```python
# -*- coding: utf-8 -*-
"""
Created on Sat Apr  6 13:47:13 2024

@author: bauma
"""
import unittest
import Hausarbeit as ha
import pandas as pd

class UnitTestHausarbeit(unittest.TestCase):

    def test_init_datensatz(self):
        '''
        Diese Methode prüft, ob ein DatensatzError ausgegeben wird, wenn der
        Instanz der Klasse Datensatz eine nichtleere Zeichenkette übergeben wird.
        '''
        with self.assertRaises(ha.DatensatzError):
            ds = ha.Datensatz('Test')

    def test_init_funktion(self):
        '''
        Diese Methode prüft, ob ein FunktionError ausgegeben wird, wenn der
        Instanz der Klasse Funktion eine nichtleere Zeichenkette übergeben wird.
        '''
        with self.assertRaises(ha.FunktionError):
            f = ha.Funktion('Test')

    def test_init_punkt(self):
        '''
        Diese Methode prüft, ob ein PunktError ausgegeben wird, wenn der
        Instanz der Klasse Punkt zwei Zeichenketten übergeben wird.
        '''
        with self.assertRaises(ha.PunktError):
            p = ha.Punkt('Test','Test2')

    def test_p_hole_x_wert(self):
        '''
        Diese Methode prüft, ob eine Instanz der Klasse Punkt den richtigen Wert
        zurück gibt, wenn der x-Wert ausgegeben werden soll.
        '''
        d_x = ha.Punkt (2,3)
        result = d_x.hole_x_wert()
        self.assertEqual(result,2,"Der X-Wert sollte 2 ergeben")


    def test_p_hole_y_wert(self):
        '''
        Diese Methode prüft, ob eine Instanz der Klasse Punkt den richtigen Wert
        zurück gibt, wenn der y-Wert ausgegeben werden soll.
        '''
        d_y = ha.Punkt (2,3)
        result = d_y.hole_y_wert()
        self.assertEqual(result,3,"Der Y-Wert sollte 3 ergeben")

    def test_pp_berechne_y_abweichung(self):
        '''
        Diese Methode prüft, ob eine Instanz der Klasse Punkt die richtige
        Y-Abweichung berechnet.
        '''
```

```
        p_1 = ha.Punkt(2,3)
        p_2 = ha.Punkt(1,2)
        result = p_1.berechne_y_abweichung(p_2)
        self.assertEqual(result,1,"Die Y - Abweichung sollte 1 betragen")

    def test_hole_y_abweichung(self):
        '''
        Diese Methode prüft, ob eine Instanz der Klasse Punkt die richtige
        Y-Abweichung zurück gibt.
        '''
        d_y = ha.Punkt (2,3)
        d_y.setze_y_abweichung(2)
        result = d_y.hole_y_abweichung()
        self.assertEqual(result, 2, 'Die Y-Abweichung sollte 2 sein!')

    def test_hole_y_spalte(self):
        '''
        Diese Methode prüft, ob eine Instanz der Klasse Funktion die richtigen
        Y-Werte zurückgibt.
        '''
        df_dic = {"x":[1,2,3],"Y1":[4,5,6]}
        df = pd.DataFrame(df_dic)
        f_y = ha.Funktion (df)
        result = []
        for i in f_y.hole_y_spalte('Y1'):
            result.append(i)
        self.assertEqual(result, [4,5,6], 'Die Y-Werte sollten 4,5,6 sein')

    def test_hole_x_spalte(self):
        '''
        Diese Methode prüft, ob eine Instanz der Klasse Funktion die richtigen
        X-Werte zurückgibt.
        '''
        df_dic = {"x":[1,2,3],"Y1":[4,5,6]}
        df = pd.DataFrame(df_dic)
        f_x = ha.Funktion (df)
        result = []
        for i in f_x.hole_x_spalte():
            result.append(i)
        self.assertEqual(result, [1,2,3], 'Die X-Werte sollten 1,2,3 sein')

    def test_hole_anzahl_zeilen(self):
        '''
        Diese Methode prüft, ob eine Instanz der Klasse Funktion die richtige
        Anzahl an Werten zurückgibt.
        '''
        df_dic = {"x":[1,2,3],"Y1":[4,5,6]}
        df = pd.DataFrame(df_dic)
        f_w = ha.Funktion (df)
        result = f_w.anzahl_zeilen()

        self.assertEqual(result,3, 'Die Anzahl der Werte sollte 3 betragen')

    def test_hole_funktionsname(self):
        '''
        Diese Methode prüft, ob eine Instanz der Klasse Funktion den richtigen
        Funktionsnamen zurückgibt
        '''
        df_dic = {"X":[1,2,3],"Y1":[4,5,6]}
        df = pd.DataFrame(df_dic)
```

```
        f_f = ha.Funktion (df)
        result = f_f.hole_funktionsname()
        self.assertEqual(result, 'Y1', 'Der Funktionsname sollte Y1 sein')

    def test_suche_idealfunktion (self):
        '''
        Diese Methode prüft, ob eine Instanz der Klasse Funktion die richtige
        Idealfunktion bestimmt
        '''
        df_dic = {"x":[1,2,3],"y1":[4,5,6]}
        df = pd.DataFrame(df_dic)
        f_i = ha.Trainingsfunktion (df)
        df_dic_ds = {"x":[1,2,3],"y1":[4.2,5.2,6.2],"y2":[5,6,7]}
        df_ds = pd.DataFrame(df_dic_ds)
        ds = ha.Datensatz(df_ds)
        f_i.suche_idealfunktion(ds)
        result = f_i.hole_idealfunktion().hole_funktionsname()

        self.assertEqual(result,'y1' , 'Der Funktionsname sollte Y1 sein')

if __name__ == '__main__':
    unittest.main()
```