Was ist eine Klassendefinition ? Eine Klassendefinition oder Klasse beschreibt die Struktur eines Objektes, welche aus Attributen und Methoden bestehen. Es ist nur die Beschreibung eines Objekttypes und nicht das Objekt selber. Beispiel

class Rechteck: def **init**(self, laenge, breite): self.laenge = laenge self.breite = breite

```
    def __str__(self):

        return f"{self.laenge}, {self.breite}"
```

Was ist eine Instanz ? Durch die Instanz ruft man die Methoden der einzelnen Klassen auf. Es ist die Ausführung einer Klasse mit Werten für die einzelnen Attributen. Beispiel:

a = Rechteck(4,2) print(a)

Was sind magische Methoden ?

Sind spezielle Methoden, die einer Klasse die besondere Fähigkeiten geben. Sie werden automatisch (indirket) abgerufen wenn eine bestimmte Instanz aufgerufen wird. z.B. die **str**. Damit wird eine Zeichenkette erstellt und wenn man dan als Instanz print(a) eingibt wird die **str** aufgerufen und die Werte ausgegeben welche dort definiert wurden siehe Beispiel von der Klasse. So können bestimmte ausdrücke vereinfacht ausgeführt werden. Ein weiters Beispiel wäre **add** so können zwei Eingaben miteinander addiert werden. -> Eingabe z.B a + b rechnet automatisch die beiden zusammen.

Was wird automatisch aufgerufen wenn eine neue Instanz einer Klasse erstellt wird ? der Konstruktor Wenn eine Klasse vererbt wird so muss die vererbte Klasse immer mit dem super().**init**() manuell aufgerufen werden damit die Attribute korrekt initialisiert werden.

In [ ]:

```
class Rechteck:
    def __init__(self, laenge, breite):
        self.laenge = laenge
        self.breite = breite

    def __str__(self):
        return f"{self.laenge}, {self.breite}"

a = Rechteck(4,2)
print(a) # wenn print von einem Objekt muss __str__ definiert sein
```

In [ ]:

```
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}, Einwohnerzahl: {self.einwohnerzahl}, L
and: {self.land}, Koordinate: {self.koordinate}"
```

```
    # Die verschiededenen Variabeln könnten auch eigene Klassen definiert w
erden. Die seperaten Klassen könnte man implementieren und in den eigenstän
digen
    # Klassen könnte man noch eigene Berechnungen oder Bedingungen definier
en. Bei den Koordinaten berechnungen LV95 zu WGS84
    # bei dem Land können einzelne Sachen einfacher geändert werden weil z.
B. meherer Städte erfasst werden können unter dem gleichen Land. Das heisst
    # sobald etwas beim Land geändert werden muss muss es überall gemacht w
erden ausser man hat eine eigene Klasse dafür
```

In [ ]:

```
class Student:
    def __init__(self,name,vorname,geschlecht,immu_nr, age = 0):
        self.name = name
        self.vorname = vorname
        self.geschlecht = geschlecht
        self.immu_nr = immu_nr
        self.setAge(age)
        self.mark = {}

    def setAge(self,age):
        self.age = age

    def setMark(self,topic,mark):
        self.mark[topic] = mark

    def display(self):
        print(self.name, self.vorname, self.geschlecht, self.immu_nr, self.
age, self.mark)

a = Student("Walliser","Fabrice", "Male", "75666116","26")
a.setAge(27)
a.setMark("Mathe",5.9)
a.display()
```

In [ ]:

```
class Roman:
    def __init__(self,roman):
        self.roman = roman

    def __str__(self):
        return f"{self.roman}"

    def __add__(self, other):
        return self.int_to_Roman(self.roman_to_int(self.roman)+self.roman_t
o_int(other.roman))

    def __int__(self):
        return  self.roman_to_int(self.roman)
```

In [ ]:

```
class Punkt:
    def __init__(self,x ,y):
        self.x = x
        self.y = y
    def __str__(self):
        return f"Punkt: ({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}"

class Dreieck:

    def __init__(self,A,B,C):
        self.A = A
        self.B = B
        self.C = C

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

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

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

    def inkreis(self):
        r = (2 * self.F) / (self.a.laenge()+self.b.laenge()+self.c.laenge()
)
        return r
```

```
X = Punkt(0,0)
Y = Punkt(1,0)
Z = Punkt(1,1)

d = Dreieck(X,Y,Z)

flaechedr = d.flaeche()
print(flaechedr)

udrei = d.umfang()
print(udrei)

dinkr = d.inkreis()
print(dinkr)
```

In [ ]:

```
class Punkt:
    def __init__(self,x ,y):
        self.x = x
        self.y = y
    def __str__(self):
        return f"Punkt: ({self.x}, {self.y})"

class Quadrat:

    def __init__(self,A,B):
        self.A = A
        self.B = B

    def umfang(self):
        self.u = (abs((self.A.x - self.B.x)))*4
        return self.u

    def flaeche(self):
        self.f = (abs((self.A.x - self.B.x)))**2
        return self.f

    def mittelpunkt(self):
        self.mx = (abs((self.A.x - self.B.x))) / 2
        self.my = (abs((self.A.y - self.B.y))) / 2
        return f"Mittelpunkt: X: {self.mx}, Y: {self.my}"

X = Punkt(0,0)
Y = Punkt(1,1)

q = Quadrat(X,Y)
um = q.umfang()
```

```
fl = q.flaeche()
mi = q.mittelpunkt()
print(um)
print(fl)
print(mi)
```

In [ ]:

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

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


        self.setWindowTitle("Datum auswählen")

        layout = QFormLayout()

        self.event = QLineEdit()
        self.calendar = QCalendarWidget()
        self.button = QPushButton("Ok")

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

        center = QWidget()
        center.setLayout(layout)

        self.setCentralWidget(center)
        self.show()

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

In [ ]:

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

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


        self.setWindowTitle("Währungsumrechner")
```

```
        layout = QFormLayout()

        self.franken = QLineEdit()
        self.euro = QLabel()
        self.umkurs = QLineEdit()
        self.umrechnen = QPushButton("Umrechnen")

        layout.addRow("Schweizer Franken", self.franken)
        layout.addRow("Umrechnungskurs", self.umkurs)
        layout.addRow("Euro", self.euro)
        layout.addRow(self.umrechnen)

        self.umrechnen.clicked.connect(self.rechner)

        center = QWidget()
        center.setLayout(layout)

        self.setCentralWidget(center)
        self.show()

    def rechner(self):
        try:
            chf = float(self.franken.text())
            kurs = float(self.umkurs.text())
            euro = chf * kurs
            self.euro.setText(str(round(euro,2)))
        except:
            QMessageBox.warning(self,"Achtung auf Eingabe", "Es müssen Zahl
en eingegeben werden")

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

In [ ]:

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

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


        self.setWindowTitle("Adresseingabe")

        layout = QFormLayout()
```

```
        self.name = QLabel()
        self.adresse = QLabel()
        self.name_eingabe = QLineEdit()
        self.adresse_eingabe = QTextEdit()
        self.button = QPushButton("OK")

        layout.addRow("Name:", self.name)
        layout.addRow(self.name_eingabe)
        layout.addRow("Adresse", self.adresse)
        layout.addRow(self.adresse_eingabe)
        layout.addRow(self.button)

        center = QWidget()
        center.setLayout(layout)

        self.setCentralWidget(center)
        self.show()

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

In [ ]:

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

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

        loadUi("showmap.ui", self)

        self.convertButton.clicked.connect(self.convert)

        self.show()

    def convert(self):
        try:
            self.chf = float(lineEditCHF)
            self.euro = 0.999 * self.ch
            lineEdit.setText(str(self.euro))

        except:
            QMessageBox.warning(self,"Achtung", "Esm müssen Zahlen eingegeb
en werden")
```

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

In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(-5,5,20)
y = (0.5 * x**2) -1
plt.plot(x,y,"ro-")
plt.title("Funktion")
plt.axis("equal")
plt.show()
```