format(i+1, week[i])
     )

1 день недели — Пн
2 день недели — Вт
3 день недели — Ср
4 день недели — Чт
5 день недели — Пт
6 день недели — Сб
7 день недели — Вс
```

Задания

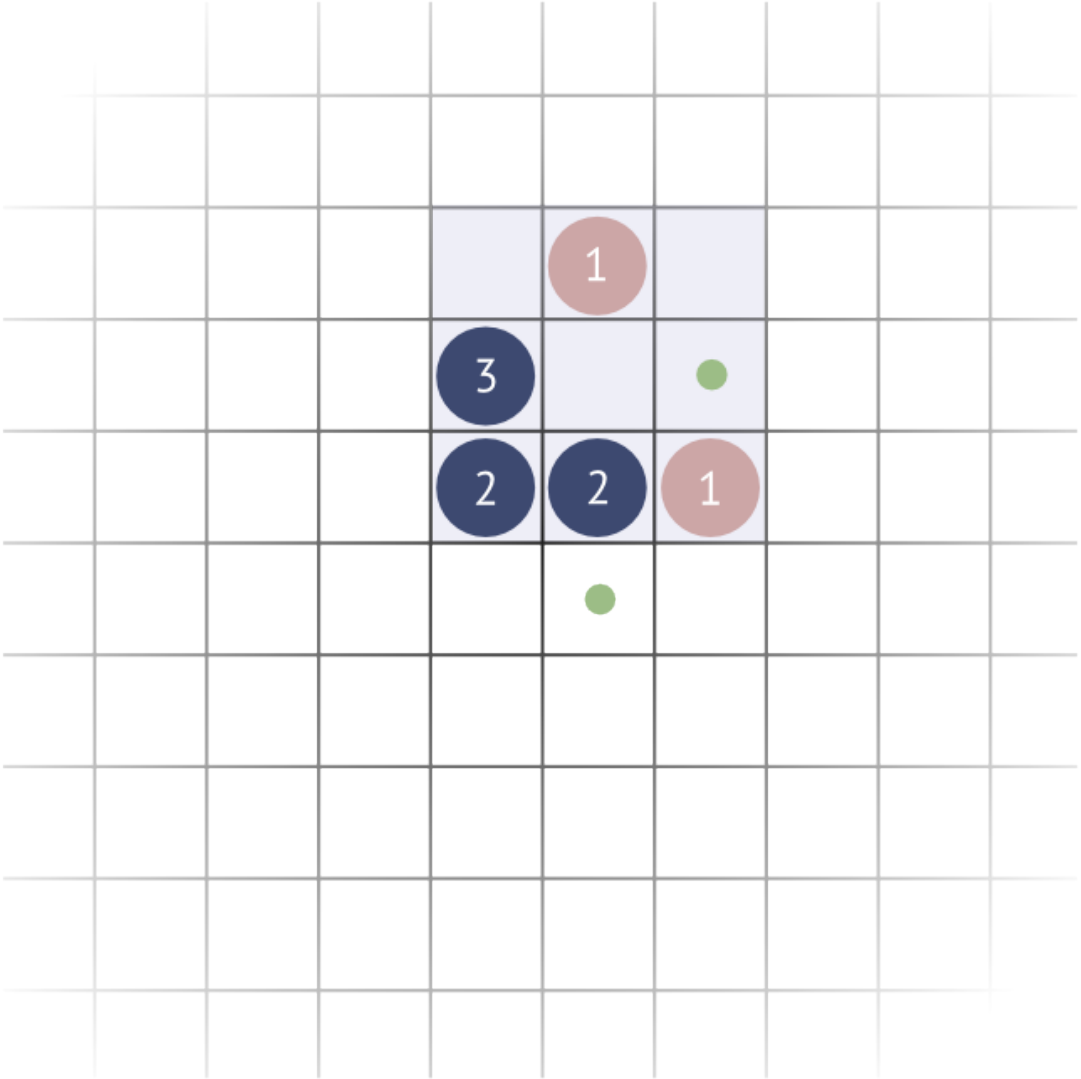
Игра “Жизнь” (автор Дж. Конвей)

1. Дано бесконечное поле на плоскости, разбитое на квадратные ячейки.
2. Каждая ячейка может быть пустой или занятой клеткой.
3. Каждая ячейка граничит с 9 пустыми или занятыми ячейками.
4. Клетка умирает если вокруг неё меньше 2 или больше 3 соседей.
5. Клетка рождается в пустой ячейке если вокруг неё ровно 3 соседа (клетки).
6. Рождение и смерть клеток происходит одновременно.

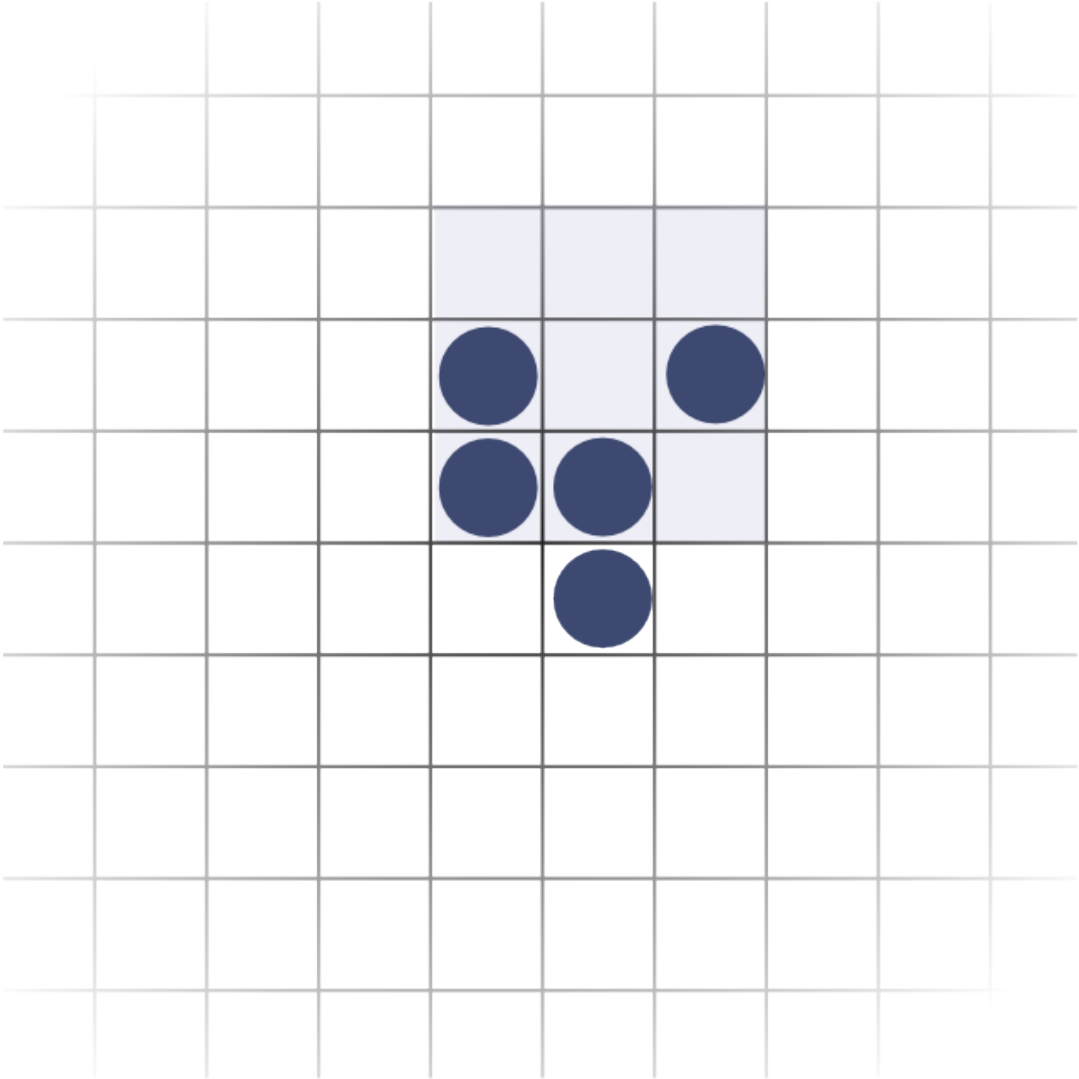
Глайдер



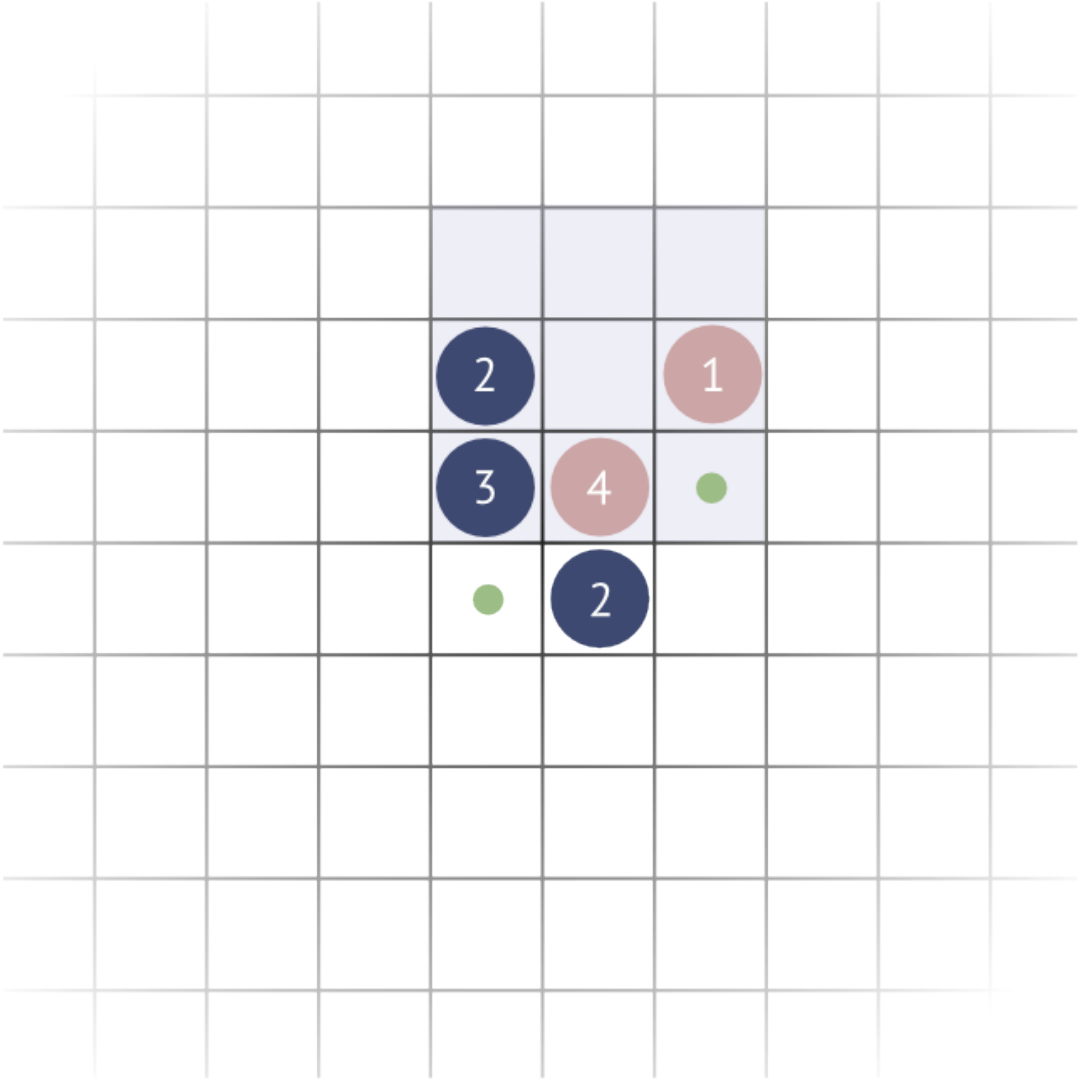
Глайдер



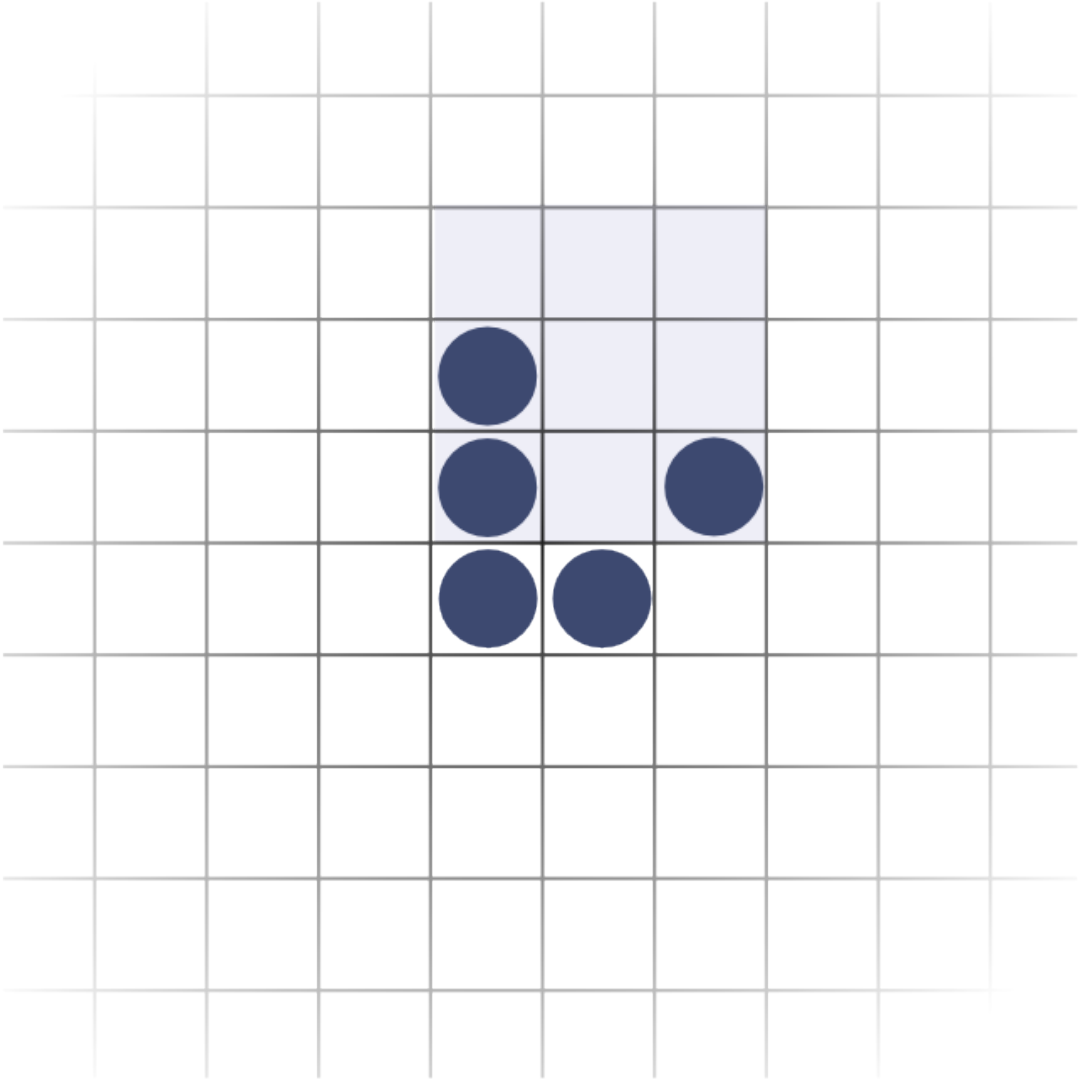
Глайдер



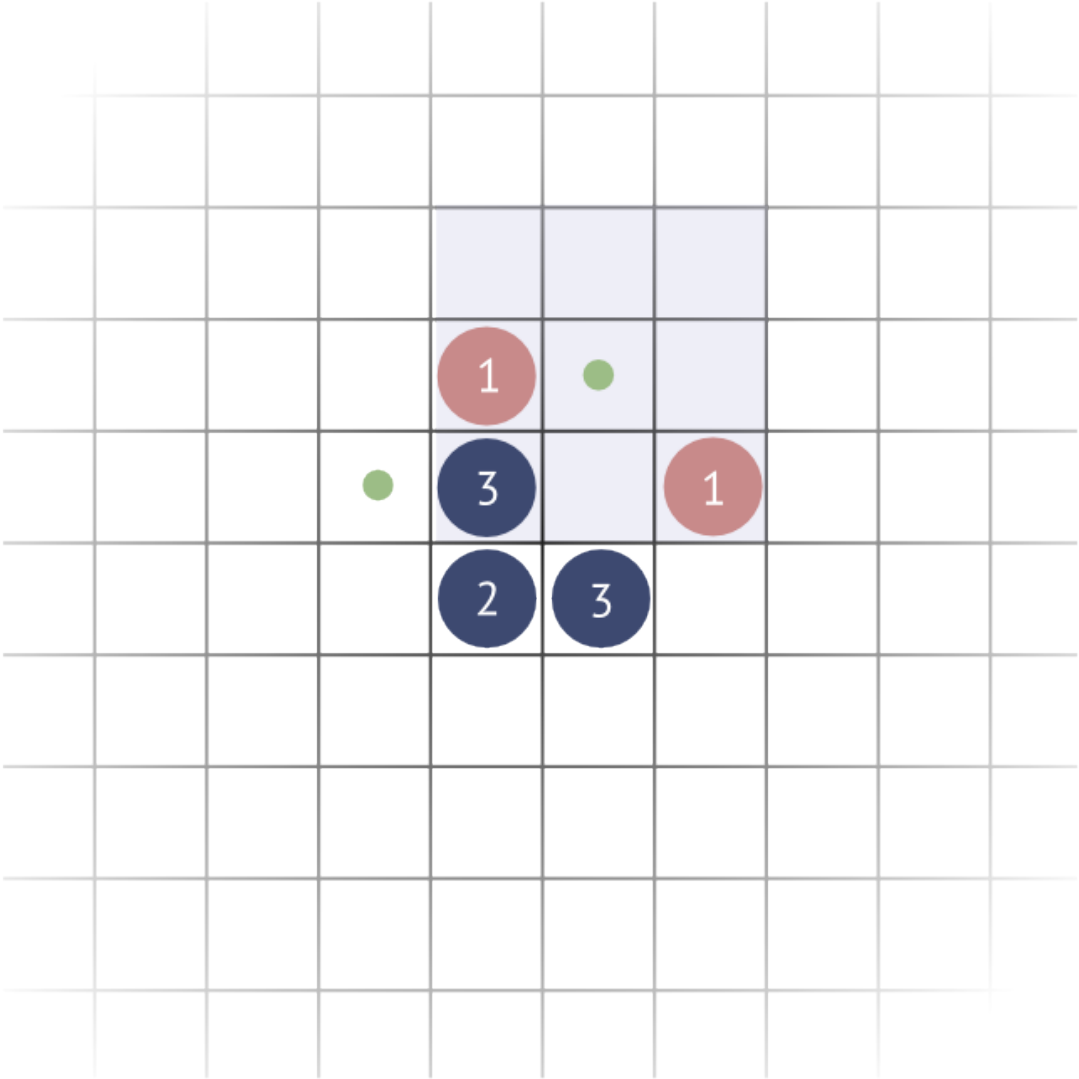
Глайдер



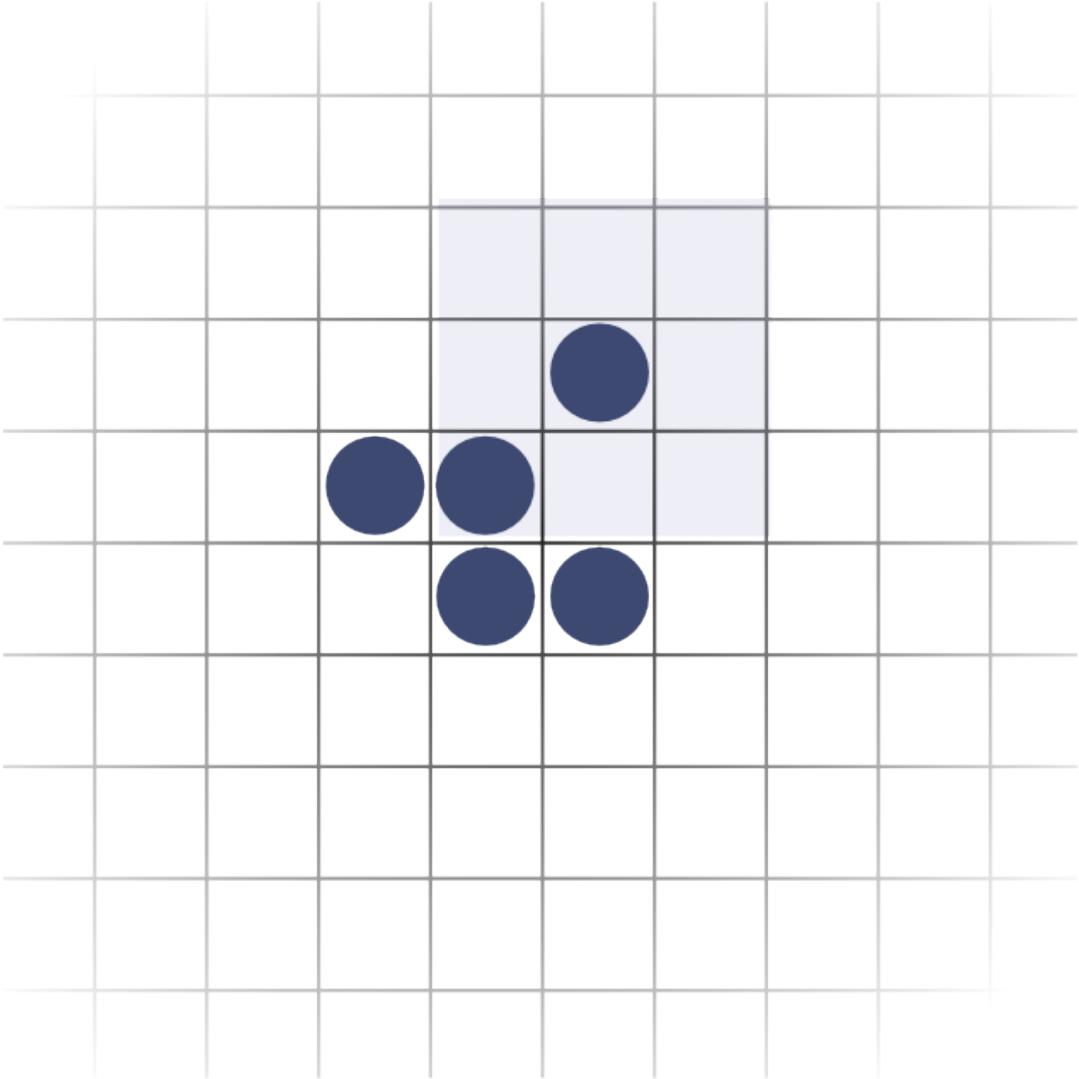
Глайдер



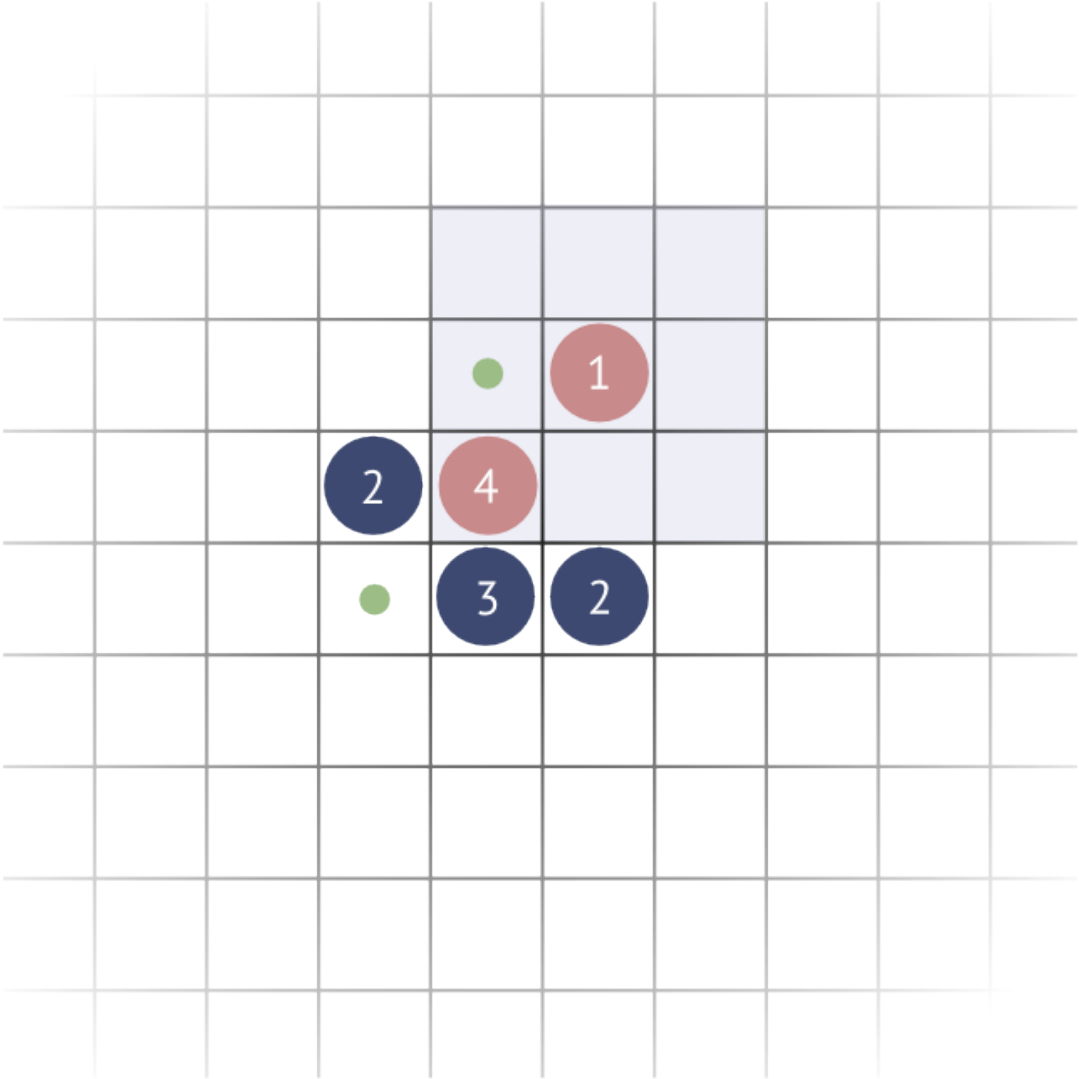
Глайдер



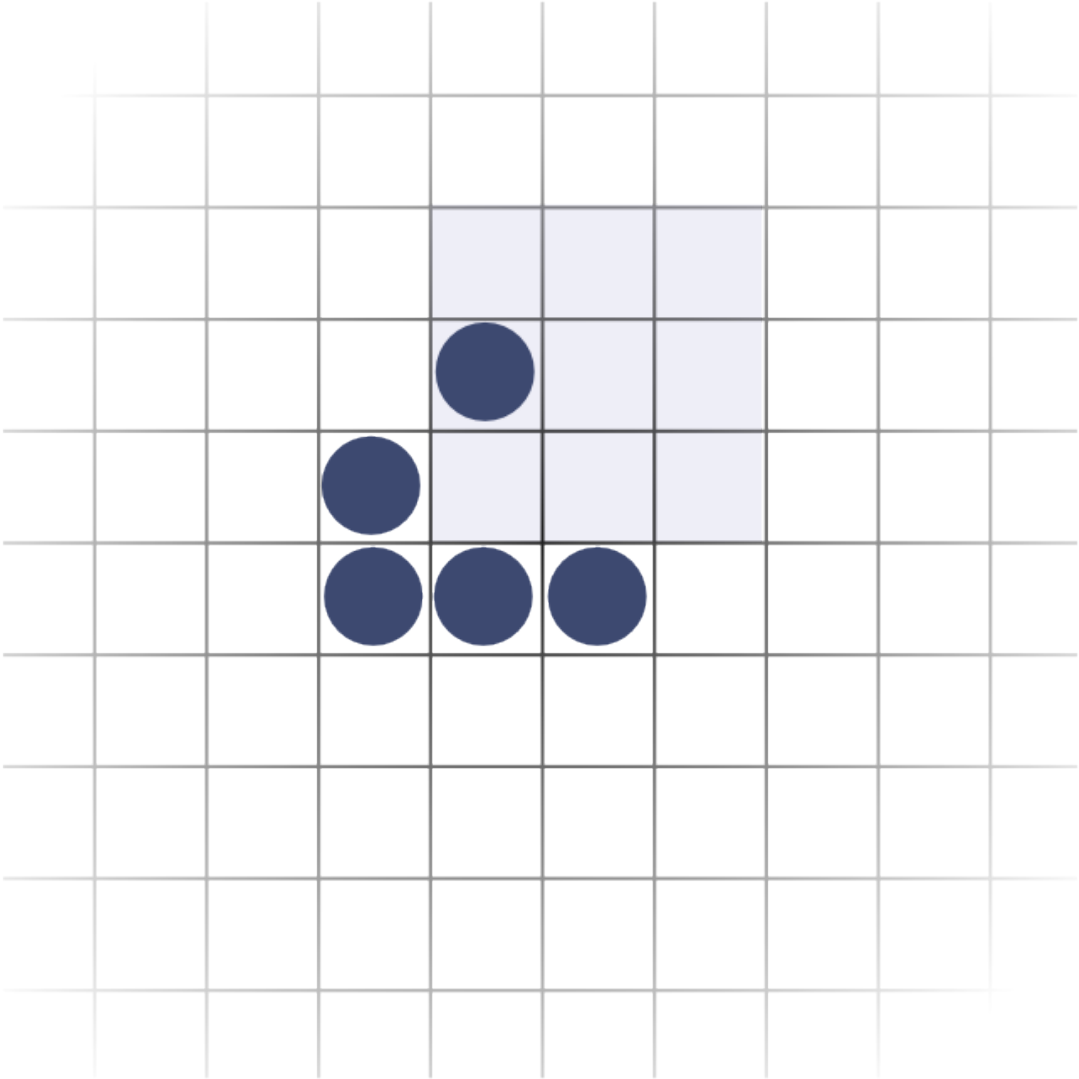
Глайдер



Глайдер



Глайдер



Задание 1.1

Координаты “живых” клеток колонии заданы списком пар координат x и y (список кортежей):

```
cells = [ (1,1), (2,1), (3,1), (3,2), (2,3) ]
```

Напишите программу, которая для заданных координат определяет занята клетка или свободна:

```
x = 1
y = 3
...
...
print(isOccuped)
```

False

Задание 1.2

Координаты “живых” клеток колонии заданы списком пар координат x и y (список кортежей):

```
cells = [ (1,1), (2,1), (3,1), (3,2), (2,3) ]
```

Напишите программу, которая для заданных координат клетки поля определяет количество её соседей:

```
x = 1
y = 1
...
...
print(neighbors)
```

Задание 1.3

Координаты “живых” клеток колонии заданы списком пар координат x и y (список кортежей):

```
cells = [ (1,1), (2,1), (3,1), (3,2), (2,3) ]
```

Напишите программу, которая определяет список неповторяющихся пар координат клеток-соседей колонии, вместе с клетками колонии:

```
...
...
print(candidates)
```

Задание 1.4

Координаты “живых” клеток колонии заданы множеством пар координат x и y (множество кортежей):

```
cells = set([ (1,1), (2,1), (3,1), (3,2), (2,3) ])
```

Напишите программу, которая для заданных координат клетки поля определяет количество её соседей:

```
x = 1
y = 1
...
...
print(neighbors)
```

Задание 1.5

Координаты “живых” клеток колонии заданы множеством пар координат x и y (множество кортежей):

```
cells = set([ (1,1), (2,1), (3,1), (3,2), (2,3) ])
```

Напишите программу, которая определяет множество пар координат клеток-соседей колонии, вместе с клетками колонии:

```
x = 1
y = 1
...
...
print(candidates)
```

Список источников

1. Простой Python. Современный стиль программирования. — СПб.: Питер, 2016. — 480 с.: ил. — (Серия «Бестселлеры O'Reilly»).
2. Python 3 для начинающих `https://pythonworld.ru`
3. Python documentation `https://docs.python.org`