Zwischenprüfung Programmieren II

25.10.2018 von 13:45 bis 14:45

FHNW GEOMATIK

Dauer: 60 Minuten

Erlaubte Hilfsmittel:

Unterlagen zur Python-Programmierung aus Papier (z.B. Skript, Übungen, ...)
Lösungen müssen **handschriftlich direkt** auf dem abgegebenen Papier erfolgen. Abgabe von Zusatzblättern ist nicht erlaubt! Bei Platzmangel kann das Zusatzblatt auf Seite 9 verwendet werden.

Laptops, Taschenrechner, Mobiltelefone, Smartphones, Tablets, etc. sind nicht erlaubt!
Bei Smartwatches ist nur die „Uhr-Funktion" erlaubt.

Name: Hofmann Tobias

Punkte: 25.5

Note: 5.9

Aufgabe 1 (4 Punkte)

a) Was ist eine Klassendefinition ?

b) Was sind Instanzen ?

Beschreiben Sie das Konzept und schreiben Sie eine Beispielklasse bei der dieses gezeigt wird.

a) Klassendefinition: "theoretischer" Bauplan einer Klasse, in dem alle Attribute und Methoden definiert sind.

b) Instanzen: "Realisation" einer Klassendefinition. ~~Jede Instanz ist~~ Für jede Instanz können die Klassenattribute eigene, unterschiedliche Werte annehmen.

Beispiel:

a)
```
class Baum:
    def __init__(self, hoehe)
        self.hoehe = hoehe
```
} Bauplan → Klassendefinition

b)
```
Baum1 = Baum(9)
Baum2 = Baum(11.2)
```
} 2 Instanzen: { Baum1 mit 9m Höhe
Baum2 mit 11.2m Höhe

4

Aufgabe 2 (2 Punkte)

Was sind magische Methoden in Python und für was können diese verwendet werden ?

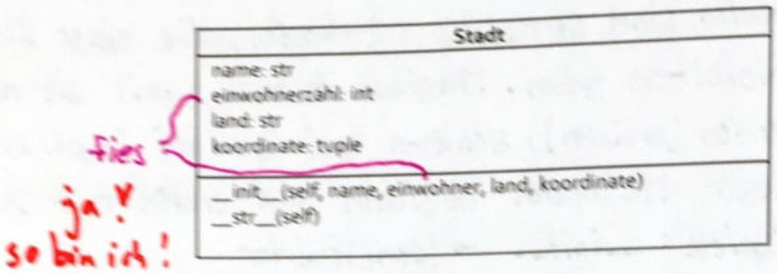
Magische Methoden sind spezielle Methoden, die einer Klasse besondere Fähigkeiten geben. Magisch deshalb, weil sie nicht direkt aufgerufen werden (müssen), sondern bei Bedarf implizit.
Die Namen dieser Methoden beginnen und enden mit zwei Unterstrichen.
Verwendungsbeispiele: __init__ → Konstruktor
__str__ → Generiert Zeichenkette, die mit print ausgegeben wird
__add__ → Wird aufgerufen, wenn Instanzen einer Klasse addiert werden.

⋮

2

Aufgabe 3 (4 Punkte)

a) Implementieren Sie eine Klasse `Stadt` in Python. Diese ist durch folgendes UML Diagramm gegeben:



b) «koordinate» und «land» sind vom Datentyp „tuple" resp. „str". Weshalb wäre es sinnvoll für diese eigene Klassen einzuführen ?

a)
```
class Stadt:
    def __init__(self, name, einwohner, land, koordinate):
        self.name = name
        self.einwohnerzahl = einwohner
        self.land = land
        self.koordinate = koordinate

    def __str__(self):
        return f"Stadt: {self.name}, Land: {self.land}"
```

b) land: Das Land wird für jede Stadt neu definiert. Würde man es in eine separate Klasse einführen, und sich die Schreibweise von Deutschland z.B. in Dütschland ändern, müsste man es nur ïn an einer Stelle ändern statt in allen deutschen Städten.

koordinate: In einer Klasse wäre es möglich, z.B. Koordinatentransformationen zu definieren, die z.B. LV95 in WGS84 automatisch transformiert und umgekehrt.

4

Aufgabe 4 (6 Punkte)

Erstellen Sie mit Python eine Klasse `Dreieck` welche mit einem sinnvollen Konstruktor instanziiert werden kann.

Gegeben sind auch die Klasse Punkt und Strecke ***(nicht abschreiben!!)***

```
class Punkt:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return f"POINT ({self.x} {self.y})"


class Strecke:
    def __init__(self, A, B):
        self.a = A
        self.b = B

    def laenge(self):
        return ((self.a.x-self.b.x)**2+(self.a.y-self.b.y)**2)**0.5

    def __str__(self):
        return f"STRECKE: {self.a} bis {self.b}"
```

Implementieren Sie neben dem Konstruktor die folgenden Methoden:

`flaeche()` : Gibt die Fläche des Dreiecks zurück
`umfang()` : Gibt den Umfang des Dreiecks zurück
`inkreisradius()`: Gibt den Radius des Inkreises zurück

Hinweis:

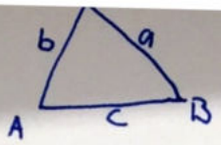
Die Fläche F des Dreicks kann z.B. über den Satz des Heron berechnet werden:

$$s = \frac{a+b+c}{2}$$

$$F = \sqrt{s(s-a)(s-b)(s-c)}$$

Wobei a,b und c die Seitenlängen des Dreiecks sind.

Der Inkreisradius r des Dreiecks berechnet sich mit $r = \frac{2F}{a+b+c}$

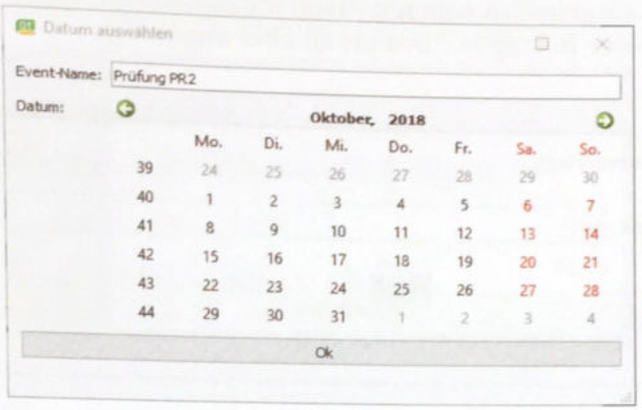


Voraussetzung hier: Klassen Punkt, Strecke, Dreieck im gleichen Modul

```
class Dreieck:
    def __init__(self, A, B, C):
        self.A; self.B, self.C = A, B, C

    def strecken(self):
        self.a = Strecke(B,C)
        self.b = Strecke(A,C)
        self.c = Strecke(A,B)

    def flaeche(self):
        self.strecken()
        s = (a.laenge() + b.laenge() + c.laenge()) / 2
        F = (s*(s - a.laenge())*(s-b.laenge())*(s- c.laenge()))**0.5
        return F

    def umfang(self):
        strecken()
        return self.a.laenge() + self.b.laenge() + self.c.laenge()

    def inkreisradius(self):
        strecken()
        return 2* flaeche() / (self.a.laenge() + self.b.laenge() + self.c.laenge())
```

Aufgabe 5 (6 Punkte)

Implementieren Sie folgendes GUI (nur Layout, ohne Funktionalität)



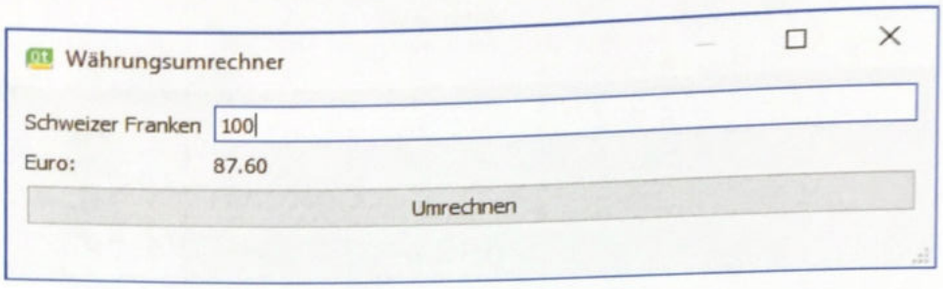

```
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *

class Fenster(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Datum ")
        self.show()

        ## Diesen Teil implementieren

        center = QWidget()
        center.setLayout(layout)
        self.setCentralWidget(center)


app = QApplication([])
fenster = Fenster()
app.exec_()
```

```
layout = QFormLayout()
event = QLineEdit()
calendar = QCalendarWidget()
button = QPushButton("Ok")

layout.addRow("Event-Name", event)
layout.addRow("Datum", calendar)
layout.addRow(button)
```

6

Aufgabe 6 (4 Punkte)

Implementieren Sie folgende GUI-Applikation. Der Betrag in Schweizer Franken kann im QLineEdit eingegeben werden. Nach klicken auf den Button «umrechnen» wird der Betrag in Euro als QLabel angezeigt.




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

```
class MyWindow(QMainWindow):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Währungsumrechner")
        self.show()

        layout = QFormLayout()

        self.chf = QLineEdit()
        self.euro = QLabel()
        self.button = QPushButton("Umrechnen")
        layout.addRow("Schweizer Franken", self.chf)
        layout.addRow("Euro:", self.euro)
        layout.addRow(self.button)

        self.button.clicked.connect(self.umrechnen)
```

```
        center = QWidget()
        center.setLayout(layout)
        self.setCentralWidget(center)

    def umrechnen(self):
        self.euro.setText(str(float(
        chf= self.chf.text()
        euro= float(chf)*0.876
        self.euro.setText(str(round(euro,2)))

app = QApplication([])
f= Fenster
f= MyWindow()
app.exec_()
```

try:

except:

```
#Verbesserungsmöglichkeit: Abfrage, ob in self.chf eine Zahl steht, sonst
#                          Fehlermeldung
#                          • Umrechnungskurz als Variable (self.kurs)
#                          • Runden auf 0.05 € Euro
#                          • Dokumentation durch Doc-Strings
```

3.5