HSLU Hochschule Luzern

PH LUZERN
PÄDAGOGISCHE
HOCHSCHULE

Die Programmierumgebung

Um das Universum und die Planeten zu erforschen, benötigen wir die richtige Ausrüstung. Hilf mit und bereite die Ausrüstung vor, die wir zum Forschen benötigen.

Ziel:
Du lernst, die Programmierumgebung zu nutzen.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- Ausrüstung vorbereiten

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

Ausrüstung vorbereiten

Bereite die Ausrüstung vor, welche wir zum Forschen benötigen.

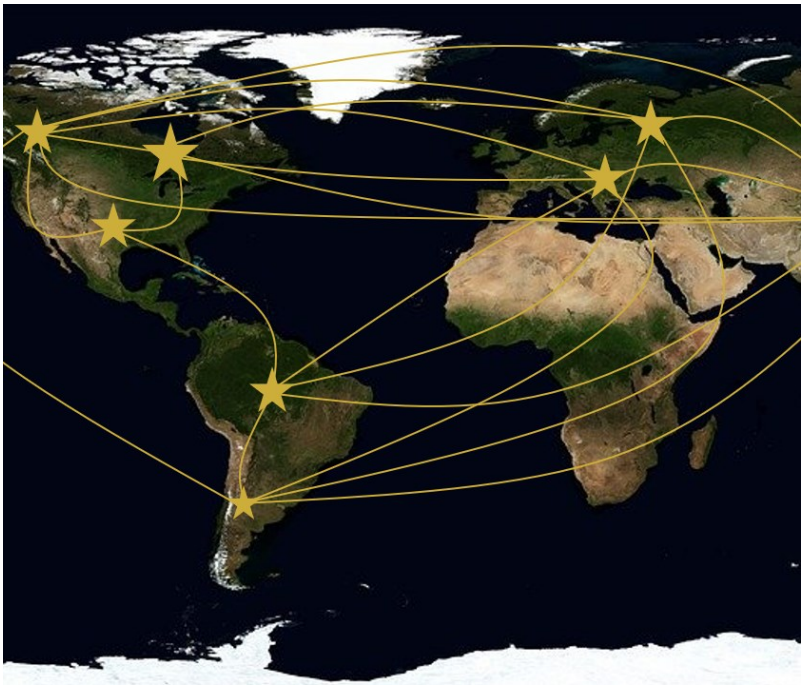
Aufgabe:

Starte den Codeeditor unter
https://pythontutor.com/visualize.html.
Probiere etwas aus, was geschieht, wenn du einige Worte schreibst und dann auf «Vizualize Execution» klickst?

Lösungsvorschlag

Aufgabenkarte 1: Forscherteams

Zu Beginn unserer Forschung, senden wir einige Texte an andere Forschungsteams. Wir wollen mit diesen Teams unsere Forschungsergebnisse austauschen.



Ziel:
Du lernst, wie dein Phyton-Programm einen Text ausgeben kann, dass eine oder mehrere Variabeln enthält.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 1.1 Forscherteams begrüssen
- 1.2 Sich anderen Forscherteams vorstellen
- 1.3 Etwas über sich erzählen
- 1.4 Frage des Forscherteams beantworten

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

1.1 Forscherteams begrüssen

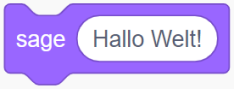
Wir begrüssen andere Forscherteams.

Aufgabe:

Erstelle ein Programm, das einen einfachen Text ausgibt.
Gib den Satz: "Hallo Forscherteams der Welt." aus.

Tipps zum Vorgehen:

Mit der print(..)-Funktion kann man Text und/oder Variablen ausgeben.
Texte müssen immer in Anführungszeichen geschrieben sein, damit diese vom Programm als Texte erkannt werden.

Beschrieb	Phyton	Scratch
Einen Text ausgeben	print ("Hallo Welt")	

1.1 Forscherteams begrüssen

Lösungsvorschlag

Befehl für die Ausgabe eines Textes.

```
1  print("Hallo Forscherteams der Welt.")
```

Text der Ausgegeben wird.
Der Text muss in Anführungszeichen stehen.

1.2 Sich anderen Forscherteams vorstellen

Wir stellen uns den anderen Forschungsteams mit unserem Namen vor. Damit sich auch andere Wissenschaftler*innen einfach vorstellen können, soll der Name im Satz einfach ausgetauscht werden können.

Aufgabe:

Erstelle ein Programm, das einen einfachen Text mit einer Variablen ausgibt. Du kannst dafür das vorhin erstellte Programm erweitern. Wähle als Variable deinen Namen und gib folgenden Satz aus:
"Hallo Forscherteams der Welt sagt name"

Tipps zum Vorgehen:

Variablen sind Wertespeicher. Man kann darin einen Wert ablegen und diesen wieder abfragen. Sie sind wie eine kleine Schublade, die einen Namen hat und einen Wert enthält. Um einer Variablen einen Wert zuzuweisen, muss sie links von einem =-Zeichen stehen.

1.2 Sich anderen Forscherteams vorstellen

Lösungsvorschlag

Name der Variable.
Der Name einer Variablen muss mit einem kleinen Buchstaben beginnen.
Der Name darf keine Sonderzeichen oder ä, ö und ü enthalten.

Text der Ausgegeben wird.

Wert welcher der Variablen zugewiesen wird.
Da der Variable ein Text zugewiesen wird, muss dieser in Anführungszeichen stehen.

```
1  name = "mein Name"
2  print("Hallo Forscherteams der Welt sagt", name)
```

Befehl für die Ausgabe eines Textes.

Mit einem Komma getrennt, können auch ein oder mehrere Werte innerhalb eines «print»-Befehls ausgegeben werden.

Variable die Ausgegeben wird.
Da es sich um eine Variable handelt, darf diese nicht in Anführungszeichen stehen.

1.3 Etwas über sich erzählen

Ein anderes Forschungsteam hat sich gemeldet, es möchte mehr über euch erfahren. Neben dem Namen, interessiert sich das andere Team auch für euer Alter und eure Grösse.

Aufgabe:

Erstelle ein Programm, das einen Text mit mehreren Variablen ausgibt. Du kannst dafür das vorhin erstellte Programm erweitern.
Wähle als Variablen deinen Namen, dein Alter und deine Grösse in Centimetern.
"Hallo Forscherteams der Welt sagt "name" ich bin "alter" Jahre alt und "grösse" cm gross."

Tipps zum Vorgehen:

Variablen sind Wertespeicher. Man kann darin einen Wert ablegen und diesen wieder abfragen. Sie sind wie eine kleine Schublade, die einen Namen hat und einen Wert enthält. Um einer Variablen einen Wert zuzuweisen, muss sie links von einem =-Zeichen stehen.
Print gibt auch mehrere Werte aus, die mit einem Komma getrennt sind.

1.3 Etwas über sich erzählen

Lösungsvorschlag

Wert welcher der Variablen zugewiesen wird.
Da der Variable ein Text zugewiesen wird, muss dieser in Anführungszeichen stehen.

Name der Variable.

Wert welcher der Variablen zugewiesen wird.
Da der Variable eine Zahl zugewiesen wird, muss diese nicht in Anführungszeichen stehen.

```
1 name = "mein Name"
2 alter = 15
3 groesse = 155
4
5 print("Hallo Forscherteams der Welt sagt", name, "ich bin", alter, "Jahre alt und" groesse, "cm gross.")
```

Text der Ausgegeben wird.

Befehl für die Ausgabe eines Textes.

Mit einem Komma getrennt, können auch ein oder mehrere Werte innerhalb eines «print»-Befehls ausgegeben werden.

Variable die Ausgegeben wird.
Da es sich um eine Variable handelt, darf diese nicht in Anführungszeichen stehen.

1.4 Frage des Forscherteams beantworten

Das andere Forschungsteam, interessiert sich mehr für euer Geburtsjahr als für euer alter.

Aufgabe:

Erstelle ein Programm, das einen Text mit mehreren Variablen ausgibt. Du kannst dafür das vorhin erstellte Programm erweitern.
Wähle als Variablen deinen Namen, dein Alter und deine Grösse in Centimetern. Berechne mithilfe der Variablen Alter dein Geburtsjahr und erstelle damit die Variable geburtsjahr. " Hallo Forscherteams der Welt sagt name bin im Jahr X Geboren und grösse cm gross. "

Tipps zum Vorgehen:

Mit Zahlen und Variablen kann normal gerechnet werden.
Um einer Variablen einen Wert zuzuweisen, muss sie links von einem = -Zeichen stehen.
Beispiel:
Einer Variablen mit dem Namen «meineZahl» wird der Wert 42 zugewiesen

```
meineZahl = 42
```

Der Wert von «meineZahl» wird um 5 erhöht

```
meineZahl = meineZahl + 5
```

HSLU Hochschule Luzern

1.4 Frage des Forscherteams beantworten

PH LUZERN PÄDAGOGISCHE HOCHSCHULE

Lösungsvorschlag

Wert welcher der Variablen zugewiesen wird.
Texte müssen in Anführungszeichen stehen.
Zahlen müssen nicht in Anführungszeichen stehen.

Berechnung der Variablen geburtsjahr.

Text der Ausgegeben wird.

```
1 name = "mein Name"
2 alter = 15
3 groesse = 155
4 geburtsjahr = 2022 - alter
5
6 print("Hallo Forscherteams der Welt sagt", name, "bin im Jahr ", geburtsjahr, "geboren und" groesse, "cm gross.")
```

Name der Variable.

Befehl für die Ausgabe eines Textes.

Mit einem Komma getrennt, können auch ein oder mehrere Werte innerhalb eines «print»-Befehls ausgegeben werden.

Variable die Ausgegeben wird. Da es sich um eine Variable handelt, darf diese nicht in Anführungszeichen stehen.

Aufgabenkarte 2: Verkehrssystem entwickeln

Wir überlegen uns, wie ein Luftschiff-Hafen ausschauen könnte, falls wir zur Erforschung den Weltraum bereisen müssen.

Ziel:
Du lernst, wie sich dein Programm anders Verhalten kann, je nachdem welche Bedingungen erfüllt sind.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 2.1 Einfache Entscheidungen treffen
- 2.2 Komplexe Entscheidungen treffen
- 2.3 Schiffe willkommen heissen
- 2.4 Schiffe ablehnen
- 2.5 Schiffe an richtigen Ort zuweisen

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

2.1 Einfache Entscheidungen treffen

Beim Forschen muss oft entschieden werden, wie es weitergeht. Welche Entscheidungen kannst du treffen?

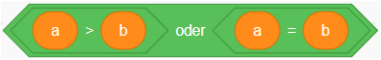
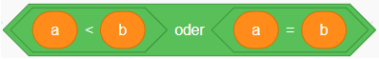
Aufgabe:

Schreibe folgende Fälle als Bedingungen auf:

1. i ist grösser als 10
2. i ist kleiner als die Variable "zahl"
3. i ist grösser gleich 100
4. i ist nicht gleich 50
5. i ist kleiner gleich 0
6. i ist gleich der Variablen "zahl"

Tipps zum Vorgehen:

Eine Bedingung ist entweder Wahr (True) oder Falsch (False).

Beschrieb	Phyton	Scratch
a ist gleich b	a == b	
a ist nicht gleich b	a != b	
a ist grösser gleich b	a >= b	
a ist kleiner gleich b	a <= b	
a ist grösser als b	a > b	
a ist kleiner als b	a < b	

Mit **not** kann eine Bedingung ins Gegenteil gedreht werden. True wird damit zu False und False wird damit zu True.

2.1 Einfache Entscheidungen treffen

Lösungsvorschlag

```
1  i > 10
2  i < zahl
3  i >= 100
4  i != 50
5  i <= 0
6  i == zahl
```

2.2 Komplexe Entscheidungen treffen

Entscheidungen können nicht nur auf einem Faktor beruhen. Manchmal sind es mehrere Faktoren, die miteinander kombiniert zu einer Entscheidung führen.

Aufgabe:

Schreibe folgende Fälle als Bedingungen auf:

1. i ist grösser als 16 und kleiner als 18
2. i ist nicht gleich 100 oder grösser gleich 20
3. i ist nicht kleiner als 25 und grösser als 17
4. i ist gleich der Variablen "anzahl" und nicht grösser als 30
5. i ist grösser gleich der Variablen "anzahl" oder kleiner gleich der Variablen "lohn"
6. i ist nicht gleich der Variablen "zahl" oder gleich der Variablen "anzahl" und grösser als 30

Tipps zum Vorgehen:

Mit **and** können mehrere Bedingungen kombiniert werden. Es müssen alle Bedingungen erfüllt werden.

Mit **or** können mehrere Bedingungen kombiniert werden. Es muss eine der Bedingungen erfüllt werden.

2.2 Komplexe Entscheidungen treffen

Lösungsvorschlag

```
1  i > 16 and i < 18
2  i != 100 or i >= 20
3  i not < 25 and i > 17
4  i == anzahl and i not > 30
5  i >= anzahl or i <= lohn
6  i != zahl or i == anzahl and i > 30
```

2.3 Schiffe willkommen heissen

Du beobachtest in einer Simulation das treiben an einem Luftschiff-Hafen. Im Hafen dürfen nur Passagierschiffe anlegen, die mehr als 20 Passagiere an Board haben. Erfüllt ein Schiff diese Bedingung, soll angezeigt werden, dass es im Hafen willkommen ist.

Aufgabe:

Schreibe ein Programm das entscheidet, ob ein Schiff in den Hafen einlaufen darf oder nicht. Erstelle dafür eine Variable "passagiere". Sind auf dem Schiff mehr als 20 Passagiere, so darf das Schiff in den Hafen einlaufen.
Wenn das Schiff in den Hafen einlaufen darf, gib die Meldung "Bitte in den Hafen einfahren" aus. Probiere das Programm aus, was passiert, wenn das Schiff weniger als 20 Passagiere oder genau 20 Passagiere enthält?

Tipps zum Vorgehen:

Beschrieb	Phyton

if-Abfragen
Eine if-Abfrage überprüft, ob eine Bedingung Wahr oder Falsch ist. Alles was nach der if-Abfrage mit einem Tab eingerückt ist gehört zum Körper der if-Abfrage und wird nur ausgeführt, wenn die Bedingung Wahr ist
	wert = 10 # Eine Variable mit einem beliebigen Wert (hier 10)
	
	if wert > 20: # wird NUR ausgeführt, wenn Bedingung wahr ist
	  print("Der Wert ist größer als 20") # (ACHTUNG: Einrückung)
	
	# Hier gehts weiter (ACHUTUNG: Nicht eingerückt)

2.3 Schiffe willkommen heissen

Lösungsvorschlag

```
1  passagiere = 21
2
3  if passagiere > 20:
4      print ("Bitte in den Hafen einfahren")
5
```

Bedingung.

if-Abfrage

Der Doppelpunkt kennzeichnet das Ende der Bedingung.

Teil der Ausgeführt wird, wenn die Bedingung stimmt. Muss eingerückt sein.

Bei weniger als 20 Passagieren oder genau 20 Passagieren geschieht nicht.

2.4 Schiffe ablehnen

Damit Schiffe mit weniger als 20 Passagieren nicht in den Hafen einlaufen, soll ihnen angezeigt werden, dass der Hafen nicht angesteuert werden kann.

Aufgabe:

Erweitere das vorhergehende Programm so, dass bei genau 20 Passagieren oder weniger als 20 Passagieren die Meldung "Der Hafen kann nicht angesteuert werden" ausgegeben wird.

Tipps zum Vorgehen:

Beschrieb	Phyton

if-else-Abfragen
Mit einer if-else Abfrage kann man auch auf eine nicht erfüllte Bedingung mit dem 'else'-Zweig reagieren. Es wird immer entweder der if-Zweig oder der else-Zweig ausgeführt.
	wert = 10 # Eine Variable mit einem beliebigen Wert (hier 10)
	
	if wert > 20: # wird NUR ausgeführt wenn Bedingung erfüllt ist
	  print("Der Wert ist größer als 20")
	else: # wird NUR ausgeführt wenn Bedingung nicht erfüllt ist
	  print("Wert ist kleiner oder gleich 20.")

2.4 Schiffe ablehnen

Lösungsvorschlag

```
1  passagiere = 20
2
3  if passagiere > 20:
4      print ("Bitte in den Hafen einfahren")
5  else:
6      print ("Der Hafen kann nicht angesteuert werden")
```

Bedingung.

Der Doppelpunkt kennzeichnet das Ende der Bedingung.

if-Abfrage

Teil der Ausgeführt wird, wenn die Bedingung stimmt. Muss eingerückt sein.

Teil der Ausgeführt wird, wenn die Bedingung nicht zutrifft. Muss eingerückt sein.

2.5 Schiffe an richtigen Ort zuweisen

In der Simulation hast du den Luftschiffhafen erweitert. Neu können auch Schiffe in den Hafen einlaufen, die zwischen 10 und 20 Passagiere an Board haben. Jedoch müssen diese Schiffe an ein spezielles Terminal. Dies soll den Schiffen angezeigt werden.

Aufgabe:

Erweitere das vorhergehende Programm so, dass

- Bei über 20 Passagieren die Meldung "Bitte in den Hafen einfahren" ausgegeben wird
- Zwischen 10 und 20 Passagieren die Meldung "Einfahrt eingeschränkt, nur an Terminal 2 möglich" ausgegeben wird
- Bei weniger als 10 Passagieren die Meldung "Der Hafen kann nicht angesteuert werden" ausgegeben wird

Tipps zum Vorgehen:

Es können mehrere Schlaufen miteinander kombiniert und ineinander Verschachtelt werden, um ein gewünschtes Resultat zu erzielen.
Im else-Teil, also dem Teil der Ausgeführt wird wenn die Bedingung nicht erfüllt war, kann eine weitere Bedingung gestellt werden.

2.5 Schiffe an richtigen Ort zuweisen

Lösungsvorschlag

```
1  passagiere = 22
2
3  if passagiere > 20:
4      print ("Bitte in den Hafen einfahren")
5  else:
6      if passagiere >= 10:
7          print ("Einfahrt eingeschränkt, nur an Terminal 2 möglich")
8      else:
9          print ("Der Hafen kann nicht angesteuert werden")
```

if-Abfrage

Die zweite if-Abfrage wird nur ausgeführt, wenn die Bedingung der vorhergehenden Abfrage nicht erfüllt war. Die zweite if-Abfrage ist deshalb unter der ersten if-Abfrage eingerückt.

Aufgabenkarte 3: Meteoriten

Du hast ein paar Meteoriten am Himmel entdeckt.

Ziel:
Du lernst, wie du Teile eines Programmes so lange ausführen kannst, bis eine Bedingung nicht mehr erfüllt ist.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 3.1 Nummerieren
- 3.2 Nummerieren (2)
- 3.3 Fracht für die Luftschiffe

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

3.1 Nummerieren

Die neu entdeckten Meteoriten musst du nummerieren, damit du diese zweifelsfrei wieder identifizieren kannst.

Aufgabe:

Schreibe ein Programm, das so lange eine Zahl hochzählt, bis die Zahl 11 beträgt.
Erstelle dafür eine Variable "zahl". Der Variablen "zahl" weist du den Startwert 1 zu. In jeder Runde soll geprüft werden, ob die Zahl den Wert 11 bereits erreicht hat. Hat sie dies nicht, so soll 1 zu der bestehenden Zahl dazugezählt werden.

Tipps zum Vorgehen:

Beschrieb	Phyton	Scratch

while-Schleife
Eine while-Schleife führt den eingerückten Code aus, solange die Bedingung erfüllt ist.
	zahl = 1
	while zahl < 10: # solange die Bedingung wahr ist wird dieser eingerückte Code wird wiederholt
	  print("Zahl:", zahl)
	  zahl = zahl + 1 # Zähler erhöhen
	# Hier gehts normal weiter (Achtung: nicht eingerückt)
		

Lösungsvorschlag

Variable «zahl» mit Startwert 1

Der Doppelpunkt kennzeichnet das Ende der Bedingung.

Solange die Variable «zahl» kleiner ist als 12, wird die Zahl ausgegeben und 1 dazugerechnet.

```
1  zahl = 1
2
3  while zahl <12:
4      print (zahl)
5      zahl = zahl + 1
```

3.2 Zählen lernen

In einem unbemerkten Moment hat sich eine Forscherin einen Scherz erlaubt und deinen Code erweitert.

Aufgabe:

Betrachte nachfolgenden Code, was geschieht, resp. Was wird ausgegeben?

```
1  zahl = 1
2
3  while zahl <12:
4      print (zahl)
5      zahl = zahl + 1
6  print (zahl)
```

Tipps zum Vorgehen:

Achte dich genau auf die Einrückungen.

Lösungsvorschlag

Es werden die Zahlen 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 ausgegeben

```
1  zahl = 1
2
3  while zahl <12:
4      print (zahl)
5      zahl = zahl + 1
6  print (zahl)
```

Die while-Schlaufe gibt die Zahlen 1 bis 11 aus.
Jeweils nachdem die Zahl ausgegeben wurde, wird diese um 1 erhöht.

Der print-Befehl hier wird ausgeführt, wenn die Bedingung in der Schleife nicht mehr erfüllt ist. Da die Schleife im letzten Schritt (nach der Ausgabe der Zahl 11) die Zahl noch um 1 erhöht, wird hier 12 ausgegeben.

3.3 Fracht für die Luftschiffe

Neben dem Transport von Personen, ist der Transport von Fracht wichtig, um das Weltall erforschen zu können. Doch die Container, welche auf die Luftschiffe kommen, dürfen nicht zu schwer werden.

Aufgabe:

Ein Container soll so lange weiter beladen werden, bis die Fracht ein Gesamtgewicht von 500kg erreicht. Der Container wird von zwei Kränen gleichzeitig beladen. Kran A belädt den Container jeweils mit 15kg Fracht, Kran B belädt den Container jeweils mit 25kg Fracht.
Schreibe ein Programm das anzeigt, ob der Container weiter beladen werden kann oder nicht. Erstelle dazu die Variablen "kranA", "kranB" und "fracht" und arbeite mit diesen. Solange der Container weiter beladen werden kann, soll die Meldung "Container weiter beladen, Gewicht der Fracht beträgt "fracht"." ausgegeben werden. Sobald der Container nicht mehr Beladen werden soll, soll die Meldung "Stop, Container ist voll, das Gewicht beträgt "fracht"." ausgegeben werden.

Tipps zum Vorgehen:

Schaue dir die Aufgabenkarten 1: Forscherteams und vorhergehende Aufgaben in diesen Aufgabenkarten nochmals an.

HSLU Hochschule Luzern

3.3 Fracht für die Luftschiffe

Lösungsvorschlag

```
1  kranA = 15
2  kranB = 25
3  fracht = 0
4
5  while fracht < 500:
6      print ("Container weiter beladen, Gewicht der Fracht beträgt", fracht)
7      fracht = fracht + kranA + kranB
8  print ("Stop, Container ist voll, das Gewicht beträgt", fracht)
```

Solange das Gewicht der Fracht unter 500 ist, wird der Container beladen.

Überschreitet das Gewicht bei der Prüfung 500, wird die while-Schlaufe nicht mehr ausgeführt. Das Programm führt den weiteren Code ausserhalb der Schlaufe aus.

Es wird erst geprüft, ob die Fracht unter 500 schwer ist, danach wird die hinzukommende Fracht dazugerechnet.

PH LUZERN PÄDAGOGISCHE HOCHSCHULE

Aufgabenkarte 4: Das Forschungszentrum

Immer mehr Menschen werden auf eure Forschung aufmerksam. Ihr entschliesst euch, euer Forschungszentrum für Besuchende zu öffnen.

Ziel:
Du übst, einfache Phyton-Programme zu erstellen und zu interpretieren.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 4.1 Billett kaufen
- 4.2 Barriere schliessen

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

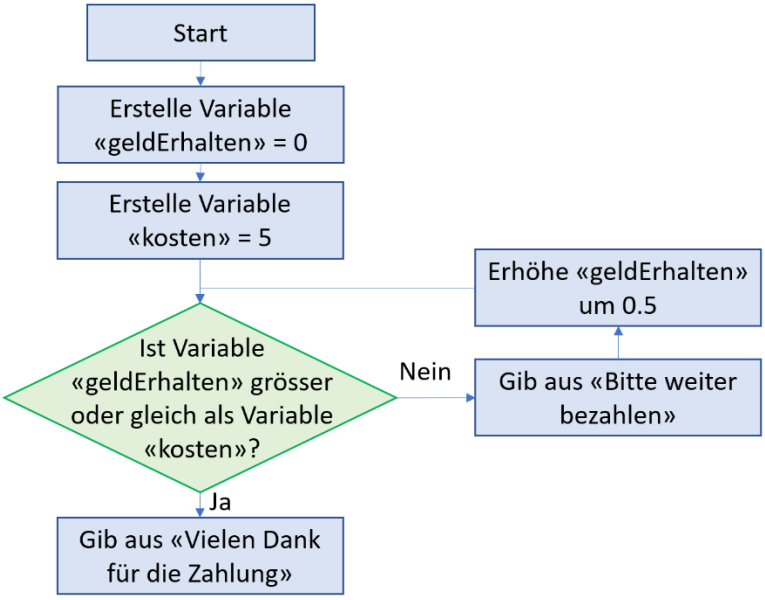
Platz für Notizen

4.1 Billett kaufen

Immer mehr Menschen interessieren sich dafür, das Weltall kennen zu lernen. Um eure Forschung zu finanzieren, bietet ihr Führungen an. Um an einer Führung teilnehmen zu können, wird ein gültiges Billett benötigt.

Aufgabe:

Übersetze das nachfolgende Ablaufdiagramm in ein Phyton-Programm. Wie oft muss das Programm die Meldung "Bitte weiter bezahlen" ausgeben?



Tipps zum Vorgehen:

Überlege dir Schritt für Schritt, was im Ablaufdiagramm geschieht und übersetze dies in Phyton.

4.1 Billett kaufen

Lösungsvorschlag

```
1  geldErhalten = 0
2  kosten = 5
3
4  while geldErhalten < kosten:
5      print ("Bitte weiter bezahlen")
6      geldErhalten = geldErhalten + 0.5
7  print ("Vielen Dank für die Zahlung")
```

Die Bedingung «grösser oder gleich» kann im Umkehrschluss auch als «kleiner» angegeben werden.

Die while-Schlaufe wird gewählt, weil etwas so lange gemacht werden muss, bis die Bedingung nicht mehr gegeben ist.
Erst wenn der erhaltene Betrag dem Preis entspricht oder diesen übersteigt, soll die Dankesmeldung ausgegeben werden.

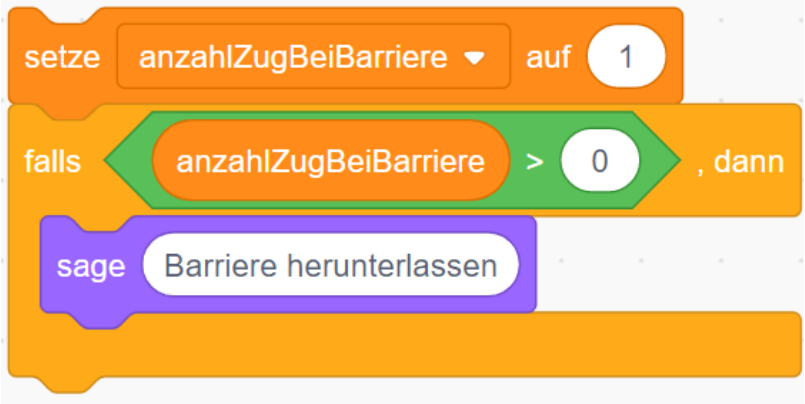
Die Meldung «Bitte weiter bezahlen» muss 11 Mal ausgegeben werden.

4.2 Barriere schliessen

Euer Forschungszentrum liegt in unmittelbarer Nähe zum Bahnhof. Dies macht ihr euch zu Nutze und erklärt euren Gästen wie eine Barriere funktioniert. Denn eine solche soll auch vor dem Luftschiffhafen installiert werden.

Aufgabe:

Übersetze das nachfolgende Scratch-Programm in ein Phyton-Programm.



Tipps zum Vorgehen:

Überlege dir Schritt für Schritt, was im Scratch-Programm geschieht. Übersetze das Scratch-Programm in Phyton.

4.2 Barriere schliessen

Lösungsvorschlag

Variable anzahlZugBeiBarriere erstellen und ihr den Wert 1 zuweisen

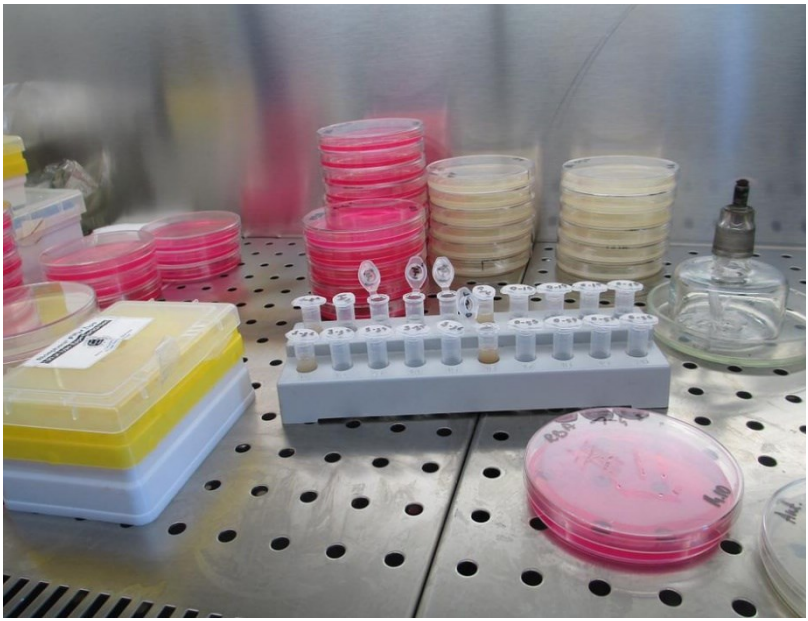
```
1  anzahlZugBeiBarriere = 1
2
3  if anzahlZugBeiBarriere > 0:
4      print ("Barriere herunterlassen")
```

Wenn mehr als 0 Züge bei der Barriere sind

Meldung die ausgegeben wird, wenn die Bedingung erfüllt ist

Aufgabenkarte 5: Gesammelte Proben

Im laufe der Forschung haben sich viele Probenbehälter angesammelt. Du überlegst dir, wir ihr es schafft den überblick über all die Proben nicht zu verlieren.



Ziel:
Du lernst, wieder verwendbare Programmteile zu erstellen und diese im Programm zu nutzen.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 5.1 Probenbehälter zählen
- 5.2 Probenbehälter zählen (2)
- 5.3 Die höchste Probe

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

5.1 Probenbehälter zählen

Immer wieder müssen die Probenbehälter gezählt werden, um sicherzustellen dass keine Probe verloren gegangen ist.

Aufgabe:

Schreibe ein Programm, dass du so oft wie du es benötigst aufrufen kannst (eine Funktion). Das Programm soll mit der Ausgabe der Zahl 1 beginnen und so lange hochzählen, bis 10 erreicht ist. Erstelle dafür eine Variable "zahl" und arbeite mit einer while-Schlaufe.
Rufe die Funktion anschliessend auf, um zu überprüfen, ob das Programm so funktioniert wie es sollte.

Tipps zum Vorgehen:

Beschrieb	Phyton

Eine Funktion definieren
Eine Funktion ist ein Block Code, der vom Haupt-Programm getrennt definiert wird. Die Funktion wird erst dann ausgeführt, wenn man sie über ihren Namen aufruft. Dafür kann man eine Funktion so oft mal will verwenden. Variablen, die innerhalb einer Funktion definiert sind, sind nur innerhalb der Funktion verwendbar!
	# Neue Funktion mit dem Namen 'sageHallo' definieren
	def sageHallo():
	  print("Hallo Welt") # Einrückung!
	
	# Funktion aufrufen (keine Einrückung)
	sageHallo()

5.1 Probenbehälter zählen

Lösungsvorschlag

Funktion zaehlen wird definiert.

```
1  def zaehlen():
2      zahl = 1
3      while zahl <= 10:
4          print (zahl)
5          zahl = zahl + 1
6
7  zaehlen()
```

Mit der while-Schleife werden nacheinander die Zahlen 1 bis 10 ausgegeben (vgl. Aufgabenkarte 3: E-Lien lernt zählen).

Hier wird die oben definierte Funktion aufgerufen.
Ohne das Aufrufen der Funktion geschieht nichts, denn oben wird sie nur definiert.
Damit die Funktion weiss, wann sie gebraucht wird, muss sie explizit Aufgerufen werden.

5.2 Probenbehälter zählen (2)

Da immer wieder neue Proben dazukommen und ältere aufgebraucht sind, musst du ein Programm zum Zählen der Proben entwickeln, bei dem die Start- und die Endzahl definiert werden kann.

Aufgabe:

Erweitere das vorhergehende Programm so, dass beim Aufrufen der Funktion zwei Zahlen eingegeben werden können. Die erste Zahl wird zur Variablen "zahl", dort wird mit dem Zählen begonnen. Die andere Zahl wird zur Variablen "schluss", dort wird mit dem Zählen aufgehört. Rufe die Funktion auf und gib der Funktion als Parameter die Zahlen 5 und 15 mit.

Tipps zum Vorgehen:

Beschrieb	Phyton

Eine Funktion mit Übergabeparametern definieren
Funktion können Parameter (Variablen, die nur innerhalb der Funktion existieren) haben. Diese müssen beim Aufrufen mitübergeben werden.
	`# Funktion definieren`
	`def sageHallo(name, alter):`
	`  print("Hallo", name)`
	`  print("Du bist", alter, "Jahre alt")`
	
	`# Funktion aufrufen`
	`sageHallo("Mark", 22)`

5.2 Probenbehälter zählen (2)

Lösungsvorschlag

Funktion zaehlen wird definiert.

Die beiden Parameter (zahl, schluss werden angelegt).

```
1  def zaehlen(zahl, schluss):
2
3      while zahl <= schluss:
4          print (zahl)
5          zahl = zahl + 1
6
7  zaehlen(5,15)
```

Mit der while-Schleife werden nacheinander die Zahlen beginnend bei «zahl» bis «schluss» ausgegeben (vgl. Aufgabenkarte 3: E-Lien lernt zählen).

Hier wird die oben definierte Funktion aufgerufen.

Der Funktion werden die Parameter 5 und 15 mitgegeben.

5.3 Die höchste Probe

Nachdem alle Proben gezählt sind, müsst ihr jeweils die Nummer der höchsten Probe wissen. Damit könnt ihr abgleichen, ob noch alle Proben vorhanden sind.

Aufgabe:

Erweitere das vorhergehende Programm so, dass es den Wert der Variable "schluss" zurückgibt.
Fange die Variable "schluss" beim Aufrufen der Funktion mit einer Variablen "hoheZahl" ab. Nutze die Variable "hoheZahl" um folgenden Satz auszugeben: Die höchste Probe ist "hohe Zahl".

Tipps zum Vorgehen:

Innerhalb der definierten Funktion kann mittels dem Befehl return ein Wert definiert werden, der Ausserhalb der Funktion genutzt werden kann.
Um den Wert ausserhalb der Funktion nutzen zu können, muss die Funktion wie folgt aufgerufen werden:
variablenname = Name der Funktion (Parameter, Parameter, …)

5.3 Die höchste Probe

Lösungsvorschlag

Funktion zaehlenlernen wird definiert.

Die beiden Parameter (zahl, schluss werden angelegt).

```
1  def zaehlen(zahl, schluss):
2      while zahl <= schluss:
3          print (zahl)
4          zahl = zahl + 1
5      return schluss
6
7  hoheZahl = zaehlen(5,15)
8  print("Die höchste Probe ist", hoheZahl,".")
```

Einrückung beachten: return muss wieder auf höhe von while sein. Sonst ist das return ein Teil der while-Schlaufe und kann nicht weiterverwendet werden.

Die Funktion zaehlenlernen wird so aufgerufen, dass der Rückgabewert in der Variablen hoheZahl aufgefangen wird.

return muss am Ende der Definition einer Funktion stehen. Denn bei return wird die Ausführung abgebrochen und der aktuelle Wert zurückgegeben.

Aufgabenkarte 6: Neue Freundschaften

Seit du mit dem Forschen begonnen hast, hast du viele neue Freundschaften geschlossen. Damit du den überblick über all die neuen Freundschaften nicht verlierst, schreibst du dir diese auf.

Ziel:
Du lernst, Listen zu nutzen. Dazu gehört das Erstellen von Listen, das hinzufügen und löschen von Einträgen und das Auslesen der Liste.

Anleitung:

Löse der Reihe nach alle Teilaufgaben.

- 6.1 Freundesliste erstellen
- 6.2 Neue Freunde hinzufügen
- 6.3 Falscher Eintrag löschen
- 6.4 Die Liste kontrollieren

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

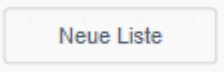
6.1 Freundesliste erstellen

Halte alle neuen Freundschaften, die du während der Forschung geschlossen hast, in einer Liste fest.

Aufgabe:

Erstelle eine Liste "meineFreundschaften" in denen du folgende Personen hinzufügst: Jane Muster, Max Doe, Fred Nurk, Juan Fulano

Tipps zum Vorgehen:

Beschrieb	Phyton	Scratch
Eine leere Liste erstellen	liste = []	
Eine Liste mit Einträgen erstellen	liste = [1, 2, 3, 4, 5]	

6.1 Freundesliste erstellen

Lösungsvorschlag

Name der Liste.

Die Namen müssen in Anführungszeichen stehen, da es sich um Text handelt.

```
1 meineFreundschaften["Jane Muster", "Max Doe", "Fred Nurk", "Juan Fulano"]
```

Die eckigen Klammern zeigen an, dass es eine Liste ist die erstellt wird.

6.2 Neue Freunde hinzufügen

Du hast noch jemand neuen kennen gelernt. Füge auch diese Person der Liste hinzu.

Aufgabe:

Erweitere das vorhergehende Programm.
Füge Newton und Lisa Bloggs, mit dem dafür vorgesehenen Befehl, zur Liste hinzu.

Tipps zum Vorgehen:

Beschrieb	Phyton	Scratch
Einen Eintrag hinzufügen	liste.append(3) # einen Eintrag anfügen	füge 3 zu Liste hinzu

6.2 Neue Freunde hinzufügen

Lösungsvorschlag

Name der Liste, zu der etwas hinzugefügt werden soll.

Die Namen müssen in Anführungszeichen stehen, da es sich um Text handelt.

```
1  meineFreundschaften["Jane Muster", "Max Doe", "Fred Nurk", "Juan Fulano"]
2
3  meineFreundschaften.append("Newton")
4  meineFreundschaften.append("Lisa Bloggs")
```

Die Einträge müssen einzeln zur Liste hinzugefügt werden.

Mit dem Befehl append kann ein Element hinzugefügt werden.

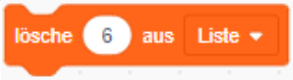
6.3 Falscher Eintrag löschen

Du hast ausversehen jemand der Liste hinzugefügt, den du gar nicht kennen gelernt hast. Lösche diesen Eintrag.

Aufgabe:

Erweitere das vorhergehende Programm.
Lösche den Eintrag Newton, mit dem dafür vorgesehenen Befehl, aus der Liste.

Tipps zum Vorgehen:

Beschrieb	Phyton	Scratch
Einen Eintrag entfernen	liste.remove(6) # löscht den Wert 6 aus der Liste	

HSLU Hochschule Luzern

6.3 Falscher Eintrag löschen

Lösungsvorschlag

```
1  meineFreundschaften["Jane Muster", "Max Doe", "Fred Nurk", "Juan Fulano"]
2
3  meineFreundschaften.append("Newton")
4  meineFreundschaften.append("Lisa Bloggs")
5
6  meineFreundschaften.remove("Newton")
```

Der Name muss in Anführungszeichen stehen, da es sich um Text handelt.

Name der Liste, aus der etwas gelöscht werden soll.

Mit dem Befehl remove kann ein einzelnes Element gelöscht werden.

PH LUZERN PÄDAGOGISCHE HOCHSCHULE

6.4 Die Liste kontrollieren

Nach all dem löschen und hinzufügen, bist du dir nicht mehr sicher ob deine Liste noch korrekt ist.

Aufgabe:

Erweitere das vorhergehende Programm, so dass die fertige Liste ausgegeben wird.

Tipps zum Vorgehen:

Die einzelnen Elemente einer Liste können mittels einer for-Schleife durchlaufen werden.

```
for element in liste:
print(element)
```

6.4 Die Liste kontrollieren

Lösungsvorschlag

```
1  meineFreundschaften["Jane Muster", "Max Doe", "Fred Nurk", "Juan Fulano"]
2
3  meineFreundschaften.append("Newton")
4  meineFreundschaften.append("Lisa Bloggs")
5
6  meineFreundschaften.remove("Newton")
7
8  for element in meineFreundschaften:
9      print(element)
```

for-Schleife

Name der Liste, die ausgegeben werden soll.

Aufgabenkarte 7: Wartezeiten

Immer wieder müsst ihr warten, bis ihr mit eurer Forschungsarbeit weitermachen könnt. Damit euch nicht langweilig wird, beginnt ihr mit Würfelspielen.

Ziel:
Du lernst einige nützliche Module kennen und anwenden.

Anleitung:
Löse der Reihe nach alle Teilaufgaben.

- 7.1 Würfeln

Lies die Teilaufgabe jeweils sorgfältig durch. Tipps zur Lösung der Aufgabe findest du direkt auf der jeweiligen Aufgabenkarte unter «*Tipps zum Vorgehen*».

Wenn du beim Lösen der Aufgabe nicht weiterkommst, kannst du die Aufgabenkarte umdrehen. Dort findest du einen Lösungsvorschlag mit Erklärung.

Platz für Notizen

7.1 Würfeln

Ihr spielt ein einfaches Würfelspiel: Jeder kann einmal werfen und derjenige mit der höchsten Zahl gewinnt.

Aufgabe:

Schreibe ein Programm, dass eine Zufallszahl zwischen 1 und 6 in der Variable "wuerfel" speichert.
Wenn der Wurf kleiner als 4 ist, gib die Meldung "Mit der Zahl "wuerfel" wird es schwierig zu gewinnen" aus. Ist der Wurf 4 oder 5, gib die Meldung "Mit der Zahl "wuerfel" hast du chancen zu gewinnen" aus. Ist der Wurf grösser als 6, gib die Meldung "Mit der Zahl "wuerfel" hast du so gut wie gewonnen" aus.

Tipps zum Vorgehen:

Beschrieb	Phyton

Zufallszahl
Will man eine Zufallszahl erzeugen, kann man dies mit dem Modul random tun.
Das Modul random muss einmal importiert werden, um verwendet werden zu können
	import random # Modul random importieren
	
	# generiert eine Zufallszahl zwischen 1 und 100
	zufallsZahl = random.randint(1,100)

Lösungsvorschlag

Das Modul random wird importiert, so dass es für das Programm verwendet werden kann.

Der Variable wird eine Zufallszahl zwischen 1 und 6 zugewiesen.

Variable wuerfel.

```
1  import random
2  wuerfel = random.randint(1,6)
3
4  if wuerfel < 3:
5      print("Mit der Zahl", wuerfel, "wird es schwierig zu gewinnen")
6  else:
7      if wuerfel <6:
8          print("Mit der Zahl", wuerfel, "hast du chancen zu gewinnen")
9      else:
10         print("Mit der Zahl", wuerfel, "hast du so gut wiegewonnen")
```

Verschachtelte if-Abfrage