Einführung: Verzweigungen

In der letzten Sitzung zu Python habt ihr euch mit Schleifen beschäftigt. Heute geht es darum, erste Programme mit **Verzweigungen** zu schreiben und zu verstehen.

1. Aufgabe

a. Gebt den Python Code unten in euren Editor ein.
Klickt auf "Run" (Play Symbol) und schaut euch das Ergebnis in der Konsole an.
b. Testet verschiedene Eingaben.
c. Lest euch den Quellcode und die Kommentare (mit # gekennzeichnet) durch.
d. Passt den Code so an, dass er **bei 0 "Du hast eine Null eingegeben."** ausgibt.
Tipp: Weitere Fälle kannst du mit dem Schlüsselwort ***elif*** hinzufügen.
e. Eine Erklärung, was hier passiert, findet ihr unten. Eine mögliche Lösung auf der nächsten Seite.

```
1  zahl = int(input('Gib eine beliebige ganze Zahl ein.'))
2
3  #Wenn eine Zahl kleiner als 0 eingegeben wird, gib aus,
   dass eine negative Zahl eingegeben wurde. Ansonsten gib
   aus, dass eine positive Zahl eingegeben wurde.
4  if(zahl < 0):
5    print('Du hast eine negative Zahl eingegeben.')
6  else:
7    print('Du hast eine positive Zahl eingegeben.')
```

```
Gib eine beliebige ganze Zahl ein.
```

Tipp: Ihr könnt parallel Sitzung 1 und Sitzung 2 öffnen, um z.B. nachzuschauen, wie ihr input() einsetzt.

Verzweigungen

Um Zahlen zu unterscheiden, habt ihr eine **Verzweigung** verwendet:

- **if 'Bedingung1':**
  'Befolge diese Anweisung'

Verzweigungen braucht ihr immer dann, wenn ihr **Fälle** unterscheiden möchtet. Mit Hilfe eines logischen Operators könnt ihr Unterschiede ausmachen. Was ein Vergleichsoperator ist, habt ihr schon in der letzten Sitzung zu Schleifen gelernt.

Bei Verzweigungen könnt ihr **beliebige Datentypen vergleichen**, zum Beispiel Zahlen. Das Ergebnis ist immer **True oder False**. Trifft eine Bedingung zu (d.h. das Ergebnis ist wahr (true)), werden die Anweisungen in dem Code-Block ausgeführt. Die Anweisungen in den anderen Bedingungen werden ignoriert.

Achtung! Verzweigungen werden häufig mit Schleifen verwechselt. Das ist aber nicht korrekt. Folgendes unterscheidet Schleifen und Verzweigungen:

- **Schleifen** werden genutzt, um Dinge zu **wiederholen**. Eine Schleife braucht eine Abbruchbedingung.
- **Verzweigungen** werden genutzt, um **Fälle zu unterscheiden**. Je nach Fall werden bestimmte Anweisungen ausgeführt. Die Anweisungen in den übrigen Fällen werden ignoriert.

Es gibt verschiedene Arten von Verzweigungen. Hier habt ihr eine sogenannte **IF-Verzweigung** verwendet.

Mögliche Lösung - Aufgabe 1

```
1  zahl = int(input('Gib eine beliebige ganze Zahl ein.'))
2
3  if(zahl < 0):
4    print('Du hast eine negative Zahl eingegeben.')
5  elif(zahl > 0):
6    print('Du hast eine positive Zahl eingegeben.')
7  else:
8    print('Du hast eine Null eingegeben.')
```

```
Gib eine beliebige ganze Zahl ein.0
Du hast eine Null eingegeben.
```

Hinweis: Sieht eure Lösung anders aus? Kein Problem! Beim Programmieren gibt es oft mehrere Ansätze, um die gleiche Lösung zu erhalten.

Python Merkzettel

Nehmt euren **Python Merkzettel** und füllt die Box zum Thema **Verzweigungen** aus. Lasst noch etwas Platz, um die Notizen später zu ergänzen.

Hier findet ihr den Merkzettel. Druckt ihn entweder aus und füllt ihn handschriftlich oder bearbeitet das pdf-Dokument direkt an eurem Rechner. Vergesst dann bitte nicht, das Dokument zu speichern und sinnvoll abzulegen.

(Alle Lehrkräfte finden den Merkzettel unter “Vorbereitung” auf der App Camps Plattform.)

2. Aufgabe

Zahlen in Verzweigungen mit Vergleichsoperatoren unterscheiden

a. Gebt den Python Code unten in euren Editor ein und klickt auf “Run”.
b. Testet verschiedene Eingaben.
c. Lest euch den Quellcode und die Kommentare (mit # gekennzeichnet) durch.
d. Eine Erklärung, was hier passiert, findet ihr auf der nächsten Seite.

```
1  uhrzeit = int(input("Gib eine Uhrzeit ein:"))
2
3  if(uhrzeit < 7 or uhrzeit > 23):
4    print('Es ist Zeit zu schlafen')
5  else:
6    print('Carpe Diem! - Genieße den Tag')
```

```
Gib eine Uhrzeit ein:
```

Hinweis: Solltet ihr mal etwas nicht verstehen, fragt eure MitschülerInnen. Gemeinsam könnt ihr natürlich auch eure Lehrkraft um Hilfe bitten.

3. Aufgabe

Zahlen in Verzweigungen mit Vergleichsoperatoren unterscheiden

a. Lest euch die **Infobox** unten aufmerksam durch.
b. Überlegt euch nun mögliche Anwendungsfälle für Verzweigungen. Tauscht euch mit euren MitschülerInnen aus.
c. Auf der nächsten Seite findet ihr Übungsaufgaben.

Logische Operatoren

Es gibt verschiedene sogenannte **logische Operatoren**. In der letzten Sitzung habt ihr bereits logische Vergleichsoperatoren kennengelernt (z.B. "kleiner als", <).

Außerdem gibt es auch **logische Verknüpfungen** (z.B. "oder"). Mit einer Verknüpfung könnt ihr Bedingungen in Abhängigkeit setzen. Die Operatoren können beliebig verschachtelt werden. Das kann schon mal ganz schön kompliziert werden.

Hier findet ihr eine **Übersicht** logischer Operatoren:

Operator	Was macht der Operator?	Beispiel
Verknüpfungen		
and	Und - Ist nur wahr, wenn beide Bedingungen zutreffen.	zahl > 5 and zahl < 10 (True für 6,7,8,9)
or	Oder - Ist wahr, wenn eine der Bedingungen oder beide zutreffen.	zahl > 5 or zahl < 10 (True für 1,2,...,6,...,10,11,...) oder zahl < 5 or zahl > 10 (True für ...,3,4,11,12,...)
not	Nicht - macht aus True, False und umgekehrt	not (1 != X) (True für x = 1)
Vergleiche		
==	Ist gleich	1 == 1 (wahr) oder 'Hallo' == 'Halli' (falsch)
!=	Ist ungleich	1 != 1 (falsch) oder 'Hallo' != 'Halli' (wahr)
>	Ist größer als	2 > 1 (wahr) oder 1 > 1 (falsch) oder 1 > 2 (falsch)
>=	Ist größer als oder gleich	2 >= 1 (wahr) oder 1 >= 1 (wahr) oder 1 >= 2 (falsch)
<	Ist kleiner als	1 < 2 (wahr) oder 1 < 1 (falsch) oder 2 < 1 (falsch)
<=	Ist kleiner als oder gleich	1 <= 2 (wahr) oder 1 <= 1 (wahr) oder 2 <= 1 (falsch)

In **Python** werden die Verknüpfungen **ausgeschrieben**. In anderen Programmiersprachen werden in der Regel **Symbole** genutzt. So wird für den UND-Operator in Python "and" geschrieben und in der Programmiersprache Java zum Beispiel &&. Der ODER-Operator heißt in Python "or" und in Java ||. Wundert euch also nicht, wenn ihr euch mal mit anderen Programmiersprachen auseinandersetzt.

Python Merkzettel

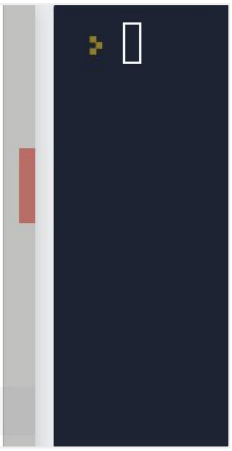
Nehmt euren **Python Merkzettel** und ergänzt die Box zum Thema **Logische Operatoren**. Lasst noch etwas Platz, um die Notizen später zu ergänzen.

4. Aufgabe

Erstellt einen Tagesplaner: Was macht ihr gerade?

a. **Überlegt zunächst handschriftlich**, was ihr zu welcher Uhrzeit macht. (z.B. Schule von 8-14 Uhr, Abendbrot um 19 Uhr usw.)
b. Befolgt nun die **Anweisungen in den #Kommentaren** unten im Quellcode. Erstellt für die jeweiligen Uhrzeiten eine Wenn-Dann-Verzweigung (if-elif-else).
c. Überlegt euch, warum es sinnvoll ist, hier eine **Verzweigung statt einer Schleife** zu verwenden. Wenn ihr möchtet, probiert es aus, damit ihr selbst seht, was passiert.
d. Auf der nächsten Seite findet ihr eine mögliche Lösung und eine Erklärung zu dem, was hier passiert.

```
1  #Erstellt eine Variable Uhrzeit. Lasst die aktuelle
   Uhrzeit im Format Stunde.Minute eingeben. (Tipp: Nutze
   dazu input() und parse den Input in einen float)
2
3  print("Da es aktuell", uhrzeit , "Uhr ist, bin ich
   (vermutlich) gerade dabei zu:")
4
5  #Erstellt eine if-elif-else-Verzweigung für deinen
   Tagesplaner
```



Beim Programmieren ist es oft hilfreich, Dinge einfach **auszuprobieren**, um zu sehen, was passiert und dann auch zu verstehen. Wichtig ist nur, dass ihr immer eine funktionierende Version eures Codes als **Backup** speichert.

Mögliche Lösung - Aufgabe 4

```
1   #Mein Tagesplaner: Was mache ich gerade, bzw. was
    sollte ich gerade tun?
2
3   uhrzeit = float(input('Gib die aktuelle Uhrzeit im
    Format Stunde.Minute ein.'))
4
5   print("Da es aktuell", uhrzeit , "Uhr ist, bin ich
    (vermutlich) gerade :")
6
7   #Ermitteln was ich laut Tagesplan gerade (wohl) mache:
8
9   if uhrzeit < 7: print('am Schlafen')
10  elif uhrzeit < 8: print('beim Frühstücken')
11  elif (uhrzeit > 12 and uhrzeit < 13): print('beim
    Mittagessen')
12  elif uhrzeit < 17: print('am Arbeiten')
13  elif uhrzeit < 18: print('auf dem Heimweg')
14  elif uhrzeit < 20: print('beim Abendessen')
15  elif uhrzeit < 22: print('beim Entspannen')
16  else: print('am Schlafen')
```

```
Gib die aktuelle Uhrzeit im Format Stunde.Minute ein.13.20
Da es aktuell 13.2 Uhr ist, bin ich (vermutlich) gerade :
am Arbeiten
```

5. Aufgabe

a. Lest euch zunächst die Beispiel-Lösung oben durch. Habt ihr alles verstanden? Erklärt einem eurer MitschülerInnen das Programm.

b. Lest euch die blaue Infobox durch.

c. Überlegt dann, wie ihr das Programm erweitern könnt. Wo könnte es dennoch sinnvoll sein, eine **Schleife** in das Programm einzubinden?

d. Auf den nächsten Seiten findet ihr zwei mögliche Lösungen.

Verzweigungen versus Schleifen

Eine kleine Erinnerung:

- **Schleifen** werden genutzt, um Dinge zu **wiederholen**. Eine Schleife braucht eine Abbruchbedingung.
- **Verzweigungen** werden genutzt, um **Fälle zu unterscheiden**. Je nach Fall werden bestimmte Anweisungen ausgeführt.

Wenn wir in unserem Beispiel also eine Schleife verwenden, sieht das so aus:

```
am Schlafen
am Schlafen
am Schlafen

Repl.it: Stopped
```

Innerhalb weniger Sekunden wird X mal der Text ausgegeben. Aber warum?

- Mit deiner Eingabe legst du die Uhrzeit fest (im Beispiel rechts 23.30 Uhr).
- Der Computer merkt sich die Uhrzeit, und da es keine Abbruchbedingung gibt, wird immer wieder der print-Befehl ausgeführt. Es entsteht eine Endlosschleife.

Python Merkzettel

Ergänzt ggf. euren **Python Merkzettel**.

Mögliche Lösung - Aufgabe 5: Dauerhafte Abfrage

```
#Mein Tagesplaner: Was mache ich gerade, bzw. was sollte ich gerade tun?

while True:
    uhrzeit = float(input('Gib die aktuelle Uhrzeit im Format Stunde.Minute ein.'))

    print("Da es aktuell", uhrzeit , "Uhr ist, bin ich (vermutlich) gerade:")

    #Ermitteln was ich laut Tagesplan gerade (wohl) mache:

    if uhrzeit < 7: print('am Schlafen')
    elif uhrzeit < 8: print('beim Frühstücken')
    elif (uhrzeit > 12 and uhrzeit < 13): print('beim Mittagessen')
    elif uhrzeit < 17: print('am Arbeiten')
    elif uhrzeit < 18: print('auf dem Heimweg')
    elif uhrzeit < 20: print('beim Abendessen')
    elif uhrzeit < 22: print('beim Entspannen')
    else: print('am Schlafen')
```

```
Gib die aktuelle Uhrzeit im Format Stunde.Minute ein.06.20
Da es aktuell 6.2 Uhr ist, bin ich (vermutlich) gerade:
am Schlafen
Gib die aktuelle Uhrzeit im Format Stunde.Minute ein.07.30
Da es aktuell 7.3 Uhr ist, bin ich (vermutlich) gerade:
beim Frühstücken
Gib die aktuelle Uhrzeit im Format Stunde.Minute ein.09.30
Da es aktuell 9.3 Uhr ist, bin ich (vermutlich) gerade:
am Arbeiten
Gib die aktuelle Uhrzeit im Format Stunde.Minute ein.
```

Habt ihr euch ähnliche Beispiele überlegt? Wenn ihr noch nicht wisst, wie ihr das, was ihr euch überlegt habt, in Code umsetzen könnt, ist das überhaupt nicht schlimm.

Generell gilt: **Pair Programming** kann sehr hilfreich sein, um auf gute Lösungen zu kommen. #Inspiration #4AugenSehenMehr

Beim Programmieren besteht ein Großteil der Zeit aus **Suchen nach Lösungen** im Internet. Es gibt (fast) immer jemanden, der das Problem schon einmal hatte und einen Forenbeitrag aufgesetzt hat. Dort findet ihr dann häufig auch die Lösung. Ein tolles Forum ist zum Beispiel **Stackoverflow**. Unter appcamps.link/python7 finet ihr die neuesten Beiträge zu Python.

6. Aufgabe

a. Lest euch die erste **Beispiel-Lösung** oben durch. Habt ihr alles verstanden? Erklärt einer/m eurer MitschülerInnen das Programm.

b. Lest euch den Text unter appcamps.link/python8 zur Kombination von Schleifen und Verzweigungen durch. Dort findet ihr ein weiteres anschauliches Beispiel, in dem eine **WHILE-Schleife** mit einer **IF-Verzweigung** kombiniert wird.
Außerdem erfahrt ihr, was das Schlüsselwort **'break'** macht.

c. Auf der nächsten Seite findet ihr eine weitere mögliche Lösung.

Mögliche Lösung - Aufgabe 6: Systemzeit

```python
#Mein Tagesplaner: Was mache ich gerade, bzw. was sollte ich gerade tun?

#Systemzeit 'importieren' und Zeitversatz anpassen
import time
jetzt = time.localtime()
uhrzeit_stunde = jetzt.tm_hour + 1
uhrzeit_minute = jetzt.tm_min

print("Da es aktuell", str(uhrzeit_stunde)+':'+str(uhrzeit_minute), "Uhr ist, bin ich (vermutlich) gerade:")

#Ermitteln was ich laut Tagesplan gerade (wohl) mache:

if uhrzeit_stunde < 7: print('am Schlafen')
elif uhrzeit_stunde < 8: print('beim Frühstücken')
elif (uhrzeit_stunde > 12 and uhrzeit_stunde < 13): print('beim Mittagessen')
elif uhrzeit_stunde < 17: print('am Arbeiten')
elif uhrzeit_stunde < 18: print('auf dem Heimweg')
elif uhrzeit_stunde < 20: print('beim Abendessen')
elif uhrzeit_stunde < 22: print('beim Entspannen')
else: print('am Schlafen')
```

```
Da es aktuell 10:9 Uhr ist, bin ich (vermutlich) gerade:
am Arbeiten
```

7. Aufgabe

a. Lest euch die zweite **Beispiel-Lösung** oben durch.

b. Habt ihr alles verstanden? Wenn nicht, ist das überhaupt nicht schlimm. Tatsächlich greifen wir hier bereits ein sehr komplexes Thema auf, nämlich **Objektorientierung.** Das ist aber ein Thema für eine andere Stunde. Falls euch das trotzdem interessiert, sucht doch mal im Internet danach.

c. Wägt **Vor- und Nachteile** der beiden Beispiel-Lösungen ab. Überlegt zum Beispiel, welche Folgen es haben könnte, dass das Programm in einer Dauerschleife ist oder, dass der Nutzer nicht mehr die Möglichkeit hat, die Zeit selbst einzugeben.

d. Lest euch unten die blaue Box mit möglichen **Verbesserungen** durch.

Wie können wir das Programm verbessern?

- Nur Zahlen bzw. Uhrzeiten zulassen, die auch **tatsächlich möglich** sind (z.B. nicht 28.66 Uhr).
- Statt eine Dauerschleife zu verwenden, kann es effizienter sein, die Abfrage z.B. **über eine Taste** zu aktivieren.
- Dem Benutzer die Möglichkeit geben, die **Zeitzone** anzupassen.
- Nicht zu vergessen: Eine schöne grafische **Benutzeroberfläche (GUI)**. Unter appcamps.link/python9 findet ihr weitere Infos zu GUIs.

Ihr seht also: Bereits eine einfache Anwendung wie diese kann beliebig erweitert werden.

1. Knobelaufgabe zu Schleifen & Verzweigungen

a. Schau euch den Code an. Um Schleifen und Verzweigungen richtig zu schreiben, müsst ihr nachvollziehen können, wie beides funktioniert.

```
1   zaehler = 0
2   ergebnis = 0
3   schleifendurchlaeufe = 0
4
5   while zaehler < 5:
6     ergebnis = ergebnis + zaehler
7     if zaehler > 2:
8       zaehler = zaehler + 2
9     else:
10      zaehler = zaehler + 1
11    print (ergebnis,zaehler)
12    schleifendurchlaeufe = schleifendurchlaeufe + 1
13
14  print("Fertig! Ergebnis = ", ergebnis)
15  print("Schleifendurchläufe: ", schleifendurchlaeufe)
16  print("Zähler: ", zaehler)
```

b. Geht die Schleifendurchläufe Schritt für Schritt durch, um die folgenden Fragen zu beantworten.

- Welche Ausgabe erhaltet ihr, wenn ihr das Programm startet? Welchen Wert hat die Variable 'ergebnis'?
- Wie oft wird die Schleife durchlaufen? Welchen Wert hat die Variable 'schleifendurchlaeufe'?

c. Nutzt die Tabelle unten als Vorlage, um alle Zwischenergebnisse zu notieren. Schreibt sie ab und füllt sie handschriftlich aus.

Schleifendurchlauf	Ergebnis	Zaehler
0 (zu Beginn)		
1		
2		
...		

d. Wenn ihr auf eine Lösung gekommen seid, geht auf die nächste Seite. Dort findet ihr die Musterlösung, um eure Antwort zu überprüfen.

Lösung 1. Knobelaufgabe

a. Schaut euch die Tabelle unten an. Sieht eure Tabelle genauso aus oder habt ihr etwas vergessen?
   - Welchen Wert hat die Variable 'ergebnis' zum Schluss?
   - Welchen Wert hat die Variable 'schleifendurchlaeufe' zum Schluss?

b. Überprüft euer Ergebnis im Editor.

c. Stimmt eure Lösung der Aufgabe mit der unten überein? Falls ja - super! Falls nicht - versucht nachzuvollziehen, wo der Fehler liegt. Tauscht euch eventuell mit euren MitschülerInnen aus.

Tipp: Im Editor könnt ihr jederzeit mit einem 'print' - Befehl das Zwischenergebnis ausgeben lassen und so überprüfen, ob das Programm macht, was ihr möchtet. Unter appcamps.link/python10 findet ihr den Code zu der Aufgabe.

Schleifendurchlauf	Ergebnis	Zaehler
0 (zu Beginn)	0	0
1	0	1
2	1	2
3	3	3
4	6	5

Antwort:

- Der Wert der Variable 'ergebnis' ist 6.
- Der Wert der Variable 'schleifendurchlaeufe' ist 4.

Die Schleife wird ausgeführt, solange der Zähler kleiner als 5 ist. Nach vier Schleifendurchläufen ist die Bedingung, dass der Zähler kleiner als fünf sein muss, nicht mehr erfüllt. Die Schleife wird also automatisch nicht mehr durchlaufen und der Code weiter unten (hier 3 print()-Befehle) wird ausgeführt.

Beim Zähler wird in jedem Schleifendurchlauf etwas hinzuaddiert. Sobald er größer als 2 ist, wird 2 addiert, davor 1. Beim Ergebnis wird zum aktuellen Wert der Zähler hinzuaddiert.

2. Knobelaufgabe zu Schleifen & Verzweigungen

a. Schau euch den Code an. Um Schleifen und Verzweigungen richtig zu schreiben, müsst ihr nachvollziehen können, wie beides funktioniert.

```
1   zaehler = -2
2   ergebnis = 0
3   schleifendurchlaeufe = 0
4
5   while zaehler < 5:
6     ergebnis = ergebnis + (zaehler * 2)
7     if zaehler > 0:
8       zaehler = zaehler + 2
9     else:
10      zaehler = zaehler + 1
11    schleifendurchlaeufe = schleifendurchlaeufe + 1
12
13  print("Fertig! Ergebnis = ", ergebnis)
14  print("Schleifendurchläufe: ", schleifendurchlaeufe)
15  print("Zähler: ", zaehler)
```

b. Geht die Schleifendurchläufe Schritt für Schritt durch, um folgende Fragen zu beantworten:

- Welche Ausgabe erhaltet ihr, wenn ihr das Programm startet? Welchen Wert hat die Variable 'ergebnis'?
- Wie oft wird die Schleife durchlaufen? Welchen Wert hat die Variable 'schleifendurchlaeufe'?

c. Nutzt die Tabelle unten als Vorlage, um alle Zwischenergebnisse zu notieren. Schreibt sie ab und füllt sie handschriftlich aus.

Schleifendurchlauf	Ergebnis	Zaehler
0 (zu Beginn)		
1		
2		
...		

d. Wenn ihr auf eine Lösung gekommen seid, geht auf die nächste Seite. Dort findet ihr die Musterlösung, um eure Antwort zu überprüfen.

Lösung 2. Knobelaufgabe

a. Schaut euch die Tabelle unten an. Sieht eure Tabelle genauso aus oder habt ihr etwas vergessen?
   - Welchen Wert hat die Variable 'ergebnis' zum Schluss?
   - Welchen Wert hat die Variable 'schleifendurchlaeufe' zum Schluss?

b. Überprüft euer Ergebnis im Editor.

c. Stimmt eure Lösung der Aufgabe mit der unten überein? Falls ja - super! Falls nicht - versucht nachzuvollziehen, wo der Fehler liegt. Tauscht euch eventuell mit euren MitschülerInnen aus.

Tipp: Im Editor könnt ihr jederzeit mit einem 'print' - Befehl das Zwischenergebnis ausgeben lassen und so überprüfen, ob das Programm macht, was ihr möchtet.

Schleifendurchlauf	Ergebnis	Zaehler
0 (zu Beginn)	0	-2
1	-4	-1
2	-6	0
3	-6	1
4	-4	3
5	2	5

Antwort:

- Der Wert der Variable 'ergebnis' ist 2.
- Der Wert der Variable 'schleifendurchlaeufe' ist 5.

Die Schleife wird ausgeführt, solange der Zähler kleiner als 5 ist. Nach fünf Schleifendurchläufen ist die Bedingung, dass der Zähler kleiner als fünf sein muss, nicht mehr erfüllt. Die Schleife wird also automatisch nicht mehr durchlaufen und der Code weiter unten (hier 3 print()-Befehle) wird ausgeführt.

Beim Zähler wird in jedem Schleifendurchlauf etwas hinzuaddiert. Solange er kleiner als 0 ist, wird 1 addiert, danach 2. Beim Ergebnis wird der Zähler, multipliziert mit 2, zum aktuellen Wert hinzuaddiert.

3. Knobelaufgabe zu Schleifen & Verzweigungen

a. Schau euch den Code an. Um Schleifen und Verzweigungen richtig zu schreiben, müsst ihr nachvollziehen können, wie Schleifen und Verzweigungen funktionieren.

```
1   zaehler = -28
2   ergebnis = 8
3   schleifendurchlaeufe = 0
4
5   while zaehler < 5:
6     ergebnis = ergebnis + (zaehler * 2)
7     if zaehler < -10:
8       zaehler += 13
9       #zaehler += 13 ist die Kurzschreibweise für zaehler
        = zaehler + 13
10    elif zaehler < 0:
11      zaehler +=  7
12    else:
13      zaehler = zaehler + 3
14    schleifendurchlaeufe += 1
15
16  print("Fertig! Ergebnis = ", ergebnis)
17  print("Schleifendurchläufe: ", schleifendurchlaeufe)
18  print("Zähler: ", zaehler)
```

b. Geht die Schleifendurchläufe Schritt für Schritt durch, um folgende Fragen zu beantworten:

- Welche Ausgabe erhaltet ihr, wenn ihr das Programm startet? Welchen Wert hat die Variable 'ergebnis'?
- Wie oft wird die Schleife durchlaufen? Welchen Wert hat die Variable 'schleifendurchlaeufe'?

c. Nutzt die Tabelle unten als Vorlage, um alle Zwischenergebnisse zu notieren. Schreibt sie ab und füllt sie handschriftlich aus.

Schleifendurchlauf	Ergebnis	Zaehler
0 (zu Beginn)		
1		
2		
...		

d. Wenn ihr auf eine Lösung gekommen seid, geht auf die nächste Seite. Dort findet ihr die Musterlösung, um eure Antwort zu überprüfen.

Lösung 3. Knobelaufgabe

a. Schaut euch die Tabelle unten an. Sieht eure Tabelle genauso aus oder habt ihr etwas vergessen?
   - Welchen Wert hat die Variable 'ergebnis' zum Schluss?
   - Welchen Wert hat die Variable 'schleifendurchlaeufe' zum Schluss?

b. Überprüft euer Ergebnis im Editor.

c. Stimmt eure Lösung der Aufgabe mit der unten überein? Falls ja - super! Falls nicht - versucht nachzuvollziehen, wo der Fehler liegt. Tauscht euch eventuell mit euren MitschülerInnen aus.

Tipp: Im Editor könnt ihr jederzeit mit einem 'print' - Befehl das Zwischenergebnis ausgeben lassen und so überprüfen, ob das Programm macht, was ihr möchtet.

Schleifendurchlauf	Ergebnis	Zaehler
0 (zu Beginn)	8	-28
1	-48	-15
2	-78	-2
3	-82	5

4. Knobelaufgabe zu Schleifen & Verzweigungen

a. Schau euch den Code an. Um Schleifen und Verzweigungen richtig zu schreiben, müsst ihr nachvollziehen können, wie Schleifen und Verzweigungen funktionieren.

```
1   zaehler = -28
2   ergebnis = 8
3   schleifendurchlaeufe = 0
4
5   while zaehler < 5:
6     if zaehler < -10:
7       zaehler += 13
8     elif zaehler < 0:
9       zaehler +=  7
10      ergebnis = ergebnis + (zaehler * 2)
11    else:
12      zaehler = zaehler + 3
13    ergebnis = ergebnis + zaehler
14    schleifendurchlaeufe += 1
15
16  print("Fertig! Ergebnis = ", ergebnis)
17  print("Schleifendurchläufe: ", schleifendurchlaeufe)
18  print("Zähler: ", zaehler)
```

b. Geht die Schleifendurchläufe Schritt für Schritt durch, um folgende Fragen zu beantworten:

- Welche Ausgabe erhaltet ihr, wenn ihr das Programm startet? Welchen Wert hat die Variable 'ergebnis'?
- Wie oft wird die Schleife durchlaufen? Welchen Wert hat die Variable 'schleifendurchlaeufe'?

c. Nutzt die Tabelle unten als Vorlage, um alle Zwischenergebnisse zu notieren. Schreibt sie ab und füllt sie handschriftlich aus.

Schleifendurchlauf	Ergebnis	Zaehler
0 (zu Beginn)		
1		
2		
...		

d. Wenn ihr auf eine Lösung gekommen seid, geht auf die nächste Seite. Dort findet ihr die Musterlösung, um eure Antwort zu überprüfen.

Lösung 4. Knobelaufgabe

a. Schaut euch die Tabelle unten an. Sieht eure Tabelle genauso aus oder habt ihr etwas vergessen?
   - Welchen Wert hat die Variable 'ergebnis' zum Schluss?
   - Welchen Wert hat die Variable 'schleifendurchlaeufe' zum Schluss?

b. Überprüft euer Ergebnis im Editor.

c. Stimmt eure Lösung der Aufgabe mit der unten überein? Falls ja - super! Falls nicht - versucht nachzuvollziehen, wo der Fehler liegt. Tauscht euch eventuell mit euren MitschülerInnen aus.

Tipp: Im Editor könnt ihr jederzeit mit einem 'print' - Befehl das Zwischenergebnis ausgeben lassen und so überprüfen, ob das Programm macht, was ihr möchtet.

Schleifendurchlauf	Ergebnis	Zaehler
0 (zu Beginn)	8	-28
1	-7	-15
2	-9	-2
3	6	5

Geschafft!

Seid ihr schon fertig? Toll!

Überlegt euch weitere Programme, die ihr mit euren bisherigen Kenntnissen umsetzen könntet.
Hier eine Anregung:

Schreibt ein Programm, das

- abfragt, welcher Text ausgegeben werden soll.
- abfragt, wie oft der Text ausgegeben werden soll.
- abfragt, ob der Text wirklich wiederholt werden soll (Ja oder Nein).

Die ersten beiden Schritte habt ihr vorher bereits in einem Programm (Sitzung 2) umgesetzt. Wenn ihr etwas Hilfe braucht, findet ihr hier unter appcamps.link/python11 den Code dazu.

Zu der Ja-/Nein-Abfrage folgende Hinweise:

1. Frage den Benutzer, ob der Text wiederholt werden soll. Lege dazu eine Variable an und nutze den 'input'- Befehl.
2. Ergänze folgendermaßen eine Verzweigung:
   a. Wenn der Nutzer "Ja" eingibt, soll die WHILE-Schleife ausgeführt werden.
   b. Wenn der Nutzer hingegen "Nein" eingibt, wird der Text nur 1x ausgegeben.
   c. Andernfalls bekommt der Nutzer die Rückmeldung, dass er keine gültige Eingabe gemacht hat.

Viel Spaß!

P.S.: Unter appcamps.link/python12 findet ihr eine mögliche Lösung. Denkt daran: Es ist nicht schlimm, wenn eure Lösung anders aussieht. Es gibt immer viele verschiedene Lösungswege.