Einführung in Python

Arne Hüffmeier

Mittelwert berechnen

Wir wollen jetzt ein kleines Programm schreiben, welches den Mittelwert berechnen kann.

Wie gehen wir vor?

Wichtige Fragen, die man sich vorher stellen sollte

1. Welchen Mittelwert?
2. Wie sieht die Eingabe aus?
3. Wie sieht die Ausgabe aus?
4. Wie ist der Ablauf?

1) Welchen Mittelwert?

Am Anfang nehmen wir das arithmetische Mittel

$$\frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

2) Wie sieht die Eingabe aus?

Um es einfach zu gestalten, nehmen wir als Eingabe einfach eine Liste an

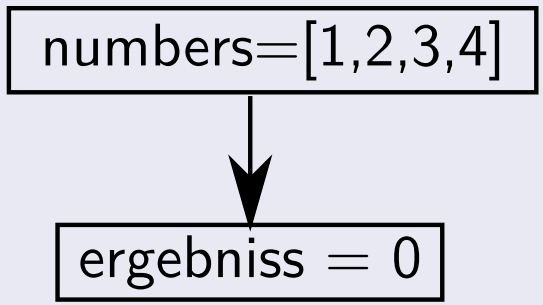
```
1 numbers= [1,2,3,4,5,6,7,8,9,10,33]
```

3) Wie sieht die Ausgabe aus?

Bei der Ausgabe reicht uns ein einfaches `print`

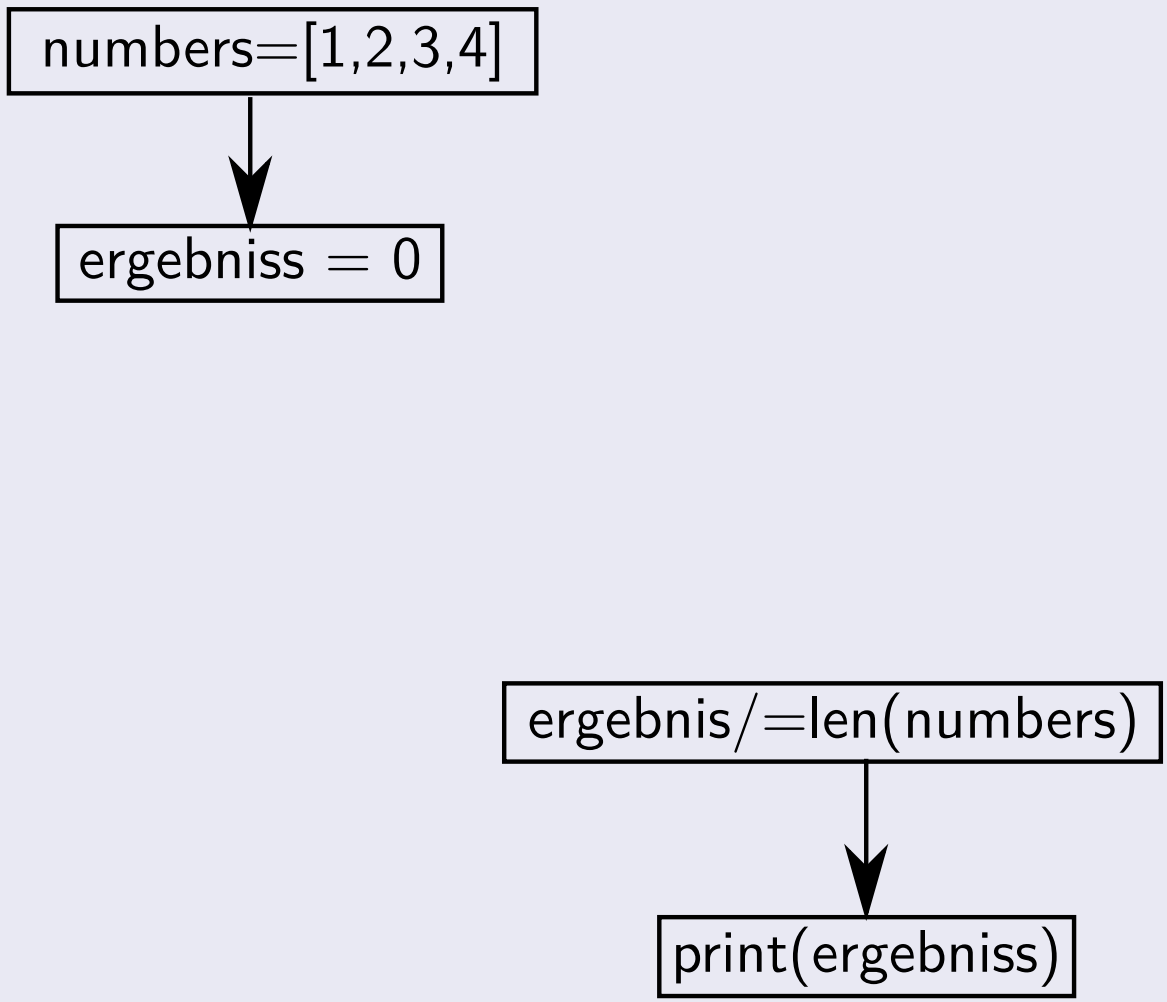
```
1 print("Das␣Arithmetische␣Mittel␣ist:", ergebnis)
```

Was wir schon wissen

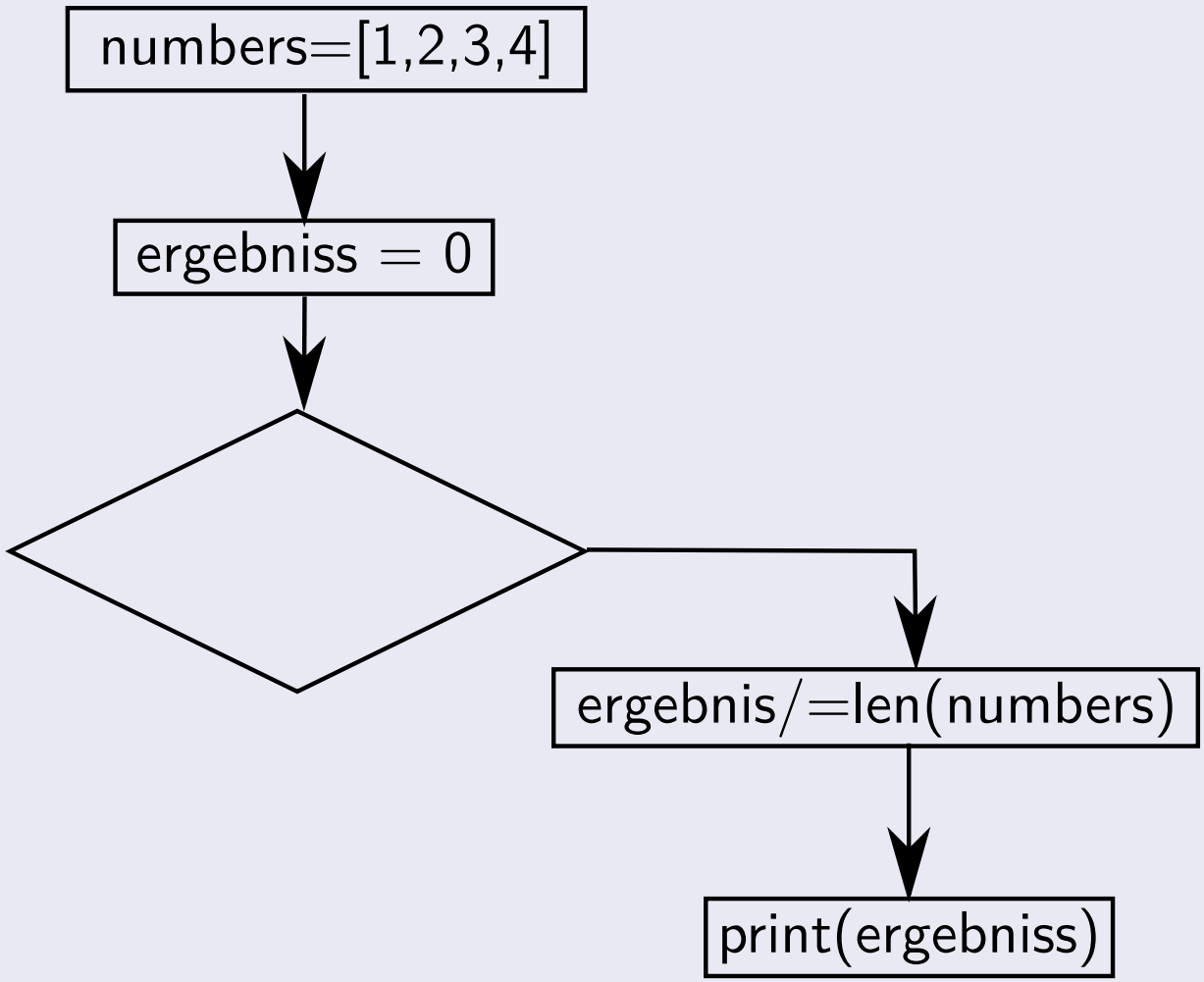


print(ergebniss)

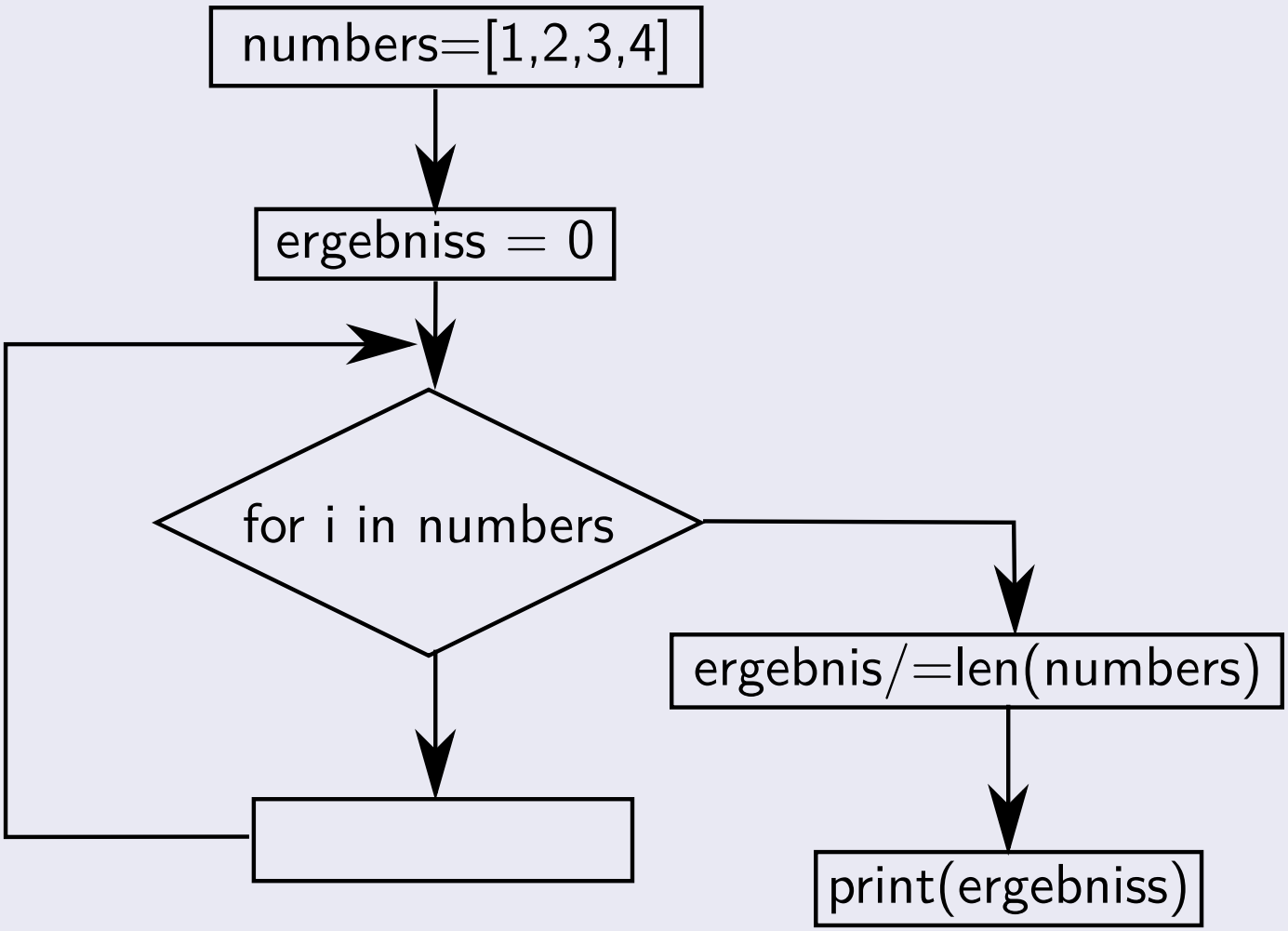
Aus der Formel ergibt sich



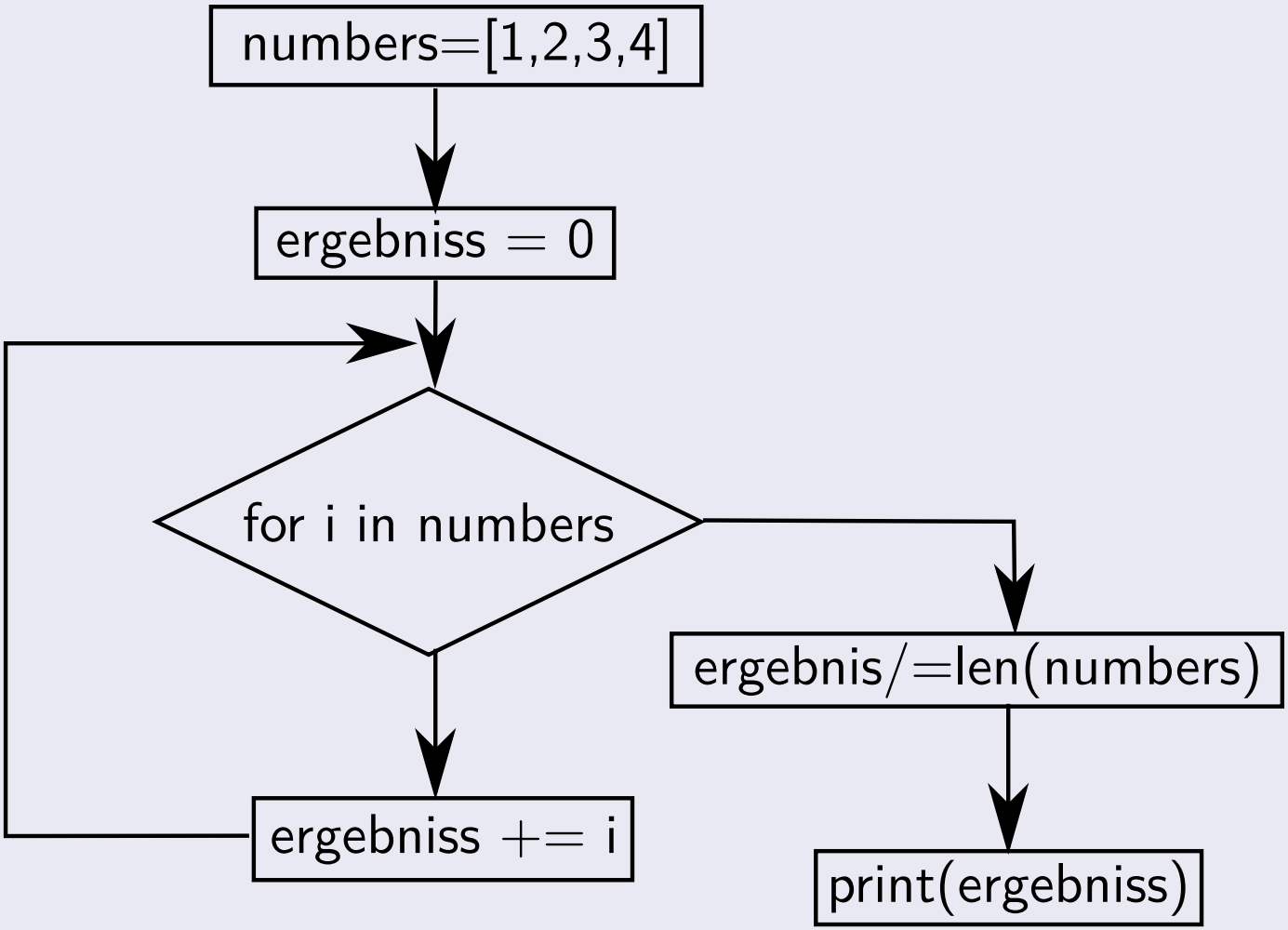
Da wir über die Liste iterieren müssen



Dafür ist for da



Eine Addition und wir sind fertig



Der erste Teil in Python

```
1 numbers= [1,2,3,4,5,6,7,8,9,10,33]
2 ergebnis = 0
3
4
5
6 print("Das Arithmetische Mittel ist:", ergebnis)
```

Das Teilen kommt dazu

```
1 numbers= [1,2,3,4,5,6,7,8,9,10,33]
2 ergebnis = 0
3
4
5 ergebnis /= len(numbers)
6 print("Das Arithmetische Mittel ist:", ergebnis)
```

for hinzugefügt

```
1 numbers= [1,2,3,4,5,6,7,8,9,10,33]
2 ergebnis = 0
3 for i in numbers:
4
5 ergebnis /= len(numbers)
6 print("Das␣Arithmetische␣Mittel␣ist:",ergebnis)
```

Jetzt ist es fertig

```
1 numbers= [1,2,3,4,5,6,7,8,9,10,33]
2 ergebnis = 0
3 for i in numbers:
4     ergebnis += i
5 ergebnis /= len(numbers)
6 print("Das Arithmetische Mittel ist:",ergebnis)
```

kleine Verbesserung

Dieses kleine Programm funktioniert zwar, jedoch würde das Programm sehr unübersichtlich werden, wenn wir weitere Mittelwerte berechnen wollen. Zudem würde jede Berechnung ausgeführt, auch wenn wir das nicht wollen.

Was passiert, wenn wir bei einem größeren Projekt an mehreren Stellen den arithmetischen Mittelwert benötigen?

Um mehrfaches Auftauchen von gleichen Codeschnipseln zu verhindern, gibt es Funktionen.

Machen wir aus unserer Berechung mal eine Funktion

```
1 def arithmetic_mean():
2     numbers= [1,2,3,4,5,6,7,8,9,10,33]
3     ergebnis = 0
4     for i in numbers:
5         ergebnis += i
6     ergebnis /= len(numbers)
7     print("Das Arithmetisches Mittel ist:",ergebniss)
```

Was ist neu?

Es ist nur eine Zeile dazu gekommen

```
1 def arithmetic_mean():
```

`def` = es handelt sich um eine Funktion
`arithmetic_mean` = Funktionsname (dieser ist frei wählbar)
`():` = Parameter und Ende

Bevor wir uns mit den Funktionen weiter beschäftigen, schauen wir uns an, wie wir die Funktion aufrufen können.

```
1  def arithmetic_mean():
2      numbers= [1,2,3,4,5,6,7,8,9,10,33]
3      ergebnis = 0
4      for i in numbers:
5          ergebnis += i
6      ergebnis /= len(numbers)
7      print("Das Arithmetische Mittel ist:",ergebnis)
8
9
10 arithmetic_mean()
```

Ein einfaches `arithmetic_mean()` reicht aus um unsere Funktion aufzurufen.

So etwas kennen wir doch schon?!

```
1 print("Hallo")
```

Das `print` ist auch eine Funktion. Diese Funktion wurde von den Python Entwicklern für uns geschrieben.

Doch bei dieser Funktion schreiben wir etwas in die Runden Klammern (Das sind die Parameter).

Einbauen von Parametern

Bis jetzt haben wir:

```
1 def arithmetic_mean():
2     numbers= [1,2,3,4,5,6,7,8,9,10,33]
3     ergebnis = 0
4     for i in numbers:
5     ergebnis += i
6     ergebnis /= len(numbers)
7     print("Das Arithmetisches Mittel ist:",ergebniss)
```

Jedoch ergibt es ja wenig Sinn, immer für die gleichen Zahlen den Mittelwert zu bestimmen.
Besser wäre es, wenn wir die Liste als Parameter bekommen würden.

```
1 def arithmetic_mean(numbers):
2     ergebnis = 0
3     for i in numbers:
4         ergebnis += i
5     ergebnis /= len(numbers)
6     print("Das Arithmetische Mittel ist:",ergebnis)
```

Jetzt sieht der Aufruf unserer Funktion etwas anders aus

```
1 zahlen = [1,2,3,4,45,5,6,8]
2 arithmetic_mean(zahlen)
```

Zudem ist es jetzt möglich, von mehreren Listen den Mittelwert zu berechnen.

```
1 liste1 = [1,23,4,5,6,2,42,6,2,43,6,87,21,76,3]
2 liste2 = [42,433,2,1,3,45,65,632,232,564,43,5,223,65]
3 arithmetic_mean(liste1)
4 arithmetic_mean(liste2)
```

Infos zu Parametern

Eine Funktion kann auch mehrere Parameter bekommen.

```
1 def add(a,b,c,d,e):
2     ergebniss = a + b + c + d + e
3     print(ergebniss)
```

Der Aufruf der Funktion sieht dann dementsprechend aus:

```
1 add(1,2,3,4,5)
```

Wichtig ist, dass die Reihenfolge stimmt. In diesem Beispiel wäre a = 1, b = 2, c = 3, d = 4 und e = 5

Was ist wenn wir mit dem Ergebnis noch weiter rechnen möchten?

Auch dafür gibt es natürlich eine Lösung ;-)

```
1 def arithmetic_mean(numbers):
2     ergebnis = 0
3     for i in numbers:
4         ergebnis += i
5     ergebnis /= len(numbers)
6     return ergebnis
```

Mit dem `return` können wir Python mitteilen, dass die Funktion hier vorbei ist und dass der Inhalt vom Ergebnis weiter gegeben werden soll.

Was müssen wir jetzt am Aufruf verändern?

Bis jetzt mussten wir folgendes Schreiben:

```
1 zahlen  = [1,2,3,4,45,5,6,8]
2 arithmetic_mean(zahlen)
```

Eigentlich nichts. Jedoch sehen wir keine Ausgabe mehr.

```
1 zahlen  = [1,2,3,4,45,5,6,8]
2 erg = arithmetic_mean(zahlen)
3 print("Das ergebnis ist", erg)
```

Infos zu return

Mit dem `return` endet die Funktion sofort.

```
1 def test():
2     print("erster Text")
3     return 1
4     print("zweiter Text")
```

Bei dieser `test`-Funktion würde das zweite `print` nicht ausgeführt werden.

Das Zweite Programm

Das erste Programm war sehr klein und zugegebenermaßen auch nicht wirklich interessant. Deswegen werden wir jetzt ein kleines Spiel programmieren.

Mastermind

Wir erinnern uns noch mal an die Fragen beim ersten Programm.

Wichtige Fragen, die man sich vorher stellen sollte:

1. Welche Regeln hat das Spiel?
2. Wie sieht die Eingabe aus?
3. Wie sieht die Ausgabe aus?
4. Wie ist der Ablauf?

1) Welche Regeln hat das Spiel?

- Es müssen 5 Farben ausgewählt werden.
- Zur Auswahl stehen 8 Farben.
- Der Spieler hat 12 Versuche, um die richtige Farbauswahl zu "erraten".
- Nach jedem Versuch bekommt er die Rückmeldung, wie häufig die richtige Farbe am richtigen Platz war und wie häufig die richtige Farbe am falschen Platz war.

2) Wie sieht die Eingabe aus?

Hier geht es leider nicht mit einer einfachen Liste, denn der Spieler muss auf die Ausgaben reagieren können.
Also benötigen wir eine Eingabe für jede Runde.

3) Wie sieht die Ausgabe aus?

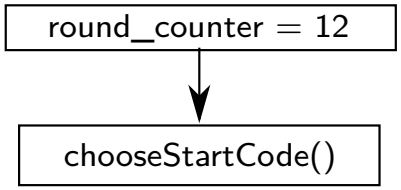
Die Ausgabe können wir mit `print` realisieren, jedoch brauchen wir Verschiedene.

1. Nach jedem Versuch des Spielers mit den Informationen: Runden-Nummer, Infos zur Eingabe
2. Wenn der Spieler richtig geraten hat.
3. Wenn der Spieler es nicht geschafft hat, in der maximalen Anzahl Runden richtig zu raten.

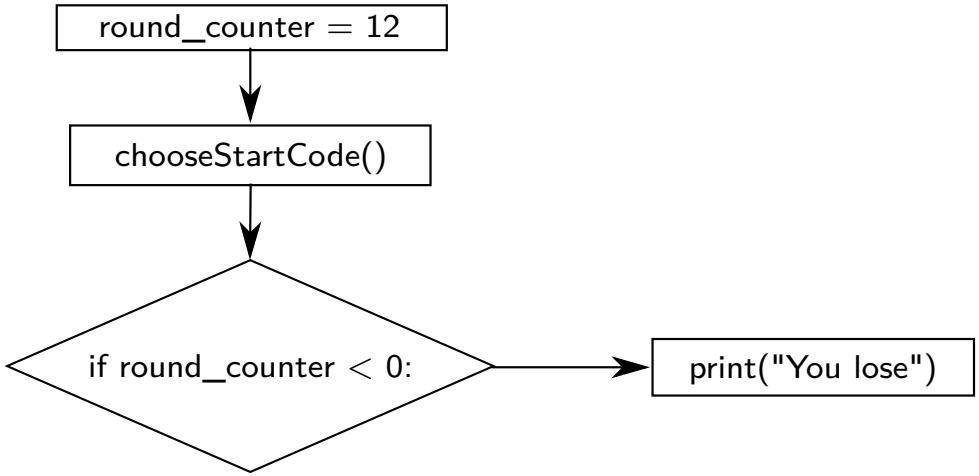
round_counter = 12

Als Erstes brauchen wir etwas, um die verbleibende Anzahl an Runden (Versuchen) zu zählen.

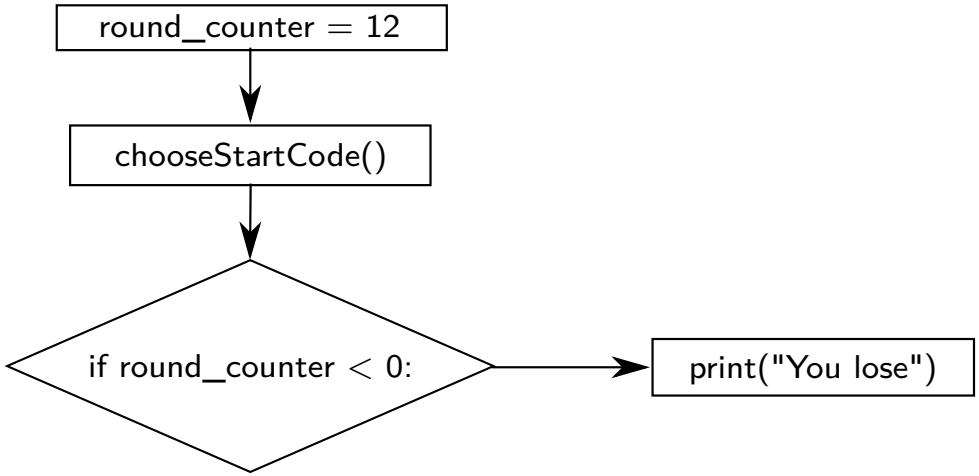
Der Ablauf

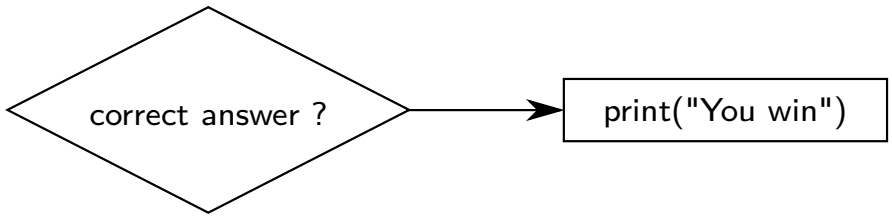


Bevor der Spieler raten kann, muss erstmal eine Farbkombination ausgewählt werden.



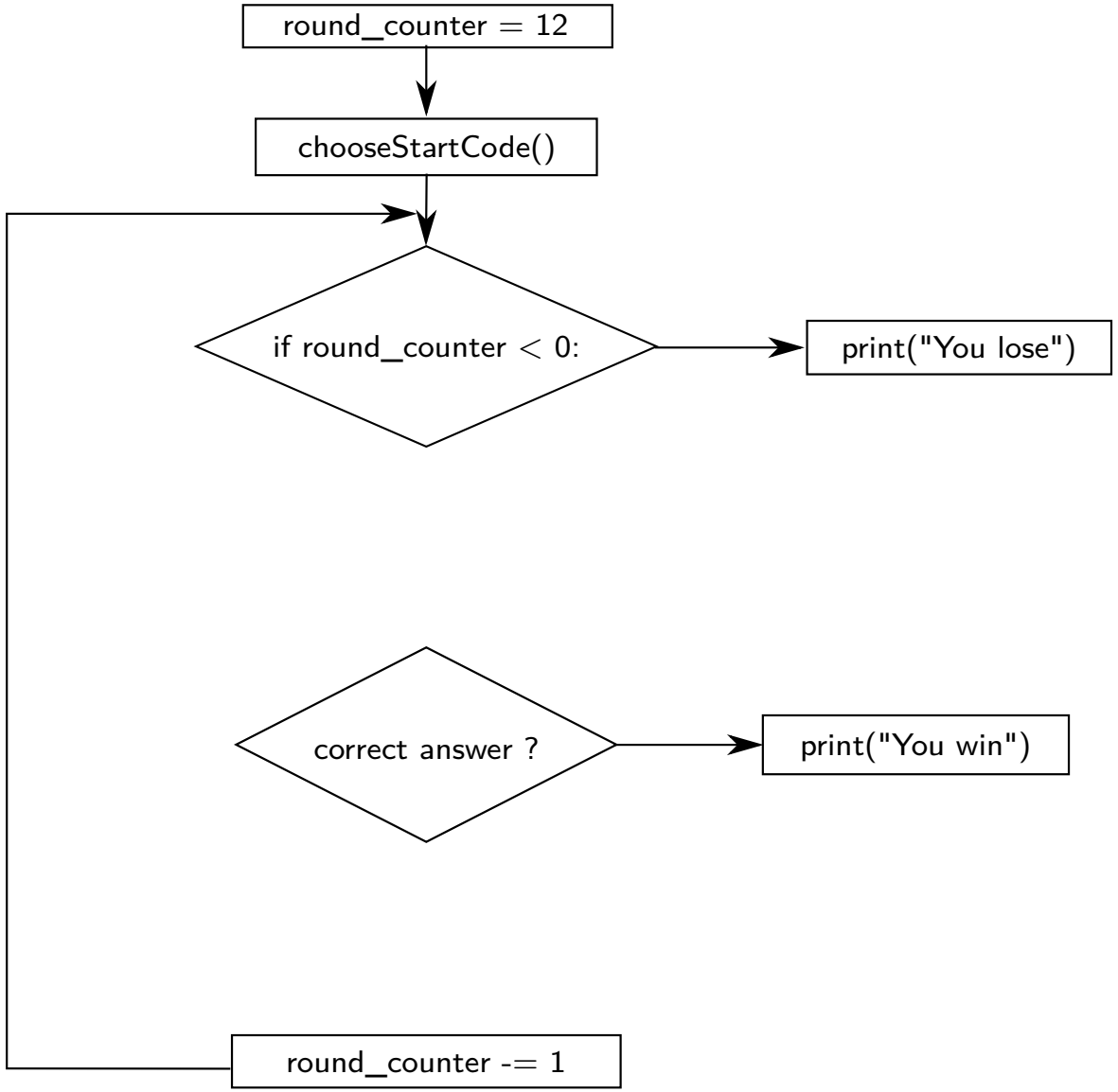
Wenn der Runden-Zähler auf 0 sinkt, hat der Spieler verloren.





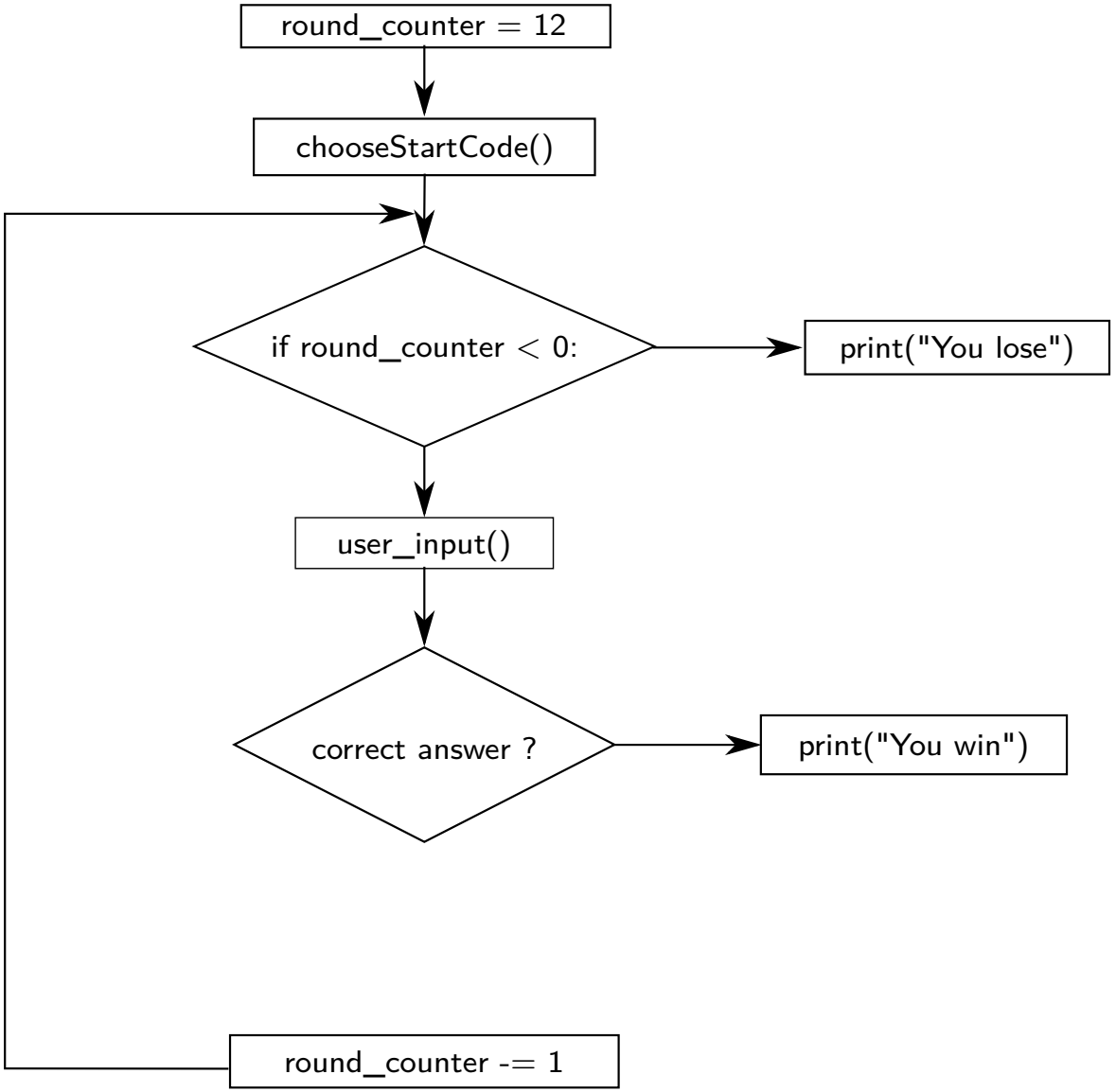
Und wenn die richtige Antwort gegeben wurde, hat er gewonnen.

Der Ablauf



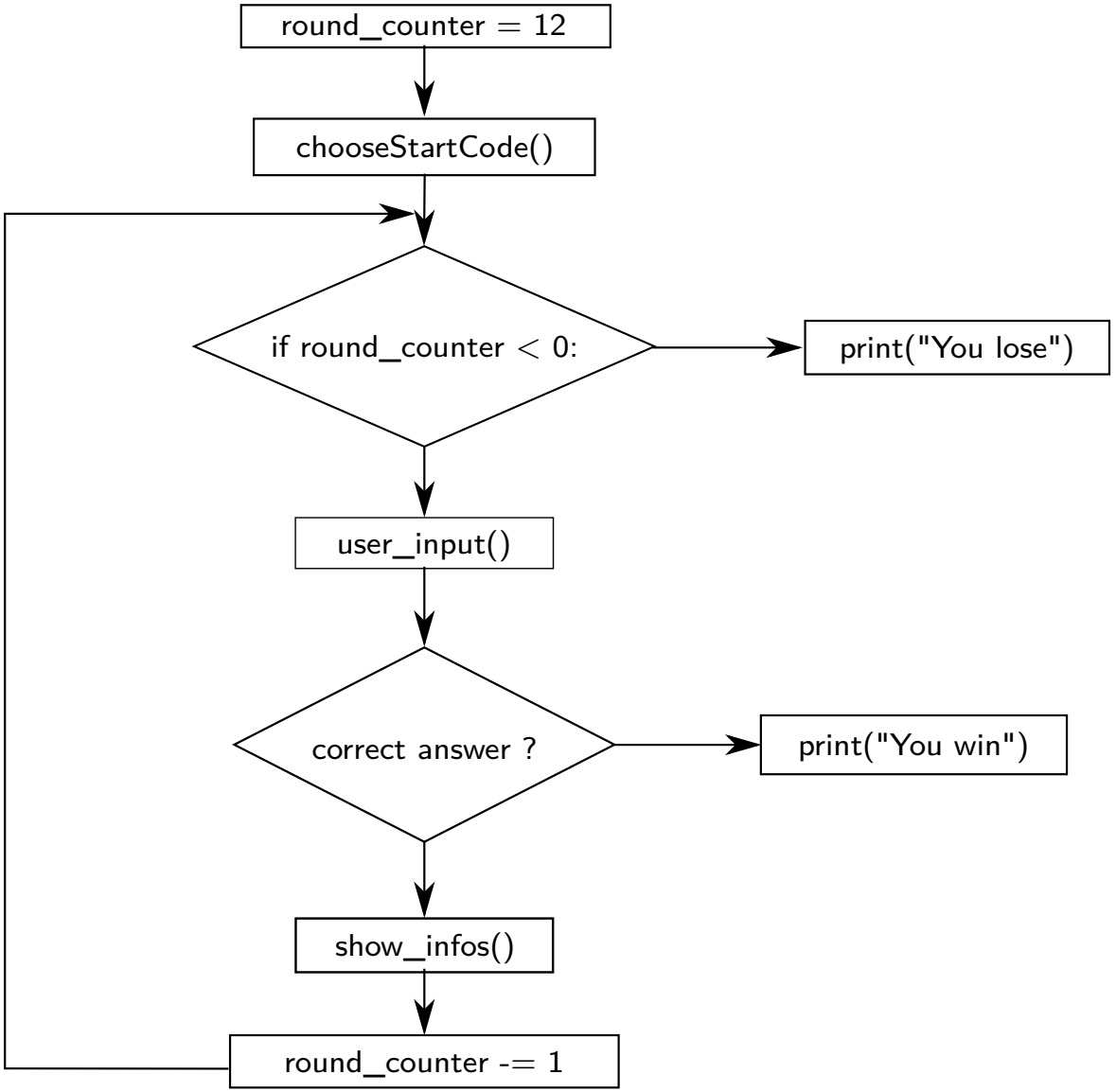
Der Runden-Zähler muss natürlich nach jeder Runde verringert werden.

Der Ablauf



Natürlich muss noch die Spielereingabe hinzukommen.

Der Ablauf



Zum Schluss fehlt noch die Ausgabe, in der die Eingabe bewertet wird.

Überführen wir unsere Grafik in Quellcode

```
1  def game_start():
2      round_counter = 12
3      color = chooseStartCode()
4
5      while 1:
6          if round_counter <= 0:
7              print("Du hast verloren. Die richtige Antwort waere:", color)
8              return 1
9          inp = user_input()
10         ret = checkInput(inp)
11         if ret == 2:
12             print("Du hast die richtige Kombination erraten und noch",
                   round_counter, "Runden uebrig." )
13             return 1
14         if ret == 1:
15             round_counter -= 1
```

Erstellen von der zu ratenden Kombination

Da die Farbeingabe etwas schwieriger ist, lassen wir den Spieler keine Farben, sondern Zahlen raten. Wir benötigen also 5 Zahlen, die zwischen 1 und 8 liegen.

```
1     def chooseStartCode():
2         color = []
3         for i in range(0,5):
4             pass
5         return color
6 ____
```

Wir benötigen jetzt noch Zufallszahlen.

random

Um Zufallszahlen zu erstellen, gibt es die Funktion `randrange`.
Diese hat als Parameter die Grenzen des Zufallszahlenbereichs.

```
1 randrange(1,8)
```

Jedoch kennt Python diesen Befehl von alleine nicht. Wir müssen ihn erst einbinden.
Das passiert über:

```
1 from random import randrange
```

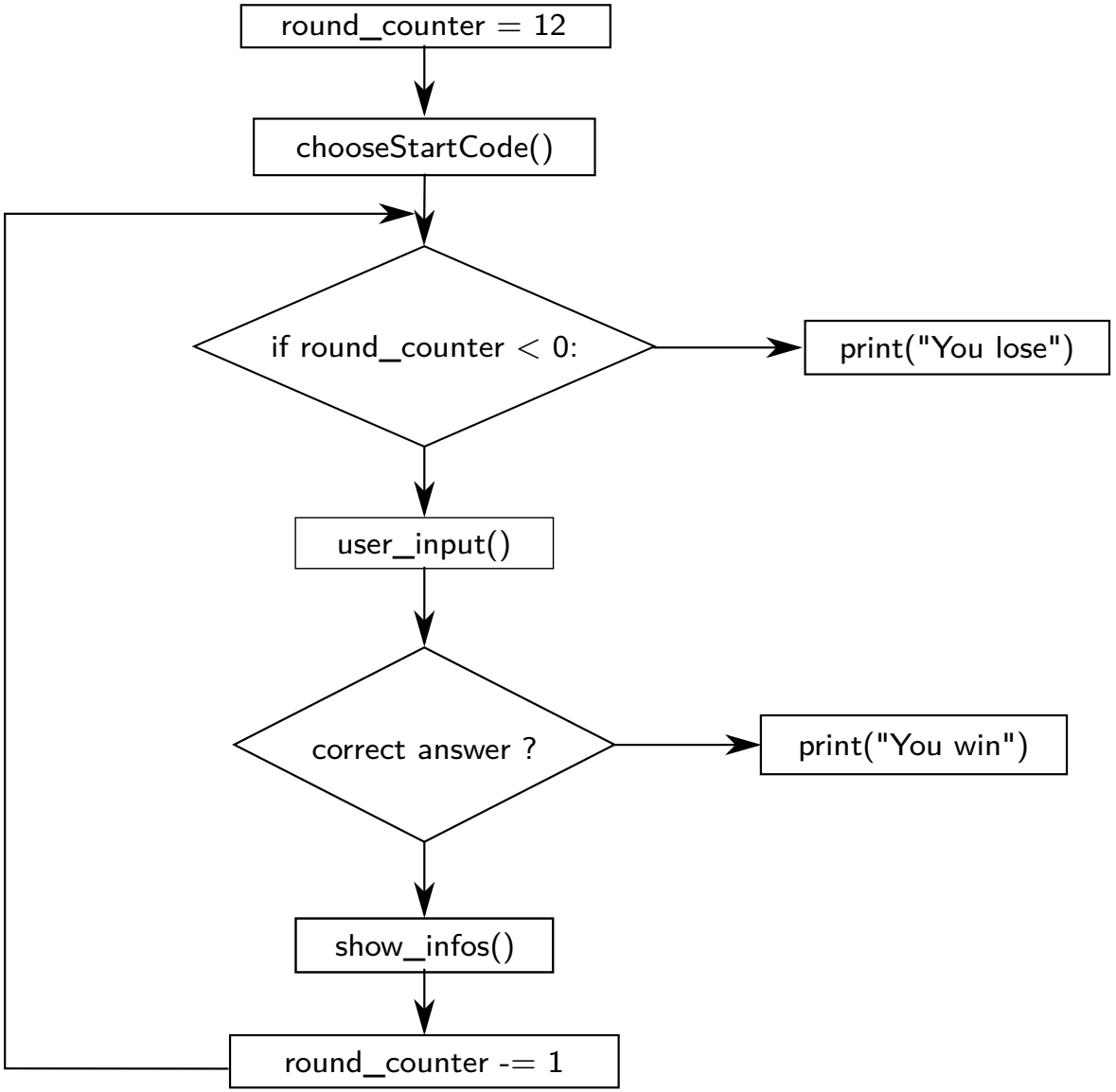
Diese Zeile kommt einfach weit nach oben ins Programm.

Mit der Zeile sieht unsere Funktion dann folgendermaßen aus:

```
1 def chooseStartCode():
2     color = []
3     for i in range(0,5):
4         color.append(randrange(1,8))
5     return color
```

Da color eine Liste ist, können wir mit dem Befehl append einfach eine weitere Zahl hinzufügen.

Weiter im Programm



Was uns jetzt noch fehlt, sind die Funktionen `user_input()` und `show_infos()`.

Wie kann der Spieler seine Zahlen-Kombination eingeben ?

Für eine einfache Benutzereingabe gibt es die Funktion `input()`

```
1 eingabe = input("Was haben Sie zu sagen?")
2 print(eingabe)
```

Als Parameter kann die Funktion einen Text bekommen, der vor der Eingabe ausgegeben werden soll.
Als Rückgabe bekommt man die Eingabe des Benutzers.

Unsere Input-Funktion

Unsere Funktion ist sehr sehr klein. Es ist also nicht nötig sie auszulagern.

```
1 def user_input():
2     inp = input("Eingabe:")
3     return inp
```

Wie sieht jetzt unsere game Funktion aus?

```
1  def game_start():
2      round_counter = 12
3      color = chouseStartCode()
4
5      while 1:
6          if round_counter <= 0:
7              print("Du hast verloren. Die richtige Antwort waere:", color)
8              return 1
9          print("Runde nr.:", round_counter)
10         inp = input("Eingabe:")
11         ret = checkInput(inp, color)
12         if ret == 2:
13             print("Du hast die richtige kombination erraten und noch",
                   round_counter, " Runden uebrig." )
14             return 1
15         if ret == 1:
16             round_counter -= 1
```

Überprüfen, ob die Eingabe richtig ist

```
1 def checkInput(inp, color):
2     pass
```

Was müssen wir testen?

- Stimmt die Eingabe überein?
- Stimmt Position und Zahl überein?
- Kommt die Zahl in der Kombination vor?
- Ist die Eingabe des Spielers fünf Zahlen lang?
- Hat der Spieler nur Zahlen eingegeben?

Eingabe Test

Am einfachsten ist zu überprüfen, ob die passende Anzahl an Zeichen eingegeben wurde.

```
1 def checkInput(inp , color):
2     if (len(inp) is not 5):
3         return 0
```

Eingabe Test

Es muss aber noch getestet werden, ob der Spieler nur Zahlen eingeben hat. Es gibt vorgefertigte Funktionen, mit denen man testen kann, ob ein String nur aus Zahlen besteht.

```
1 def checkInput(inp, color):
2     if (len(inp) is not 5):
3         return 0
4     if inp.isdigit() is not True:
5         return 0
```

Eingabe Test

Da wir für jede Zahl testen müssen, ob sie in unserer Kombination vorkommt, brauchen wir eine for Schleife.

```
1 def checkInput(inp, color):
2     if (len(inp) is not 5):
3         return 0
4     if inp.isdigit() is not True:
5         return 0
6     answer = []
7     for i in range(0,5):
8         pass
```

Eingabe Test

Als Erstes testen wir, ob die Antwort richtig ist. Wenn sie richtig ist, speichern wir uns ein "P" ab.

```
1 def checkInput(inp, color):
2     if (len(inp) is not 5):
3         return 0
4     if inp.isdigit() is not True:
5         return 0
6     answer = []
7     for i in range(0,5):
8         if (inp[i] == str(color[i])):
9             answer.append("P")
```

Eingabe Test

Wenn die Zahl nicht richtig ist, dann müssen wir noch überprüfen, ob die Zahl an einer anderen Stelle vorkommt. Um das einfach zu realisieren, können wir unsere Liste fragen, wie häufig ein Element darin vorkommt.
Da unsere Liste aus Zahlen besteht, müssen wir die Eingabe in eine Zahl umwandeln.

```
1  def checkInput(inp, color):
2      if (len(inp) is not 5):
3          return 0
4      if inp.isdigit() is not True:
5          return 0
6      answer = []
7      for i in range(0,5):
8          if (inp[i] == str(color[i])):
9              answer.append("P")
10         elif (color.count(int(inp[i])) > 0):
11             answer.append("C")
12 ____.____
```

Eingabe-Test

Um zu wissen, ob die Eingabe komplett richtig ist, können wir einfach zählen, wie viele "P" in unserer Auswertung vorkommen. Zudem müssen wir den Tipp für den Benutzer ausgeben.

```
1  def checkInput(inp, color):
2      if (len(inp) is not 5):
3          return 0
4      if inp.isdigit() is not True:
5          return 0
6      answer = []
7      for i in range(0,5):
8          if (inp[i] == str(color[i])):
9              answer.append("P")
10         elif (color.count(int(inp[i])) > 0):
11             answer.append("C")
12     if (answer.count("P") == 5 ):
13         return 2
14     print(answer)
15
16     return 1
17 ____.____
```

Jetzt fehlt nur noch eine Sache ...

... und zwar das Ganze mal auszuprobieren.

```
1  def game_start():
2      round_counter = 12
3      color = chouseStartCode()
4      while 1:
5          if round_counter < 0:
6              print("Du hast verlohren die richtige Antwort waere:", color)
7              return 1
8          print("Runde nr.:", round_counter)
9          inp = input("Eingabe:")
10         ret = checkInput(inp, color)
11         if ret == 2:
12             print("Du hast die Richtige kombination erraten und noch",
                   round_counter, "runden uebrig." )
13             return 1
14         if ret == 1:
15             round_counter -= 1
```

Herzlichen Glückwunsch.
Wir haben jetzt ein Python-Spiel programmiert.

Geschafft

Nun habt ihr einen etwas tieferen Einstieg in Python

Viel Spaß im Tutorium