

# Python: Übungsblatt 8 (reguläre Ausdrücke)

## Erzeugung regulärer Ausdrücke

Entwickeln und Testen Sie die regulären mithilfe eines Online-Auswerters (z. B. regex101.com).

- Gegeben sei die rechts stehende Namensliste. Schreiben Sie einen regulären Ausdruck, der die Namen (Vornamen plus Nachname) für alle Personen mit dem Nachnamen Hoppenstedt matcht. Annahmen: 1) Der Vorname ist das Wort unmittelbar vor dem Nachnamen; 2) Der Vorname beginnt mit einem Großbuchstaben. (Die in der Liste *hervorgehobenen* Zeichenketten sollen gematcht werden.)

**Lösung:** `([A-Z][a-z]+) Hoppenstedt`

Erwin Lindemann  
*Walter Hoppenstedt*  
*Lieselotte Hoppenstedt*  
 Heinrich Loose  
*Dicki Hoppenstedt*  
 Gwyneth Molesworth

- Gegeben sei der rechts stehende Text. Schreiben Sie einen regulären Ausdruck, der alle separat stehenden Zahlen matcht (Ziffernfolgen ohne Buchstabe unmittelbar vor/nach der Zahl).

**Lösung:** `\b\d+\b`

Match eins: *1*  
 Match zwei: *20*  
 Match drei: *314*  
 Non-match eins: 1EU  
 Non-match zwei: x2  
 Non-match drei: 1x2

- Gegeben sei der rechts stehende Text. Schreiben Sie einen regulären Ausdruck der Wörter markiert, wenn sie zweimal direkt nebeneinander vorkommen (im Beispieltext: ich, die, Herauf, an).

**Lösung:** `(\b\w+\b)\s\1`

Da steh ich nun, *ich ich* armer Tor!  
 Und bin so klug als wie zuvor;  
 Heiße Magister, heiße Doktor gar  
 Und ziehe schon an *die die* zehen Jahr  
*Herauf*  
*Herauf*, herab und quer und krumm  
 Meine Schüler *an an* der Nase herum-  
 Und sehe, daß wir nichts wissen können!  
 Das will mir schier das Herz verbrennen.  
 Goethe (Faust: Der Tragödie erster Teil)

- Gegeben sei die links stehende Wegbeschreibung (zur FOM Essen). **a)** Schreiben Sie einen regulären Ausdruck, der alle Straßennamen der Form ...straße matcht (z.B. Steelerstraße). Anderen Straßennamen sollen nicht gematcht werden (z.B. Breite Straße oder Deutz-Mühlheimer-Straße. **b)** Erweitern Sie den Ausdruck so, dass eine evtl. gegebene Hausnummer (hier sollen der Einfachheit halber nur Zahlen erlaubt sein) ebenfalls gematcht wird (hier ein erweiterter Match: *Herkulesstraße 32*).

**Lösung a):** `\w+straße`

**Lösung b):** `\w+straße(\s\d+)?`

Wenn Sie den HBF in Richtung Innenstadt verlassen, orientieren Sie sich rechts und folgen Sie dem Verlauf der *Hollestraße* über die Kreuzung (*Steelerstraße*) weiter in die *Herkulesstraße*. Die Dauer des Gehwegs ab Hauptbahnhof beträgt ca. zehn Minuten.

Anschrift:  
*Herkulesstraße* 32  
 45127 Essen

5. Gegeben sei der rechts stehende Text. Schreiben Sie einen regulären Ausdruck, der alle Zahlen, die den Punkt (.) als Zifferngruppierung (Tausendertrennung) nutzen, zu matcht.

**Lösung a):** `\b\d{1,3}(\.\d{3})+\b`

# gematcht werden sollen:

*1.000.000 t, 5.000 t, 1.320 t*

# nicht gematcht werden sollen:

1000000, 1000, 150, 15.0

1.50, 1.000000, x1.000, 1.000x

6. Gegeben sei der rechts stehende Text. Schreiben Sie einen regulären Ausdruck der die folgenden Zeichen matcht: **a)** alle Zahlen (Preise) mit genau zwei Nachkommastellen denen genau ein EU folgt. Als Dezimaltrennzeichen kann neben dem Komma auch ein Punkt stehen. Das EU kann ohne, mit einem oder mit mehreren Leerzeichen von der Zahl getrennt sein. **b)** wie a), aber mit der Einschränkung, dass ein Preis nur dann gematcht werden sollen, wenn außer ihm keine weiteren Zeichen in der Zeile vorhanden sind (im gegebenen Text soll also nur die 9,99 EU gematcht werden).

**Lösung a):** `\d+[,\.]\d\d *EU`

**Lösung b):** `^\d+[,\.]\d\d *EU$`

# gematcht werden sollen:

*9,99 EU*

*19,95 EU, 13.50 EU*

*120,99EU, 1,99 EU*

# nicht gematcht werden sollen:

44 EU (\*), 15,999 EU

EU 9,99, 13,519 EU

20 EU, 1 EU

### Nutzung regulärer Ausdrücke mit Python

7. Gegeben sei der rechts stehende Text. **a)** Suchen Sie alle Telefonnummern und speichern Sie diese in einer Liste des folgenden Formats:

`['<Vorwahl>/<Nummer>-<Durchwahl>', ...]`

Ergebnis für das Textbeispiel:

```
telefonnummern = ['0800/77889900',
                   '01803/12341234-11', '02232/222-3']
```

**b)** Suchen Sie alle Email-Adressen und speichern Sie diese in einer Liste emailadressen.

Ergebnis für das Textbeispiel:

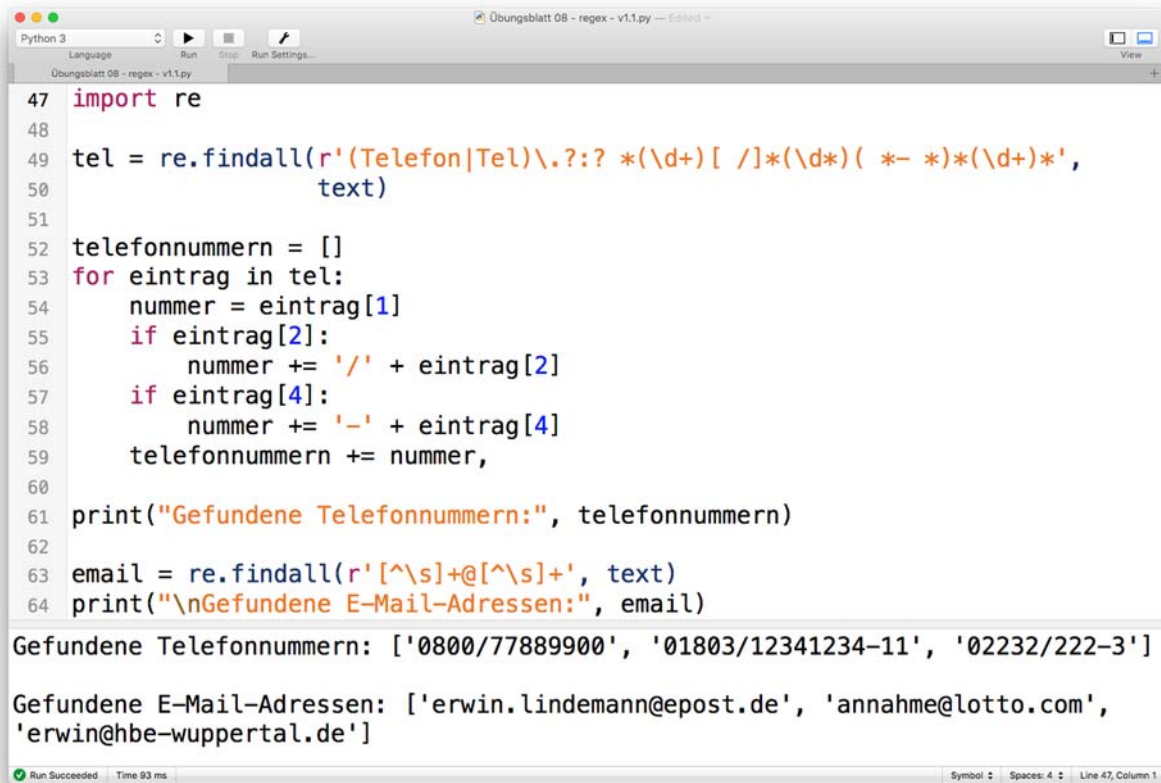
```
emailadressen = ['erwin.lindemann@epost.de',
                  'annahme@lotto.com', 'erwin@hbe-
                  wuppertal.de']
```

Erwin Lindemann  
Tel.: 0800/77889900  
Fax: 0800/77889999  
erwin.lindemann@epost.de (privat)

Lottoannahmestelle  
Telefon 01803 / 12341234-11  
Fax 01803 / 12341234-22  
annahme@lotto.com

Herren-Boutique Erwin  
Tel: 02232 222 - 3  
Email: erwin@hbe-wuppertal.de (Geschäft)

## Lösung



```
47 import re
48
49 tel = re.findall(r'(Telefon|Tel)\.?:? *(\d+)[ /]*(\d*)( *- )*(\d+)*',
50                 text)
51
52 telefonnummern = []
53 for eintrag in tel:
54     nummer = eintrag[1]
55     if eintrag[2]:
56         nummer += '/' + eintrag[2]
57     if eintrag[4]:
58         nummer += '-' + eintrag[4]
59     telefonnummern += nummer,
60
61 print("Gefundene Telefonnummern:", telefonnummern)
62
63 email = re.findall(r'^\s]+@[^\s]+', text)
64 print("\nGefundene E-Mail-Adressen:", email)
```

Gefundene Telefonnummern: ['0800/77889900', '01803/12341234-11', '02232/222-3']

Gefundene E-Mail-Adressen: ['erwin.lindemann@epost.de', 'annahme@lotto.com', 'erwin@hbe-wuppertal.de']

Run Succeeded Time 93 ms Symbol Spaces: 4 Line 47, Column 1