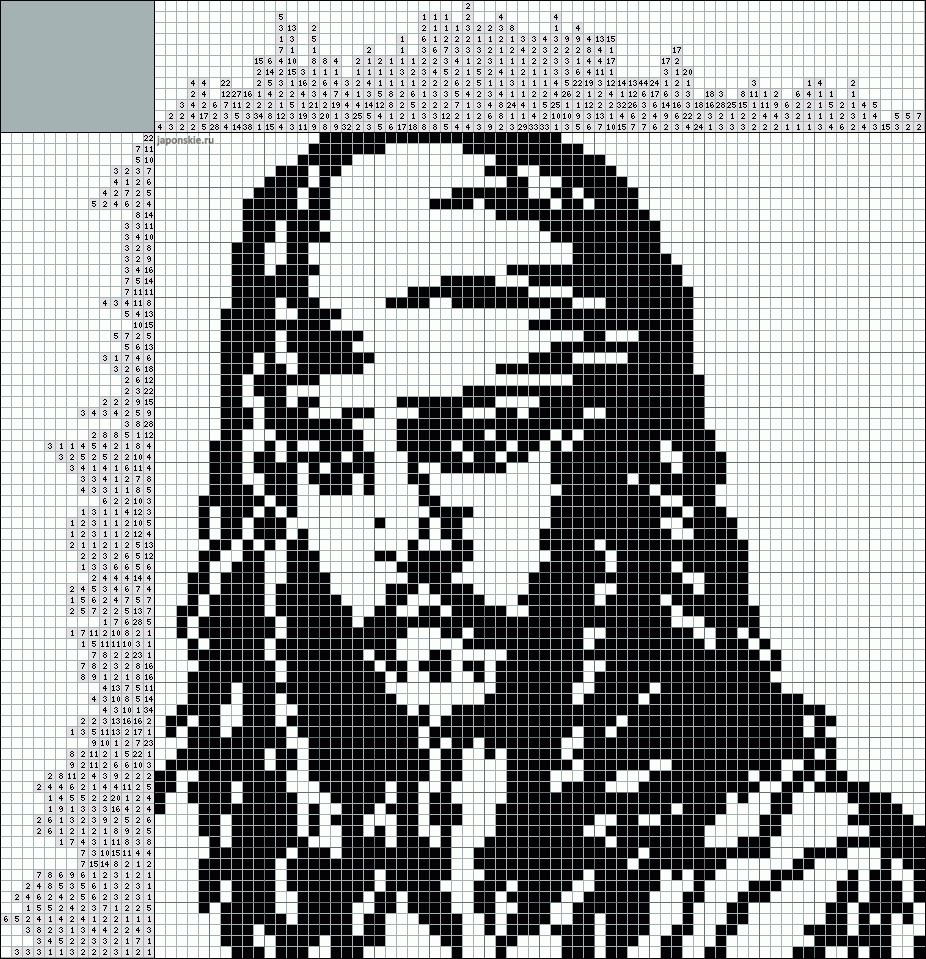
|  |
| --- |
| **5BHIT** |
| Nonogramm in Python |
| Software Development |

|  |
| --- |
| Daniel Scheuch - Florian Dienesch  **30.12.2014** |



**Content**

[READ ME 1](#_Toc411449551)

[Tasks 2](#_Toc411449552)

[Cost estimation 3](#_Toc411449553)

[Accopmplishment 4](#_Toc411449554)

[Code 4](#_Toc411449555)

[Starter 4](#_Toc411449556)

[Controller 4](#_Toc411449557)

[VIEW 7](#_Toc411449558)

[MODEL 10](#_Toc411449559)

[OBJEKTE 10](#_Toc411449560)

[Sphinx 13](#_Toc411449561)

# READ ME

You can play our Nonogramm if you download our Git repo and open the Starter Folder an execute the starter.py script. <https://github.com/scheichi/Nonogramm.git> its public! There you are also going to find our sphinx documentation (Protocoll/Doc/\_build/html).

# Tasks

Nachdem Sie einige Designpattern in Python betrachtet haben, wollen wir uns einem wichtigen Entwurfsmuster spielerisch nähern:

In einem Team (2) soll das Spiel Nonogramm umgesetzt werden.

* Spielfeld: 15 x 15
* Eine Statusleiste mit Anzeige der noch gesuchten Felder,
* Button zur sofortigen Lösung
* Button zum Neustart

Auswahlfeld zur Einstellung der Schwierigkeit (EASY/200; MEDIUM/150; HARD/125; EXPERT/90; IMPOSSIBLE/50) auf Basis der gesuchten Felder!

Die Farbe rosa ist natürlich nicht Pflicht und könnte vielleicht vom User variabel eingestellt werden.

Viel Erfolg!

Ressourcen:

Unterlagen zu GUI-Programmierung in Python

https://de.wikipedia.org/wiki/Nonogramm

# Cost estimation

|  |  |  |
| --- | --- | --- |
| Pattern | Should-Hours | Be-Hours |
| Model | Scheuch | 0.5 | 0.5 |
| Algorithmus | Scheuch | 1.5 | 2 |
| View | Dienesch | 0.5 | 1 |
| Controller | Dienesch | 0.5 | 1 |
|  |  | 0 |
| Protocoll | Dienesch + Scheuch | 0,5 | 0.5 |
| **General** | **3,5** | **~5** |

# Accopmplishment

## Code

### Starter

\_\_author\_\_ = **'daniel'  
import** sys  
**from** PyQt4 **import** QtCore, QtGui  
**from** Controller.nonoControl **import** NonoController  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 app = QtGui.QApplication(sys.argv)  
 myapp = NonoController()  
 myapp.show()  
 sys.exit(app.exec\_())

### Controller

\_\_author\_\_ = **'floriandienesch-danielscheuch'  
  
import** sys, random  
**from** random **import** randint  
**from** PyQt4 **import** QtCore, QtGui  
**from** View.nonoView **import** Ui\_Dialog  
**from** Model.nonoModel **import** NonoModel  
**from** Objekte.spielfeld **import** Spielfeld  
  
**class** NonoController(QtGui.QMainWindow):  
 *"""  
 MVC pattern: Creates a controller - mvc pattern.  
 """* **def** \_\_init\_\_(self, parent=**None**):  
 *"""  
 Create a new controller with a object MyView  
 and a object MyModel using the mvc pattern.* **:param** *parent:  
 """* QtGui.QWidget.\_\_init\_\_(self, parent)  
 self.ui = Ui\_Dialog()  
 self.model = NonoModel()  
 self.offeneFelder = self.model.nonogram.felder  
 self.ui.setupUi(self, self)  
 **for** x **in** range(16):  
 **for** y **in** range(16):  
 self.ui.spielfeld.setItem(x, y, QtGui.QTableWidgetItem())  
  
 **def** setRechts(self, rechts):  
 *"""  
 Setzt die rechten Angaben für den User* **:param** *rechts: Liste der Angabe* **:return***:  
 """* **for** x **in** range(len(rechts)):  
 **for** y **in** range(len(rechts[x])):  
 self.ui.rechts.setItem(x, y, QtGui.QTableWidgetItem(str(rechts[x][y])))  
  
 **def** setOben(self, oben):  
 *"""  
 Setzt die oberen Angaben für den User* **:param** *oben: Liste der Angabe* **:return***:  
 """* **for** x **in** range(len(oben)):  
 **for** y **in** range(len(oben[x])):  
 self.ui.oben.setItem(8 - len(oben[x]) + y, x, QtGui.QTableWidgetItem(str(oben[x][y])))  
  
 **def** changeBg(self, row, column):  
 *"""  
 Methode changeBg  
 Diese Methode aendert die Hintergrundfarbe von einer spezifischen  
 Flaeche wenn man darauf klickt  
 führt die statistik* **:param** *row: row of that tile* **:param** *column: column of that tile* **:return***:  
 """  
 #print("Zeile %d und Spalte %d" % (row, column))* **if** self.ui.spielfeld.item(row, column).backgroundColor() == QtGui.QColor(**'white'**):  
 self.model.feld[row \* self.model.nonogram.laenge + column] = 1  
 self.ui.spielfeld.item(row, column).setBackground(QtGui.QColor(**'blue'**))  
  
 **elif** self.ui.spielfeld.item(row, column).backgroundColor() == QtGui.QColor(**'blue'**):  
 self.model.feld[row \* self.model.nonogram.laenge + column] = 0  
 self.ui.spielfeld.item(row, column).setBackground(QtGui.QColor(**'orange'**))  
  
 **elif** self.ui.spielfeld.item(row, column).backgroundColor() == QtGui.QColor(**'orange'**):  
 self.model.feld[row \* self.model.nonogram.laenge + column] = **None** self.ui.spielfeld.item(row, column).setBackground(QtGui.QColor(**'white'**))  
  
 **else**:  
 self.model.feld[row \* self.model.nonogram.laenge + column] = 1  
 self.ui.spielfeld.item(row, column).setBackground(QtGui.QColor(**'blue'**))  
  
 self.offeneFelder = self.model.nonogram.felder  
 **for** x **in** range(0, len(self.model.feld)):  
 **if** self.model.feld[x] == self.model.nonogram.spielfeld[x]:  
 self.offeneFelder -= 1  
 self.ui.textBrowserFelderOffen.setText(str(self.offeneFelder))  
  
 **def** loesung(self):  
 *"""  
 Method loesung  
 Zeigt die Loesung an indem die Spielflaechen gefaerbt werden* **:return***:  
 """* **for** x **in** range(self.model.nonogram.laenge):  
 **for** y **in** range(self.model.nonogram.laenge):  
 **if** self.model.nonogram.spielfeld[x \* self.model.nonogram.laenge + y] == 1:  
 self.ui.spielfeld.item(x, y).setBackground(QtGui.QColor(**'blue'**))  
 **elif** self.model.nonogram.spielfeld[x \* self.model.nonogram.laenge + y] == 0:  
 self.ui.spielfeld.item(x, y).setBackground(QtGui.QColor(**'orange'**))  
  
 **def** neustart(self):  
 *"""  
 Methode neustart  
 Setzt alle Spielzuege sowie Statisiken zurueck* **:return***:  
 """* self.ui.spielfeld.clear()  
 **for** x **in** range(16):  
 **for** y **in** range(16):  
 self.ui.spielfeld.setItem(x, y, QtGui.QTableWidgetItem())  
 self.ui.oben.setItem(x, y, QtGui.QTableWidgetItem())  
 self.ui.rechts.setItem(x, y, QtGui.QTableWidgetItem())  
  
 self.offeneFelder = self.model.nonogram.felder  
 self.ui.textBrowserFelderOffen.setText(str(self.offeneFelder))  
  
 self.model.nonogram = Spielfeld(self.model.difficult)  
 self.model.angabe = self.model.nonogram.checkSpielfeld()  
  
 self.setOben(self.model.angabe[1])  
 self.setRechts(self.model.angabe[0])  
  
 **def** changeDifficulty(self, item):  
 *"""  
 Methode changeDifficulty  
 Aendert den Schwierigkeitsgrad* **:param** *item:* **:return***:  
 """* **if** item == **"EASY"**:  
 self.model.difficult = 1  
  
 **elif** item == **"MEDIUM"**:  
 self.model.difficult = 2  
  
 **elif** item == **"HARD"**:  
 self.model.difficult = 3  
  
 **elif** item == **"EXPERT"**:  
 self.model.difficult = 4  
  
 **elif** item == **"IMPOSSIBLE"**:  
 self.model.difficult = 5  
  
 **def** flash(self, grad):  
 self.model.gradR = grad  
 self.setStyleSheet((**"#Dialog {""background: rgb("**+str(self.model.gradR)+**", "**+str(self.model.gradG)+**", "**+str(self.model.gradB)+**");""}"**))  
 **return** grad  
  
 **def** flashG(self, grad):  
 self.model.gradG = grad  
 self.setStyleSheet((**"#Dialog {""background: rgb("**+str(self.model.gradR)+**", "**+str(self.model.gradG)+**", "**+str(self.model.gradB)+**");""}"**))  
  
 **def** flashB(self, grad):  
 self.model.gradB = grad  
 self.setStyleSheet((**"#Dialog {""background: rgb("**+str(self.model.gradR)+**", "**+str(self.model.gradG)+**", "**+str(self.model.gradB)+**");""}"**))  
  
 **def** stock(self):  
 self.setStyleSheet((**"#Dialog {background: rgb(255, 183, 227);""}"**))

### VIEW

*# -\*- coding: utf-8 -\*-  
  
# Form implementation generated from reading ui file '/Users/floriandienesch/Desktop/nonoUI.ui'  
#  
# Created: Sun Jan 25 17:26:55 2015  
# by: PyQt4 UI code generator 4.11.3  
#  
# WARNING! All changes made in this file will be lost!***from** PyQt4 **import** QtCore, QtGui  
  
**try**:  
 \_fromUtf8 = QtCore.QString.fromUtf8  
**except** AttributeError:  
 **def** \_fromUtf8(s):  
 **return** s  
  
**try**:  
 \_encoding = QtGui.QApplication.UnicodeUTF8  
 **def** \_translate(context, text, disambig):  
 **return** QtGui.QApplication.translate(context, text, disambig, \_encoding)  
**except** AttributeError:  
 **def** \_translate(context, text, disambig):  
 **return** QtGui.QApplication.translate(context, text, disambig)  
  
**class** Ui\_Dialog(object):  
 **def** setupUi(self, Dialog, nonoControl):  
 Dialog.setObjectName(\_fromUtf8(**"Dialog"**))  
 Dialog.resize(750, 680)  
 Dialog.setStyleSheet(\_fromUtf8(**"#Dialog {\n" " background: rgb(255, 183, 227);\n""}"**))  
 self.oben = QtGui.QTableWidget(Dialog)  
 self.oben.setEnabled(**False**)  
 self.oben.setGeometry(QtCore.QRect(20, 10, 451, 241))  
 self.oben.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)  
 self.oben.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)  
 self.oben.setRowCount(8)  
 self.oben.setColumnCount(15)  
 self.oben.setObjectName(\_fromUtf8(**"oben"**))  
 self.oben.horizontalHeader().setVisible(**False**)  
 self.oben.horizontalHeader().setDefaultSectionSize(30)  
 self.oben.horizontalHeader().setHighlightSections(**True**)  
 self.oben.verticalHeader().setVisible(**False**)  
 self.spielfeld = QtGui.QTableWidget(Dialog)  
 self.spielfeld.setGeometry(QtCore.QRect(20, 270, 451, 361))  
 self.spielfeld.setFocusPolicy(QtCore.Qt.NoFocus)  
 self.spielfeld.setAcceptDrops(**False**)  
 self.spielfeld.setAutoFillBackground(**False**)  
 self.spielfeld.setFrameShape(QtGui.QFrame.NoFrame)  
 self.spielfeld.setFrameShadow(QtGui.QFrame.Plain)  
 self.spielfeld.setLineWidth(0)  
 self.spielfeld.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)  
 self.spielfeld.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)  
 self.spielfeld.setEditTriggers(QtGui.QAbstractItemView.NoEditTriggers)  
 self.spielfeld.setSelectionMode(QtGui.QAbstractItemView.NoSelection)  
 self.spielfeld.setSelectionBehavior(QtGui.QAbstractItemView.SelectItems)  
 self.spielfeld.setRowCount(15)  
 self.spielfeld.setColumnCount(15)  
 self.spielfeld.setObjectName(\_fromUtf8(**"spielfeld"**))  
 self.spielfeld.horizontalHeader().setVisible(**False**)  
 self.spielfeld.horizontalHeader().setDefaultSectionSize(30)  
 self.spielfeld.horizontalHeader().setMinimumSectionSize(4)  
 self.spielfeld.verticalHeader().setVisible(**False**)  
 self.spielfeld.verticalHeader().setDefaultSectionSize(24)  
 self.spielfeld.verticalHeader().setMinimumSectionSize(20)  
 self.rechts = QtGui.QTableWidget(Dialog)  
 self.rechts.setEnabled(**False**)  
 self.rechts.setGeometry(QtCore.QRect(490, 270, 241, 361))  
 self.rechts.setAutoFillBackground(**False**)  
 self.rechts.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)  
 self.rechts.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)  
 self.rechts.setRowCount(15)  
 self.rechts.setColumnCount(8)  
 self.rechts.setObjectName(\_fromUtf8(**"rechts"**))  
 self.rechts.horizontalHeader().setVisible(**False**)  
 self.rechts.horizontalHeader().setDefaultSectionSize(30)  
 self.rechts.horizontalHeader().setMinimumSectionSize(4)  
 self.rechts.verticalHeader().setVisible(**False**)  
 self.rechts.verticalHeader().setDefaultSectionSize(24)  
 self.rechts.verticalHeader().setMinimumSectionSize(20)  
 self.buttonLoesung = QtGui.QPushButton(Dialog)  
 self.buttonLoesung.setGeometry(QtCore.QRect(390, 640, 85, 32))  
 self.buttonLoesung.setObjectName(\_fromUtf8(**"buttonLoesung"**))  
 self.buttonNeustart = QtGui.QPushButton(Dialog)  
 self.buttonNeustart.setGeometry(QtCore.QRect(640, 640, 92, 32))  
 self.buttonNeustart.setObjectName(\_fromUtf8(**"buttonNeustart"**))  
 self.comboSchwierigkeit = QtGui.QComboBox(Dialog)  
 self.comboSchwierigkeit.setGeometry(QtCore.QRect(490, 640, 133, 26))  
 self.comboSchwierigkeit.setObjectName(\_fromUtf8(**"comboSchwierigkeit"**))  
 self.comboSchwierigkeit.addItem(\_fromUtf8(**""**))  
 self.comboSchwierigkeit.addItem(\_fromUtf8(**""**))  
 self.comboSchwierigkeit.addItem(\_fromUtf8(**""**))  
 self.comboSchwierigkeit.addItem(\_fromUtf8(**""**))  
 self.comboSchwierigkeit.addItem(\_fromUtf8(**""**))  
 self.labelFelderOffen = QtGui.QLabel(Dialog)  
 self.labelFelderOffen.setGeometry(QtCore.QRect(20, 640, 71, 21))  
 self.labelFelderOffen.setObjectName(\_fromUtf8(**"labelFelderOffen"**))  
 self.labelFarbeR = QtGui.QLabel(Dialog)  
 self.labelFarbeR.setGeometry(QtCore.QRect(600, 180, 71, 21))  
 self.labelFarbeR.setObjectName(\_fromUtf8(**"labelFelderOffen"**))  
 self.labelFarbeG = QtGui.QLabel(Dialog)  
 self.labelFarbeG.setGeometry(QtCore.QRect(650, 180, 71, 21))  
 self.labelFarbeG.setObjectName(\_fromUtf8(**"labelFelderOffen"**))  
 self.labelFarbeB = QtGui.QLabel(Dialog)  
 self.labelFarbeB.setGeometry(QtCore.QRect(700, 180, 71, 21))  
 self.labelFarbeB.setObjectName(\_fromUtf8(**"labelFelderOffen"**))  
 self.textBrowserFelderOffen = QtGui.QTextBrowser(Dialog)  
 self.textBrowserFelderOffen.setEnabled(**False**)  
 self.textBrowserFelderOffen.setGeometry(QtCore.QRect(180, 640, 181, 31))  
 self.textBrowserFelderOffen.setObjectName(\_fromUtf8(**"textBrowserFelderOffen"**))  
 self.dialFarbeR = QtGui.QDial(Dialog)  
 self.dialFarbeR.setGeometry(QtCore.QRect(580, 200, 50, 64))  
 self.dialFarbeR.setMaximum(255)  
 self.dialFarbeR.setPageStep(1)  
 self.dialFarbeR.setObjectName(\_fromUtf8(**"dialFarbe"**))  
 self.dialFarbeG = QtGui.QDial(Dialog)  
 self.dialFarbeG.setGeometry(QtCore.QRect(630, 200, 50, 64))  
 self.dialFarbeG.setMaximum(255)  
 self.dialFarbeG.setPageStep(1)  
 self.dialFarbeG.setObjectName(\_fromUtf8(**"dialFarbeG"**))  
 self.dialFarbeB = QtGui.QDial(Dialog)  
 self.dialFarbeB.setGeometry(QtCore.QRect(680, 200, 50, 64))  
 self.dialFarbeB.setMaximum(255)  
 self.dialFarbeB.setPageStep(1)  
 self.dialFarbeB.setObjectName(\_fromUtf8(**"dialFarbeB"**))  
 self.buttonFarbeStock = QtGui.QPushButton(Dialog)  
 self.buttonFarbeStock.setGeometry(QtCore.QRect(480, 225, 100, 32))  
 self.buttonFarbeStock.setObjectName(\_fromUtf8(**"buttonFarbe"**))  
  
 self.retranslateUi(Dialog)  
 QtCore.QObject.connect(self.spielfeld, QtCore.SIGNAL(\_fromUtf8(**"cellClicked(int,int)"**)), nonoControl.changeBg)  
 QtCore.QObject.connect(self.buttonNeustart, QtCore.SIGNAL(\_fromUtf8(**"clicked()"**)), nonoControl.neustart)  
 QtCore.QObject.connect(self.comboSchwierigkeit, QtCore.SIGNAL(\_fromUtf8(**"currentIndexChanged(QString)"**)), nonoControl.changeDifficulty)  
 QtCore.QObject.connect(self.buttonLoesung, QtCore.SIGNAL(\_fromUtf8(**"clicked()"**)), nonoControl.loesung)  
 QtCore.QObject.connect(self.dialFarbeR, QtCore.SIGNAL(\_fromUtf8(**"valueChanged(int)"**)), nonoControl.flash)  
 QtCore.QObject.connect(self.dialFarbeG, QtCore.SIGNAL(\_fromUtf8(**"valueChanged(int)"**)), nonoControl.flashG)  
 QtCore.QObject.connect(self.dialFarbeB, QtCore.SIGNAL(\_fromUtf8(**"valueChanged(int)"**)), nonoControl.flashB)  
 QtCore.QObject.connect(self.buttonFarbeStock, QtCore.SIGNAL(\_fromUtf8(**"clicked()"**)), nonoControl.stock)  
 QtCore.QMetaObject.connectSlotsByName(Dialog)  
  
 **def** retranslateUi(self, Dialog):  
 Dialog.setWindowTitle(\_translate(**"Dialog"**, **"Dialog"**, **None**))  
 self.buttonLoesung.setText(\_translate(**"Dialog"**, **"Lösung"**, **None**))  
 self.buttonNeustart.setText(\_translate(**"Dialog"**, **"Neustart"**, **None**))  
 self.comboSchwierigkeit.setItemText(0, \_translate(**"Dialog"**, **"EASY"**, **None**))  
 self.comboSchwierigkeit.setItemText(1, \_translate(**"Dialog"**, **"MEDIUM"**, **None**))  
 self.comboSchwierigkeit.setItemText(2, \_translate(**"Dialog"**, **"HARD"**, **None**))  
 self.comboSchwierigkeit.setItemText(3, \_translate(**"Dialog"**, **"EXPERT"**, **None**))  
 self.comboSchwierigkeit.setItemText(4, \_translate(**"Dialog"**, **"IMPOSSIBLE"**, **None**))  
 self.labelFelderOffen.setText(\_translate(**"Dialog"**, **"Felder offen:"**, **None**))  
 self.labelFarbeR.setText(\_translate(**"Dialog"**, **"R"**, **None**))  
 self.labelFarbeG.setText(\_translate(**"Dialog"**, **"G"**, **None**))  
 self.labelFarbeB.setText(\_translate(**"Dialog"**, **"B"**, **None**))  
 self.buttonFarbeStock.setText(\_translate(**"Dialog"**, **"Stock Farbe"**, **None**))

### MODEL

\_\_author\_\_ = **'daniel'  
from** Objekte.spielfeld **import** Spielfeld  
**class** NonoModel():  
  
 *"""  
 MVC pattern: Creates a model - mvc pattern.  
 """* **def** \_\_init\_\_(self, parent=**None**):  
 self.difficult = 1  
 self.gradR = 0  
 self.gradG = 0  
 self.gradB = 0  
 self.nonogram = Spielfeld(self.difficult)  
 self.feld = [**None**] \* self.nonogram.felder  
 self.offeneFelder = 0  
 self.angabe = self.nonogram.checkSpielfeld()  
 **pass**

### OBJEKTE

\_\_author\_\_ = **'danielscheuch'  
from** random **import** randint  
  
**class** Spielfeld():  
 *"""  
 Spielfeldobjekt  
 """* laenge = 15  
 felder = laenge\*laenge  
  
 **def** \_\_init\_\_(self, schwierigkeit):  
 *"""  
 Konstruktor  
 """* self.spielfeld = self.makeSpielfeld(schwierigkeit)  
  
 @property  
 **def** getspielfeld(self):  
 *"""  
 Spielfeld* **:return***: SPielfeld  
 """* **return** self.spielfeld  
  
 **def** makeSpielfeld(self, schwierigkeit):  
 *"""  
 Generiert durch Zufall das gewünschte Feld* **:param** *schwierigkeit: Schwierigkeit des Felds* **:return***: das generierte Spielfeld  
 """* spielfeld = []  
 **for** x **in** range(self.felder):  
 random = randint(0,10)  
 **if** schwierigkeit == 1:  
 **if** random > 9:  
 spielfeld.append(1)  
 **else**:  
 spielfeld.append(0)  
  
 **elif** schwierigkeit == 2:  
 **if** random > 7:  
 spielfeld.append(1)  
 **else**:  
 spielfeld.append(0)  
  
 **elif** schwierigkeit == 3:  
 **if** random > 5:  
 spielfeld.append(1)  
 **else**:  
 spielfeld.append(0)  
  
 **elif** schwierigkeit == 4:  
 **if** random > 3:  
 spielfeld.append(1)  
 **else**:  
 spielfeld.append(0)  
  
 **elif** schwierigkeit == 5:  
 **if** random > 2:  
 spielfeld.append(1)  
 **else**:  
 spielfeld.append(0)  
 **return** spielfeld  
  
 **def** checkHorizontal(self):  
 *"""  
 Prüft das SPielfeld für die Zahlen horizontal die für die User als Angabe nötig sind* **:return***:werte für die angabe horizontal  
  
 """* out = []  
 row = []  
 z = 0  
 **for** x **in** range(1, len(self.spielfeld)+1):  
 **if** self.spielfeld[x-1] == 1:  
 z = z+1  
 **elif** self.spielfeld[x-1] == 0:  
 **if** z != 0:  
 row.append(z)  
 z = 0  
 **if** x%self.laenge == 0 **and** x != 0:  
 **if** z != 0:  
 row.append(z)  
 z = 0  
 out.append(row)  
 row = []  
 **return** out  
  
 **def** checkVertikal(self):  
 *"""  
 Prüft das SPielfeld für die Zahlen vertikal die für die User als Angabe nötig sind* **:return***: werte für die angabe vertikal  
 """* out = []  
 row = []  
 z = 0  
 **for** x **in** range(self.laenge):  
 **for** y **in** range(self.laenge):  
 **if** self.spielfeld[x+y\*self.laenge] == 1:  
 z = z+1  
 **elif** self.spielfeld[x+y\*self.laenge] == 0:  
 **if** z != 0:  
 row.append(z)  
 z = 0  
 **else**:  
 **if** z != 0:  
 row.append(z)  
 z = 0  
 out.append(row)  
 row = []  
 **return** out  
  
 **def** checkSpielfeld(self):  
 *"""  
 prüft generiertes spielfeld für die eingabe* **:return***: array mit 2 stellen das array enthaltet mit jeweiligen angaben  
 """* **return** [self.checkHorizontal(), self.checkVertikal()]  
  
 **def** \_\_str\_\_(self):  
 *"""  
 toString Methode* **:return***: Output String (x/y)  
 """* out = **""  
 for** x **in** range(1, len(self.spielfeld)+1):  
 **if** x%self.laenge == 0 **and** x **is not** 0:  
 out += str(self.spielfeld[x-1]) + **"\n"  
 else**:  
 out += str(self.spielfeld[x-1])  
 **return** out

## Sphinx

Because of troubles at executeing quickstart the first time we had to fix some things in the conf.py file:

In row 1461 we changed “for k, v in d.items():" to "for k, v in list(d.items()):" and “extensions = []” to and “extensions = ["spinx.ext.autodoc"]” and added “sys.path.insert(0, os.pathabspath("../.."))”

After them changes everything runned without problems.

As example:

