Dokumentation Junioraufgabe 1

Jugendwettbewerb Informatik 3.Runde

Aufgabenstellung:

Es soll ein Programm geschrieben werden, um den besten Weg einen Garten in möglichst gleich große, quadratische Flächen einzuteilen zu finden.

Die Regel:

Es gibt eine Anzahl von Interessenten und die muss man in ein Feld von lang mal breit so einteilen, dass sie so quadratisch wie möglich sind. Es dürfen höchstens zehn Prozent mehr Felder als Interessenten sein. Die Option mit den quadratischsten Feldern wird ausgegeben.

Umsetzung:

```
import math
import sys
```

Hiermit importiere ich zwei Bibliotheken, die im Code nützlich sein könnten:

Ich brauche math, damit ich später die Wurzel ziehen kann.

Ich brauche sys, um den Filenamen der Eingabedatei (Text) auf der Kommandozeile zu bekommen.

```
def get_divisors(n):
    divisors = set()
```

Das ist die Definition get_divisors. Sie braucht eine Zahl als Argument und gibt dann die Teiler von der Zahl zurück. Als erstes brauchen wir ein Set, damit keine Teiler doppelt vorkommen.

```
for i in range(1, int(math.sqrt(n)) + 1):
        if n % i == 0:
        divisors.add(i)
        divisors.add(n // i)
return sorted(divisors)
```

Hier wiederhole ich eine Schleife, bis sie zum Mittelpunkt von der Zahl n gekommen ist. Ich wiederhole zu schauen ob die Zahl n durch i ohne Rest teilbar ist, wenn ja ergänze ich die Zahl i und deren Zahl die mit i zusammen multipliziert n ergeben. Dann werde ich das set „divisors“ wieder zurückgeben.

```
def quadratisch_praktisch_grün(file_name):
file = open(file_name, "r")
INputliste = []
factorpairs = []
dictio = {}
```

Das ist die Funktion quadratisch, praktisch, grün(). Sie nimmt eine Datei als Argument ein und öffnet sie. Die Datei wird als „file“ gespeichert. Dann werden sämtliche Listen und Dictionaries erstellt. „INputliste“ ist für das Einlesen der Datei wichtig. „Factorpairs“ speichert später die Faktorenpaare und „dictio“ speichert die Faktorenpaare mit ihren Differenzwerten.

```
for line in file.readlines():
```

```
        INputliste.append(int(line))
[Interessant, lang, breit] = INputliste
```

Hier lese ich die Datei ein und speichere jede Zahl in „INputliste“. Dann werde ich die 1.Zahl in „INputliste“ „Interessant“ nennen. Das ist die Anzahl von Interessenten. Die 2. Zahl nenne ich „lang“ und die 3. nenne ich „breit“. Sie sind die Länge und Breite von dem Feld, das Herr grün hat.

```
if breit > lang:
        größere_seite = breit
        kleinere_seite = lang
else:
        größere_seite = lang
        kleinere_seite = breit
```

Hier schaue ich ob breite Seite länger ist als die lange Seite oder andersrum, und dann ordne ich der längeren Seite „größere_seite“ und der kürzeren Seite „kleinere_seite“ zu.

```
PARZELlen = int(Interessant * 1.1)
for n in range(Interessant, PARZELlen+1):
factors = get_divisors(n)
factorpairs = []
```

Hier erstelle ich die Variable „PARZELlen“, die die höchste Anzahl von Interessenten angebe. Danach wiederhole ich für jede Zahl zwischen „Interessant“(also die Mindestanzahl von Interessenten) und „PARZELlen“(also die Höchstanzahl von Interessenten) das:

Ich muss zuerst die Faktoren von der Zahl bekommen, indem ich die Funktion get_divisors (siehe oben) hervorrufe. Dann erstelle ich eine

leere Liste, die „factorpairs“ heißt. Sie wird später die Teilerfaktoren speichern.

```
for i in range(len(factors)):
        if n/factors[i] < factors[i]:
        break
        factorpairs.append((factors[i], int(n/factors[i])))
```

Hier wiederholen wir für jeden Teiler von der Anzahl von Interessenten, dass wir schauen ob es größer ist als wenn man die Zahl von Interessenten von sich selbst teilt, wenn ja, gibt es die Paarung schon und man bricht ab, wenn nicht wird die Paarung zu „factorpairs“ hinzugefügt.

```
for pair in factorpairs:
        differenz = kleinere_seite / pair[0] - größere_seite / pair[1]
        dictio[abs(differenz)] = pair
```

Hier gehen wir bei jedem Paar in den Faktorenpaaren durch die Schleife. Wir wollen dann die Faktorenpaarungen auf die Seiten des Feldes verteilen. Die kleinere Seite wird durch die kleinere Zahl geteilt und die größere Seite wird durch die größere Zahl geteilt. Davon speichere ich die Differenz in „differenz“. Und dann speichere ich das Absolut von „differenz“ in „dictio“ mit einem Wert von pair.

```
ausgabe = dictio[min(dictio.keys())]
text = f"Die beste Aufteilung ergibt {ausgabe[0]} mal {ausgabe[1]} = {ausgabe[0] * ausgabe[1]} Parzellen für {Interessant} Interessenten."
return text
```

Die kleinste Differenz hat das perfekteste Quadrat, und deswegen such ich das heraus und speichere das in „ausgabe“. Dann formuliere ich ein Text und gebe den Text aus.

```
if __name__ == "__main__":
print(quadratisch_praktisch_grün(sys.argv[1]))
```

Das ist der „main“-code. Wenn man das Programm auf der Kommandozeile aufruft, nimmt es die Definition quadratisch_praktisch_grün() mit dem Argument sys.argv[1] (das 1.Argument auf der Kommandozeile) und gebe dann den zurückgegebenen string (message) aus.

ENDE UMSETZUNG

Aufrufen:

Das Programm ruft man auf, indem man das Terminal benutzt und die Datei Das_ägyptische_Grabmal mit python3 aufruft und nach einem Leerzeichen noch eine der grabmal0.txt bis grabmal5.txt hinschreibt. Beispiel: `python3 Das_ägyptische_Grabmal grabmal2.txt`

Beispiele:

garten0.txt

23
42
66

Output: Die beste Aufteilung ergibt 4 mal 6 = 24 Parzellen für 23 Interessenten.

garten1.txt

19
15
12

Output: Die beste Aufteilung ergibt 4 mal 5 = 20 Parzellen für 19 Interessenten.

garten2.txt

36
55
77

Output: Die beste Aufteilung ergibt 6 mal 6 = 36 Parzellen für 36 Interessenten.

garten3.txt

101
15
15

Output: Die beste Aufteilung ergibt 10 mal 11 = 110 Parzellen für 101 Interessenten.

garten4.txt

1200
37
2000

Output: Die beste Aufteilung ergibt 5 mal 264 = 1320 Parzellen für 1200 Interessenten.

garten5.txt

35000
365

937

Output: Die beste Aufteilung ergibt 120 mal 308 = 36960 Parzellen für 35000 Interessenten.

Code:

```
import math
import sys

def get_divisors(n):
        divisors = set()
        for i in range(1, int(math.sqrt(n)) + 1):
        if n % i == 0:
        divisors.add(i)
        divisors.add(n // i)
        return sorted(divisors)

def quadratisch_praktisch_grün(file_name):
        file = open(file_name, "r")
        INputliste = []
        factorpairs = []
        dictio = {}
        for line in file.readlines():
        INputliste.append(int(line))
        [Interessant, lang, breit] = INputliste
        if breit > lang:
        größere_seite = breit
        kleinere_seite = lang
        else:
        größere_seite = lang
        kleinere_seite = breit
        PARZELlen = int(Interessant * 1.1)
        for n in range(Interessant, PARZELlen+1):
        factors = get_divisors(n)
        factorpairs = []
        for i in range(len(factors)):
        if n/factors[i] < factors[i]:
        break
        factorpairs.append((factors[i], int(n/factors[i])))
        for pair in factorpairs:
        differenz = kleinere_seite / pair[0] - größere_seite / pair[1]
```

```
            dictio[abs(differenz)] = pair
            ausgabe = dictio[min(dictio.keys())]
            text = f"Die beste Aufteilung ergibt {ausgabe[0]} mal {ausgabe[1]} = {ausgabe[0] * ausgabe[1]} Parzellen
für {Interessant} Interessenten."
            return text

if __name__ == "__main__":
            print(quadratisch_praktisch_grün(sys.argv[1]))
```