PMR3201 Computação para Automação
Aula de Laboratório 1

O Jogo da Vida / The Game of Life

Newton Maruyama
Thiago de Castro Martins
Marcos S. G. Tsuzuki
Rafael Traldi Moura

12 de março de 2019

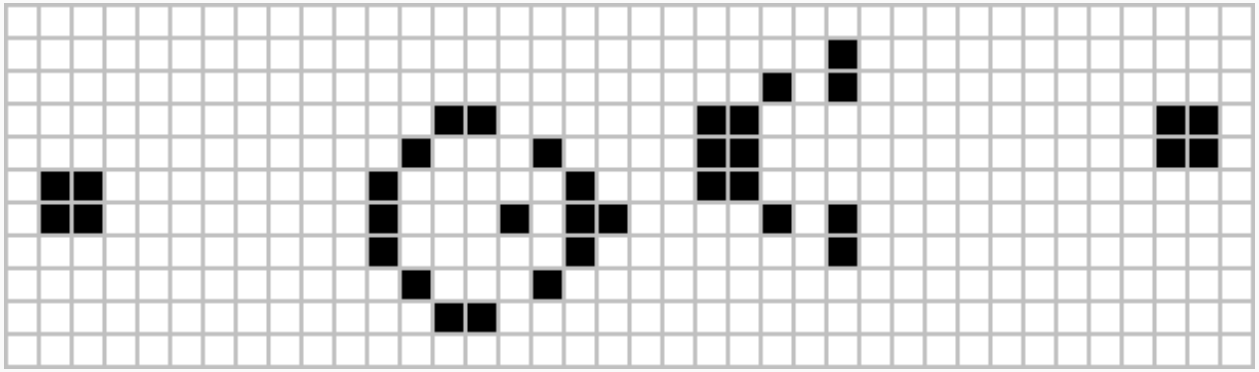
PMR-EPUSP

Conteúdo

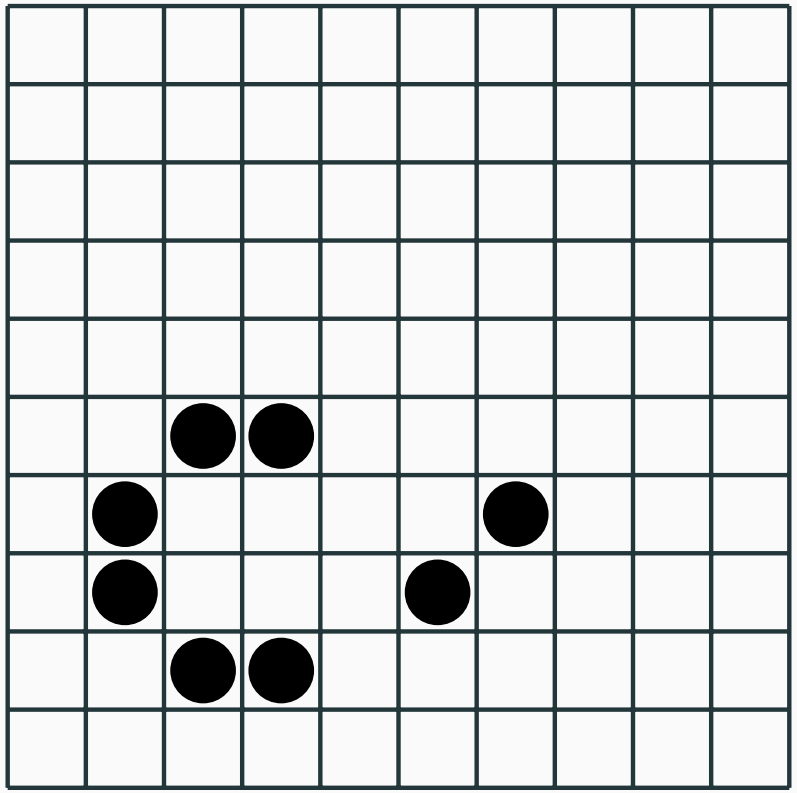
Apresentação do problema

O Jogo da Vida

- O *Jogo da Vida*, em Inglês, *Game of Life*, foi desenvolvido em 1970 pelo matemático J. H. Conway para a modelagem de organismos vivos, e consiste em uma simulação do desenvolvimento de uma colônia de células.
- O *Jogo da Vida* pode ser classificado como um automato celular, conceito este, desenvolvido por Stanislaw Ulam e John Von Neumann nos anos 40.
- O interesse teórico pelo *Jogo da Vida* se deve ao fato de que algoritmo é uma *Máquina Universal de Turing*, ou seja, qualquer problema computável pode ser computado pelo Jogo da Vida.



- O Universo do *Jogo da Vida* consiste de um *grid* bidimensional de células. Cada célula pode se encontrar em dois estados, *viva* ou *morta*.
- Cada célula interage com oito células vizinhas (horizontais, verticais e diagonais).

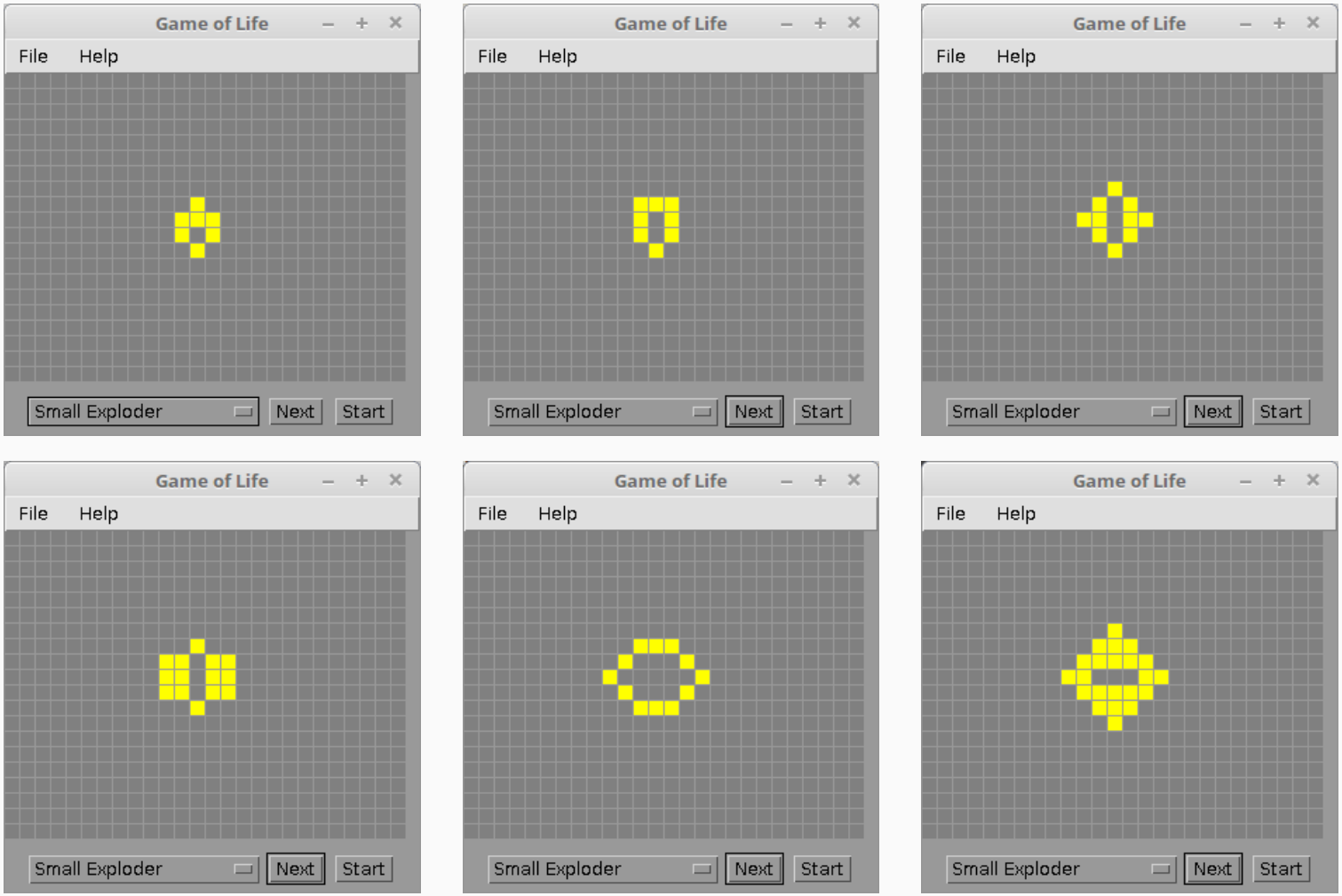


As regras do jogo

1. A colônia consiste em uma matriz quadrada contendo células vivas e mortas,
2. As células vivem ou morrem de geração em geração,
3. A mudança de geração acontece instantaneamente.
   A mudança futura no estado de uma célula não interfere no nascimento ou morte de outras células,
4. Uma célula tem oito vizinhos (verticais, horizontais e diagonais),
5. A “borda” da colônia está sempre morta,
6. Uma célula viva com apenas um ou nenhum vizinho vivo deverá morrer na próxima geração devido a isolamento,
7. Uma célula viva com quatro ou mais vizinhos vivos deverá morrer na próxima geração devido à superpopulação,
8. Se uma célula estiver morta e tiver três vizinhos vivos, haverá nascimento de uma célula na próxima geração,
9. Todos os outros casos continuam sem mudança de geração a geração.

- As regras descritas acima correspondem ao *Jogo da Vida padrão*.
- Tais regras são usualmente resumidas através dos símbolos B3/S23 (Born 3, Stays alive 2 and 3).
- Várias outras regras já foram testadas como por exemplo B6/S16, B36/S23, B3/S236, B3/S2367, etc.

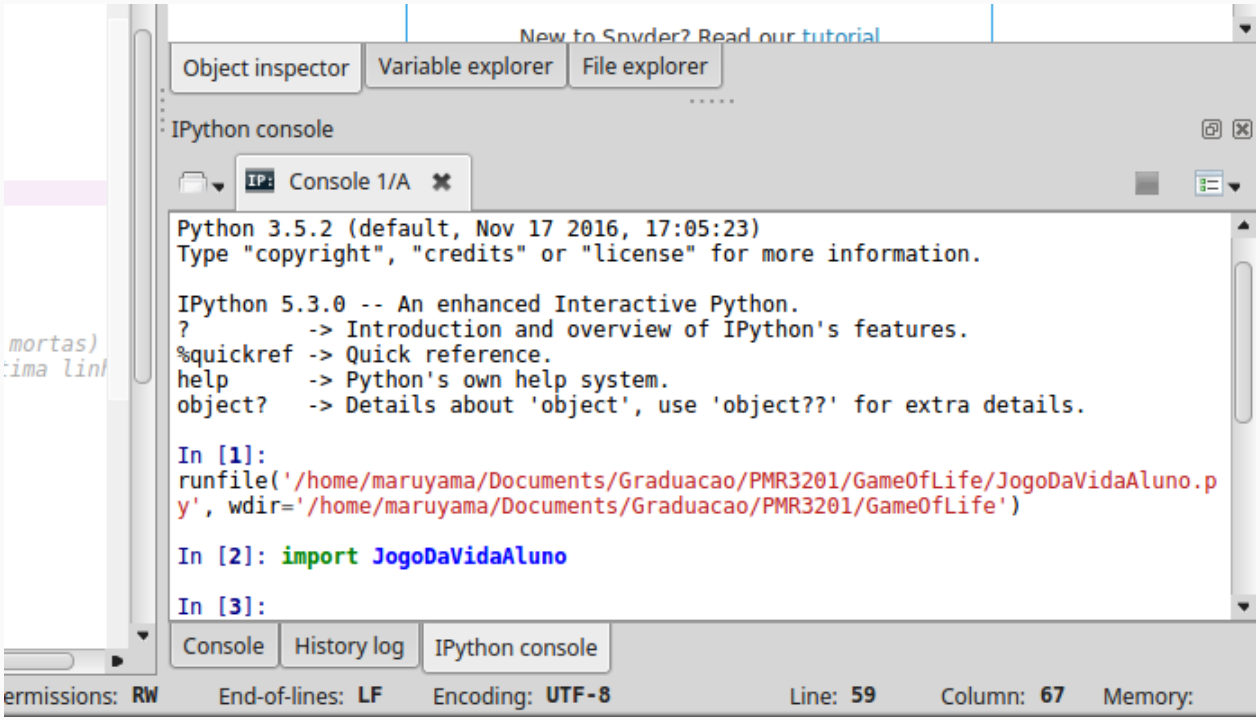
Evolução do padrão denominado *Small Exploder*



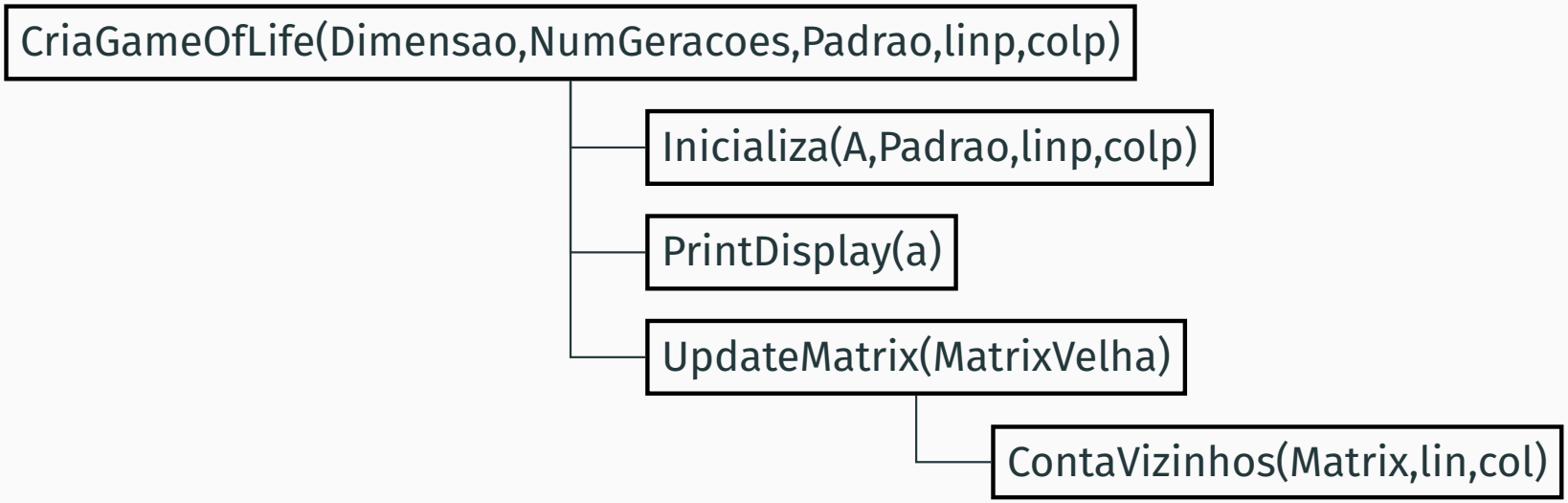
Descrição do programa

Descrição do programa

- O módulo do código do programa de simulação do *Jogo da Vida* está implementado no arquivo **JogoDaVidaAluno.py**.
- O módulo deve ser importado no console da IDE *Spyder*.



Árvore de funções do programa



Função CriaGameOfLife

- A função principal é denominada **CriaGameOfLife**.
- O código de **CriaGameOfLife** é apresentado a seguir.

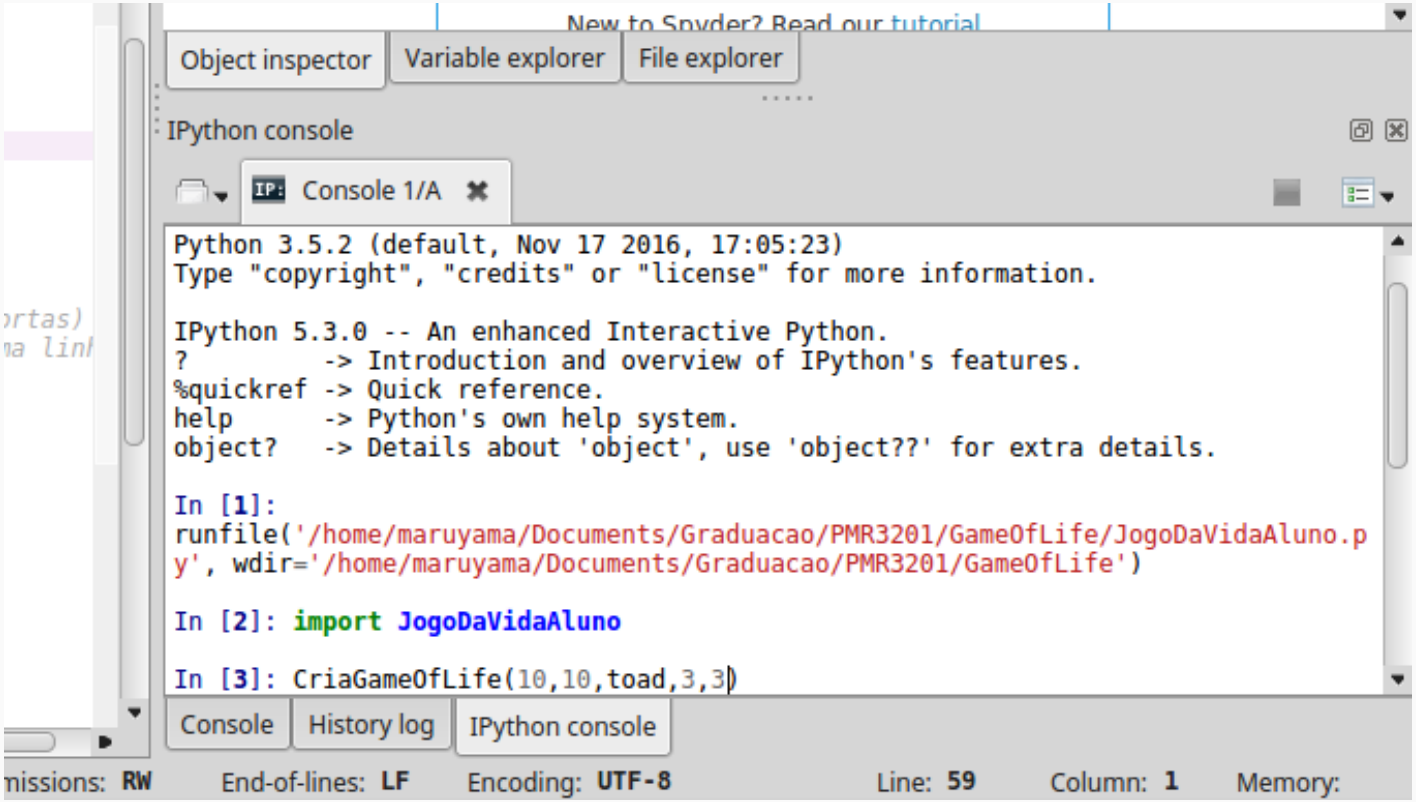
```
def CriaGameOfLife(Dimensao,NumGeracoes,Padrao,linp,colp):
    flag=False
    Matrix=[[False for y in range(Dimensao)] for x in range(Dimensao)]
    if Inicializa(Matrix,Padrao,linp,colp):
      PrintDisplay(Matrix)              # estado inicial
      for k in range(1,NumGeracoes+1):
         input('Press enter to continue: ')
         Matrix = UpdateMatrix(Matrix)
         PrintDisplay(Matrix)
         # print Matrix
      flag=True
    return(flag)
```

Parâmetros de entrada

- Os parâmetros de entrada de **CriaGameOfLife** são descritos a seguir:
  - **Dimensao**: dimensão do *grid*.
  - **NumGeracoes**: número gerações que devem ser simuladas.
  - **Padrao**: matriz que contêm o padrão inicial da colônia de células que deve ser simulada.
  - **(linp,colp)**: coordenadas do grid aonde o canto superior esquerdo do padrão deve ser colocado.
  - **linp**: índice da linha no grid aonde o padrão deve ser colocado.
  - **colp**: índice da coluna no grid aonde o padrão deve ser colocado.

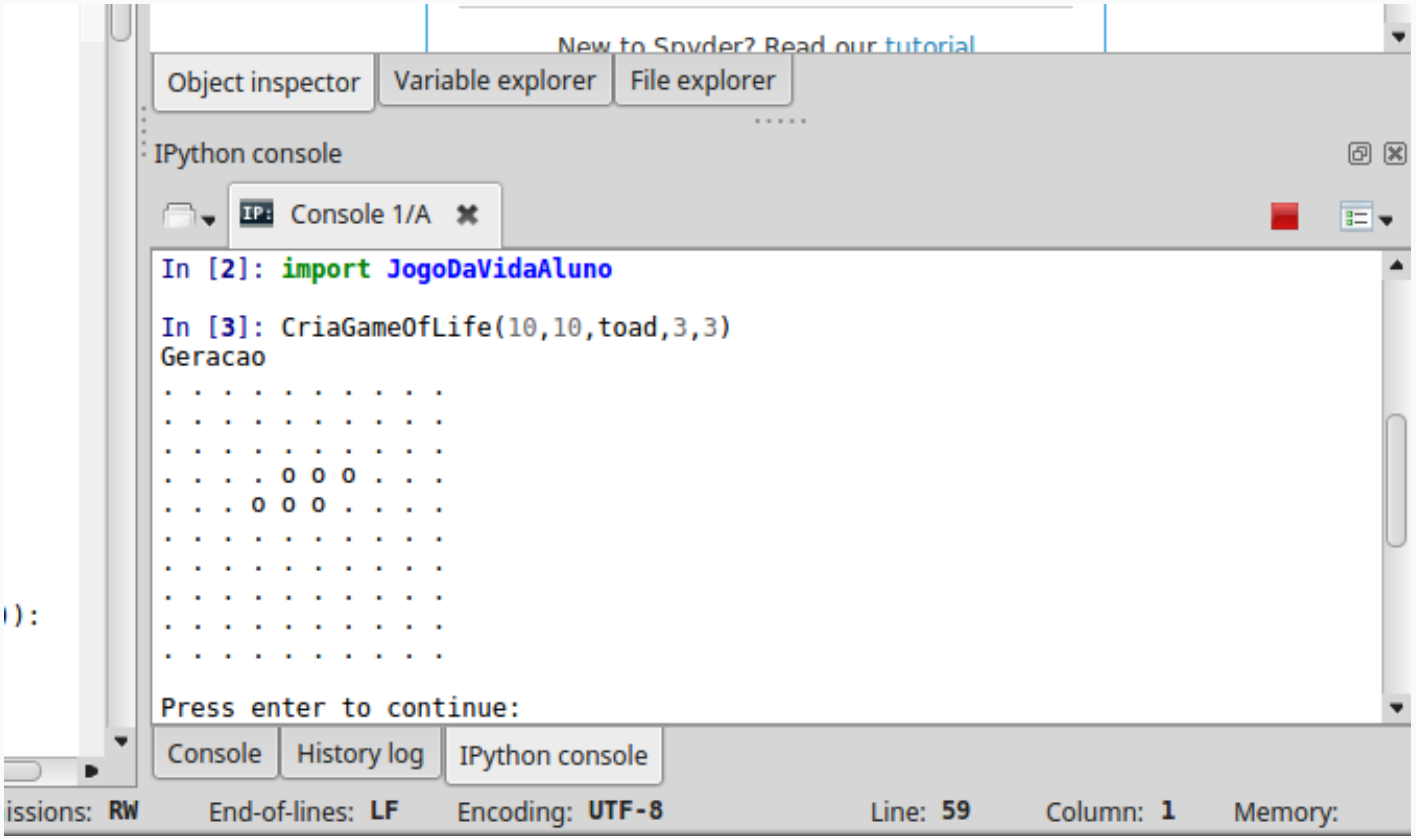
Como utilizar a função CriaGameOfLife

- Exemplo de utilização da função `CriaGameOfLife`.
- `Dimensao=10, NumGeracoes=10,Padrao=toad,linp=3,colp=3`
- Ao final do arquivo `JogoDaVidaAluno.py` existem padrões pré-definidos.



Como utilizar a função CriaGameOfLife

- Ao pressionar *Enter* o programa imprime na tela o estado incial da colônia de células e espera novamente um outro *Enter* para calcular a nova geração.



Estrutura de dados

- A principal estrutura de dados é denominada `Matrix` que define um `grid` de células. A estrutura é uma lista de listas formando um arranjo equivalente a um `array` bi-dimensional. Os elementos dessa variável são do tipo booleano.
- Apresenta-se a seguir a linha de código correspondente à criação de `Matrix`.

```
Matrix=[[False for y in range(Dimensao)] for x in range(Dimensao)]
```

  `Matrix` é inicializada com o valor `False`.
- `Matrix` é carregada na posicao (`linp`,`colp`), através da chamada da função `Inicializa`. com um array denominado `Padrao` que contêm um padrão espacial de uma colônia de células vivas. No `array` `Padrao` as células mortas são representadas por '0' e as células vivas são representadas por '1'.

```
   Inicializa(Matrix,Padrao,linp,colp)
```

- A função `Inicializa` troca os valor inteiro '0' por 'False' e o o valor '1' por 'True'.

Algoritmo principal

```
Cria matriz Matrix
flag = False          % indica erro no carregamento de Padrao
Inicializa Matrix com Padrao
Se a inicialização foi possível
   {
   Imprime na Tela
   Para k=1 até NumGeracoes faça
      {
      espera tecla Enter
      Calcula a nova Matrix
      Imprime na Tela
      }
   flag = True
   }
retorna(flag)
```

Outras funções

- `UpdateMatrix(Matrix)`: A partir do *grid* de células representado por `Matrix` o algoritmo da função calcula o *grid* de células correspondente à proxima geração da colônia de células.
- `ContaVizinhos(Matrix,lin,col)`: Aado uma célula cuja posição no `grid` é dada por `(lin,col)` a algoritmo da função deve contar o número de células vizinhas que estão vivas.
- `PrintDisplay(a)`: Imprime o *grid* de células representado por `Matrix` representando 'True' por 'o' e 'False' por '.'.

Para você fazer

Funções que você deve projetar

- Modifique a função `UpdateMatrix(MatrixVelha)`

```
def UpdateMatrix(MatrixVelha): # Calcula a nova geracao de celulas
   # recupero as dimensoes da matriz
   numlines = len(MatrixVelha)
   numcols = len(MatrixVelha[0])
   # defino uma nova matrix de celulas com as mesmas dimensoes
   MatrixNova=[[False for y in range(numcols)] for x in range(numlines)]
                                  # a borda da matriz e' sempre False
                                  # (celulas mortas)
   for i in range(1,numlines-1):  # comeca da segunda linha e
                                  # termina na penultima linha
      for j in range(1,numcols-1): # idem para colunas
        # Coloque aqui o seu codigo
        # contendo as regras de vida e morte
        # Utilize a funcao ContaVizinhos()
        MatrixNova[i][j]=MatrixVelha[i][j]
   return MatrixNova
```

Funções que você deve projetar

- Modifique a função `ContaVizinhos(Matrix, lin, col)`

```
def ContaVizinhos(Matrix, lin, col): # Conta o numero de celulas
                                     # vizinhas que estao vivas
                                     # (lin,col) posicao da celula
    contador = 0
        # (lin,col) posicao da celula
        # Dimensoes do array Matrix nao sao conhecidos
        # Pressuposto que indices (lin,col) nao se referem a uma celula
        # na borda do array

    # coloque o seu codigo aqui
    return contador
```