APSSDC

Andhra Pradesh State Skill Development Corporation

(https://apssdc.in)

Day 15 Regular Expressions in Python

Recap

- Modules and Packages
  - Builtin Packages/Modules
    - math, random, calendar, os
  - User Defined packages/ Modules

Today Objectives

- Regular Expressions (Regex)

In [2]:

```
s = """Python is an interpreted high-level general-purpose programming language. Pythor
Developer: Python Software Foundation
Stable release: 3.9.5 / 3 May 2021; 29 days ago
Preview release: 3.10.0b1 / 3 May 2021; 29 days ago
Typing discipline: Duck, dynamic, strong typing; gradual (since 3.5, but ignored in CPy
First appeared: February 1991; 30 years ago
Paradigm: Multi-paradigm: object-oriented, procedural (imperative), functional, structu
```

In [3]:

```
print(list(s))
```

```
['P', 'y', 't', 'h', 'o', 'n', ' ', 'i', 's', ' ', 'a', 'n', ' ', 'i', 'n',
't', 'e', 'r', 'p', 'r', 'e', 't', 'e', 'd', ' ', 'h', 'i', 'g', 'h', '-',
'l', 'e', 'v', 'e', 'l', ' ', 'g', 'e', 'n', 'e', 'r', 'a', 'l', '-', 'p',
'u', 'r', 'p', 'o', 's', 'e', ' ', 'p', 'r', 'o', 'g', 'r', 'a', 'm', 'm',
'i', 'n', 'g', ' ', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e', '.', ' ', 'P',
'y', 't', 'h', 'o', 'n', '"', 's', ' ', 'd', 'e', 's', 'i', 'g', 'n', ' ',
'p', 'h', 'i', 'l', 'o', 's', 'o', 'p', 'h', 'y', ' ', 'e', 'm', 'p', 'h',
'a', 's', 'i', 'z', 'e', 's', ' ', 'c', 'o', 'd', 'e', ' ', 'r', 'e', 'a',
'd', 'a', 'b', 'i', 'l', 'i', 't', 'y', ' ', 'w', 'i', 't', 'h', ' ', 'i',
't', 's', ' ', 'n', 'o', 't', 'a', 'b', 'l', 'e', ' ', 'u', 's', 'e', ' ',
'o', 'f', ' ', 's', 'i', 'g', 'n', 'i', 'f', 'i', 'c', 'a', 'n', 't', ' ',
'i', 'n', 'd', 'e', 'n', 't', 'a', 't', 'i', 'o', 'n', '.', ' ', 'W', 'i',
'k', 'i', 'p', 'e', 'd', 'i', 'a', '\n', 'D', 'e', 'v', 'e', 'l', 'o', 'p',
'e', 'r', ':', ' ', 'P', 'y', 't', 'h', 'o', 'n', ' ', 'S', 'o', 'f', 't',
'w', 'a', 'r', 'e', ' ', 'F', 'o', 'u', 'n', 'd', 'a', 't', 'i', 'o', 'n',
'\n', 'S', 't', 'a', 'b', 'l', 'e', ' ', 'r', 'e', 'l', 'e', 'a', 's', 'e',
':', ' ', '3', '.', '9', '.', '5', ' ', '/', ' ', '3', ' ', 'M', 'a', 'y', '
', '2', '0', '2', '1', ';', ' ', '2', '9', ' ', 'd', 'a', 'y', 's', ' ',
'a', 'g', 'o', '\n', 'P', 'r', 'e', 'v', 'i', 'e', 'w', ' ', 'r', 'e', 'l',
'e', 'a', 's', 'e', ':', ' ', '3', '.', '1', '0', '.', '0', 'b', '1', ' ',
'/', ' ', '3', ' ', 'M', 'a', 'y', ' ', '2', '0', '2', '1', ';', ' ', '2',
'9', ' ', 'd', 'a', 'y', 's', ' ', 'a', 'g', 'o', '\n', 'T', 'y', 'p', 'i',
'n', 'g', ' ', 'd', 'i', 's', 'c', 'i', 'p', 'l', 'i', 'n', 'e', ':', ' ',
'D', 'u', 'c', 'k', ',', ' ', 'd', 'y', 'n', 'a', 'm', 'i', 'c', ',', ' ',
's', 't', 'r', 'o', 'n', 'g', ' ', 't', 'y', 'p', 'i', 'n', 'g', ';', ' ',
'g', 'r', 'a', 'd', 'u', 'a', 'l', ' ', '(', 's', 'i', 'n', 'c', 'e', ' ',
'3', '.', '5', ',', ' ', 'b', 'u', 't', ' ', 'i', 'g', 'n', 'o', 'r', 'e',
'd', ' ', 'i', 'n', ' ', 'C', 'P', 'y', 't', 'h', 'o', 'n', ')', '\n', 'F',
'i', 'r', 's', 't', ' ', 'a', 'p', 'p', 'e', 'a', 'r', 'e', 'd', ':', ' ',
'F', 'e', 'b', 'r', 'u', 'a', 'r', 'y', ' ', '1', '9', '9', '1', ';', ' ',
'3', '0', ' ', 'y', 'e', 'a', 'r', 's', ' ', 'a', 'g', 'o', '\n', 'P', 'a',
'r', 'a', 'd', 'i', 'g', 'm', ':', ' ', 'M', 'u', 'l', 't', 'i', '-', 'p',
'a', 'r', 'a', 'd', 'i', 'g', 'm', ':', ' ', 'o', 'b', 'j', 'e', 'c', 't',
'-', 'o', 'r', 'i', 'e', 'n', 't', 'e', 'd', ',', ' ', 'p', 'r', 'o', 'c',
'e', 'd', 'u', 'r', 'a', 'l', ' ', '(', 'i', 'm', 'p', 'e', 'r', 'a', 't',
'i', 'v', 'e', ')', ',', ' ', 'f', 'u', 'n', 'c', 't', 'i', 'o', 'n', 'a',
'l', ',', ' ', 's', 't', 'r', 'u', 'c', 't', 'u', 'r', 'e', 'd', ',', ' ',
'r', 'e', 'f', 'l', 'e', 'c', 't', 'i', 'v', 'e']
```

In [4]:

```
li = []

for char in s:
    if char.isupper():
        li.append(char)
print(li)
```

```
['P', 'P', 'W', 'D', 'P', 'S', 'F', 'S', 'M', 'P', 'M', 'T', 'D', 'C', 'P',
'F', 'F', 'P', 'M']
```

In [5]:

```
li = []

for char in s:
    if not char.isalnum():
        li.append(char)
print(li)
```

```
[' ', ' ', ' ', ' ', '-', ' ', '-', ' ', ' ', '.', ' ', "'", ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '.', ' ', '\n', ':', ' ', ' ',
' ', '\n', ' ', ':', ' ', '.', '.', ' ', '/', ' ', ' ', ' ', ';', ' ', ' ',
' ', '\n', ' ', ':', ' ', '.', '.', ' ', '/', ' ', ' ', ' ', ';', ' ', ' ',
' ', '\n', ' ', ':', ' ', ',', ' ', ',', ' ', ' ', ';', ' ', ' ', '(', ' ',
'.', ',', ' ', ' ', ' ', ' ', ')', '\n', ' ', ':', ' ', ' ', ';', ' ', ' ',
' ', '\n', ':', ' ', '-', ':', ' ', '-', ',', ' ', ' ', '(', ')', ',', ' ',
',', ' ', ',', ' ']
```

In [7]:

```
print(len(s))
```

```
565
```

replace all special characters with **

In [8]:

```
print(s)
```

```
Python is an interpreted high-level general-purpose programming language. Py
thon's design philosophy emphasizes code readability with its notable use of
significant indentation. Wikipedia
Developer: Python Software Foundation
Stable release: 3.9.5 / 3 May 2021; 29 days ago
Preview release: 3.10.0b1 / 3 May 2021; 29 days ago
Typing discipline: Duck, dynamic, strong typing; gradual (since 3.5, but ign
ored in CPython)
First appeared: February 1991; 30 years ago
Paradigm: Multi-paradigm: object-oriented, procedural (imperative), function
al, structured, reflective
```

In [9]:

```
s1 = ""

for char in s:
    if not char.isalnum():
        s1 += '**'
    else:
        s1 += char


print(s1)
```

```
Python**is**an**interpreted**high**level**general**purpose**programming**lan
guage****Python**s**design**philosophy**emphasizes**code**readability**with*
*its**notable**use**of**significant**indentation****Wikipedia**Developer****
Python**Software**Foundation**Stable**release****3**9**5******3**May**2021**
**29**days**ago**Preview**release****3**10**0b1******3**May**2021****29**day
s**ago**Typing**discipline****Duck****dynamic****strong**typing****gradual**
**since**3**5****but**ignored**in**CPython****First**appeared****February**1
991****30**years**ago**Paradigm****Multi**paradigm****object**oriented****pr
ocedural****imperative******functional****structured****reflective
```

In [10]:

```
s1 = ""

for char in s:
    if not char.isalnum() and char != ' ' and not char.isspace():
        s1 += '**'
    else:
        s1 += char


print(s1)
```

```
Python is an interpreted high**level general**purpose programming language**
Python**s design philosophy emphasizes code readability with its notable use
of significant indentation** Wikipedia
Developer** Python Software Foundation
Stable release** 3**9**5 ** 3 May 2021** 29 days ago
Preview release** 3**10**0b1 ** 3 May 2021** 29 days ago
Typing discipline** Duck** dynamic** strong typing** gradual **since 3**5**
but ignored in CPython**
First appeared** February 1991** 30 years ago
Paradigm** Multi**paradigm** object**oriented** procedural **imperative****
functional** structured** reflective
```

In [12]:

```
ss = s.split()
li = []

for char in ss:
    if char[0].islower():
        li.append(char)
print(li)
```

```
['is', 'an', 'interpreted', 'high-level', 'general-purpose', 'programming',
'language.', 'design', 'philosophy', 'emphasizes', 'code', 'readability', 'w
ith', 'its', 'notable', 'use', 'of', 'significant', 'indentation.', 'releas
e:', 'days', 'ago', 'release:', 'days', 'ago', 'discipline:', 'dynamic,', 's
trong', 'typing;', 'gradual', 'but', 'ignored', 'in', 'appeared:', 'years',
'ago', 'object-oriented,', 'procedural', 'functional,', 'structured,', 'refl
ective']
```

Regular Expression

a group of character/special characters for matching the pattern

In [13]:

```
import re
```

In [15]:

```
1 print(re.__doc__)
```

```
Support for regular expressions (RE).

This module provides regular expression matching operations similar to
those found in Perl.  It supports both 8-bit and Unicode strings; both
the pattern and the strings being processed can contain null bytes and
characters outside the US ASCII range.

Regular expressions can contain both special and ordinary characters.
Most ordinary characters, like "A", "a", or "0", are the simplest
regular expressions; they simply match themselves.  You can
concatenate ordinary characters, so last matches the string 'last'.

The special characters are:
    "."      Matches any character except a newline.
    "^"      Matches the start of the string.
    "$"      Matches the end of the string or just before the newline at
             the end of the string.
    "*"      Matches 0 or more (greedy) repetitions of the preceding RE.
             Greedy means that it will match as many repetitions as possibl
e.
    "+"      Matches 1 or more (greedy) repetitions of the preceding RE.
    "?"      Matches 0 or 1 (greedy) of the preceding RE.
    *?,+?,?? Non-greedy versions of the previous three special characters.
    {m,n}    Matches from m to n repetitions of the preceding RE.
    {m,n}?   Non-greedy version of the above.
    "\\"     Either escapes special characters or signals a special sequenc
e.
    []       Indicates a set of characters.
             A "^" as the first character indicates a complementing set.
    "|"      A|B, creates an RE that will match either A or B.
    (...)    Matches the RE inside the parentheses.
             The contents can be retrieved or matched later in the string.
    (?aiLmsux) The letters set the corresponding flags defined below.
    (?:...)  Non-grouping version of regular parentheses.
    (?P<name>...) The substring matched by the group is accessible by name.
    (?P=name)     Matches the text matched earlier by the group named name.
    (?#...)  A comment; ignored.
    (?=...)  Matches if ... matches next, but doesn't consume the string.
    (?!...)  Matches if ... doesn't match next.
    (?<=...) Matches if preceded by ... (must be fixed length).
    (?<!...) Matches if not preceded by ... (must be fixed length).
    (?(id/name)yes|no) Matches yes pattern if the group with id/name matche
d,
                       the (optional) no pattern otherwise.

The special sequences consist of "\\" and a character from the list
below.  If the ordinary character is not on the list, then the
resulting RE will match the second character.
    \number  Matches the contents of the group of the same number.
    \A       Matches only at the start of the string.
    \Z       Matches only at the end of the string.
    \b       Matches the empty string, but only at the start or end of a wor
d.
    \B       Matches the empty string, but not at the start or end of a wor
d.
    \d       Matches any decimal digit; equivalent to the set [0-9] in
```

```
            bytes patterns or string patterns with the ASCII flag.
            In string patterns without the ASCII flag, it will match the wh
ole
            range of Unicode digits.
    \D      Matches any non-digit character; equivalent to [^\d].
    \s      Matches any whitespace character; equivalent to [ \t\n\r\f\v] i
n
            bytes patterns or string patterns with the ASCII flag.
            In string patterns without the ASCII flag, it will match the wh
ole
            range of Unicode whitespace characters.
    \S      Matches any non-whitespace character; equivalent to [^\s].
    \w      Matches any alphanumeric character; equivalent to [a-zA-Z0-9_]
            in bytes patterns or string patterns with the ASCII flag.
            In string patterns without the ASCII flag, it will match the
            range of Unicode alphanumeric characters (letters plus digits
            plus underscore).
            With LOCALE, it will match the set [0-9_] plus characters defin
ed
            as letters for the current locale.
    \W      Matches the complement of \w.
    \\      Matches a literal backslash.

This module exports the following functions:
    match     Match a regular expression pattern to the beginning of a strin
g.
    fullmatch Match a regular expression pattern to all of a string.
    search    Search a string for the presence of a pattern.
    sub       Substitute occurrences of a pattern found in a string.
    subn      Same as sub, but also return the number of substitutions made.
    split     Split a string by the occurrences of a pattern.
    findall   Find all occurrences of a pattern in a string.
    finditer  Return an iterator yielding a Match object for each match.
    compile   Compile a pattern into a Pattern object.
    purge     Clear the regular expression cache.
    escape    Backslash all non-alphanumerics in a string.

Each function other than purge and escape can take an optional 'flags' argum
ent
consisting of one or more of the following module constants, joined by "|".
A, L, and U are mutually exclusive.
    A  ASCII       For string patterns, make \w, \W, \b, \B, \d, \D
                   match the corresponding ASCII character categories
                   (rather than the whole Unicode categories, which is the
                   default).
                   For bytes patterns, this flag is the only available
                   behaviour and needn't be specified.
    I  IGNORECASE  Perform case-insensitive matching.
    L  LOCALE      Make \w, \W, \b, \B, dependent on the current locale.
    M  MULTILINE   "^" matches the beginning of lines (after a newline)
                   as well as the string.
                   "$" matches the end of lines (before a newline) as well
                   as the end of the string.
    S  DOTALL      "." matches any character at all, including the newline.
    X  VERBOSE     Ignore whitespace and comments for nicer looking RE's.
    U  UNICODE     For compatibility only. Ignored for string patterns (it
                   is the default), and forbidden for bytes patterns.

This module also defines an exception 'error'.
```

Character(s)	What it does
.	A period. Matches any single character except the newline character.
^	A caret. Matches a pattern at the start of the string.
\A	Uppercase A. Matches only at the start of the string.
$	Dollar sign. Matches the end of the string.
\Z	Uppercase Z. Matches only at the end of the string.
[ ]	Matches the set of characters you specify within it.
\	· If the character following the backslash is a recognized escape character, then the special meaning of the term is taken. · Else the backslash () is treated like any other character and passed through. · It can be used in front of all the metacharacters to remove their special meaning.
\w	Lowercase w. Matches any single letter, digit, or underscore.
\W	Uppercase W. Matches any character not part of `\w` (lowercase w).
\s	Lowercase s. Matches a single whitespace character like: space, newline, tab, return.
\S	Uppercase S. Matches any character not part of `\s` (lowercase s).
\d	Lowercase d. Matches decimal digit 0-9.
\D	Uppercase D. Matches any character that is not a decimal digit.
\t	Lowercase t. Matches tab.
\n	Lowercase n. Matches newline.
\r	Lowercase r. Matches return.
\b	Lowercase b. Matches only the beginning or end of the word.
+	Checks if the preceding character appears one or more times.
*	Checks if the preceding character appears zero or more times.
?	· Checks if the preceding character appears exactly zero or one time. · Specifies a non-greedy version of +, *
{ }	Checks for an explicit number of times.
( )	Creates a group when performing matches.
< >	Creates a named group when performing matches.

syntax

- method(pattern, string)

In [16]:

```
1 print(re.findall('P', s))
```

```
['P', 'P', 'P', 'P', 'P', 'P']
```

In [17]:

```
print(re.findall('P..', s))
```

```
['Pyt', 'Pyt', 'Pyt', 'Pre', 'Pyt', 'Par']
```

In [18]:

```
print(re.findall('a..', s))
```

```
['an ', 'al-', 'amm', 'ang', 'age', 'asi', 'ada', 'abl', 'ant', 'ati', 'ar
e', 'ati', 'abl', 'ase', 'ay ', 'ays', 'ago', 'ase', 'ay ', 'ays', 'ago', 'a
mi', 'adu', 'al ', 'app', 'are', 'ary', 'ars', 'ago', 'ara', 'ara', 'al ',
'ati', 'al,']
```

In [20]:

```
print(s)
```

```
Python is an interpreted high-level general-purpose programming language. Py
thon's design philosophy emphasizes code readability with its notable use of
significant indentation. Wikipedia
Developer: Python Software Foundation
Stable release: 3.9.5 / 3 May 2021; 29 days ago
Preview release: 3.10.0b1 / 3 May 2021; 29 days ago
Typing discipline: Duck, dynamic, strong typing; gradual (since 3.5, but ign
ored in CPython)
First appeared: February 1991; 30 years ago
Paradigm: Multi-paradigm: object-oriented, procedural (imperative), function
al, structured, reflective
```

In [22]:

```
print(re.findall('^P..', s))
```

```
['Pyt']
```

In [23]:

```
print(re.findall('\A..', s))
```

```
['Py']
```

In [27]:

```
print(re.findall('..e$', s))
```

```
['ive']
```

In [28]:

```
print(re.findall('....e$', s))
```

```
['ctive']
```

In [32]:

```
print(re.findall('....E$', s))
```

```
[]
```

In [29]:

```
s
```

Out[29]:

```
"Python is an interpreted high-level general-purpose programming language. P
ython's design philosophy emphasizes code readability with its notable use o
f significant indentation. Wikipedia\nDeveloper: Python Software Foundation
\nStable release: 3.9.5 / 3 May 2021; 29 days ago\nPreview release: 3.10.0b1
/ 3 May 2021; 29 days ago\nTyping discipline: Duck, dynamic, strong typing;
gradual (since 3.5, but ignored in CPython)\nFirst appeared: February 1991;
30 years ago\nParadigm: Multi-paradigm: object-oriented, procedural (imperat
ive), functional, structured, reflective"
```

In [31]:

```
st = 'kjbdvkjsdbvkjbvkbZ'


print(re.findall('....\Z', st))
```

```
['vkbZ']
```

In [33]:

```
print(re.findall('...', s))
```

```
['Pyt', 'hon', ' is', ' an', ' in', 'ter', 'pre', 'ted', ' hi', 'gh-', 'le
v', 'el ', 'gen', 'era', 'l-p', 'urp', 'ose', ' pr', 'ogr', 'amm', 'ing', '
la', 'ngu', 'age', '. P', 'yth', "on'", 's d', 'esi', 'gn ', 'phi', 'los',
'oph', 'y e', 'mph', 'asi', 'zes', ' co', 'de ', 'rea', 'dab', 'ili', 'ty ',
'wit', 'h i', 'ts ', 'not', 'abl', 'e u', 'se ', 'of ', 'sig', 'nif', 'ica',
'nt ', 'ind', 'ent', 'ati', 'on.', ' Wi', 'kip', 'edi', 'Dev', 'elo', 'per',
': P', 'yth', 'on ', 'Sof', 'twa', 're ', 'Fou', 'nda', 'tio', 'Sta', 'ble',
' re', 'lea', 'se:', ' 3.', '9.5', ' / ', '3 M', 'ay ', '202', '1; ', '29 ',
'day', 's a', 'Pre', 'vie', 'w r', 'ele', 'ase', ': 3', '.10', '.0b', '1 /',
' 3 ', 'May', ' 20', '21;', ' 29', ' da', 'ys ', 'ago', 'Typ', 'ing', ' di',
'sci', 'pli', 'ne:', ' Du', 'ck,', ' dy', 'nam', 'ic,', ' st', 'ron', 'g t',
'ypi', 'ng;', ' gr', 'adu', 'al ', '(si', 'nce', ' 3.', '5, ', 'but', ' ig',
'nor', 'ed ', 'in ', 'CPy', 'tho', 'Fir', 'st ', 'app', 'ear', 'ed:', ' Fe',
'bru', 'ary', ' 19', '91;', ' 30', ' ye', 'ars', ' ag', 'Par', 'adi', 'gm:',
' Mu', 'lti', '-pa', 'rad', 'igm', ': o', 'bje', 'ct-', 'ori', 'ent', 'ed,',
' pr', 'oce', 'dur', 'al ', '(im', 'per', 'ati', 've)', ', f', 'unc', 'tio',
'nal', ', s', 'tru', 'ctu', 'red', ', r', 'efl', 'ect', 'ive']
```

In [35]:

```
1 print(re.findall('..[a-z]', s))
```

```
['Pyt', 'hon', ' is', ' an', ' in', 'ter', 'pre', 'ted', ' hi', 'h-l', 'ev
e', 'l g', 'ene', 'ral', '-pu', 'rpo', 'e p', 'rog', 'ram', 'min', 'g l', 'a
ng', 'uag', ' Py', 'tho', "n's", ' de', 'sig', 'n p', 'hil', 'oso', 'phy', '
em', 'pha', 'siz', 's c', 'ode', ' re', 'ada', 'bil', 'ity', ' wi', 'h i',
's n', 'ota', 'ble', ' us', 'e o', 'f s', 'ign', 'ifi', 'can', 't i', 'nde',
'nta', 'tio', ' Wi', 'kip', 'edi', 'Dev', 'elo', 'per', ' Py', 'tho', ' So',
'ftw', 'are', ' Fo', 'und', 'ati', 'Sta', 'ble', ' re', 'lea', ' Ma', '9 d',
'ays', ' ag', 'Pre', 'vie', 'w r', 'ele', 'ase', '.0b', ' Ma', '9 d', 'ays',
' ag', 'Typ', 'ing', ' di', 'sci', 'pli', ' Du', ', d', 'yna', 'mic', ', s',
'tro', 'g t', 'ypi', '; g', 'rad', 'ual', ' (s', 'inc', ', b', 't i', 'gno',
'red', ' in', 'CPy', 'tho', 'Fir', 't a', 'ppe', 'are', ' Fe', 'bru', 'ary',
'0 y', 'ear', 's a', 'Par', 'adi', ' Mu', 'lti', '-pa', 'rad', 'igm', ': o',
'bje', 't-o', 'rie', 'nte', ', p', 'roc', 'edu', 'ral', ' (i', 'mpe', 'rat',
'ive', ', f', 'unc', 'tio', 'nal', ', s', 'tru', 'ctu', 'red', ', r', 'efl',
'ect', 'ive']
```

In [36]:

```
1 print(re.findall('[a-z].[a-z]', s))
```

```
['yth', 'n i', 's a', 'n i', 'nte', 'rpr', 'ete', 'd h', 'igh', 'lev', 'l
g', 'ene', 'ral', 'pur', 'pos', 'e p', 'rog', 'ram', 'min', 'g l', 'ang', 'u
ag', 'yth', "n's", 'des', 'ign', 'phi', 'los', 'oph', 'y e', 'mph', 'asi',
'zes', 'cod', 'e r', 'ead', 'abi', 'lit', 'y w', 'ith', 'its', 'not', 'abl',
'e u', 'e o', 'f s', 'ign', 'ifi', 'can', 't i', 'nde', 'nta', 'tio', 'iki',
'ped', 'eve', 'lop', 'yth', 'oft', 'war', 'oun', 'dat', 'ion', 'tab', 'e r',
'ele', 'ase', 'day', 's a', 'rev', 'iew', 'rel', 'eas', 'day', 's a', 'ypi',
'g d', 'isc', 'ipl', 'ine', 'uck', 'dyn', 'ami', 'str', 'ong', 'typ', 'ing',
'gra', 'dua', 'sin', 'but', 'ign', 'ore', 'd i', 'yth', 'irs', 't a', 'ppe',
'are', 'ebr', 'uar', 'yea', 's a', 'ara', 'dig', 'ult', 'i-p', 'ara', 'dig',
'obj', 'ect', 'ori', 'ent', 'pro', 'ced', 'ura', 'imp', 'era', 'tiv', 'fun',
'cti', 'ona', 'str', 'uct', 'ure', 'ref', 'lec', 'tiv']
```

In [37]:

```
1 print(re.findall('[A-Za-z].[a-z]', s))
```

```
['Pyt', 'hon', 's a', 'n i', 'nte', 'rpr', 'ete', 'd h', 'igh', 'lev', 'l
g', 'ene', 'ral', 'pur', 'pos', 'e p', 'rog', 'ram', 'min', 'g l', 'ang', 'u
ag', 'Pyt', 'hon', 's d', 'esi', 'n p', 'hil', 'oso', 'phy', 'emp', 'has',
'ize', 's c', 'ode', 'rea', 'dab', 'ili', 'y w', 'ith', 'its', 'not', 'abl',
'e u', 'e o', 'f s', 'ign', 'ifi', 'can', 't i', 'nde', 'nta', 'tio', 'Wik',
'ipe', 'dia', 'Dev', 'elo', 'per', 'Pyt', 'hon', 'Sof', 'twa', 'Fou', 'nda',
'tio', 'Sta', 'ble', 'rel', 'eas', 'May', 'day', 's a', 'Pre', 'vie', 'w r',
'ele', 'ase', 'May', 'day', 's a', 'Typ', 'ing', 'dis', 'cip', 'lin', 'Duc',
'dyn', 'ami', 'str', 'ong', 'typ', 'ing', 'gra', 'dua', 'sin', 'but', 'ign',
'ore', 'd i', 'CPy', 'tho', 'Fir', 't a', 'ppe', 'are', 'Feb', 'rua', 'yea',
's a', 'Par', 'adi', 'Mul', 'i-p', 'ara', 'dig', 'obj', 'ect', 'ori', 'ent',
'pro', 'ced', 'ura', 'imp', 'era', 'tiv', 'fun', 'cti', 'ona', 'str', 'uct',
'ure', 'ref', 'lec', 'tiv']
```

In [38]:

```
print(re.findall('[A-Za-z][a-z][a-z]', s))
```

```
['Pyt', 'hon', 'int', 'erp', 'ret', 'hig', 'lev', 'gen', 'era', 'pur', 'po
s', 'pro', 'gra', 'mmi', 'lan', 'gua', 'Pyt', 'hon', 'des', 'ign', 'phi', 'l
os', 'oph', 'emp', 'has', 'ize', 'cod', 'rea', 'dab', 'ili', 'wit', 'its',
'not', 'abl', 'use', 'sig', 'nif', 'ica', 'ind', 'ent', 'ati', 'Wik', 'ipe',
'dia', 'Dev', 'elo', 'per', 'Pyt', 'hon', 'Sof', 'twa', 'Fou', 'nda', 'tio',
'Sta', 'ble', 'rel', 'eas', 'May', 'day', 'ago', 'Pre', 'vie', 'rel', 'eas',
'May', 'day', 'ago', 'Typ', 'ing', 'dis', 'cip', 'lin', 'Duc', 'dyn', 'ami',
'str', 'ong', 'typ', 'ing', 'gra', 'dua', 'sin', 'but', 'ign', 'ore', 'Pyt',
'hon', 'Fir', 'app', 'ear', 'Feb', 'rua', 'yea', 'ago', 'Par', 'adi', 'Mul',
'par', 'adi', 'obj', 'ect', 'ori', 'ent', 'pro', 'ced', 'ura', 'imp', 'era',
'tiv', 'fun', 'cti', 'ona', 'str', 'uct', 'ure', 'ref', 'lec', 'tiv']
```

In [39]:

```
print(re.findall('[A-Z][a-z][a-z]', s))
```

```
['Pyt', 'Pyt', 'Wik', 'Dev', 'Pyt', 'Sof', 'Fou', 'Sta', 'May', 'Pre', 'Ma
y', 'Typ', 'Duc', 'Pyt', 'Fir', 'Feb', 'Par', 'Mul']
```

In [40]:

```
d = 'Today I recieved 10$ from my boss'
```

In [46]:

```
print(re.findall('..\$', d))
```

```
['10$']
```

In [47]:

```
print(re.findall('..$', d))
```

```
['ss']
```

In [48]:

```
print(re.findall('..e$', d))
```

```
[]
```

In [49]:

```
1 print(re.findall('.', s))
```

```
['P', 'y', 't', 'h', 'o', 'n', ' ', 'i', 's', ' ', 'a', 'n', ' ', 'i', 'n',
't', 'e', 'r', 'p', 'r', 'e', 't', 'e', 'd', ' ', 'h', 'i', 'g', 'h', '-',
'l', 'e', 'v', 'e', 'l', ' ', 'g', 'e', 'n', 'e', 'r', 'a', 'l', '-', 'p',
'u', 'r', 'p', 'o', 's', 'e', ' ', 'p', 'r', 'o', 'g', 'r', 'a', 'm', 'm',
'i', 'n', 'g', ' ', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e', '.', ' ', 'P',
'y', 't', 'h', 'o', 'n', '"', 's', ' ', 'd', 'e', 's', 'i', 'g', 'n', ' ',
'p', 'h', 'i', 'l', 'o', 's', 'o', 'p', 'h', 'y', ' ', 'e', 'm', 'p', 'h',
'a', 's', 'i', 'z', 'e', 's', ' ', 'c', 'o', 'd', 'e', ' ', 'r', 'e', 'a',
'd', 'a', 'b', 'i', 'l', 'i', 't', 'y', ' ', 'w', 'i', 't', 'h', ' ', 'i',
't', 's', ' ', 'n', 'o', 't', 'a', 'b', 'l', 'e', ' ', 'u', 's', 'e', ' ',
'o', 'f', ' ', 's', 'i', 'g', 'n', 'i', 'f', 'i', 'c', 'a', 'n', 't', ' ',
'i', 'n', 'd', 'e', 'n', 't', 'a', 't', 'i', 'o', 'n', '.', ' ', 'W', 'i',
'k', 'i', 'p', 'e', 'd', 'i', 'a', 'D', 'e', 'v', 'e', 'l', 'o', 'p', 'e',
'r', ':', ' ', 'P', 'y', 't', 'h', 'o', 'n', ' ', 'S', 'o', 'f', 't', 'w',
'a', 'r', 'e', ' ', 'F', 'o', 'u', 'n', 'd', 'a', 't', 'i', 'o', 'n', 'S',
't', 'a', 'b', 'l', 'e', ' ', 'r', 'e', 'l', 'e', 'a', 's', 'e', ':', ' ',
'3', '.', '9', '.', '5', ' ', '/', ' ', '3', ' ', 'M', 'a', 'y', ' ', '2',
'0', '2', '1', ';', ' ', '2', '9', ' ', 'd', 'a', 'y', 's', ' ', 'a', 'g',
'o', 'P', 'r', 'e', 'v', 'i', 'e', 'w', ' ', 'r', 'e', 'l', 'e', 'a', 's',
'e', ':', ' ', '3', '.', '1', '0', '.', '0', 'b', '1', ' ', '/', ' ', '3', '
', 'M', 'a', 'y', ' ', '2', '0', '2', '1', ';', ' ', '2', '9', ' ', 'd',
'a', 'y', 's', ' ', 'a', 'g', 'o', 'T', 'y', 'p', 'i', 'n', 'g', ' ', 'd',
'i', 's', 'c', 'i', 'p', 'l', 'i', 'n', 'e', ':', ' ', 'D', 'u', 'c', 'k',
',', ' ', 'd', 'y', 'n', 'a', 'm', 'i', 'c', ',', ' ', 's', 't', 'r', 'o',
'n', 'g', ' ', 't', 'y', 'p', 'i', 'n', 'g', ';', ' ', 'g', 'r', 'a', 'd',
'u', 'a', 'l', ' ', '(', 's', 'i', 'n', 'c', 'e', ' ', '3', '.', '5', ',', '
', 'b', 'u', 't', ' ', 'i', 'g', 'n', 'o', 'r', 'e', 'd', ' ', 'i', 'n', '
', 'C', 'P', 'y', 't', 'h', 'o', 'n', ')', 'F', 'i', 'r', 's', 't', ' ',
'a', 'p', 'p', 'e', 'a', 'r', 'e', 'd', ':', ' ', 'F', 'e', 'b', 'r', 'u',
'a', 'r', 'y', ' ', '1', '9', '9', '1', ';', ' ', '3', '0', ' ', 'y', 'e',
'a', 'r', 's', ' ', 'a', 'g', 'o', 'P', 'a', 'r', 'a', 'd', 'i', 'g', 'm',
':', ' ', 'M', 'u', 'l', 't', 'i', '-', 'p', 'a', 'r', 'a', 'd', 'i', 'g',
'm', ':', ' ', 'o', 'b', 'j', 'e', 'c', 't', '-', 'o', 'r', 'i', 'e', 'n',
't', 'e', 'd', ',', ' ', 'p', 'r', 'o', 'c', 'e', 'd', 'u', 'r', 'a', 'l', '
', '(', 'i', 'm', 'p', 'e', 'r', 'a', 't', 'i', 'v', 'e', ')', ',', ' ',
'f', 'u', 'n', 'c', 't', 'i', 'o', 'n', 'a', 'l', ',', ' ', 's', 't', 'r',
'u', 'c', 't', 'u', 'r', 'e', 'd', ',', ' ', 'r', 'e', 'f', 'l', 'e', 'c',
't', 'i', 'v', 'e']
```

In [50]:

```
1 print(re.findall('\.', s))
```

```
['.', '.', '.', '.', '.', '.', '.']
```

In [51]:

```
1 print(re.findall('\w', s))
```

```
['P', 'y', 't', 'h', 'o', 'n', 'i', 's', 'a', 'n', 'i', 'n', 't', 'e', 'r',
'p', 'r', 'e', 't', 'e', 'd', 'h', 'i', 'g', 'h', 'l', 'e', 'v', 'e', 'l',
'g', 'e', 'n', 'e', 'r', 'a', 'l', 'p', 'u', 'r', 'p', 'o', 's', 'e', 'p',
'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g', 'l', 'a', 'n', 'g', 'u',
'a', 'g', 'e', 'P', 'y', 't', 'h', 'o', 'n', 's', 'd', 'e', 's', 'i', 'g',
'n', 'p', 'h', 'i', 'l', 'o', 's', 'o', 'p', 'h', 'y', 'e', 'm', 'p', 'h',
'a', 's', 'i', 'z', 'e', 's', 'c', 'o', 'd', 'e', 'r', 'e', 'a', 'd', 'a',
'b', 'i', 'l', 'i', 't', 'y', 'w', 'i', 't', 'h', 'i', 't', 's', 'n', 'o',
't', 'a', 'b', 'l', 'e', 'u', 's', 'e', 'o', 'f', 's', 'i', 'g', 'n', 'i',
'f', 'i', 'c', 'a', 'n', 't', 'i', 'n', 'd', 'e', 'n', 't', 'a', 't', 'i',
'o', 'n', 'W', 'i', 'k', 'i', 'p', 'e', 'd', 'i', 'a', 'D', 'e', 'v', 'e',
'l', 'o', 'p', 'e', 'r', 'P', 'y', 't', 'h', 'o', 'n', 'S', 'o', 'f', 't',
'w', 'a', 'r', 'e', 'F', 'o', 'u', 'n', 'd', 'a', 't', 'i', 'o', 'n', 'S',
't', 'a', 'b', 'l', 'e', 'r', 'e', 'l', 'e', 'a', 's', 'e', '3', '9', '5',
'3', 'M', 'a', 'y', '2', '0', '2', '1', '2', '9', 'd', 'a', 'y', 's', 'a',
'g', 'o', 'P', 'r', 'e', 'v', 'i', 'e', 'w', 'r', 'e', 'l', 'e', 'a', 's',
'e', '3', '1', '0', '0', 'b', '1', '3', 'M', 'a', 'y', '2', '0', '2', '1',
'2', '9', 'd', 'a', 'y', 's', 'a', 'g', 'o', 'T', 'y', 'p', 'i', 'n', 'g',
'd', 'i', 's', 'c', 'i', 'p', 'l', 'i', 'n', 'e', 'D', 'u', 'c', 'k', 'd',
'y', 'n', 'a', 'm', 'i', 'c', 's', 't', 'r', 'o', 'n', 'g', 't', 'y', 'p',
'i', 'n', 'g', 'g', 'r', 'a', 'd', 'u', 'a', 'l', 's', 'i', 'n', 'c', 'e',
'3', '5', 'b', 'u', 't', 'i', 'g', 'n', 'o', 'r', 'e', 'd', 'i', 'n', 'C',
'P', 'y', 't', 'h', 'o', 'n', 'F', 'i', 'r', 's', 't', 'a', 'p', 'p', 'e',
'a', 'r', 'e', 'd', 'F', 'e', 'b', 'r', 'u', 'a', 'r', 'y', '1', '9', '9',
'1', '3', '0', 'y', 'e', 'a', 'r', 's', 'a', 'g', 'o', 'P', 'a', 'r', 'a',
'd', 'i', 'g', 'm', 'M', 'u', 'l', 't', 'i', 'p', 'a', 'r', 'a', 'd', 'i',
'g', 'm', 'o', 'b', 'j', 'e', 'c', 't', 'o', 'r', 'i', 'e', 'n', 't', 'e',
'd', 'p', 'r', 'o', 'c', 'e', 'd', 'u', 'r', 'a', 'l', 'i', 'm', 'p', 'e',
'r', 'a', 't', 'i', 'v', 'e', 'f', 'u', 'n', 'c', 't', 'i', 'o', 'n', 'a',
'l', 's', 't', 'r', 'u', 'c', 't', 'u', 'r', 'e', 'd', 'r', 'e', 'f', 'l',
'e', 'c', 't', 'i', 'v', 'e']
```

In [52]:

```
1 print(re.findall('\W', s))
```

```
[' ', ' ', ' ', ' ', '-', ' ', '-', ' ', ' ', '.', ' ', "'", ' ', ' ', ' ',
' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '.', ' ', '\n', ':', ' ', ' ',
' ', '\n', ' ', ':', ' ', '.', '.', ' ', '/', ' ', ' ', ' ', ';', ' ', ' ',
' ', '\n', ' ', ':', ' ', '.', '.', ' ', '/', ' ', ' ', ' ', ';', ' ', ' ',
' ', '\n', ' ', ':', ' ', ',', ' ', ',', ' ', ' ', ';', ' ', ' ', '(', ' ',
'.', ',', ' ', ' ', ' ', ' ', ')', '\n', ' ', ':', ' ', ' ', ';', ' ', ' ',
' ', '\n', ':', ' ', '-', ':', ' ', '-', ',', ' ', ' ', '(', ')', ',', ' ',
',', ' ', ',', ' ']
```

In [53]:

```
1 print(re.findall('\d', s))
```

```
['3', '9', '5', '3', '2', '0', '2', '1', '2', '9', '3', '1', '0', '0', '1',
'3', '2', '0', '2', '1', '2', '9', '3', '5', '1', '9', '9', '1', '3', '0']
```

In [54]:

```
1 print(re.findall('\D', s))
```

```
['P', 'y', 't', 'h', 'o', 'n', ' ', 'i', 's', ' ', 'a', 'n', ' ', 'i', 'n',
't', 'e', 'r', 'p', 'r', 'e', 't', 'e', 'd', ' ', 'h', 'i', 'g', 'h', '-',
'l', 'e', 'v', 'e', 'l', ' ', 'g', 'e', 'n', 'e', 'r', 'a', 'l', '-', 'p',
'u', 'r', 'p', 'o', 's', 'e', ' ', 'p', 'r', 'o', 'g', 'r', 'a', 'm', 'm',
'i', 'n', 'g', ' ', 'l', 'a', 'n', 'g', 'u', 'a', 'g', 'e', '.', ' ', 'P',
'y', 't', 'h', 'o', 'n', '"', 's', ' ', 'd', 'e', 's', 'i', 'g', 'n', ' ',
'p', 'h', 'i', 'l', 'o', 's', 'o', 'p', 'h', 'y', ' ', 'e', 'm', 'p', 'h',
'a', 's', 'i', 'z', 'e', 's', ' ', 'c', 'o', 'd', 'e', ' ', 'r', 'e', 'a',
'd', 'a', 'b', 'i', 'l', 'i', 't', 'y', ' ', 'w', 'i', 't', 'h', ' ', 'i',
't', 's', ' ', 'n', 'o', 't', 'a', 'b', 'l', 'e', ' ', 'u', 's', 'e', ' ',
'o', 'f', ' ', 's', 'i', 'g', 'n', 'i', 'f', 'i', 'c', 'a', 'n', 't', ' ',
'i', 'n', 'd', 'e', 'n', 't', 'a', 't', 'i', 'o', 'n', '.', ' ', 'W', 'i',
'k', 'i', 'p', 'e', 'd', 'i', 'a', '\n', 'D', 'e', 'v', 'e', 'l', 'o', 'p',
'e', 'r', ':', ' ', 'P', 'y', 't', 'h', 'o', 'n', ' ', 'S', 'o', 'f', 't',
'w', 'a', 'r', 'e', ' ', 'F', 'o', 'u', 'n', 'd', 'a', 't', 'i', 'o', 'n',
'\n', 'S', 't', 'a', 'b', 'l', 'e', ' ', 'r', 'e', 'l', 'e', 'a', 's', 'e',
':', ' ', '.', '.', ' ', '/', ' ', ' ', 'M', 'a', 'y', ' ', ';', ' ', ' ',
'd', 'a', 'y', 's', ' ', 'a', 'g', 'o', '\n', 'P', 'r', 'e', 'v', 'i', 'e',
'w', ' ', 'r', 'e', 'l', 'e', 'a', 's', 'e', ':', ' ', '.', '.', 'b', ' ',
'/', ' ', ' ', 'M', 'a', 'y', ' ', ';', ' ', ' ', 'd', 'a', 'y', 's', ' ',
'a', 'g', 'o', '\n', 'T', 'y', 'p', 'i', 'n', 'g', ' ', 'd', 'i', 's', 'c',
'i', 'p', 'l', 'i', 'n', 'e', ':', ' ', 'D', 'u', 'c', 'k', ',', ' ', 'd',
'y', 'n', 'a', 'm', 'i', 'c', ',', ' ', 's', 't', 'r', 'o', 'n', 'g', ' ',
't', 'y', 'p', 'i', 'n', 'g', ';', ' ', 'g', 'r', 'a', 'd', 'u', 'a', 'l', '
', '(', 's', 'i', 'n', 'c', 'e', ' ', '.', ',', ' ', 'b', 'u', 't', ' ',
'i', 'g', 'n', 'o', 'r', 'e', 'd', ' ', 'i', 'n', ' ', 'C', 'P', 'y', 't',
'h', 'o', 'n', ')', '\n', 'F', 'i', 'r', 's', 't', ' ', 'a', 'p', 'p', 'e',
'a', 'r', 'e', 'd', ':', ' ', 'F', 'e', 'b', 'r', 'u', 'a', 'r', 'y', ' ',
';', ' ', ' ', 'y', 'e', 'a', 'r', 's', ' ', 'a', 'g', 'o', '\n', 'P', 'a',
'r', 'a', 'd', 'i', 'g', 'm', ':', ' ', 'M', 'u', 'l', 't', 'i', '-', 'p',
'a', 'r', 'a', 'd', 'i', 'g', 'm', ':', ' ', 'o', 'b', 'j', 'e', 'c', 't',
'-', 'o', 'r', 'i', 'e', 'n', 't', 'e', 'd', ',', ' ', 'p', 'r', 'o', 'c',
'e', 'd', 'u', 'r', 'a', 'l', ' ', '(', 'i', 'm', 'p', 'e', 'r', 'a', 't',
'i', 'v', 'e', ')', ',', ' ', 'f', 'u', 'n', 'c', 't', 'i', 'o', 'n', 'a',
'l', ',', ' ', 's', 't', 'r', 'u', 'c', 't', 'u', 'r', 'e', 'd', ',', ' ',
'r', 'e', 'f', 'l', 'e', 'c', 't', 'i', 'v', 'e']
```

In [55]:

```
1 print(re.findall('\d.', s))
```

```
['3.', '9.', '5 ', '3 ', '20', '21', '29', '3.', '10', '0b', '1 ', '3 ', '2
0', '21', '29', '3.', '5,', '19', '91', '30']
```

In [56]:

```
1 print(re.findall('\d\d', s))
```

```
['20', '21', '29', '10', '20', '21', '29', '19', '91', '30']
```

In [57]:

```
print(re.findall('\d\d\d', s))
```

```
['202', '202', '199']
```

In [58]:

```
print(re.findall('\d\d\d\d', s))
```

```
['2021', '2021', '1991']
```

In [59]:

```
print(re.findall('\d\d\d\d\d', s))
```

```
[]
```

In [60]:

```
n = '4556'
n2 = '365'


print(re.findall('^\d\d\d$', n))
```

```
[]
```

In [61]:

```
print(re.findall('^\d\d\d$', n2))
```

```
['365']
```

In [62]:

```
print(re.findall('^\d\d\d$', '3o5'))
```

```
[]
```

In [64]:

```
print(re.findall('\s', s))
```

```
[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
 ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\n', ' ', ' ', ' ', ' ', '\n', ' ', ' ', ' ', ' ',
 ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\n', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
'\n', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', '\n', ' ',
 ' ', ' ', ' ', ' ', ' ', '\n', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']
```

In [65]:

```
1 print(re.findall('\S', s))
```

```
['P', 'y', 't', 'h', 'o', 'n', 'i', 's', 'a', 'n', 'i', 'n', 't', 'e', 'r',
'p', 'r', 'e', 't', 'e', 'd', 'h', 'i', 'g', 'h', '-', 'l', 'e', 'v', 'e',
'l', 'g', 'e', 'n', 'e', 'r', 'a', 'l', '-', 'p', 'u', 'r', 'p', 'o', 's',
'e', 'p', 'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g', 'l', 'a', 'n',
'g', 'u', 'a', 'g', 'e', '.', 'P', 'y', 't', 'h', 'o', 'n', "'", 's', 'd',
'e', 's', 'i', 'g', 'n', 'p', 'h', 'i', 'l', 'o', 's', 'o', 'p', 'h', 'y',
'e', 'm', 'p', 'h', 'a', 's', 'i', 'z', 'e', 's', 'c', 'o', 'd', 'e', 'r',
'e', 'a', 'd', 'a', 'b', 'i', 'l', 'i', 't', 'y', 'w', 'i', 't', 'h', 'i',
't', 's', 'n', 'o', 't', 'a', 'b', 'l', 'e', 'u', 's', 'e', 'o', 'f', 's',
'i', 'g', 'n', 'i', 'f', 'i', 'c', 'a', 'n', 't', 'i', 'n', 'd', 'e', 'n',
't', 'a', 't', 'i', 'o', 'n', '.', 'W', 'i', 'k', 'i', 'p', 'e', 'd', 'i',
'a', 'D', 'e', 'v', 'e', 'l', 'o', 'p', 'e', 'r', ':', 'P', 'y', 't', 'h',
'o', 'n', 'S', 'o', 'f', 't', 'w', 'a', 'r', 'e', 'F', 'o', 'u', 'n', 'd',
'a', 't', 'i', 'o', 'n', 'S', 't', 'a', 'b', 'l', 'e', 'r', 'e', 'l', 'e',
'a', 's', 'e', ':', '3', '.', '9', '.', '5', '/', '3', 'M', 'a', 'y', '2',
'0', '2', '1', ';', '2', '9', 'd', 'a', 'y', 's', 'a', 'g', 'o', 'P', 'r',
'e', 'v', 'i', 'e', 'w', 'r', 'e', 'l', 'e', 'a', 's', 'e', ':', '3', '.',
'1', '0', '.', '0', 'b', '1', '/', '3', 'M', 'a', 'y', '2', '0', '2', '1',
';', '2', '9', 'd', 'a', 'y', 's', 'a', 'g', 'o', 'T', 'y', 'p', 'i', 'n',
'g', 'd', 'i', 's', 'c', 'i', 'p', 'l', 'i', 'n', 'e', ':', 'D', 'u', 'c',
'k', ',', 'd', 'y', 'n', 'a', 'm', 'i', 'c', ',', 's', 't', 'r', 'o', 'n',
'g', 't', 'y', 'p', 'i', 'n', 'g', ';', 'g', 'r', 'a', 'd', 'u', 'a', 'l',
'(', 's', 'i', 'n', 'c', 'e', '3', '.', '5', ',', 'b', 'u', 't', 'i', 'g',
'n', 'o', 'r', 'e', 'd', 'i', 'n', 'C', 'P', 'y', 't', 'h', 'o', 'n', ')',
'F', 'i', 'r', 's', 't', 'a', 'p', 'p', 'e', 'a', 'r', 'e', 'd', ':', 'F',
'e', 'b', 'r', 'u', 'a', 'r', 'y', '1', '9', '9', '1', ';', '3', '0', 'y',
'e', 'a', 'r', 's', 'a', 'g', 'o', 'P', 'a', 'r', 'a', 'd', 'i', 'g', 'm',
':', 'M', 'u', 'l', 't', 'i', '-', 'p', 'a', 'r', 'a', 'd', 'i', 'g', 'm',
':', 'o', 'b', 'j', 'e', 'c', 't', '-', 'o', 'r', 'i', 'e', 'n', 't', 'e',
'd', ',', 'p', 'r', 'o', 'c', 'e', 'd', 'u', 'r', 'a', 'l', '(', 'i', 'm',
'p', 'e', 'r', 'a', 't', 'i', 'v', 'e', ')', ',', 'f', 'u', 'n', 'c', 't',
'i', 'o', 'n', 'a', 'l', ',', 's', 't', 'r', 'u', 'c', 't', 'u', 'r', 'e',
'd', ',', 'r', 'e', 'f', 'l', 'e', 'c', 't', 'i', 'v', 'e']
```

In [66]:

```
1 d
```

Out[66]:

```
'Today I recieved 10$ from my boss'
```

In [71]:

```
1 print(re.findall('\b.', 'hello'))
```

```
[]
```

In [74]:

```python
print(re.findall('P*', s))
```

```
['P', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', 'P',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', 'P', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', 'P', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', 'P', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', 'P', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '',
'', '', '', '', '', '', '', '', '', '', '', '', '', '', '', '']
```

In [75]:

```python
print(re.findall('P\d*', s))
```

```
['P', 'P', 'P', 'P', 'P', 'P']
```

In [77]:

```python
email = 'anilkumar_t@apssdc.in'
```

In [79]:

```python
print(re.findall('[a-z]*', email))
```

```
['anilkumar', '', 't', '', 'apssdc', '', 'in', '']
```

In [80]:

```
print(re.findall('[a-z_]*', email))
```

```
['anilkumar_t', '', 'apssdc', '', 'in', '']
```

In [82]:

```
print(re.findall('[a-z]+', email))
```

```
['anilkumar', 't', 'apssdc', 'in']
```

In [83]:

```
print(re.findall('[a-z]?', email))
```

```
['a', 'n', 'i', 'l', 'k', 'u', 'm', 'a', 'r', '', 't', '', 'a', 'p', 's',
's', 'd', 'c', '', 'i', 'n', '']
```

In [84]:

```
print(re.findall('[a-z]?.', email))
```

```
['an', 'il', 'ku', 'ma', 'r_', 't@', 'ap', 'ss', 'dc', '.', 'in']
```

In [85]:

```
print(re.findall('[a-z]?[a-z]', email))
```

```
['an', 'il', 'ku', 'ma', 'r', 't', 'ap', 'ss', 'dc', 'in']
```

In [92]:

```
print(re.findall('[a-z]{2}', 'emmacs'))
```

```
['em', 'ma', 'cs']
```

In [94]:

```
print(re.findall('[a-z]{3}', 'emmacs'))
```

```
['emm', 'acs']
```

In [96]:

```
ip = '163.125.255.125'

ip2 = '172.160.250.5'
```

In [97]:

```
re.match('163.125.255.[0-2][0-5][0-5]', ip)
```

Out[97]:

```
<re.Match object; span=(0, 15), match='163.125.255.125'>
```

In [99]:

```
print(re.match('163.125.255.[0-2][0-5][0-5]', ip2))
```

```
None
```

In [100]:

```
obj = re.match('163.125.255.[0-2][0-5][0-5]', ip)


obj.span()
```

Out[100]:

```
(0, 15)
```

In [102]:

```
ip3 = '163.125.255.125opa'

re.match('163.125.255.[0-2][0-5][0-5]', ip3)
```

Out[102]:

```
<re.Match object; span=(0, 15), match='163.125.255.125'>
```

In [114]:

```
ip3 = '163.125.255.125opa'

print(re.match('^163.125.255.[0-2][0-5][0-5]$', ip3))
```

```
None
```

In [115]:

```
re.match('^163.125.255.[0-2][0-5][0-5]$', ip)
```

Out[115]:

```
<re.Match object; span=(0, 15), match='163.125.255.125'>
```

In [113]:

```
obj.string
```

Out[113]:

```
'163.125.255.125'
```

- +91
- 6789
- 10

In [117]:

```
m1 = '9876543210'
m2 = '3216549870'
m3 = '987654321o'
```

In [118]:

```
pattern = '^[6789]\d{9}$'


re.match(pattern, m1)
```

Out[118]:

```
<re.Match object; span=(0, 10), match='9876543210'>
```

In [119]:

```
print(re.match(pattern, m2), re.match(pattern, m3))
```

```
None None
```

In [120]:

```
m4 = '+919876543210'


pattern = '+91^[6789]\d{9}'
```

In [121]:

```
1 re.match(pattern, m4)
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
<ipython-input-121-48189cd206fb> in <module>
----> 1 re.match(pattern, m4)

~\anaconda3\lib\re.py in match(pattern, string, flags)
    189     """Try to apply the pattern at the start of the string, returnin
g
    190     a Match object, or None if no match was found."""
--> 191     return _compile(pattern, flags).match(string)
    192
    193 def fullmatch(pattern, string, flags=0):

~\anaconda3\lib\re.py in _compile(pattern, flags)
    302     if not sre_compile.isstring(pattern):
    303         raise TypeError("first argument must be string or compiled p
attern")
--> 304     p = sre_compile.compile(pattern, flags)
    305     if not (flags & DEBUG):
    306         if len(_cache) >= _MAXCACHE:

~\anaconda3\lib\sre_compile.py in compile(p, flags)
    762     if isstring(p):
    763         pattern = p
--> 764         p = sre_parse.parse(p, flags)
    765     else:
    766         pattern = None

~\anaconda3\lib\sre_parse.py in parse(str, flags, state)
    946
    947     try:
--> 948         p = _parse_sub(source, state, flags & SRE_FLAG_VERBOSE, 0)
    949     except Verbose:
    950         # the VERBOSE flag was switched on inside the pattern.  to b
e

~\anaconda3\lib\sre_parse.py in _parse_sub(source, state, verbose, nested)
    441     start = source.tell()
    442     while True:
--> 443         itemsappend(_parse(source, state, verbose, nested + 1,
    444                            not nested and not items))
    445         if not sourcematch("|"):

~\anaconda3\lib\sre_parse.py in _parse(source, state, verbose, nested, firs
t)
    666                 item = None
    667             if not item or item[0][0] is AT:
--> 668                 raise source.error("nothing to repeat",
    669                                    source.tell() - here + len(this))
    670             if item[0][0] in _REPEATCODES:

error: nothing to repeat at position 0
```

In [123]:

```
pattern = '^[+]91[6789]\d{9}'


re.match(pattern, m4)
```

Out[123]:

```
<re.Match object; span=(0, 13), match='+919876543210'>
```

In [124]:

```
m3 = '+9164578d9656'

print(re.match(pattern, m3))
```

```
None
```

task

[userName@domain.extension (mailto:userName@domain.extension)](mailto:userName@domain.extension)

- userName
  - a-zA-Z0-9._
  - it shouldn't startwith .
  - min 8 max 15
- @
- Domain
  - a-zA-Z0-9
  - min 2 max 10
- Extension
  - a-zA-Z
  - 2, 5

sub -> replace

syntax

sub(Pat, repla, string)

In [125]:

```
1 print(re.sub('\s', '-', s))
```

```
Python-is-an-interpreted-high-level-general-purpose-programming-language.-Py
thon's-design-philosophy-emphasizes-code-readability-with-its-notable-use-of
-significant-indentation.-Wikipedia-Developer:-Python-Software-Foundation-St
able-release:-3.9.5-/-3-May-2021;-29-days-ago-Preview-release:-3.10.0b1-/-3-
May-2021;-29-days-ago-Typing-discipline:-Duck,-dynamic,-strong-typing;-gradu
al-(since-3.5,-but-ignored-in-CPython)-First-appeared:-February-1991;-30-yea
rs-ago-Paradigm:-Multi-paradigm:-object-oriented,-procedural-(imperative),-f
unctional,-structured,-reflective
```

In [126]:

```
1 print(re.sub('\d', '0', s))
```

```
Python is an interpreted high-level general-purpose programming language. Py
thon's design philosophy emphasizes code readability with its notable use of
significant indentation. Wikipedia
Developer: Python Software Foundation
Stable release: 0.0.0 / 0 May 0000; 00 days ago
Preview release: 0.00.0b0 / 0 May 0000; 00 days ago
Typing discipline: Duck, dynamic, strong typing; gradual (since 0.0, but ign
ored in CPython)
First appeared: February 0000; 00 years ago
Paradigm: Multi-paradigm: object-oriented, procedural (imperative), function
al, structured, reflective
```

In [127]:

```
1 print(re.subn('\d', '0', s))
```

```
("Python is an interpreted high-level general-purpose programming language.
Python's design philosophy emphasizes code readability with its notable use
of significant indentation. Wikipedia\nDeveloper: Python Software Foundation
\nStable release: 0.0.0 / 0 May 0000; 00 days ago\nPreview release: 0.00.0b0
/ 0 May 0000; 00 days ago\nTyping discipline: Duck, dynamic, strong typing;
gradual (since 0.0, but ignored in CPython)\nFirst appeared: February 0000;
00 years ago\nParadigm: Multi-paradigm: object-oriented, procedural (imperat
ive), functional, structured, reflective", 30)
```

In [128]:

```
print(re.sub('\W', '***', s))
```

```
Python***is***an***interpreted***high***level***general***purpose***programm
ing***language******Python***s***design***philosophy***emphasizes***code***r
eadability***with***its***notable***use***of***significant***indentation****
**Wikipedia***Developer******Python***Software***Foundation***Stable***relea
se******3***9***5*********3***May***2021******29***days***ago***Preview***re
lease******3***10***0b1*********3***May***2021******29***days***ago***Typing
***discipline******Duck******dynamic******strong***typing******gradual******
since***3***5******but***ignored***in***CPython******First***appeared******F
ebruary***1991******30***years***ago***Paradigm******Multi***paradigm******o
bject***oriented******procedural******imperative*********functional******str
uctured******reflective
```

In [130]:

```
print(re.sub('\S', '*', s))
```

```
****** ** ** ********** ********** ********** ********** ********* **
****** ****** ********** ********** **** ********** **** *** ******* *** **
********** ********** *********
********** ****** ******** **********
****** ******** ***** * * *** ***** ** **** ***
******* ******** ******** * * *** ***** ** **** ***
****** ********** ***** ******** ****** ******* ******* ****** **** *** ***
**** ** ********
***** ********* ******** ***** ** ***** ***
********* ********** ********** ********** ********** ********
*** ********** **********
```

In [131]:

```
print(re.subn('\S', '*', s))
```

```
('****** ** ** ********** ********** ********** ********** *********
******** ****** ********** ********** **** ********** **** *** ******* ***
** ********** ********** *********\n********** ****** ******** **********
\n****** ******** ***** * * *** ***** ** **** ***\n******* ******** ********
* * *** ***** ** **** ***\n****** ********** ***** ******** ****** *******
******* ****** **** *** ******* ** ********\n***** ********* ******** *****
** ***** ***\n********* ********** ********** ********** ********* ********
***** ********** ********** **********', 492)
```

In [132]:

```
print(re.subn('[aeiou]', '*', s))
```

```
("Pyth*n *s *n *nt*rpr*t*d h*gh-l*v*l g*n*r*l-p*rp*s* pr*gr*mm*ng l*ng**g*.
Pyth*n's d*s*gn ph*l*s*phy *mph*s*z*s c*d* r**d*b*l*ty w*th *ts n*t*bl* *s*
*f s*gn*f*c*nt *nd*nt*t**n. W*k*p*d**\nD*v*l*p*r: Pyth*n S*ftw*r* F**nd*t**n
\nSt*bl* r*l**s*: 3.9.5 / 3 M*y 2021; 29 d*ys *g*\nPr*v**w r*l**s*: 3.10.0b1
/ 3 M*y 2021; 29 d*ys *g*\nTyp*ng d*sc*pl*n*: D*ck, dyn*m*c, str*ng typ*ng;
gr*d**l (s*nc* 3.5, b*t *gn*r*d *n CPyth*n)\nF*rst *pp**r*d: F*br**ry 1991;
30 y**rs *g*\nP*r*d*gm: M*lt*-p*r*d*gm: *bj*ct-*r**nt*d, pr*c*d*r*l (*mp*r*t
*v*), f*nct**n*l, str*ct*r*d, r*fl*ct*v*", 163)
```

In [134]:

```
print(re.subn('[^aeiou\s]', '*', s))
```

```
('****o* i* a* i**e***e*e* *i****e*e* *e*e*a***u**o*e **o**a**i** *a**ua*e*
****o*** *e*i** **i*o*o*** e***a*i*e* *o*e *ea*a*i*i** *i** i** *o*a**e u*e
o* *i**i*i*a** i**e**a*io** *i*i*e*ia\n*e*e*o*e** ****o* *o***a*e *ou**a*io*
\n**a**e *e*ea*e* ***** * * *a* ***** ** *a** a*o\n**e*ie* *e*ea*e* ********
* * *a* ***** ** *a** a*o\n***i** *i**i**i*e* *u*** ***a*i** ***o** ***i***
**a*ua* **i**e **** *u* i**o*e* i* *****o**\n*i*** a**ea*e** *e**ua** *****
** *ea** a*o\n*a*a*i*** *u**i**a*a*i*** o**e***o*ie**e** **o*e*u*a* *i**e*a*
i*e** *u***io*a** ***u**u*e** *e**e**i*e', 329)
```

In [135]:

```
d1 = 'abc'
d2 = 'a'
```

In [141]:

```
print(re.findall('[a-z][a-z]+', d2))
```

```
['abc']
```

In [142]:

```
print(re.findall('[a-z][a-z]+', d2))
```

```
[]
```

In [140]:

```
print(re.findall('[a-z][a-z]*', d2))
```

```
['a']
```

In [143]:

```
1 '{abs:val}'
```

Out[143]:

'{abs:val}'

In [145]:

```
1 print(re.subn('[^aeiou\S]','@',s))
```

("Python@is@an@interpreted@high-level@general-purpose@programming@language.@
Python's@design@philosophy@emphasizes@code@readability@with@its@notable@use@
of@significant@indentation.@Wikipedia@Developer:@Python@Software@Foundation@
Stable@release:@3.9.5@/@3@May@2021;@29@days@ago@Preview@release:@3.10.0b1@/@
3@May@2021;@29@days@ago@Typing@discipline:@Duck,@dynamic,@strong@typing;@gra
dual@(since@3.5,@but@ignored@in@CPython)@First@appeared:@February@1991;@30@y
ears@ago@Paradigm:@Multi-paradigm:@object-oriented,@procedural@(imperative),
@functional,@structured,@reflective", 73)

\s -> [^\S]