```
__author__ = "7093700, Schenk"
```

GPR-Blatt 6:
Aufgabe 2:
In binären Systemen können nur Brüche exakt dargestellt werden, deren Nenner eine Potenz von 2 ist (also z.B. : 1/4 oder 1/8). Da 5 keine Potenz von 2 ist, kann dieser Bruch nicht exakt als binärer Bruch dargestellt werden.

Die Binärapproximation von 1/5 bis zur achten Stelle nach dem Komma ist `00110011`. (siehe Aufgabe aus GPR 5 mit 8 Bit statt 32)
0*0.5+0*0.25+1*0.125+1*0.0625+0+0+1*0.0078125+1*0.00390625
= 0.125 + 0.0625 + 0.0078125 + 0.00390625 = 0.19921875

In dezimaler Form ist dies 0.19921875
Der Unterschied beträgt somit 0.2 - 0.19921875 = 0.00078125

https://github.com/philippschenk2000/gpr

```
# gpr6 Aufgabe 3 -- ANALYSE
Aufgabe ÜE-03
Berechnung von der Bonuspunkte für die Klausur mittels zweier Zahlen in
python==3.0 oder neuer

Ein- und Ausgabeformat:
------------------------------
Ein: Zwei Strings, die durch Benutzereingabe erhalten werden. Der erste
String (needle) ist der zu suchende String, und der zweite String
(haystack) ist der String, in dem gesucht wird.
Aus: Gibt den Index des ersten Vorkommens von needle in haystack aus. Wenn
needle nicht gefunden wird, gibt das Programm -1 aus.

Annahmen:
------------------------------
Der Benutzer gibt gültige Strings ein.

Entwurfsmuster:
------------------------------
Ich werde auch in Zukunft nach einem ähnlichen Muster vorgehen, nämlich
dass die Überfunktion "def main" regelmäßig weitere Funktionen aufruft und
somit als "oberste Funktion" gilt.
Aus Gründen der Übersichtlichkeit füge ich mehrere ähnliche Aufgaben dann
zu kleinen Unterfunktionen zusammen, sodass diese von "def main" aus
gesteuert werden.
Iterativ: Das Skript durchläuft haystack Zeichen für Zeichen und prüft bei
jedem Schritt, ob die folgenden Zeichen mit needle übereinstimmen.
Früher Abbruch: Sobald das erste Vorkommen von needle in haystack gefunden
wird, beendet das Skript die Suche und gibt den Index aus.

# gpr6 Aufgabe 3 -- TESTS
Code: siehe epr2_exc2_Schenk_7093700.py
------------------------------
Test1:
IN: du, hallo wie gehts dir du knecht
SHOULD: 20
```

```
OUT: 20

Test2:
IN: 124, 1389502105205380457930q
SHOULD: -1
OUT: -1

Test3:
IN: meine amen dha, amen
SHOULD: -1
OUT: -1
```