

Python Grundlagen

Grundelemente einer Programmiersprache

Bisher haben wir mit Turtle gearbeitet. Diese Befehle stammen aus dem Turtle-Package (Modul) und dienen nur dazu, eine kleine Schildkröte zu steuern. Python kann jedoch viel mehr als nur dieses Paket.

Wie jede Programmiersprache folgt auch Python bestimmten Regeln und Mustern, damit wir sinnvoll mit dem Computer arbeiten können. Diese Elemente werden wir uns genauer anschauen.

Grundelemente einer Programmiersprache

1. Variablen und Zuweisungen

Der Computer muss sich Sachen merken können.

2. Ein- und Ausgabe

Etwas muss in den Computer rein, etwas kommt raus.

3. Datentypen

Der Computer hat Regeln, was er mit welchen Variablen machen darf (mit Text kann er nicht dasselbe machen wie mit Zahlen).

4. Operatoren

Der Computer muss wissen, was er mit den Variablen machen kann (rechnen, vergleichen,...).

5. Kontrollstrukturen

Der Computer kann ein Programm (oder Teile davon) wiederholen oder je nach Resultat sich für einen von mehreren Wegen entscheiden.

6. Funktionen

Wiederverwendbare Unterprogramme, die eine bestimmte Aufgabe erfüllen.

All diese Elemente sind eng miteinander verbunden und manchmal ist es schwierig das eine zu verstehen ohne die anderen. Darum versuchen wir in kleinen Etappen sie alle näher zu begreifen.

Inhaltsverzeichnis

Grundelemente einer Programmiersprache	1
1. Variablen und Zuweisungen	2
2. Ein- und Ausgabe	3
3. Datentypen	4
4. Operatoren	5
4.1. Arithmetische Operatoren	5
4.2. Vergleichsoperatoren	5
4.3. Logische Operatoren	5
4.4. Text-Operatoren	5
4.5. Zuweisungsoperatoren	6
5. Kontrollstrukturen	7
5.1. Verzweigungen: <code>if else</code>	7
5.2. Schleifen: <code>while</code> und <code>for</code>	7
5.3. Bedingungen formulieren	7
6. Funktionen	9
Lösungen	10

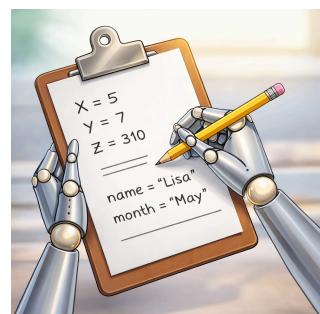
1. Variablen und Zuweisungen

Damit der Computer ein Programm ausführen kann, muss er sich Sachen merken können. Die gespeicherten Werte nennt man **Variablen**.

Variablen und Zuweisungen

Eine **Variable** benutzt man, um einen Wert zu speichern (wie ein Platzhalter).

Sie wird mit einer **Zuweisung** (=) erstellt, wobei links der **Name** und rechts der **Wert** der Variable steht.



Zuweisungen



```
1 x = 5
2 monat = "Januar"
3 summe = 1 + 256 + 42
```

Die **Werte der Variablen** können Zahlen, Text oder auch Berechnungen sein.

In der ersten Zeile erhält die Variable `x` den Wert `5`. In der zweiten Zeile bekommt die Variable `monat` den Wert `"Januar"`. Dies ist ein Text, der immer in Anführungszeichen stehen muss, damit der Computer ihn als solchen versteht.

Den **Namen der Variablen** darf man (fast) frei wählen:

- erlaubt sind Buchstaben (klein und gross), Zahlen und der Unterstrich _
- der Name darf nicht mit einer Zahl beginnen

Im Verlauf des Programms können sich die Werte von Variablen ändern (man kann sie also überschreiben).

Variablen überschreiben



```
1 x = 5
2 x = 7 # der Wert von x ist neu 7
```

Mit dem Symbol `#` können wir Code in Python **kommentieren**. Alles, was nach dem `#` auf derselben Zeile folgt, wird vom Computer ignoriert. **Kommentare** helfen uns, den Code lesbarer (verständlicher) zu machen.

Aufgabe 1

Gegeben ist eine Liste von möglichen **Variablennamen**. Welche sind laut den Regeln für Python gültig?

`Summe_1 = 433`

`summe_1 = 123`

`summel = 63`

`*myname = "Glurak"`

`_my_name = "Pikachu"`

`my name = "Flegmon"`

`8Bit = 256`

`L337Code = 42`

`Byte8Numb3r = 16`

Wie kann man schnell überprüfen, ob ein Variablenname okay ist oder nicht?

Aufgabe 2

Arbeiten Sie die Seite [Variablen](#) im **Turtle-Tutorial** durch. Probieren Sie die **Musterbeispiele** aus und lösen Sie die Aufgaben 1-3 im Abschnitt **Zum selbst Lösen**.

2. Ein- und Ausgabe

Damit wir mit dem Computer arbeiten können, braucht eine Programmiersprache zwei Dinge: Wir müssen **Eingaben (Input)** machen können, und der Computer muss diese verarbeiten und uns danach **Resultate (Output) zurückgeben**.

In Python gibt es zwei Funktionen (Befehle) dafür.

`print()` Damit können Sie beliebige Sachen in die Kommandozeile (Ausgabefenster) ausdrucken lassen.

`input()` Damit fragt der Computer nach einem Input vom Benutzer.

Benutzer-Eingabe und Resultat-Ausgabe



```
1 erste_zahl = input("Erste Zahl:")
2 zweite_zahl = input("Zweite Zahl:")
3 print(erste_zahl + zweite_zahl)
```

Um Programme wirklich nützlich machen zu können, braucht es meistens **Inputs** vom Benutzer und ein Resultat als **Output**, das für den Benutzer hilfreich ist.

Aufgabe 3

In der Fahrschule lernt man folgende Faustregel für den Anhalteweg:

$$\text{Reaktionsweg (in m)} = \frac{\text{Geschwindigkeit (in km/h)}}{10} \cdot 3$$

$$\text{Bremsweg (in m)} = \frac{\text{Geschwindigkeit (in km/h)}}{10} \cdot \frac{\text{Geschwindigkeit (in km/h)}}{10}$$

$$\text{Anhalteweg (in m)} = \text{Reaktionsweg (in m)} + \text{Bremsweg (in m)}$$

Erstellen Sie ein Programm, das die Geschwindigkeit als Input vom Benutzer bekommt, und die restlichen Variablen berechnen Sie mit den Formeln oben. Am Schluss geben Sie das Resultat mit `print(anhalteweg)` aus.

3. Datentypen

Wenn der Computer mit Variablen zu tun hat, muss er wissen, zu welchem Typ sie gehören. Wenn er zum Beispiel weiss, dass es sich um einen Text handelt, dann weiss er auch sofort, dass nicht alle Operatoren funktionieren werden.

Name	Abkürzung	Beschreibung
Integer	int	Das ist der Datentyp für ganze Zahlen . Alle Zahlen in einem Programm, die keine Dezimalstellen haben, sind ganze Zahlen.
Float	float	Floats sind Zahlen mit Dezimalstellen , welche auch Gleitkommazahlen genannt werden. Jede Zahl, die Dezimalstellen in einem Programm hat, gehört zu diesem Datentyp.
String	str	Strings sind Ketten von beliebigen Zeichen. Einfacher gesagt: Text . Alles, was in einem Programm in Anführungszeichen geschrieben steht, ist vom Datentyp String.
Boolean	bool	Boolean sind Wahrheitswerte . Davon gibt es nur zwei: True und False .

Datentypen

Variablen gehören in Python jeweils einem **Datentyp** an. Dieser bestimmt dann, was man alles mit der Variable tun kann. Man kann zum Beispiel bei einem String nicht dividieren.

Wenn Sie eine Variable `x` haben, können wir mit dem Befehl `print(type(x))` den Datentyp herausfinden.

Datentyp herausfinden

```
1 alter = 17
2 print(type(alter))
3 name = "harry"
4 print(type(name))
```

Bestimmter Patenttyp als Input

```
1 inputInt("Ganze Zahl eingeben:")
2 inputFloat("Zahl eingeben:")
3 inputString("Bitte Text eingeben:")
```

Nicht alle Datentypen sind miteinander kompatibel. Zum Beispiel kann Python einen Integer und einen String nicht miteinander addieren. Einen Integer mit einem Float hingegen schon. Darum wollen wir bei den Inputs meistens nur ganz bestimmte Datentypen zulassen.

Manchmal möchte man Datentypen «ineinander umwandeln». Z.B. wenn man in einer `print`-Ausgabe Text und Zahlen verknüpfen möchte (um beides auf derselben Zeile darzustellen). Zwei Texte lassen sich mit einem `+` aneinanderhängen (siehe Zeile 1). Mit Text und Zahl funktioniert dies aber nicht (siehe Zeile 3), denn dies führt zu einem Fehler. Man kann die Zahl `x` jedoch mit `str(x)` in einen Text umwandeln um mit `print` auszugeben (Zeile 4) oder Text und Zahl (ohne Umwandlung) mit einem Komma trennen (Zeile 5).

Text und Zahlen im Output kombinieren

```
1 print("hallo" + " wie gehts?")
2 x = 5
3 print("Das Resultat lautet " + x)
4 print("Das Resultat lautet " + str(x))
5 print("Das Resultat lautet " + x)
```

Aufgabe 4

Passen Sie den Output von der Aufgabe mit dem Anhalteweg so an, dass er wie folgt aussieht:

Der Anhalteweg ist 40.0 Meter lang

Aufgabe 5

Gegeben sind verschiedene Variablen. Bestimmen Sie zuerst ohne Computer, welchem Datentyp sie jeweils zugehören und überprüfen Sie Ihre Lösung dann mit WebTigerPython.

```
"2+3"      2+3      2.0+3      2.0+3.0
"2" * 3    "2" + "2"  "True"    True
```

Aufgabe 6

Welche der untenstehenden Ausdrücke sind gültig in Python?

```
3**2+4**2      ("go"*3)+"yes"      3.14*10*10  
"go"+3        "go"*3            3_14+10
```

Aufgabe 7

Erklären Sie in Ihren eigenen Worten, warum `print("15+10")` nicht das gleiche gibt wie `print(15+10)`.

4. Operatoren

Die Operatoren bestimmen, was man mit den Variablen machen kann. Sie verknüpfen Variablen oder Werte und liefern ein Resultat. Es gibt sehr viele Operatoren. Einige davon schauen wir uns an.

Berechnungen mit Operatoren



```
1 celcius = 15
2 fahrenheit = celcius * 9 / 5 + 32
```

4.1. Arithmetische Operatoren

- Durch normales Rechnen werden zwei Zahlen zu einer neuen Zahl verknüpft.

Operator	Bedeutung	Beispiel	Ergebnis
+	Addition	3 + 2	5
-	Subtraktion	7 - 4	3
*	Multiplikation	6 * 2	12
/	Division (immer als Kommazahl)	7 / 2	3.5
//	Ganzzahl-Division (Abrunden)	7 // 2	3
%	Modulo (Rest)	7 % 2	1
**	Potenz	2 ** 3	8

4.2. Vergleichsoperatoren

- Operatoren können als Resultat auch Wahrheitswerte (`True/False`) liefern.
- Es werden zwei Ausdrücke miteinander verglichen.
- Stimmt die notierte Aussage, ergibt der Operator `True` und sonst `False`.

Operator	Bedeutung	Beispiel	Ergebnis
==	gleich	3 == 3	True
!=	ungleich	4 != 4	False
<	kleiner als	2 < 5	True
<=	kleiner gleich als	7 <= 5	False
>	grösserals als	10 > 7	True
>=	grösser gleich als	3 >= 4	False

Wichtig

In Python verwendet man `==` (Doppel-Gleich) für «gleich», weil das Symbol `=` schon für die Zuweisung von Variablen verwendet wird!

4.3. Logische Operatoren

- Bedingungen werden kombiniert und das Resultat ist ein Wahrheitswert.

Operator	Bedeutung	Wann ergibt der Operator <code>True</code> ?	Beispiel	Ergebnis
<code>and</code>	UND	wenn beide Bedingungen True sind	<code>(3 > 1) and (2 < 5)</code>	True
<code>or</code>	ODER	wenn (mindestens) eine der Bedingungen True ist	<code>(3 > 5) or (2 < 5)</code>	True
<code>not</code>	NICHT	wenn die Bedingungen False ist	<code>not (3 > 5)</code>	True

4.4. Text-Operatoren

- Für den Datentyp String gibt es besondere Operatoren. Sie verhalten sich anders als jende von den Zahlen.

Operator	Bedeutung	Beispiel	Ergebnis
<	alphabetische Sortierung	<code>"Apfel" < "Banane"</code>	True
==	gleich	<code>"Hallo" == "Tschüss"</code>	False
!=	nicht gleich	<code>"Kathe" != "katze"</code>	True
+	Verkettung	<code>"hallo" + "hallo"</code>	<code>"hallohallo"</code>
*	Wiederholung	<code>"ha" * 3</code>	<code>"hahaha"</code>

4.5. Zuweisungsoperatoren

- Wir wissen bereits, dass das Zeichen `=` bedeutet «weise den Wert rechts der Variablen links zu» (Zuweisung).
- In Python gibt es Kurzformen, die eine Rechnung und eine Zuweisung kombinieren:

Operator	Bedeutung	Beispiel	Ergebnis
<code>+=</code>	Addition und Zuweisung	<code>x += 1</code>	<code>x = x + 1</code>
<code>-=</code>	Subtraktion und Zuweisung	<code>x -= 2</code>	<code>x = x - 2</code>
<code>*=</code>	Multiplikation und Zuweisung	<code>x *= 3</code>	<code>x = x * 3</code>
<code>/=</code>	Division und Zuweisung	<code>x /= 2</code>	<code>x = x / 2</code>
<code>//=</code>	Ganzzahl-Division und Zuweisung	<code>x //= 2</code>	<code>x = x // 12</code>
<code>%=</code>	Modulo und Zuweisung	<code>x %= 2</code>	<code>x = x % 2</code>
<code>**=</code>	Potenz und Zuweisung	<code>x **= 3</code>	<code>x = x ** 3</code>

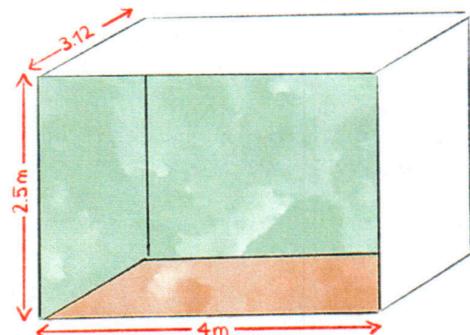
Aufgabe 8

Sie haben folgendes ausgemessenes Zimmer, bei dem an den Wänden langsam die Farbe abblättert.

Erstellen Sie in WebTigerPython passende Variablen und berechnen Sie die Gesamtfläche, die neu gestrichen werden muss.

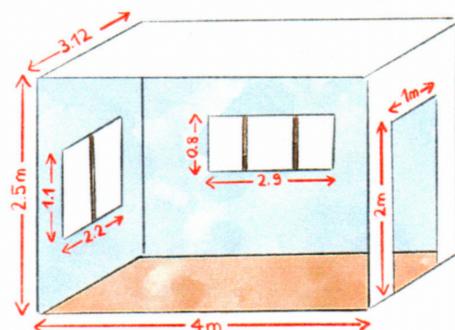
Sie können mit folgendem Code starten und dann weiter ausbauen:

```
1 laenge = 4
2 hoehe = 2.5
3 breite = ...
4 wand_links = hoehe * breite
5 ...
6 gesamtflaeche = ...
7 print(gesamtflaeche)
```



Aufgabe 9

Das Zimmer wurde renoviert und es hat jetzt zwei Fenster und eine Tür. Die Farbe an der Wand gefällt uns aber nicht mehr und wir möchten sie wieder neu streichen. Passen Sie Ihr Programm so an, dass Sie die neue Wandfläche berechnen.



Aufgabe 10

Probieren Sie ein paar der Operatoren aus den Tabellen aus. Überlegen Sie sich jeweils vor dem Ausführen, was Sie als Resultat erwarten. Zum Beispiel:

```
1 name = "jo"
2 print(name*3)
```

```
1 a = 16
2 b = 3
3 print(a/b)
4 print(a*b)
```

```
1 print(0 == 0)
2 print(0 != 0)
```

5. Kontrollstrukturen

Mit Verzweigungen oder Schleifen kann der Programmablauf gesteuert werden.

5.1. Verzweigungen: `if else`

Verzweigungen

```
1 if punkte > 20:  
2   print("Gewonnen!")
```

 Py

Verzweigungen

```
1 if punkte > 20:  
2   print("Gewonnen!")  
3 else:  
4   print('Verloren.')
```

 Py

Verzweigungen

```
1 if punkte > 20:  
2   print("Gewonnen!")  
3 elif punkte < 10:  
4   print("Verloren")  
5 else:  
6   print("Nochmals versuchen!")
```

 Py

5.2. Schleifen: `while` und `for`

Das Einrücken ist wichtig. Dadurch sagen wir Python was alles zur Wiederholung gehört und was nicht mehr.

Schleife

```
1 while punkte < 10:  
2   x = 5
```

 Py

5.3. Bedingungen formulieren

Bei der Formulierung von Bedingungen wird oft der Wert einer Variablen mit einem Vorgegebenen Wert verglichen.

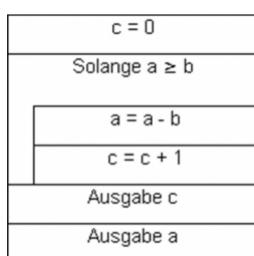
Bedingung in Worten	Bedingung in Python
Punkte sind grösser als 50	<code>punkte > 50</code>
Level ist kleiner oder gleich als 3	<code>level <= 3</code>
Zeit ist gleich 0	<code>zeit == 0</code>

 **Merke**
Das Zeichen `=` hat in Python nicht dieselbe Bedeutung wie in der Mathematik.

Aufgabe 11

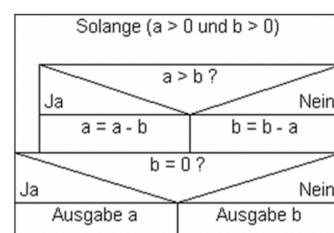
Übersetzen Sie das Struktogramm in Python Code.

Überprüfen Sie den Code einmal mit `a = 14` und `b = 4` und einmal mit `a = 38` und `b = 5`.



Aufgabe 12

Übersetzen Sie das Struktogramm in Python Code und prüfen Sie Ihren Code ob er richtig funktioniert.



Aufgabe 13

Schreiben Sie ein Programm in Python, welches für eine Benutzer-Eingabe die sogenannte Fakultät berechnet. Die Fakultät berechnet für eine natürlichen Zahl das Produkt aller natürlichen Zahlen kleiner oder gleich dieser Zahl. Das Zeichen für die Fakultät ist ein Ausrufezeichen (!).

Beispiel:

$1! = 1$
 $2! = 2 \cdot 1 = 2$
 $3! = 3 \cdot 2 \cdot 1 = 6$
 $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$
 $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$
...

Formatieren Sie die Ausgabe folgendermassen:

5! = 120

Aufgabe 14

Schreiben Sie ein Programm in Python, welches die grösste von drei Zahlen findet. Dabei sollen die drei Zahlen als Benutzerinput eingelesen werden. Der Output soll folgendermassen aussehen:

max(3, 15, -7) = 15

Aufgabe 15

Schreiben Sie ein Python-Programm, welches den Zeitraum zwischen zwei Uhrzeiten bestimmt.

Es soll angegeben werden, wie viel Zeit zwischen der Startzeit (Startstunde, Startminute) und der Endzeit (Endstunde, Endminute) liegt. Die Angabe soll in Stunden und Minuten erfolgen. Beide Uhrzeiten liegen innerhalb des gleichen Tages.

Beispiele:

- Von 6:46 bis 13:28 sind es 6 Stunden und 42 Minuten.
- Von 14:20 bis 15:07 sind es 47 Minuten.

6. Funktionen

Wir können in **Funktionen** einen Ablauf speichern und diese dann mit verschiedenen Parametern aufrufen.

Auch hier ist das **Einrücken** wichtig.

Funktionen

```
1 def fahrenheit_to_celsius(fahrenheit):
2     celsius = (fahrenheit - 32) * 5/9
3     return celsius
4
5 fahrenheit_to_celsius(77)
6 fahrenheit_to_celsius(86)
7 fahrenheit_to_celsius(50)
```



Aufgabe 16

Arbeiten Sie die Seite [Funktionen](#) im **Turtle-Tutorial** durch. Probieren Sie die **Musterbeispiele** aus und lösen Sie die Aufgaben 1-3c im Abschnitt **Zum selbst Lösen**.

Aufgabe 17

Arbeiten Sie die Seite [Parameter](#) im **Turtle-Tutorial** durch. Probieren Sie die **Musterbeispiele** aus und lösen Sie die Aufgaben 1-5 im Abschnitt **Zum selbst Lösen**.

Hinweise:

- Bei Aufgabe 1 ist der Drehwinkel 10 Grad.

Lösungen

Lösung 1

gültig:

```
Summe_1 = 433
summe_1 = 123
_my_name = "Pikachu"
L337Code = 42
summel = 63
Byte8Numb3r = 16
```

ungültig:

```
*myname = "Glurak"
8Bit = 256
my name = "Flegmon"
```

Variablenname auf Gültigkeit überprüfen:

Zuweisung mit dem Variablenamen in WebTigerPython eingeben und den Code laufen lassen. Bei einem ungültigen Namen wird eine Fehlermeldung erscheinen.

Lösung 2

Aufgabe 1

```
1 from gturtle import *
2 setPenColor("grey")
3 hideTurtle()
4 n = inputInt("Anzahl Perlen?")
5 repeat n:
6     dot(20)
7     forward(22)
8     right(360/n)
```

Aufgabe 2

```
1 from gturtle import *
2 hideTurtle()
3 s = 5
4 w = inputInt("Gib den Drehwinkel an")
5 repeat 100:
6     forward(s)
7     right(w)
8     s += 1
```

Aufgabe 3

```
1 from gturtle import *
2 hideTurtle()
3 setPenColor("blue")
4 s = 5
5 repeat 100:
6     repeat 4:
7         forward(s)
8         right(90)
9         right(6)
10        s += 2
```

Lösung 3

Ein-/Ausgabe

```
1 geschwindigkeit = inputInt("Bitte
2 Geschwindigkeit in km/h eingeben:")
3 reaktionsweg = geschwindigkeit/10 * 3
4 bremsweg = (geschwindigkeit/10) ** 2
5 anhalteweg = reaktionsweg + bremsweg
6 print(anhalteweg)
```

Lösung 4

Diese Zeile ist anzupassen

```
5 print("Der Anhalteweg ist " +
str(anhalteweg) + " Meter lang")
```

Alternative

```
5 print("Der Anhalteweg ist", anhalteweg,
"Meter lang")
```

Lösung 5

"2+3" → str	2+3 → int
2.0+3 → float	2.0+3.0 → float
"2"*3 → str	"2"+"2" → str
"True" → str	True → bool

Lösung 6

Nur "go"+3 ist ungültig. Alle anderen sind gültig.

Lösung 7

Alles in Anführungszeichen wird als Text behandelt. Deswegen ist für Python der Ausdruck "15+10" reiner Text und wird nicht berechnet. Der Ausdruck `print("15+10")` liefert den Output 15+10 (als Text) und der Ausdruck `print(15+10)` liefert 25 (als Zahl).

Lösung 8

Gesamtfläche aller vier Wände

```
1 laenge = 4
2 hoehe = 2.5
3 breite = 3.12
4 wand_links = hoehe * breite
5 wand_hinten = hoehe * laenge
6 gesamtflaeche = 2 * wand_links + 2 *
7 print(gesamtflaeche)
```

Lösung 9

Fläche der Wände ohne Fenster und Tür  Py

```

1 laenge = 4
2 hoehe = 2.5
3 breite = 3.12
4 wand_links = hoehe * breite
5 wand_hinten = hoehe * laenge
6 fenster_links = 1.1 * 2.2
7 fenster_hinten = 0.8 * 2.9
8 tuere = 2 * 1
9 gesamtflaeche = 2 * wand_links + 2 *
wand_hinten - fenster_links -
fenster_hinten - tuere
10 print(gesamtflaeche)

```

Lösung 10

Zum Beispiel:

Arithmetische Operatoren  Py

```

1 a = 16
2 b = 3
3 print(a + b) # 19
4 print(a - b) # 13
5 print(a * b) # 48
6 print(a / b) # 5.333333333333333
7 print(a // b) # 5
8 print(a % b) # 1
9 print(a ** b) # 4096

```

Vergleichsoperatoren  Py

```

1 print(0 == 0) # True
2 print(0 != 0) # False
3 print(2 < 3) # True
4 print(2 > 2) # False
5 print(2 <= 2) # True
6 print(-2 >= 2) # False

```

Logische Operatoren  Py

```

1 a = True
2 b = False
3 print(a and b) # False
4 print(a or b) # True
5 print(not a) # False
6 print(not b) # True

```

Text-Operatoren  Py

```

1 name = "jo"
2 print(name * 3) # jojojo
3 print(name + "ghurt") # joghurt
4 print("gymthun" != "GymThun") # True
5 print("W" * 3 == "WWW") # True

```

Lösung 11

```

1 a = inputInt("a=")
2 b = inputInt("b=")
3 c = 0
4 while a >= b:
5     a = a - b
6     c = c + 1
7 print("c=" + str(c))
8 print("a=" + str(a))

```

Lösung 12

```

1 a = inputInt("a=")
2 b = inputInt("b=")
3 while a > 0 and b > 0 :
4     if a > b :
5         a = a - b
6     else:
7         b = b - a
8 if b == 0:
9     print(a)
10 else:
11     print(b)

```

Lösung 13

```

1 zahl = inputInt("Zahl eingeben:")
2 produkt = 1
3 output = str(zahl)+"!" = "
4 while zahl > 1 :
5     produkt *= zahl
6     zahl -= 1
7 print(output + str(produkt))

```

Lösung 14

```

1 zahl1 = inputInt("erste Zahl")
2 zahl2 = inputInt("zweite Zahl")
3 zahl3 = inputInt("dritte Zahl")
4 if zahl1 > zahl2:
5     if zahl1 > zahl3:
6         resultat = zahl1
7     else:
8         resultat = zahl3
9 else:
10    if zahl2 > zahl3:
11        resultat = zahl2
12    else:
13        resultat = zahl3

```

```

print("max(" + str(zahl1) + ", " +
14 str(zahl2) + ", " + str(zahl3) + ") = "
+ str(resultat))

```

Lösung 15

```

1 startzeit_stunden =
  inputInt("Startzeit Stunden:")
2 startzeit_minuten = inputInt("Startzeit
  Minuten:")
3 endzeit_stunden = inputInt("Endzeit
  Stunden:")
4 endzeit_minuten = inputInt("Endzeit
  Minuten:")
5 differenz_stunden = endzeit_stunden -
  startzeit_stunden
6 differenz_minuten = endzeit_minuten -
  startzeit_minuten
7 if differenz_minuten < 0:
8     differenz_stunden -= 1
9     differenz_minuten += 60
10 print(str(differenz_stunden) + " Stunden
  und " + str(differenz_minuten) + " Minuten")

```

Lösung 16

Aufgabe 1

```

1 from gturtle import *
2 def sechseck():
3     repeat 6:
4         forward(60)
5         left(60)
6     setPenColor("blue")
7     hideTurtle()
8     repeat 10:
9         sechseck()
10        right(36)

```

Aufgabe 2

```

1 from gturtle import *
2 def square():
3     right(45)
4     repeat 6:
5         forward(60)
6         right(90)
7         right(135)
8     hideTurtle()
9     setPenColor("blue")
10    square()
11    setPenColor("red")
12    square()
13    setPenColor("lime")

```

```

14 square()
15 setPenColor("black")
16 square()

```

Aufgabe 3a-c

```

1 from gturtle import *
2 def bogen():
3     repeat 90:
4         forward(1)
5         right(1)
6     def blumenblatt():
7         repeat 2:
8             bogen()
9             right(90)
10    def blume():
11        repeat 8:
12            blumenblatt()
13            right(360/8)
14    hideTurtle()
15    blume()

```

Lösung 17

Aufgabe 1

```

1 from gturtle import *
2 def square(s):
3     repeat 4:
4         forward(s)
5         right(90)
6     hideTurtle()
7     s = 180
8     repeat 100:
9         square(s)
10        s *= 0.95
11        left(10)

```