

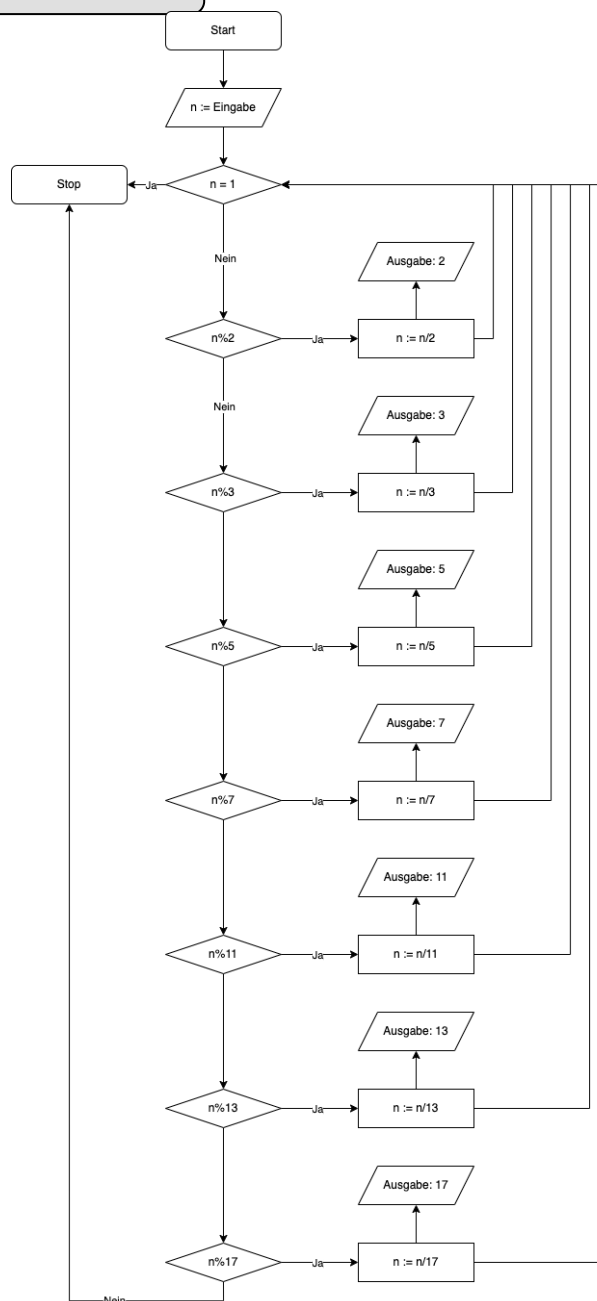
Programmbeschreibung und Flussdiagramm

**Hinweis**

Dieses Programm zerlegt eine Zahl in ihre Primfaktoren. Hierzu wird die Berechnung mit „Modulo“ (Restwert) verwendet.

Die Umsetzung des Programm verwendet:

- if-Abfrage
- while-Schleife
- Variablen
- Modulo



Schritt-für-Schritt-Anleitung

In dem folgenden Teil führt „Schritt-für-Schritt“ durch das Flussdiagramm.

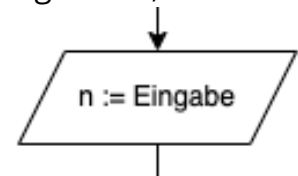
Am Ende dieser Anleitung steht das vollständige Programm passend zum Flussdiagramm.

1. Schritt: Die Eingabe (Zeile 1 und 2).

Die Variable **n** wird deklariert. „**input**“ fordert den „Nutzer“ auf, zu dieser Variablen einen Werte anzugeben. „**int()**“ wandelt den Typ in „integer“ um, da die Eingabe als „String“ ausgegeben wird.

Python

```
1 n = int(input('Gebe eine Zahl an: '))
```



2. Schritt: Die while-Schleife (Zeile 3).

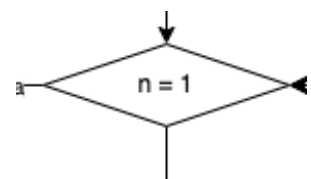
Sobald **n** **gleich 1** ist, werden die Abfragen nicht mehr durchlaufen. (**oder anders formuliert**)

Solange **n** **nicht gleich 1** ist, werden die Abfragen durchlaufen.

Hintergrund: Von 1 lässt sich keine Primfaktorzerlegung durchführen.

Python

```
1 n = int(input('Gebe eine Zahl an: '))
2
3 while n != 1:
```

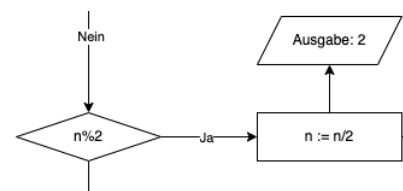


3. Schritt: Die Abzweigung (Zeile 4 bis 6).

Wenn **n Modulo 2** gleich null ist, also $n/2$ ohne Rest, dann soll „2“ ausgegeben und **n** durch 2 geteilt werden. **Hinweis:** Unterscheidung zum Flussdiagramm.

Python

```
1 n = int(input('Gebe eine Zahl an: '))
2
3 while n != 1:
4     if n % 2 == 0:
5         print("2")
6         n = n / 2
```



4. Schritt: Die erste Abzweigung. (Zeile 7 bis 26)

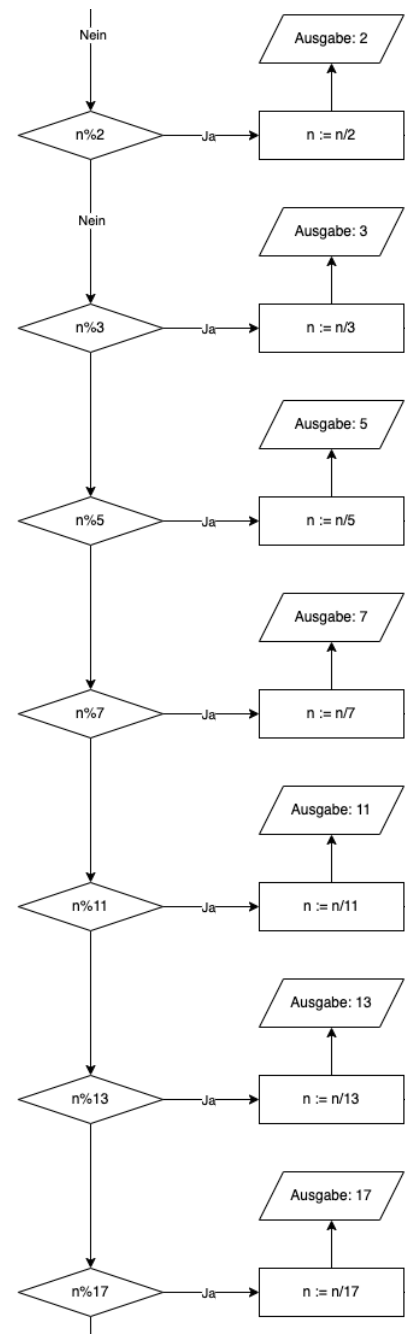
Der Schritt 3 wird für die Primzahlen 3, 5, 7, 11, 13 und 17 ebenfalls durchgeführt. Nach jeder Ausgabe und Berechnung geht es zurück zur while-Schleife, bis $n = 1$ ist.

Python

```

1 n = int(input('Gebe eine Zahl an: '))
2
3 while n != 1:
4     if n % 2 == 0:
5         print("2")
6         n = n / 2
7     elif n % 3 == 0:
8         print("3")
9         n = n / 3
10    elif n % 5 == 0:
11        print("5")
12        n = n / 5
13    elif n % 7 == 0:
14        print("7")
15        n = n / 7
16    elif n % 11 == 0:
17        print("11")
18        n = n / 11
19    elif n % 13 == 0:
20        print("13")
21        n = n / 13
22    elif n % 17 == 0:
23        print("17")
24        n = n / 17

```



ENDE

Aufgaben

- ① „Baue“ das Programm mit Hilfe der Schritt-für-Schritt-Anleitung nach. Füge in dem Quellcode passend zu den Schritten Kommentare ein, die den jeweiligen Teil in seiner Funktion beschreiben.

Python

```
1 # Ein "Hashtag" ergibt einen Zeilenkommentar
2
3 """Drei Anführungszeichen am Anfang und Ende ergeben
4 einen Kommentar auf mehreren Zeilen"""
```

- ② Einige Fragen, um das Verständnis zum Programm zu überprüfen.
- Die variable „n“ beginnt mit dem Wert, den der User einträgt. Was passiert „im Laufe“ des Programmes mit diesem Wert?
 - Dieses Programm hat „Grenzen“ in der Zerlegung in Primzahlen. Im Grunde stellt dieses Programm keinen vollständigen Algorithmus dar. Begründe, warum dies speziell bei Primfaktorzerlegung dies so ist.
- ③ Überlege Dir mögliche Verbesserungen für das Programm. Beispielsweise:
- Könnte der Text der Eingabe und der Ausgabe verbessert werden.
 - Besteht die Möglichkeit, eine andere Berechnung zu verwenden.
 - Wird am Ende eine Else-If-Anweisung benötigt?

- ④ Passe das Flussdiagramm und den Quelltext entsprechend Deiner Überlegungen aus Aufgabe 3 an.