Python

Das erste Programm ist in der Regel das „Hallo Welt – Programm“. In Python gibt es für eine Ausgabe den Befehl ***print***.

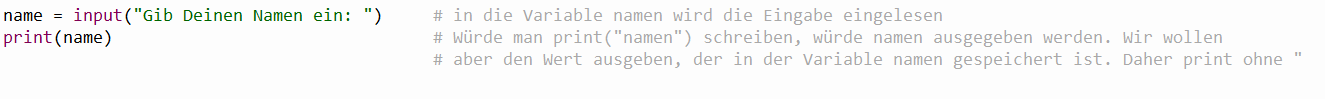
print("Hallo Welt")

Oft ist es sinnvoll, für andere oder für sich selbst, Anmerkungen in das Programm zu schreiben. Das macht man in Python mittels #. Alles was in der Zeile nach # kommt wird nicht als Programm ausgeführt.

Projekt:  
1. Es soll der Text *Heute ist ein schöner Tag* ausgegeben werden  
2. Zusätzlich soll ein Kommentar zu dieser Befehlszeile erscheinen.

Wichtig ist, dass der auszugebende Text in Gänsefüßchen steht.

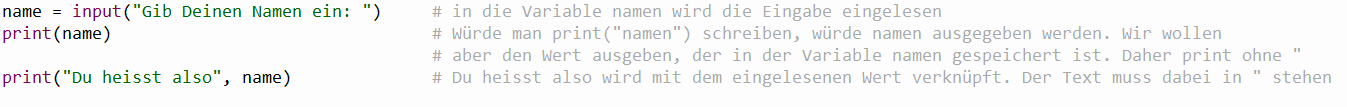
Sicher möchte man aber auch etwas in den Computer eingeben, es verarbeiten und dann eine entsprechende Ausgabe veranlassen. Der Befehl ***input*** liest Daten ein. Doch wo kommen die eingegebenen Daten hin? Hierfür brauchen wir noch eine Variable. In diese kann eben variabel alles mögliche reingeschrieben werden. Immer wenn man auf diese Variable zurückgreift holt man sich den Wert, den diese **Variable** gerade annimmt.



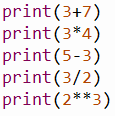
Bei der Wahl des Namens der Variablen ist darauf zu achten, dass diese auch etwas mit dem Inhalt zu tun hat. Soll das Alter eingelesen werden, wählt man als Variablennamen z. B. alter.

Projekt:  
Es sollen die Werte zweier Werte vertauscht werden.

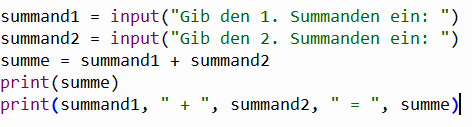
Noch eleganter wird eine Ausgabe, wenn man einen festen Text mit dem variablen Wert verknüpft. Diese Verknüpfung wird mittels Komma realisiert. 🡪 Text , Variable, Text, Variable. Es ist darauf zu achten, dass zwischen dem festen Text und der Variablen immer ein Komma steht.



Projekt:  
1. Es soll dein Alter eingelesen und ausgegeben werden.  
2. Es soll das eingelesene Alter wie folgt ausgegeben werden: *Du bist … Jahre alt.*3. Lies zusätzlich noch deinen Namen ein uns gib folgendes aus:   
 *Du heißt … und bist … Jahre alt*

Mit Python kann man auch **rechnen**. Entweder man gibt die Rechnung direkt in print ein. Dann wird nur das Ergebnis ausgegeben.   


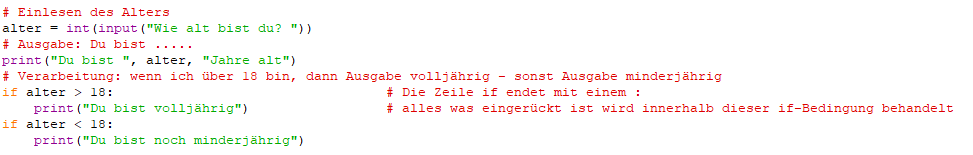
Die Multiplikation wird in den meisten Programmiersprachen mittels \* berechnet. Die Division mit /. Potenziert wird mit \*\*

Schöner wäre es aber, wenn der Nutzer z. B. die Zahlen 5 und 7 addieren möchte und man als Ausgabe erhält 5+7=12  


Bei der Variablenbezeichnung ist darauf zu achten, dass der Name ohne Leerzeichen gewählt ist, also summand1 statt summand 1.  
In die Variable summe wird das Ergebnis der Addition von summand1 und summand2 gespeichert.   
mit print(summe) wird nur das Ergebnis der Addition ausgegeben.  
Mit print(summand1, " + ", summand2, " = ", summe) wird die ganze Aufgabe mitsamt Ergebnis angezeigt. Es ist darauf zu achten, dass zwischen dem Text, in Gänsefüßchen, ein Komma stehen muss. Wird in die Gänsefüßchen ein blank (Leerzeichen) gesetzt, dann erfolgt das ebenso in der Ausgabe 🡪 3 + 4 = 7 statt 3+4=7.

Projekt:  
Berechnung des BMI: Der BMI wird nach folgender Formel berechnet: Körpergewicht (in kg) geteilt durch Größe (in m) zum Quadrat.  
1. Es sollen die benötigten Werte eingelesen werden.  
2. Es soll die Ausgabe erfolgen: *Dein BMI liegt bei …..*

Richtig interessant wird es, wenn man abhängig von einer Eingabe eine entsprechende Ausgabe erhält. Wenn man z. B. sein Alter eingibt und entsprechend die Ausgabe erhält *Du bist volljährig* oder eben *Du bist minderjährig*.  
Der Pseudocode *wenn ich über 18 bin, dann Ausgabe volljährig - sonst Ausgabe minderjährig* soll nun in Pythoncode übersetzt werden. Mit dem if-Befehl erhalten wir folgenden Code:

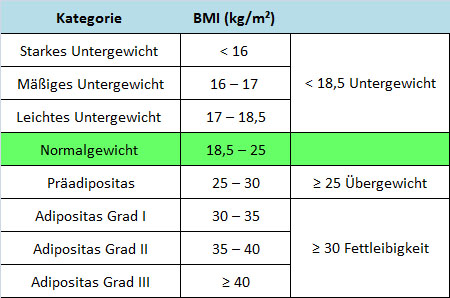


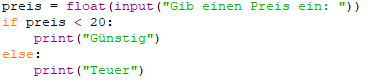
In obigem Programm erhält man für die Eingabe 18 keine Ausgabe. Hierzu muss der Code abgeändert werden in: if alter >= 18, damit bei genau 18 die Ausgabe *Du bist volljährig* erscheint.   
Will man z. B. ausgehend vom Alter ausgeben ob jemand wahrscheinlich im Kindergarten, in der Schule oder berufstätig ist, erhalten, so muss man das Alter von 6 bis 17 in den Griff bekommen. Alle Zahlen zwischen 6 **und** 17 erfüllen diese Bedingung. Daher lautet der entsprechende Programmcode:  
 

Ebenfalls ist zu beachten, dass wenn die Aussage bzgl. Zahlen genau zutreffen soll zwei Gleichheitszeichen zu verwenden sind:  

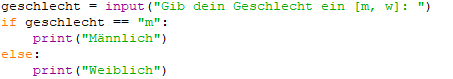

Hier nochmal die logischen Bedingungen:   
(Quelle: https://www.w3schools.com/python/python\_conditions.asp)

* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b

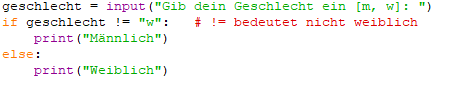
Projekt:  
Berechnung des BMI und die entsprechende Ausgabe.   
Der BMI wird nach folgender Formel berechnet: Körpergewicht (in kg) geteilt durch Größe (in m) zum Quadrat.  
  
Quelle: http://www.fitnessform.de/bmi-tabelle-und-interpretation/  
1. Berechne den BMI  
2. Je nach berechnetem Wert soll die Ausgabe Untergewicht, Normalgewicht, Übergewicht oder Fettleibigkeit erscheinen.

Wenn es nur zwei Entscheidungsmöglichkeiten gibt, kann man den if-Befehl auch mit *else* schreiben. Der Pseudocode *wenn Preis > 20, dann teuer, sonst günstig* sieht in Python wie folgt aus:  


Projekt:  
Realisiere dieses Projekt mit dem else-Befehl  
1. Lies die Temperatur ein.  
2. Wenn die Temperatur unter 30°C liegt soll *Das ist angenehm*, sonst *Das ist mir zu heiß* ausgegeben werden.

Soll ein Text verglichen werden, so muss dieser im if-Befehl in Gänsefüßchen stehen.  


Man sieht auch hier nochmal, dass == benutzt wurde.   
Die gleiche Idee könnte man auch mit der obigen Verneinung umsetzen:

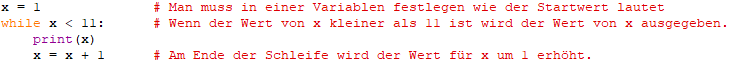


Projekt:  
Da der Nutzer nicht genau weiß, ob er m oder M eingeben muss, bzw. w oder W, soll für jeden Fall der Eingabe entsprechend männlich oder weiblich ausgegeben werden.

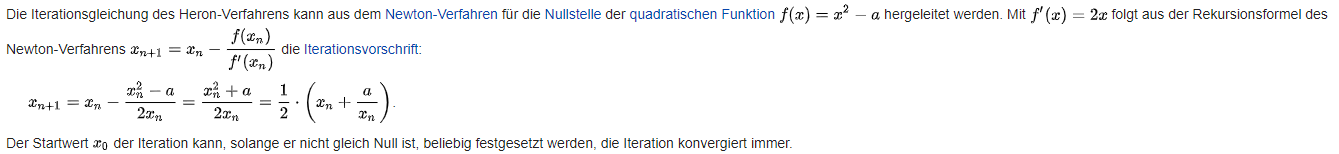
Leserlicher werden Kommazahlen mit vielen Dezimalen, wenn man sie rundet. Hierfür gibt es den Befehl *round*.  
round(Zahl, Dezimalen)

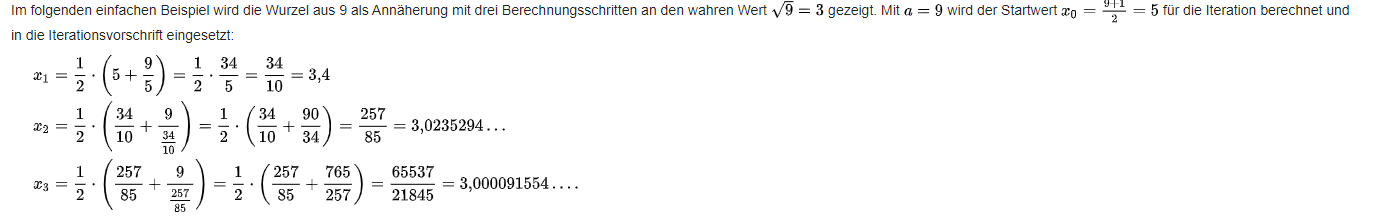
rundet die Zahl 2,3344 auf 2 Dezimalen

Projekt:  
Schreibe ein Programm, mit dem man bzgl. Der Funktion f(x) = x² + px + q die Parameter p und q einlesen kann.   
1. Davon ausgehend soll, mithilfe der p-q-Formel, die Lösung der Gleichung ausgegeben   
 werden.  
2. Wenn der Radikand negativ ist, soll eine Meldung erscheinen, dass es wegen des negativen   
 Radikanden keine Lösung der quadratischen Gleichung gibt.  
3. Runde das Ergebnis auf jeweils 3 Dezimalen

In jeder Programmiersprache gibt es die Möglichkeit auf- bzw. abwärts zu zählen. So kann man sich von Python ganz einfach die Zahlen von z. B. 3 bis 45 ausgeben lassen. Hierzu benutzt man sogenannte Schleifen. Die meisten Schleifen enthalten einen Zähler oder ganz allgemein Variablen, die im Verlauf der Berechnungen innerhalb des Schleifenkörpers ihre Werte ändern. Außerhalb, d.h. noch vor dem Beginn der Schleife, werden diese Variablen initialisiert. Vor jedem Schleifendurchlauf wird geprüft, ob ein Ausdruck, in dem diese Variable oder Variablen vorkommen, wahr ist. Dieser Ausdruck bestimmt das Endekriterium der Schleife. Solange die Berechnung dieses Ausdrucks "True" liefert wird der Rumpf der Schleife ausgeführt. Nachdem alle Anweisungen des Schleifenkörpers durchgeführt worden sind, springt die Programmsteuerung automatisch zum Anfang der Schleife, also zur Prüfung des Endekriteriums zurück und prüft wieder, ob diese nochmals erfüllt ist.  
Wenn ja, geht es wie oben beschrieben weiter, ansonsten wird der Schleifenkörper nicht mehr ausgeführt und es wird mit dem Rest des Skriptes fortgefahren. Das nebenstehende Diagramm zeigt dies schematisch. (<https://www.python-kurs.eu/schleifen.php>)   
Schauen wir uns die while Schleife etwas genauer an. So wird ausgeführt solange ein Kriterium erfüllt bzw. nicht erfüllt ist.  
Es sollen die Zahlen von 1 bis 10 ausgegeben werden.  


Projekt:  
1. Gib die Zahlen von 5 bis 15 aus.  
2. Gib die Zahlen von 20 bis 0 aus.

Projekt: Heronverfahren  
Setze folgenden Algorithmus mithilfe von Python um, um die Quadratwurzel aus einer beliebigen Zahl zu berechnen.  




Quelle: <https://de.wikipedia.org/wiki/Heron-Verfahren>