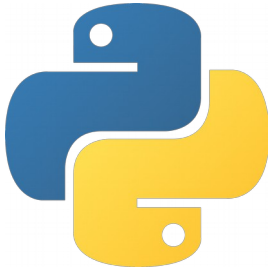
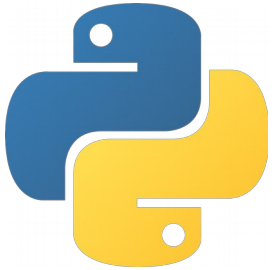
Python Strings



- Reguläre Ausdrücke – re
  - Es gibt zwei große Anwendungsbereiche von regulären Ausdrücken (regular expressions)
  - Im ersten Bereich, beim sogenannten Matching, wird geprüft, ob ein Textabschnitt auf das Muster des regulären Ausdrucks passt oder nicht
    - Ein häufiges Beispiel für Matching ist ein Test, ob eine eingegebene E-Mail-Adresse syntaktisch gültig ist
  - Die zweite Einsatzmöglichkeit von regulären Ausdrücken ist das sogenannte Searching
    - Innerhalb eines größeren Textes wird nach Textfragmenten gesucht, die auf einen regulären Ausdruck passen

Python Strings



- Reguläre Ausdrücke – re
  - Ein regulärer Ausdruck ist in Python ein String, der die entsprechenden Regeln enthält
  - Im Gegensatz zu manch anderen Programmiersprachen existiert hier kein eigenes Literal zu diesem Zweck

  - Empfehlenswert
  - Auf Pythons Raw-Strings zurückgreifen, in denen keine Escape-Sequenzen möglich sind
  - Zur Erinnerung: Raw-Strings werden in Python durch ein vorangestelltes r gekennzeichnet

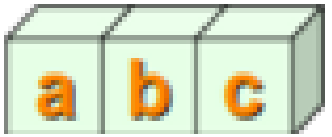
  - Beispiel
    r"\Hallo Welt"

Python Strings

- Reguläre Ausdrücke – re
  - Funktionalität wie ein RegEx arbeitet
  - Grundlage für das Beispiel

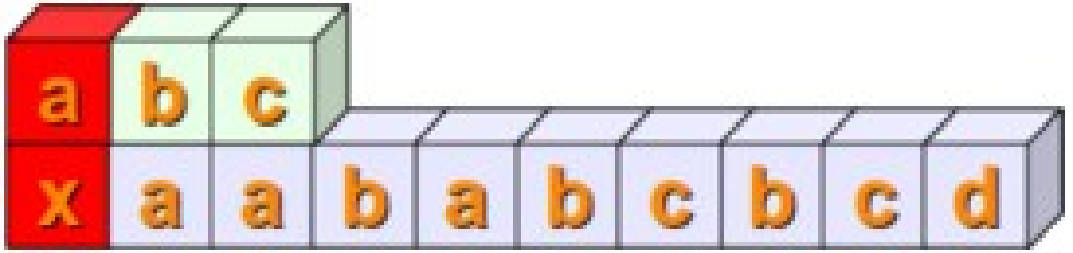
  - String s = “xaababcbcd“

  - Substring sub = “abc“

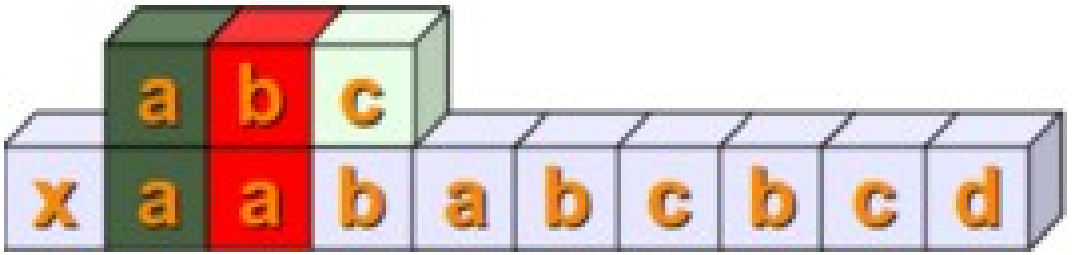


Python Strings

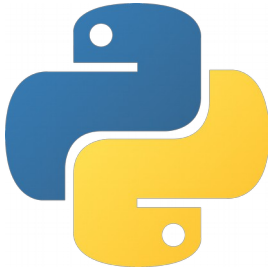
- Reguläre Ausdrücke – re
  - Zuerst wird geprüft, ob die ersten Positionen übereinstimmen, also s[0] == sub[0]
  - Schritt 1



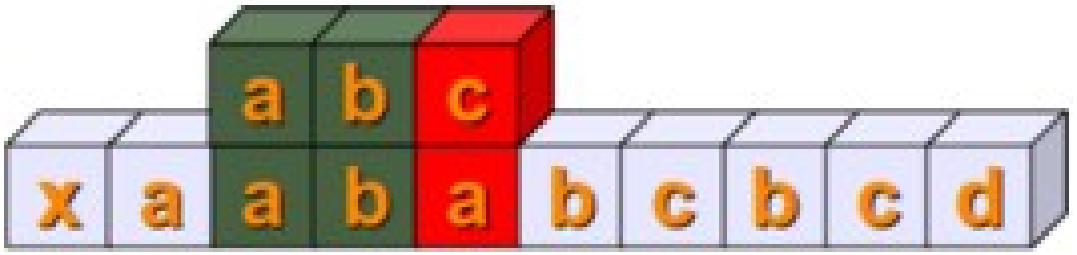
  - Schritt 2



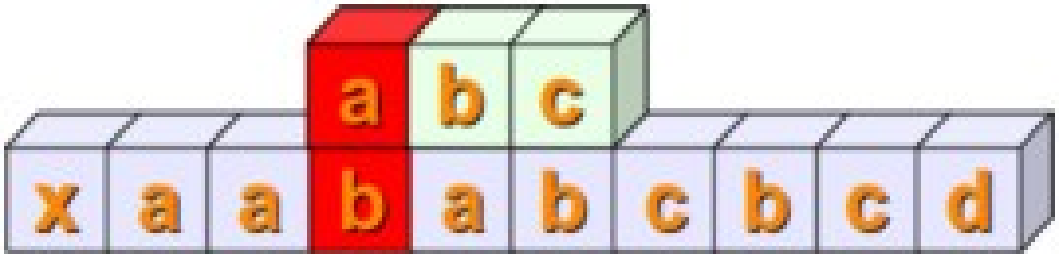
Python Strings



- Reguläre Ausdrücke – re
  - Schritt 3

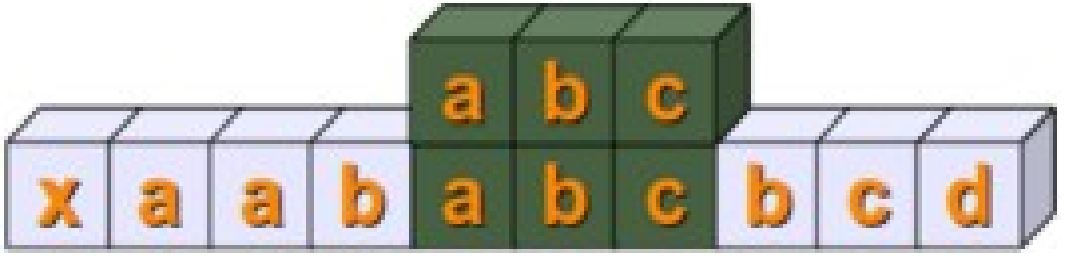


  - Schritt 4

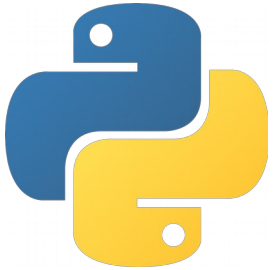


Python Strings

- Reguläre Ausdrücke – re
  - Schritt 5



Python Strings



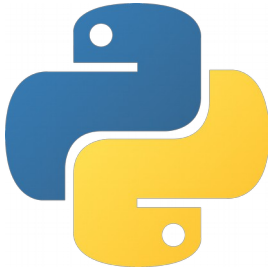
- Reguläre Ausdrücke – re
  - Syntax regulärer Ausdrücke

  - Der String → exakte Übereinstimmung
    r"python" → exakt auf den String "python" passen

  - Der Punkt (.) → Ein beliebige Zeichen
    r".ython" → "python", "Python" und "Jython", nicht "Blython"

  - Eckige Klammern [ ] → Zeichenklassen
    r"[jp]ython" → "jython", "python"
    → nicht "Python", "jpython" oder "ython"

Python Strings



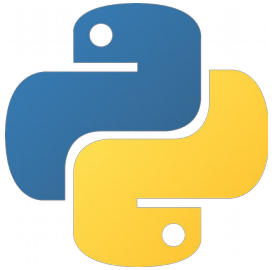
- Reguläre Ausdrücke – re
  - Syntax regulärer Ausdrücke

  - Eckige Klammern [A-Z] → Zeichenklassen mit Bereich r"[A-Z]ython"
  - Dieser reguläre Ausdruck lässt jeden Großbuchstaben als Anfangsbuchstaben des Wortes durch
  - Nicht aber Kleinbuchstaben und Zahlen

  - Eckige Klammern [A-Ra-r] → Zeichenklassen mit mehreren r"[A-Ra-r]ython"

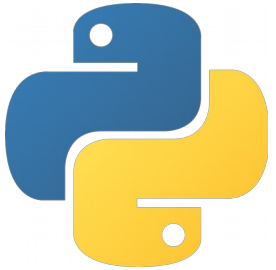
  - Eckige Klammern [0-9] → Zeichenklassen mit Zahlenbereich r"[0-9]ython"

Python Strings



- Reguläre Ausdrücke – re
  - Syntax regulärer Ausdrücke
  - Eckige Klammern [^pP] → Zeichenklassen nicht p oder P
    r"[^pP]ython"
  - Hier würden sowohl "Sython" als auch "wython" passen
  - Während "Python" und "python" außenvor bleiben

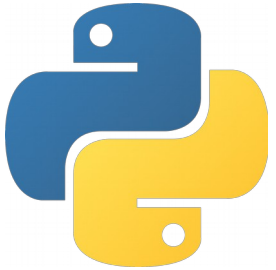
Python Strings



- Reguläre Ausdrücke – re
  - Quantoren
  - Das sind spezielle Zeichen, die hinter ein einzelnes Zeichenliteral oder eine Zeichenklasse geschrieben werden und kennzeichnen, wie oft diese auftreten dürfen

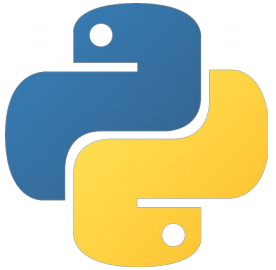
Quantor	Bedeutung
?	Das vorangegangene Zeichen bzw. die vorangegangene Zeichenklasse darf entweder keinmal oder einmal vorkommen.
*	Das vorangegangene Zeichen bzw. die vorangegangene Zeichenklasse darf beliebig oft hintereinander vorkommen, das heißt unter anderem, dass sie auch weggelassen werden kann.
+	Das vorangegangene Zeichen bzw. die vorangegangene Zeichenklasse darf beliebig oft hintereinander vorkommen, mindestens aber einmal. Sie darf also nicht weggelassen werden.

Python Strings



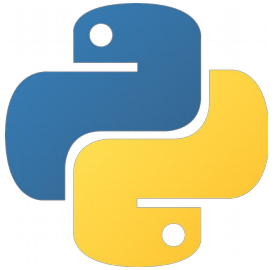
- Reguläre Ausdrücke – re
  - Quantoren – Allgemein
  - r"P[Yy]?thon"
  - Dieser reguläre Ausdruck erwartet an der zweiten Stelle des Wortes ein höchstens einmaliges Auftreten des großen oder kleinen »Y«
  - Damit passt der Ausdruck beispielsweise auf die Wörter "Python" und "Pthon", jedoch nicht auf "Pyython"

Python Strings



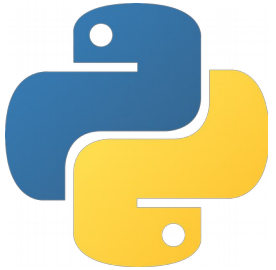
- Reguläre Ausdrücke – re
  - Quantoren – Allgemein
  - r"P[Yy]*thon"
  - Dieser reguläre Ausdruck erwartet an der zweiten Stelle des Wortes ein beliebig häufiges Auftreten des großen oder kleinen »Y«
  - Damit passt der Ausdruck beispielsweise auf die Wörter "Python" , "Pthon" und "PyyYYYyython", jedoch nicht auf "Pzthon"

Python Strings



- Reguläre Ausdrücke – re
  - Quantoren – Allgemein
  - r"P[Yy]+thon"
  - Dieser reguläre Ausdruck erwartet an der zweiten Stelle des Wortes ein mindestens einmaliges Auftreten des großen oder kleinen »Y«
  - Damit passt der Ausdruck beispielsweise auf die Wörter "Python", "PYthon" und "PyyYYYyython", jedoch nicht auf "Pthon"

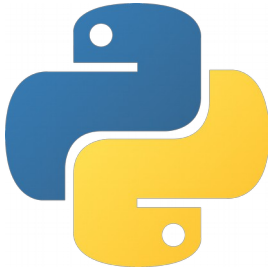
Python Strings



- Reguläre Ausdrücke – re
  - Quantoren – Unter- und Obergrenzen für Wiederholungen

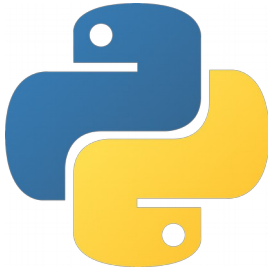
Quantor	Bedeutung
`{anz}`	Das vorangegangene Zeichen bzw. die vorangegangene Zeichenklasse muss exakt `anz`-mal vorkommen.
`{min,}`	Das vorangegangene Zeichen bzw. die vorangegangene Zeichenklasse muss mindestens `min`-mal vorkommen.
`{,max}`	Das vorangegangene Zeichen bzw. die vorangegangene Zeichenklasse darf maximal `max`-mal vorkommen.
`{min,max}`	Das vorangegangene Zeichen bzw. die vorangegangene Zeichenklasse muss mindestens `min`-mal und darf maximal `max`-mal vorkommen.

Python Strings



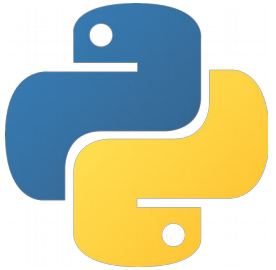
- Reguläre Ausdrücke – re
  - Quantoren – Unter- und Obergrenzen für Wiederholungen
  - r"P[Yy]{2}thon"
  - Dieser reguläre Ausdruck erwartet an der zweiten Stelle des Wortes exakt zwei jeweils große oder kleine »Y«
  - Damit passt der Ausdruck beispielsweise auf die Wörter "Pyython" oder "PYython" , jedoch nicht auf "Pyyython"

Python Strings



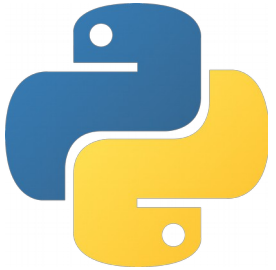
- Reguläre Ausdrücke – re
  - Quantoren – Unter- und Obergrenzen für Wiederholungen
  - r"P[Yy]{2,}thon"
  - Dieser reguläre Ausdruck erwartet an der zweiten Stelle des Wortes mindestens zwei jeweils große oder kleine »Y«
  - Damit passt der Ausdruck beispielsweise auf die Wörter "Pyython", "PYython" und "PyyYYYyython", jedoch nicht auf "Python"

Python
Strings



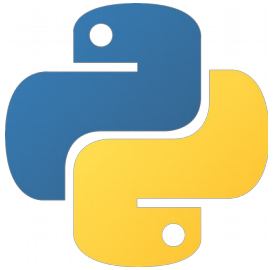
- Reguläre Ausdrücke – re
  - Quantoren – Unter- und Obergrenzen für Wiederholungen
  - r"P[Yy]{,2}thon"
  - Dieser reguläre Ausdruck erwartet an der zweiten Stelle des Wortes maximal zwei jeweils große oder kleine »Y«
  - Damit passt der Ausdruck beispielsweise auf die Wörter "Python", "Pthon" und "PYYthon", jedoch nicht auf "Pyyython"

Python Strings



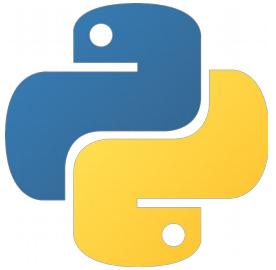
- Reguläre Ausdrücke – re
  - Quantoren – Unter- und Obergrenzen für Wiederholungen
  - r"P[Yy]{1,2}thon"
  - Dieser reguläre Ausdruck erwartet an der zweiten Stelle des Wortes mindestens ein und maximal zwei große oder kleine »Y«
  - Damit passt der Ausdruck beispielsweise auf die Wörter "Python" oder "PYython", jedoch nicht auf "Pthon" oder "PYYYthon"

Python Strings



- Reguläre Ausdrücke – re
  - Vordefinierte Zeichenklassen
  - Damit Sie nicht bei jedem regulären Ausdruck das Rad neu erfinden müssen
    - beispielsweise alle Ziffern oder alle alphanumerischen Zeichen umfassen
  - Werden bei der Arbeit mit regulären Ausdrücken häufig benötigt und können deswegen durch einen speziellen Code abgekürzt werden

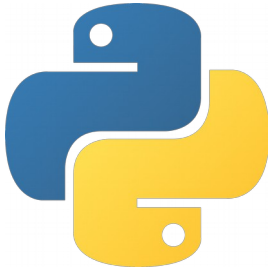
Python Strings



- Reguläre Ausdrücke – re
  - Vordefinierte Zeichenklassen

Zeichenklasse	Bedeutung
\d	Passt auf alle Zeichen, die Ziffern des Dezimalsystems sind. Ist äquivalent zu [0-9].
\D	Passt auf alle Zeichen, die nicht Ziffern des Dezimalsystems sind. Ist äquivalent zu [^0-9].
\s	Passt auf alle Whitespace-Zeichen. Ist äquivalent zu [\t\n\r\f\v].
\S	Passt auf alle Zeichen, die kein Whitespace sind. Ist äquivalent zu [^ \t\n\r\f\v].

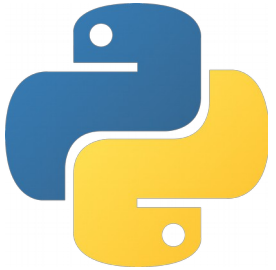
Python Strings



- Reguläre Ausdrücke – re
  - Vordefinierte Zeichenklassen

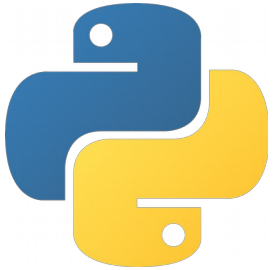
Zeichenklasse	Bedeutung
\w	Passt auf alle alphanumerischen Zeichen und den Unterstrich. Ist äquivalent zu [a-zA-Z0-9_].
\W	Passt auf alle Zeichen, die nicht alphanumerisch und kein Unterstrich sind. Ist äquivalent zu [^a-zA-Z0-9_].

Python Strings



- Reguläre Ausdrücke – re
  - Vordefinierte Zeichenklassen
  - r"P\w*th\dn"
  - Passt auf die Wörter "Pyth0n" oder "P_th1n" , beispielsweise jedoch nicht auf "Python"

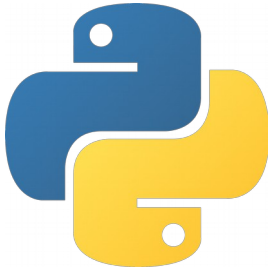
Python Strings



- Reguläre Ausdrücke – re
  - Vordefinierte Zeichenklassen

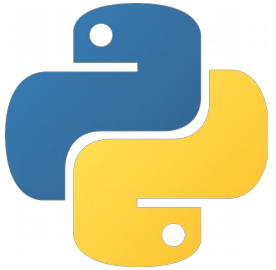
  - Beachten Sie, dass die üblichen Escape-Sequenzen, die innerhalb eines Strings verwendet werden können, auch innerhalb eines regulären Ausdrucks – selbst wenn er in einem Raw-String geschrieben wird – ihre Bedeutung behalten und nicht mit den hier vorgestellten Zeichenklassen interferieren. Gebräuchlich sind hier vor allem \n, \t , \r oder \\, insbesondere aber auch \x

Python Strings



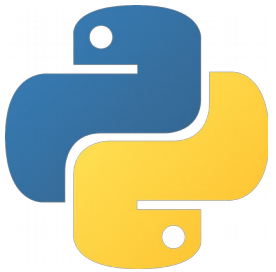
- Reguläre Ausdrücke – re
  - Vordefinierte Zeichenklassen
  - Zudem ist es mit dem Backslash möglich, einem Sonderzeichen die spezielle Bedeutung zu nehmen, die es innerhalb eines regulären Ausdrucks trägt
  - Auf diese Weise können Sie zum Beispiel mit den Zeichen »*« oder »+« arbeiten, ohne dass diese als Quantoren angesehen werden

Python Strings



- Reguläre Ausdrücke – re
  - Vordefinierte Zeichenklassen
  - r"*Py\.\.\.on*"
  - Passt allein auf den String "*Py...on*"

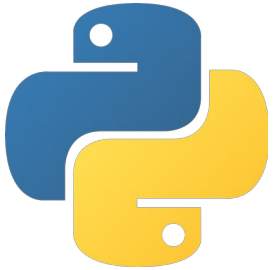
Python Strings



- Reguläre Ausdrücke – re
  - Weitere Sonderzeichen
  - Für gewisse Einsatzgebiete wird verlangt, Regeln aufstellen zu können, die über die bloße Zeichenebene hinausgehen

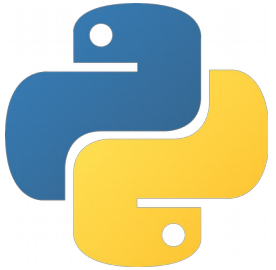
Sonderzeichen	Bedeutung
\A	Passt nur am Anfang eines Strings.
\b	Passt nur am Anfang oder Ende eines Wortes. Ein Wort kann aus allen Zeichen der Klasse \w bestehen und wird durch ein Zeichen der Klasse \s begrenzt.
\B	Passt nur, wenn es sich nicht um den Anfang oder das Ende eines Wortes handelt.
\Z	Passt nur am Ende eines Strings.
^	Passt nur am Anfang eines Strings. Beachten Sie, dass das Zeichen ^ zwei Bedeutungen hat und innerhalb einer Zeichenklasse die aufgelisteten Zeichen ausschließt. Wenn das MULTILINE-Flag gesetzt wurde, passt ^ auch direkt nach jedem Newline-Zeichen innerhalb des Strings.[4]
$	Passt nur am Ende eines Strings. Wenn das MULTILINE-Flag gesetzt wurde, passt $ auch direkt vor jedem Newline-Zeichen innerhalb des Strings.

Python Strings



- Reguläre Ausdrücke – re
  - Weitere Sonderzeichen
  - r"Python\Z"
  - Passt nur bei dem String "… Python", nicht jedoch bei "Python rockt"
  - Beispiele beziehen sich hauptsächlich auf das Matching
  - Diese Sonderzeichen sind aber gerade beim Searching von Interesse
    - Stellen Sie sich einmal vor, Sie würden in einem Text nach allen Vorkommen einer bestimmten Zeichenkette am Zeilenanfang suchen wollen
      - Dies wäre nur durch Einsatz des Sonderzeichens ^ möglich

Python Strings



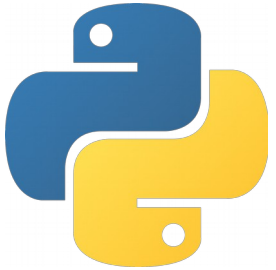
- Reguläre Ausdrücke – re
  - Genügsame Quantoren

  - Quantoren werden in der Terminologie regulärer Ausdrücke als »gefräßig« bezeichnet
  - Diese Klassifizierung ist nur beim Searching von Bedeutung

  - Beispiel → r"Py.*on"

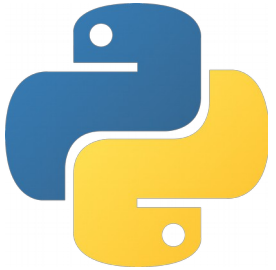
  - Dieser Ausdruck passt auf jeden Teilstring, der mit Py beginnt und mit on endet
  - Dazwischen können beliebig viele, nicht näher spezifizierte Zeichen stehen

Python Strings



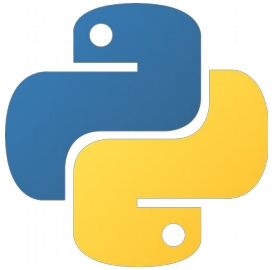
- Reguläre Ausdrücke – re
  - Genügsame Quantoren
  - Beispiel → r"Py.*on"
  - den regulären Ausdruck gedanklich auf den folgenden String anwenden
  - "Python Python Python"
  - Wieviele Treffer?

Python Strings



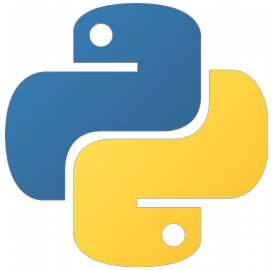
- Reguläre Ausdrücke – re
  - Genügsame Quantoren
  - Beispiel → r"Py.*on"
  - den regulären Ausdruck gedanklich auf den folgenden String anwenden
  - "Python Python Python"
  - Wieviele Treffer?
    - Sie denken es wären 3? ← Leider falsch
    - Es handelt sich um exakt ein Ergebnis, nämlich den Teilstring "Python Python Python"

Python Strings



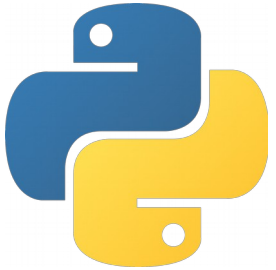
- Reguläre Ausdrücke – re
  - Genügsame Quantoren
  - Beispiel → r"Py.*on" → "Python Python Python"
  - Es wurde der »gefräßige« Quantor * eingesetzt
  - Ein solcher gefräßiger Quantor hat die Ambition, die maximal mögliche Anzahl Zeichen zu »verschlingen«
  - Beim Searching wird also, solange die »gefräßigen« Quantoren eingesetzt werden, stets der größtmögliche passende String gefunden

Python Strings



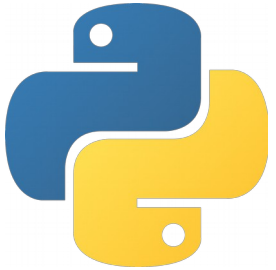
- Reguläre Ausdrücke – re
  - Genügsame Quantoren
  - Beispiel → r"Py.*on" → "Python Python Python"
  - Dieses Verhalten lässt sich umkehren, sodass immer der kleinstmögliche passende String gefunden wird
  - Dazu können Sie an jeden Quantor ein Fragezeichen anfügen
  - Dadurch wird der Quantor »genügsam«

Python Strings



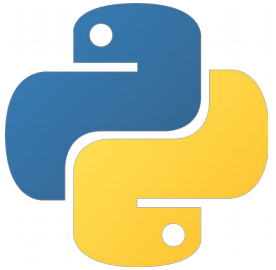
- Reguläre Ausdrücke – re
  - Genügsame Quantoren
  - Beispiel → r"Py.*?on"
  - Den angepassten regulären Ausdruck gedanklich auf den folgenden String anwenden
  - "Python Python Python"
  - Wieviele Treffer?

Python Strings



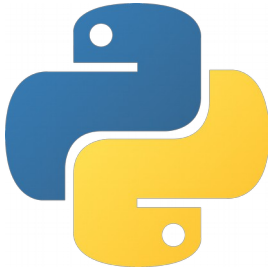
- Reguläre Ausdrücke – re
  - Genügsame Quantoren
  - Beispiel → r"Py.*on"
  - Den angepassten regulären Ausdruck gedanklich auf den folgenden String anwenden
  - "Python Python Python"
  - Wieviele Treffer?
    - Richtig → Jetzt sind es 3
  - Dies funktioniert für die Quantoren ?, *, + und {}

Python Strings



- Reguläre Ausdrücke – re
  - Gruppen
    - Ein Teil eines regulären Ausdrucks kann durch runde Klammern zu einer sogenannten Gruppe zusammengefasst werden
    - Eine solche Gruppierung hat im Wesentlichen drei Vorteile

Python Strings



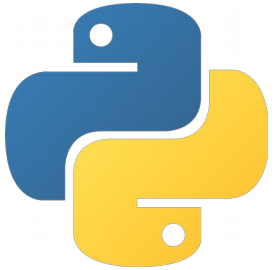
- Reguläre Ausdrücke – re
  - Gruppen

  - Vorteil 1
  - Eine Gruppe kann als Einheit betrachtet und als solche mit einem Quantor versehen werden. Auf diese Weise lässt sich beispielsweise das mehrmalige Auftreten einer bestimmten Zeichenkette erlauben:

    r"(?Python)+ ist gut"

    In diesem Ausdruck existiert eine Gruppe um den Teilausdruck r" ?Python". Dieser Teilausdruck passt auf den String "Python" mit einem optionalen Leerzeichen zu Beginn. Die gesamte Gruppe kann nun beliebig oft vorkommen, womit der obige reguläre Ausdruck sowohl auf "Python ist gut" als auch auf "Python Python Python ist gut" passt. Die Gruppe muss aber mindestens einmal auftreten, der Ausdruck passt nicht auf den String " ist gut"

Python Strings

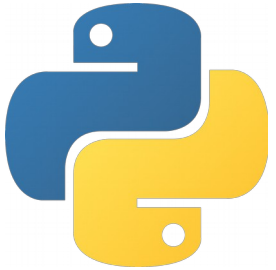


- Reguläre Ausdrücke – re
  - Gruppen

  - Vorteil 2
  - Der zweite Vorteil einer Gruppe ist, dass Sie auf sie zugreifen können, nachdem das Searching bzw. Matching durchgeführt wurde. Das heißt, Sie könnten beispielsweise überprüfen, ob eine eingegebene URL gültig ist, und gleichzeitig Subdomain, Domain und TLD herausfiltern

  - Vorteil 3
  - Es gibt Gruppen, die in einem regulären Ausdruck häufiger gebraucht werden. Um diese nicht jedes Mal erneut schreiben zu müssen, werden Gruppen, mit 1 beginnend, durchnummeriert und können dann anhand ihres Index referenziert werden. Eine solche Referenz besteht aus einem Backslash, gefolgt von dem Index der jeweiligen Gruppe, und passt auf den gleichen Teilstring, auf den die Gruppe gepasst hat. So passt der reguläre Ausdruck r"(Python) \1" auf "Python Python"

Python Strings



- Reguläre Ausdrücke – re
  - Alternativen

    - Eine weitere Möglichkeit, die die Syntax regulärer Ausdrücke vorsieht, sind sogenannte Alternativen
    - Im Prinzip handelt es sich dabei um nichts anderes als um eine ODER-Verknüpfung zweier Zeichen oder Zeichengruppen, wie Sie sie bereits von dem Operator or her kennen
    - Diese Verknüpfung wird durch den senkrechten Strich |, auch Pipe genannt, durchgeführt

      r"P(ython|eter)"

    - Dieser reguläre Ausdruck passt sowohl auf den String "Python" als auch auf "Peter"
    - Durch die Gruppe kann später ausgelesen werden, welche der beiden Alternativen aufgetreten ist.