Eingangstext:

> „Herzlich willkommen zu unserem Python-Lerntutorial!
> Immer wenn du ein Abenteuer bestanden hast wirst du mit einer Python Lektion belohnt und kannst so spielerisch Programmieren lernen.
> Dabei wirst du Schritt für Schritt in eine der meist verwendeten Skriptsprachen, Python, eingeführt

Lektion 1:

| **Aufgabe 1.0 (print)** | |
|---|---|
| Text | Willkommen zu deiner ersten Lektion.<br>Jedes Programm muss mit der Welt kommunizieren können, dazu gibt es in Python die Funktion „print“(ausdrucken/ ausgeben).“<br><br>Das einfachste Beispiel hierfür ist sich „Hello World“ ausgeben zu lassen.<br>Hierfür muss der Befehl “print” und die gewünschte Ausgabe in Anführungszeichen "Hello World" eingegeben werden. |
| Aufgabe | Gib folgenden Satz aus: Python ist KEINE Schlange |
| Vorgegebener Code | print "Hello World" |
| Lösungs Code | print "Python ist KEINE Schlange" |
| Lösung Variablenbelegung | {entfällt} |

| **Aufgabe 1.1(Int, Float, )** | |
|---|---|
| Text | Um Werte und Ergebnisse in einem Programm wieder verwenden zu können, gibt es in Programmiersprachen Variablen.<br>Diese haben einen Namen und können mit verschiedenen Arten von Wert belegt werden.<br>Für den Namen können Groß- und Kleinbuchstaben, Ziffern und das Zeichen _ (Unterstrich) verwendet werden.<br>Das erste Zeichen darf aber keine Ziffer sein und es dürfen keine reservierten Wörter der Programmiersprache verwendet werden.<br><br>Wie in der Mathematik kann man mit diesen Variablen die Grundrechenarten<br>Addition ( + ), Subtraktion ( - ), Multiplikation ( * ) und Division ( / ) verwenden.<br>Dabei kann sowohl mit ganzen Zahlen (Integer) als auch mit Fließkomma Zahlen (Floats) gerechnet werden. Bei Floats wird wie im Englischen üblich der . (Punkt) statt dem Komma verwendet. |
| Aufgabe | Weise der Variablen x die Differenz aus 14.56 - 7 zu und gib mit print das Ergebnis aus. Berechne ebenfalls für y das Produkt von 35 * 49 und für z den Quotienten von 50.4 / 3.6 und gib jeweils das Ergebnis aus. |
| Vorgegebener Code | x = 3 + 1732<br>print x |
| Lösungs Code | x = 14.56 – 7<br>print x<br><br>y = 35 * 49<br>print y<br><br>z = 50.4 / 3.6<br>print z |
| Lösung Variablenbelegung | x = 7.56<br>y = 1715<br>z = 14 |

| **Bonus zu 1.1** | |
|---|---|
| Text | Zusätzlich zu dem normalen Divisionsoperator ( / ) gibt es die ganzzahlige Division. Der Operator dazu ist der //. Mit diesem wird das ganzzahlige Ergebnis der Division berechnet. Zusätzlich gibt es den Modulo Operator ( % ) mit dem der Rest berechnet wird. |
| Beispiel | 7//5 ergibt 1<br>7%5 ergibt 2<br><br>Speziell in Python 2 liefert auch schon 7/5 das ganzzahlige Ergebnis 1, dies kann aber mit 7.0/5 oder 7/5.0 umgangen werden. |

| **Aufgabe 1.2 (Arithmetik)** | |
|---|---|
| Text | Bevor wir nun richtig loslegen folgen noch ein paar einfache Rechenaufgaben |
| Aufgabe | Monty möchte demnächst seinen Freund Brian besuchen fahren und wissen, wie viel ihn das kostet. Brian wohnt 42km entfernt. Monty weiß, dass er für jeden Kilometer 25 cent mit der Bahn zahlen muss. Zusätzlich benötigt er noch ungefähr 15 Euro um sich für die Reise ein Buch und eine Cola zu kaufen. Lege eine Variable mit dem Namen 'fahrpreis' an und berechne wieviel Monty's Hin- und Rückreise kostet. Wir nehmen an, dass Monty das Buch und die Cola nur einmal kauft. Dann lasse den Wert der Variable auf dem Bildschirm ausgeben.<br><br>Monty zahlt grundsätzlich nur mit 5 Euro scheinen und er will auch kein Wechselgeld, somit kann er die Reise nur antreten, wenn sich der Fahrpreis mit 5 Euro scheinen bezahlen lässt -- notfalls kauft er so viele Fahrten bis er bei einem Betrag ist, der sich mit 5Euro Scheinen zahlen lässt. Finde mittels einer Modulo Rechnung heraus, wie oft Brian mit dem Besuch von Monty rechnen kann. |
| Vorgegebener Code | {entfällt nach den vorherigen Aufgaben} |
| Lösungs Code | fahrpreis = (42 * 0.25) * 2 + 15<br>print(fahrpreis)<br>anzahl_besuche = fahrpreis % 5 |
| Lösung Variablenbelegung | fahrpreis = 36<br>anzahl_besuche = 5 |

Glückwunsch du hast die erste Lektion bestanden.

Lektion 2:

| **Aufgabe 2.1 (Kommentare)** | |
|---|---|
| Text | Mit den Rechenoperationen hast du nun eigentlich schon das wichtigste gelernt. An dieser Stelle wollen wir uns einem anderen, auch sehr wichtigem Thema annehmen -- Kommentaren. Manchmal kann es sehr nützlich sein, im Code Erklärungen hinzuzufügen, damit andere Leute verstehen, was der Code tut oder was sich der Programmierer dabei gedacht hat. Dieser Text soll natürlich bei der Ausführung des Programmes vom Computer ignoriert werden. Kommentare werden in Python mit dem Symbol # eingeleitet. Sofern dieses Symbol auftaucht, wird der restliche Text in dieser Zeile vom Computer ignoriert. Als Beispiel hierfür kannst du dir nochmal das obige Code Beispiel anschauen. |
| Aufgabe | |
| Vorgegebener Code | |
| Lösungs Code | |
| Lösung Variablenbelegung | |

Lektion 3:

| **Aufgabe 3.1 (Listen)** | |
|---|---|
| Text | Mit Kommentaren selbst kannst du natürlich nichts programmieren, deswegen nehmen wir uns jetzt wieder einem sehr essentiellem Thema an, und zwar Listen. Listen sind besondere Datenstrukturen. In ihnen können mehrere Werte gespeichert werden. Der Zugriff auf diese Werte erfolgt mit einem Index (die Stelle in der Liste, an der der Wert gespeichert ist — beginnend bei 0). Eine Liste wird mit [] definiert. |
| Aufgabe | Monty hat nun seine Fahrkarten gekauft und muss jetzt seinen Koffer packen. Er will folgende Dinge mitnehmen: Eine Hose, ein Hemd, eine Jacke, Socken, Schuhe und natürlich seine Zahnbürste. Lege eine Variable mit dem Namen 'gepaeck' an und füge sämtliche Gegenstände hinzu. Lasse anschließend die Liste auf dem Bildschirm ausgeben |
| Vorgegebener Code | liste = [0, 1, 2, 3]<br>print liste[2] |
| Lösungs Code | gepaeck = [Hose, Hemd, Jacke, Socken, Schuhe, Zahnbürste]<br>print(gepaeck) |
| Lösung Variablenbelegung | gepaeck = [Hose, Hemd, Jacke, Socken, Schuhe, Zahnbürste] |

Lektion 4:

<table>
<tr><th colspan="2">Aufgabe 4.1 (Manipulation von Listen)</th></tr>
<tr><td>Text</td><td>Hoffentlich hast du gemerkt, dass Listen an sich gar nicht schwer sind. Natürlich wäre es ziemlich sinnlos, wenn wir nur Listen anlegen, aber nichts mit ihnen tun könnten. Deswegen lernst du jetzt Möglichkeiten, wie du mit Listen coole Dinge machen kannst. Für die Manipulation von Listen haben wir folgende Möglichkeiten:
<table>
<tr><th>Funktion</th><th>Ergebnis</th></tr>
<tr><td>liste.len</td><td>gibt die Länge der Liste zurück</td></tr>
<tr><td>liste.del(index)</td><td>löscht de angegebenen Index</td></tr>
<tr><td>liste.append(x)</td><td>fügt x an die Liste an</td></tr>
<tr><td>liste.extend([ ])</td><td>fügt mehrere Elemente an die Liste an</td></tr>
<tr><td>liste.remove(x)</td><td>entfernt x aus der Liste</td></tr>
<tr><td>liste.index(x)</td><td>gibt das Element an Index x aus (angefangen bei 0)</td></tr>
<tr><td>liste.insert(index, x)</td><td>fügt Element x an gegebenem index ein</td></tr>
</table>
</td></tr>
<tr><td>Aufgabe</td><td>Mit erschrecken stellt Monty fest, dass seine Reisegesellschaft nur 5 Kilo Freigepäck erlaubt. Lege eine Variable mit dem Namen 'gewicht' an, die das Gewicht von Monty's Gepäck speichert. Er weiß, dass im Durchschnitt alle seine Sachen 400 Gramm wiegen.<br><br>Beim letzten Blick in den Wetterbericht hat er festgestellt, dass es sonnig und warm wird und er somit seine Jacke nicht brauchen wird. Entferne die Jacke wieder aus seinem Gepäck<br><br>Brian ruft Monty kurz vor der Abfahrt an und fragt ihn ob er denn Lust hätte mit ihm schwimmen zu gehen. Nun muss Monty noch seine Badehose, Badelatschen und seine Taucherbrille mitnehmen. Füge diese Dinge zu seinem Reisegepäck hinzu<br><br>Unterwegs fragt sich Monty ob der denn auch an seine Socken gedacht hat. Lege eine Variable mit dem Namen 'socken' an, welche den Index des Elements 'Socken' aus der Gepäck-Liste speichert</td></tr>
<tr><td>Vorgegebener Code</td><td>gepaeck = [Hose, Hemd, Jacke, Socken, Schuhe, Zahnbürste]</td></tr>
<tr><td>Lösungs Code</td><td>gewicht = gepaeck.len * 400<br>gepaeck.remove(‘Jacke')</td></tr>
<tr><td>Lösung Variablenbelegung</td><td>gepaeck = [Hose, Hemd, Jacke, Socken, Schuhe, Zahnbürste]</td></tr>
</table>

| **Aufgabe 4.2 (Manipulation von Listen)** | |
|---|---|
| Text | Es folgen nun noch 2 weitere kleine Aufgaben zur Listenmanipulation |
| Aufgabe | Brian ruft Monty kurz vor der Abfahrt an und fragt ihn ob er denn Lust hätte mit ihm schwimmen zu gehen. Nun muss Monty noch seine Badehose, Badelatschen und seine Taucherbrille mitnehmen. Füge diese Dinge zu seinem Reisegepäck hinzu<br><br>Unterwegs fragt sich Monty ob der denn auch an seine Socken gedacht hat. Lege eine Variable mit dem Namen 'socken' an, welche den Index des Elements 'Socken' aus der Gepäck-Liste speichert |
| Vorgegebener Code | gepaeck = [Hose, Hemd, Jacke, Socken, Schuhe, Zahnbürste] |
| Lösungs Code | gepaeck.append('Badehose')<br>gepaeck.append('Badelatschen')<br>gepaeck.append('Taucherbrille')<br>socken = gepaeck.index('Socken') |
| Lösung Variablenbelegung | gepaeck = [Hose, Hemd, Socken, Schuhe, Zahnbürste, Badehose, Badelatschen, Taucherbrille]<br>socken = 2 |

Lektion 5:

| **Aufgabe 5.1 (Kontrollstrukturen: Verzweigung mit if/else)** | |
|---|---|
| Text | Herzlichen Glückwunsch - du hast nun schon ein ganz schönes Stück Arbeit hinter dir! Zeit für eine kleine Verschnaufpause. Trink einen Schluck und Streck dich einmal, dann gehts weiter mit dem spannenden Themenkomplex der Kontrollstrukturen. Was das genau ist erfährst du jetzt: In einem Programm möchte man oftmals in Abhängigkeit von einer bestimmten Bedingung weiter verfahren. Wir könnten zum Beispiel ein Programm schreiben, welches den Fahrkartenautomaten simuliert, an dem Monty sein Ticket gekauft hat. Wenn ein Kunde eine Fahrkarte kauft, so müssen wir entscheiden ob (und wieviel) Wechselgeld er bekommt. Wenn der Wert des eingeworfenen Geldes genau mit dem Wert der Fahrkarte übereinstimmt, so bekommt er kein Wechselgeld, wenn zuviel eingeworfen wurde, dann ja. In Python können wir diesen Sachverhalt mit if und else abbilden:<br><br>if geldeinwurf > fahrpreis:<br>  Rückgabe = fahrpreis - geldeinwurf<br>else:<br>  rueckgabe = 0<br><br>Nach dem wort if (‘wenn’) folgt immer eine Bedingung die ausgewertet wird. Ist diese wahr (true) so wird der code ausgeführt, der direkt im anschluss folgt. Ist die Bedingung falsch (false), so wird der Teil hinter dem wort else (‘sonst’) ausgeführt. |
| Aufgabe | Monty möchte gerne noch ein Geschenk für Brian mitbringen. Er überlegt ob er lieber einen Strauß Blumen oder eine Tafel von Brians Lieblingsschokolade kaufen soll. Die Tafel Schokolade kostet 2 Euro und der Blumenstrauß 5 Euro. Finde mittels eines if/else Konstrukts heraus ob Monty Blumen, Schokolade oder sogar beides kaufen kann. Gebe auf dem Bildschrim die Gegenstände aus, die Monty kauft und wieviel Geld er danach noch übrig hat. |
| Vorgegebener Code | blumenstrauß = 5<br>schokolade = 2<br>taschengeld = 10 |
| Lösungs Code | if taschengeld > blumenstrauß<br>  print('Blumenstrauß')<br>  taschengeld = taschengeld - blumenstrauß<br>if taschengeld > schokolade<br>  print('Schokolade')<br>  taschengeld = taschengeld - schokolade<br>print(taschengeld) |
| Lösung Variablenbelegung | taschengeld = 3 |

Lektion 6:

| **Aufgabe 6.1 (Kontrollstrukturen: Schleifen)** | |
|---|---|
| Text | Verzweigungen waren gar nicht so schwer, oder? Oftmals möchte man jedoch auch Code wiederholt ausführen — hierfür verwenden Programmierer sogenannte Schleifen (engl. ‘loops’).<br><br>Im vorherigen Kapitel haben wir schon das Konstrukt der Liste kennengelernt. Angenommen wir haben eine Liste mit ganzzahligen werten und möchten zu jedem Wert 1 addieren, dann sähe das wie folgt aus:<br><br>liste = [0, 1, 2, 3]<br><br>for zahl in liste<br>  zahl = zahl + 1 |
| Aufgabe | Monty ist nach stundenlanger Fahrt endlich bei Brian angekommen (die Bahn hatte mal wieder Verspätung). Nun möchte er seinen Koffer auspacken. Lege eine Variable 'kleiderschrank' an und füge ihr alle Gegenstände aus der Gepäck-Liste hinzu. |
| Vorgegebener Code | gepaeck = [Hose, Hemd, Socken, Schuhe, Zahnbürste, Badehose, Badelatschen, Taucherbrille]<br>kleiderschrank = [] |
| Lösungs Code | for gegenstand in gepaeck<br>  kleiderschrank.append(gegenstand) |
| Lösung Variablenbelegung | kleiderschrank = [Hose, Hemd, Socken, Schuhe, Zahnbürste, Badehose, Badelatschen, Taucherbrille] |

Lektion 7:

| **Aufgabe 7.1 (Funktionen)** | |
|---|---|
| Text | Wow, du kannst Stolz auf dich sein, dass du schon bis hierhin gekommen bist. Nun folgt der letzte Teil unserer Reise, an dem es um Funktionen geht -- quasi die Masterclass beim programmieren lernen. Mit Hilfe von Funktionen lässt sich der Quellcode eines Programms strukturieren und wiederverwenden. Angenommen wir schreiben ein Programm, welches an mehreren Stellen die Summe von Werten in einer Liste bilden muss. Der dazugehörige Code wäre folgender:<br><br>summe = 0<br>for x in liste<br>  summe = summe + x<br><br>Beim programmieren sollte man darauf achten, möglichst wenig Code mehrfach zu schreiben. Die Berechnung der Summe der Zahlen soll deswegen in eine Funktion geschrieben werden, die man dann an den entsprechenenden Stellen aufrufen kann, und die das Ergebnis der Summe zurück gibt:<br><br>def listensumme(liste)<br>  summe = 0<br>  for x in liste<br>  summe = summe + x<br>  return summe<br><br>Die Funktion listensumme können wir nun an den Stellen aufrufen, wo sie gebraucht wird.<br><br>liste = [0, 1, 2, 3]<br>summe = listensumme(liste)  # summe = 6<br><br>Eine Funktion besteht immer aus einer Definition, Parametern, welche an die Funktion übergeben werden und einem optionalem Rückgabewert. In unserem obigen Beispiel ist die Definition ist der Code, aus welchem die Funktion besteht. Die Parameter befinden in den Klammern dahinter. In unserem Fall handelt es sich um eine Liste mit Werten. Der Rückgabewert befindet sich hinter dem return statement gekennzeichnet. |
| Aufgabe | Da Monty mehrere Fahrten zu Brian kaufen musste wird er wohl häufiger sein Gepäck in den Kleiderschrank räumen. Schreibe eine Funktion mit dem Namen 'auspacken', die eine Gepäck-Liste übergeben bekommt und den Inhalt des Kleiderschranks zurückgibt und überprüfe ihre Funktion mit einer neuen Gepäck-Liste |

| **Aufgabe 7.1 (Funktionen)** | |
|---|---|
| Vorgegebener Code | gepaeck = [Hose, Hemd, Socken, Schuhe, Zahnbürste, Badehose, Badelatschen, Taucherbrille] |
| Lösungs Code | def auspacken(gepaeck)<br>  kleiderschrank = []<br>  for gegenstand in gepaeck<br>  kleiderschrank.append(gegenstand)<br>  return kleiderschrank<br><br>kleiderschrank = auspacken(gepaeck) |
| Lösung Variablenbelegung | kleiderschrank = [Hose, Hemd, Socken, Schuhe, Zahnbürste, Badehose, Badelatschen, Taucherbrille] |