Modulabschlussprüfung Programmierung II

TRUE	FALSE	
X		In numpy können einzelne Arrays (sprich Listen) nur einen einzigen Datentypen enthalten.
X		Modul matplotlib muss zusätzlich installiert werden, es gehört nicht zum Standartumfang von Python.
X		Mit dem Qt Designer können GUI erstellt werden. Diese werden in XML basierten .ui-Files gespeichert.
X		Für QGIS können Plugins mittels Python programmiert werden.
X		Mit QGIS können unter anderem Shapefiles und GeoTIFF geöffnet werden.
X		Das Jupyter Notebook ist eine Web-Applikation. Dabei kann unter anderem Python-Code eingegeben werden und das Resultat wird direkt im Webbrowser angezeigt.
X		Das Modul numpy muss zusätzlich installiert werden, es gehört nicht zum Standardumfang von Python.
X		Das Submodul matplotlib.patches ermöglicht das Zeichnen von Figuren wie Ellipsen, Polygone, Pfeile etc.
X		QGIS ist Open Source, das heisst, der gesamte Quelltext von QGIS ist frei verfügbar und kann gegebenenfalls angepasst oder erweitert werden.
X		Eine Klasse diest als Bauplan für die Abbildung von realen Objekten in Softwareobjekte und beschreibt Attribute (Eigenschaften) und Methoden (Verhaltensweisen) der Objekte.
X		Klassen können miteinander in hierarchischen Beziehungen stehen und dabei zu komplexen Strukturen werden

X		Ist die Klasse so korrekt:
		class Modulabschlussprüfung:
		`def __init__(self, note):`
		`self.note = note`
		`def genuegend(self):`
		`return self.note >=3.75`

X		https://www.python.org/ ist die offizielle Homepage der Python Programmiersprache (von der Python Software Foundation).
	X	Das Programm stürtzt nach der Installation von numpy ab: import numpy as np `s = np.array([1,2,3])`
	X	Magische Methoden könne auch ausserhalb von Klassen definiert werden.
	X	In Numpy wird mit np.arange(1,5) folgendes array erstellt: `array([1, 2, 3, 4, 5])`

Aufgabe 2
Schreiben sie eine Klasse „QESolver“, welche eine Quadratische Gleichung lösen soll.

Zur Erinnerung: Die quadratische Gleichung $ax^2 + bx + c = 0$ hat im allgemeinen die beiden Lösungen:

$$x_{1/2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a} \qquad (a \neq 0)$$

- Schreiben Sie dazu eine Klasse mit sinnvollen Methoden
- Die Lösung soll als tuple zurückgegeben werden. Gibt es nur eine Lösung, so gibt es im Tuple nur einen Wert. Gibt es keine Lösung, wird ein leeres Tuple zurückgegeben, ansonsten zwei Werte.
- Erstellen Sie eine Instanz mit konkretem Beispiel

```
class QESolver:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
        if self.a == 0:
            raise ValueError("a darf nicht Null sein")

    def rechnen (self):
        diskriminante = ((self.b**2 - 4 * self.a * self.b)**0.5 / 2 * self.a)

        if diskriminante > 0:
            x1 = (-self.b + diskriminante)
            x2 = (-self.b - diskriminante)
            return x1, x2

        elif diskriminante == 0:
            x1 = (-self.b + diskriminante)
            return x1

        else:
            x1 = ()
            x2 = ()
            return x1, x2

z = QESolver(2.0, 0, 0)

print(z.rechnen())
```

Aufgabe 3
Schreiben Sie unter Verwendung von PyQt5 ein GUI Programm, welches in einem Fenster eine zufällige Weisheit des Konuzius ausgibt. Durch Drücken eines Ok-Buttons wird das Programm beendet. Die Weisheiten sind alle in einem File „weisheiten.txt“ utf-8 codiert gespeichert. Jede Weisheit ist dabei genau in einer Zeile.

Die ersten Zeilen des Files sind z.B.:
Wenn du hasst, bedeutet das, dass sie dich besiegt haben Bevor du dich rächst, grabe zwei Gräber aus. Eigentlich ist das Leben einfach, aber wir bemühen uns es zu komplizieren. Es ist besser, ein Licht zu entzünden, als auf die Dunkelheit zu schimpfen.

```
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
import random

class Window (QMainWindow):
    def __init__ (self):
        super().__init__()

        self.setWindowTitle("Konfizus' Weisheiten")

#---------------- LAYOUT ----------------------
        layout = QFormLayout()

        self.text = QLabel()
        self.button1 = QPushButton("Laden")
        self.button2 = QPushButton("OK")

        layout.addRow(self.text)
        layout.addRow(self.button1)
        layout.addRow(self.button2)

        center = QWidget()
        center.setLayout(layout)

        self.setCentralWidget(center)
        self.show()
```

```
#------------------ BUTTONS MIT FUNKTION VERBINDEN ----

        self.button1.clicked.connect(self.load)
        self.button2.clicked.connect(self.quit)

#---------------- FUNKTONEN ----------------------

    def load(self):
        file = open("Modulap/weisheiten.txt", "r",
encoding="utf-8")
        for num, weisheit in enumerate(file, 2):
            if random.randrange(num): continue
            self.text.setText(weisheit)

        file.close()

    def quit(self):
        exit(0)

# -----------------------------------------------------

app = QApplication([])
fenster = Window()
app.exec()
```

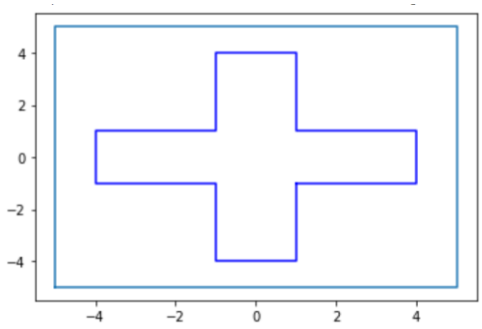
Aufgabe 4
Was geschieht, wenn die folgende Zelle ausgeführt wird ?
(Auch hier soll jede Zeile kommentiert werden).

```
import numpy as np                      # import des numpy-Moduls als np, np damit nicht immer ausgeschrieben werden muss

v1 = np.array([1,2,3,4])                # Erstellen und speichern eines numpa-Array, Datentyp aufgrund Eingabe integer
v2 = np.array([1,2,3,4],dtype=np.float) # Erstellen und speichern eines 2. numpa-Arrays, Datentyp float (Fliesskommaz.)

v1+v2                                 # Addition der beiden arrays, Datentyp float, Resultat wird in Form array ausgegeben
```

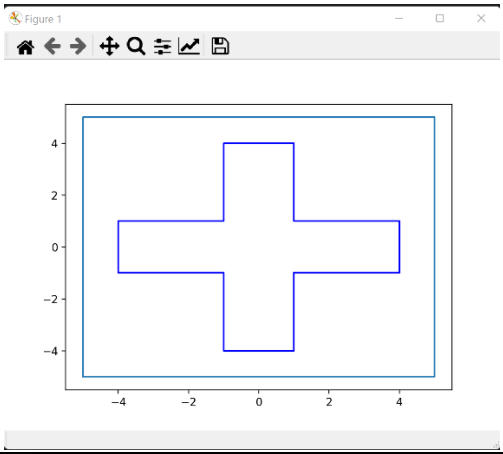
Aufgabe 5
Gegeben ist ein Polygon mit einem Loch als WKT-String, z.B:
`s = "POLYGON ((-5 -5, -5 5, 5 5, 5 -5, -5 -5), (1 -1, 4 -1, 4 1, 1 1, 1 4, -1 4, -1 1, -4 1, -4 -1, -1 -1, -1 -4, 1 -4, 1 -1))"`

a) Wie kann dieses im Jupyter Notebook dargestellt werden ?
b) Wie kann dieses in einem Python-Script dargestellt werden?

a)
```
import shapely.wkt
import matplotlib.pyplot as plt

s = "POLYGON ((-5 -5, -5 5, 5 5, 5 -5, -5 -5), (1 -1, 4 -1, 4 1, 1 1, 1 4, -1 4, -1 1, -4 1, -4 -1, -1 -1, -1 -4, 1 -4, 1 -1))"
polygon = shapely.wkt.loads(s)
x1,y1 = polygon.exterior.xy
x2,y2 = polygon.interiors[0].xy

plt.plot(x1,y1,x2,y2, 'b-')
```



b)
```
import matplotlib.pyplot as plt
import shapely.wkt

s = "POLYGON ((-5 -5, -5 5, 5 5, 5 -5, -5 -5), (1 -1, 4 -1, 4 1, 1 1, 1 4, -1 4, -1 1, -4 1, -4 -1, -1 -1, -1 -4, 1 -4, 1 -1))"
polygon = shapely.wkt.loads(s)
x1,y1 = polygon.exterior.xy
x2,y2 = polygon.interiors[0].xy

plt.plot(x1,y1,x2,y2, 'b-')
jupyterplt.show()      < ------EINZIGER UNERSCHIED
```



Aufgabe 6

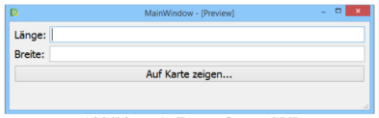


Abbildung 1: Das gefragte GUI

Mit dem Qt-Designer wurde das abgebildete GUI mit Namen maptool.ui erstellt.

Mit dem Button "Auf Karte zeigen..." soll der Standard-Webbrowser mit Google Maps geöffnet werden. Der Link kann mithilfe der Koordinate zusammengesetzt werden. Das Format dazu ist: *https://www.google.ch/maps/place/breite,länge*

```python
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *

class Window (QMainWindow):
    def __init__ (self):
        super().__init__()

        self.setWindowTitle("Aufgabe 6 Koordinaten")

#--------------- LAYOUT ----------------------
        layout = QFormLayout()

        self.laenge = QLineEdit()
        self.breite = QLineEdit()
        self.button = QPushButton("Auf Karte anzeigen...")

        layout.addRow("Länge:", self.laenge)
        layout.addRow("Breite:", self.breite)
        layout.addRow(self.button)

        center = QWidget()
        center.setLayout(layout)

        self.setCentralWidget(center)
        self.show()

#------------------ BUTTON MIT FUNKTION VERBINDEN --------------------

        self.button.clicked.connect(self.load)

#--------------- FUNKTON ----------------------

    def load(self):
        lon= self.laenge.text()
        lat = self.breite.text()

        if -90 <= float(lon) <= 90 and -180 <= float(lat) <= 180:
            QDesktopServices.openUrl(QUrl(f"https://www.google.ch/maps/place/{lat},{lon}"))
        else:
            self.laenge.setText("ungültiger Wert")
            self.breite.setText("ungültiger Wert")

# ------------------------------------------------------

app = QApplication([])
fenster = Window()
app.exec()
```