A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Basic Rules to create the pattern:

* Regex is a procedure in any language to look for specified patterns in a given text.
* “re” is the module to perform regex in Python. (Use import in scripts)
* There are different operations in re like:

Search,match,finditer,findall,split,sub etc

🡪

Re.search(pattern,text)

Re.match(pattern,text)

Re.finditer(pattern,text)

Re.findall(pattern,text)

* Pattern is a sequence of characters,which represent multiple strings
* Simple examples for pattern

“Python”

“Python[23]” 🡪 “Python2” “Python3”

* Findall

Import re

* print(re.findall(patten,text))
* a,x,9🡪 Ordinary characters that match themselves
* [abc] 🡪 a or b or c

A screenshot of a computer

AI-generated content may be incorrect.

i.

my\_pat="is"

text="This is a python and it is easy to learn"

print(re.findall(my\_pat,text))

# ['is', 'is', 'is']

2.

my\_pat="i[st]"

text="This is a python and it is easy to learn"

print(re.findall(my\_pat,text))

# ['is', 'is', 'it', 'is']

3.

# a x  9

my\_pat="a"

text="This is a python and it is easy to learn"

print(re.findall(my\_pat,text))

# single character:::::: ['a', 'a', 'a', 'a']

4.

# [abc] --> a or b or c

my\_pat="[$%^&]"

text="This is a python and $%^& it is easy to learn$%^&"

print(re.findall(my\_pat,text))

#['$', '%', '^', '&', '$', '%', '^', '&']

5.

## # [a-f] --> a or b or c

my\_pat="[a-f]"

text="This is a python and $%^& it is easy to learn$%^&"

print(re.findall(my\_pat,text))

# ['a', 'a', 'd', 'e', 'a', 'e', 'a']

6.

## # [a-f] --> a or b or c

# [a-zA-Z0-9]

my\_pat="[a-fA-Z0-9]"

text="This is a python and $%^& 0 2 3 4 5 it is easy to learn$%^&"

print(re.findall(my\_pat,text))

# ['T', 'a', 'a', 'd', '0', '2', '3', '4', '5', 'e', 'a', 'e', 'a']

7.

# # \w --> Any a-z,A-Z,0-9 \_ except space

my\_pat="\w"

text="This is a python and it is easy to learn"

print(re.findall(my\_pat,text))

# #['T', 'h', 'i', 's', 'i', 's', 'a', 'p', 'y', 't', 'h', 'o', 'n', 'a', 'n', 'd', 'i', 't', 'i', 's', 'e', 'a', 's', 'y', 't', 'o', 'l', 'e', 'a', 'r', 'n']

#### two \w\w

my\_pat="\w\w"

text="This is a python and it is easy to learn"

print(re.findall(my\_pat,text))

['Th', 'is', 'is', 'py', 'th', 'on', 'an', 'it', 'is', 'ea', 'sy', 'to', 'le', 'ar']

8.

my\_pat="\w\w\w"

text="This is a python and it is easy to learn \_ara $$$$$$$"

print(re.findall(my\_pat,text))

# ['Thi', 'pyt', 'hon', 'and', 'eas', 'lea', '\_ar']

9.

# #\W -- Matches any characters not part of \w

my\_pat="\W"

text="This is a python #$%^& \*&^%"

print(re.findall(my\_pat,text))

# # \w  ---> ['T', 'h', 'i', 's', 'i', 's', 'a', 'p', 'y', 't', 'h', 'o', 'n']

# # \W ---> [' ', ' ', ' ', ' ', '#', '$', '%', '^', '&', ' ', '\*', '&', '^', '%']

10.

# \d --Only numbers

# my\_pat="python\d"

my\_pat="\d\d"

text="This is a python1 python2 123456"

print(re.findall(my\_pat,text))

# "\d" --> ['1', '2', '3', '4', '5', '6', '7', '8']

# "python\d" ---> ['python1', 'python2']

# "\d\d"  --> ['12', '34', '56']

11.

# . matches single character except new line

# my\_pat="\."

# text="This is a python1. python2."

# print(re.findall(my\_pat,text))

# # . --> ['T', 'h', 'i', 's', ' ', 'i', 's', ' ', 'a', ' ', 'p', 'y', 't', 'h', 'o', 'n', '1']

# #  .. --> ['Th', 'is', ' i', 's ', 'a ', 'py', 'th', 'on']

# #  ...>  ['Thi', 's i', 's a', ' py', 'tho']

# # text="This is a python1. python2"

# # "\." --> ['.', '.']

12.

my\_pat="\d\d\d\.\d\d\d\.\d\d\d\.\d\d\d"

text="This is a python1. ip : 192.168.255.101 122 323 324 This is "

print(re.findall(my\_pat,text))

# "\d\d\d\.\d\d\d\.\d\d\d\.\d\d\d" --> ['192.168.255.101']

CLASS 3:

Rules to create a pattern –2:

A screenshot of a computer

AI-generated content may be incorrect.

^ 🡪 Start of the string ( and start of the line in-case of multiline string )

text="its is a python andlearn it is easy to learn "

my\_pat="^i[st]"

print(re.findall(my\_pat,text))

#"i[st]"---> ['it', 'is', 'it', 'is']

# "^i[st]" --> ['it']

OUTPUT: ['it']

* End of the String

Code 3:

# $ ---> End of the string ( and new line character in case multi string)

text="its is a python andlearn it is easy to learn"

my\_pat="learn$"

print(re.findall(my\_pat,text))

# ['learn']

##### \b

Code 4:

text="its is a python andlearn it learn is easy to learn"

my\_pat=r"\blearn"

print(re.findall(my\_pat,text))

# output:

#      ['learn', 'learn']

ii.

# text="its is a python andlearn it learnis easy to learn"

# my\_pat=r"\blearn\b"

#['learn']

# \B

# \B

text="its is a python andlearn it itlearnis easy to learn"

my\_pat=r"\Blearn\B"

print(re.findall(my\_pat,text))

#output:

# ['learn'] --> Withtou spaces "itlearnis"

* \t and \n:

my\_pat=r"\n"

text='  Hello    Hello1  \n   Hello2'

print(re.findall(my\_pat,text))

# ['\n']

Output:

iii. Rules to create pattern -3

A screenshot of a computer

AI-generated content may be incorrect.

{2} 🡪 Exactly two times

text="This is a pythonn and python3"

my\_pat=r"python"

print(re.findall(my\_pat,text))

# my\_pat=r"python" --> ['python', 'python']

OUTPUT:

['python', 'python']

text="This is a pythonnnn and python3"

my\_pat=r"python{4}"

print(re.findall(my\_pat,text))

OUTPUT:

['pythonnnn']

text="xaz xaaz xaaaz xaaaaz xaaaaaz"

print(re.findall(my\_pat,text))

Output:

# my\_pat=r"xa{2}" -> ['xaa', 'xaa', 'xaa', 'xaa'] 🡪 Atleast 2

# my\_pat=r"xa{3,5}"['xaaa', 'xaaaa', 'xaaaaa'] 🡪 3 to 5

# my\_pat=r"xa{2,}" --> ['xaa', 'xaaa', 'xaaaa', 'xaaaaa'] 🡪 2 to all

# my\_pat=r"xa+" ---> ['xa', 'xaa', 'xaaa', 'xaaaa', 'xaaaaa'] 🡪 “a” more

# r"xa\*" --> ['xa', 'xaa', 'xaaa', 'xaaaa', 'xaaaaa'] 🡪 one ore more

# my\_pat=r"xa?z" --> ['xaz'] 🡪 one or none

REGEX with Flags:

A screenshot of a computer

AI-generated content may be incorrect.

import re

text="This is a string this is a new starting THIS"

my\_pat=r'this'

print(re.findall(my\_pat,text)) # this

print(re.findall(my\_pat,text,re.I)) # ['This', 'this', 'THIS']

Multi lines

text="""this is a multiline END

This is also a find END

This also a end

this ia an end"""

my\_pat=r"this"

print(re.findall(my\_pat,text)) # ['this', 'this']

my\_pat=r"^this" # start of the string

print(re.findall(my\_pat,text)) #['this']

print(re.findall(my\_pat,text,re.M)) #['this', 'this']

print(re.findall(my\_pat,text,re.M|re.I)) # ['this', 'This', 'This', 'this']

my\_pat=r"end$"

print(re.findall(my\_pat,text,re.M|re.I)) #['END', 'END', 'end', 'end']

Working with search and Match

text="This is for python and there are two major vers python2 and python3 in future python4"

my\_pat=r"\bpython[23]?\b"

print(re.findall(my\_pat,text))

OUTPUT:

# ['python', 'python2', 'python3']

CODE 2:

text="This is for python2 and there are two major vers python2 and python3 in future python4"

match\_obj=re.search(my\_pat,text)

#print(match\_obj)

# <re.Match object; span=(12, 19), match='python2'>

if match\_obj:

    print(match\_obj.group()) # python2

    print(match\_obj.start()) #12

    print(match\_obj.end()-1) #18

    print("length:",match\_obj.end()-match\_obj.start()) # 7

else:

    print("No match found")

OUTPUT:

#python2

#12

#18

#length: 7

CODE 3:

MULTI LINE SEARCH

text="""This is for

python2 and there are two major vers

python2 and

python3 in future python4"""

my\_pat=r"\bpython[23]?\b"

match\_obj=re.search(my\_pat,text)

#print(match\_obj)

# <re.Match object; span=(12, 19), match='python2'>

if match\_obj:

    print(match\_obj.group()) # python2

    print(match\_obj.start()) #12

    print(match\_obj.end()-1) #18

    print("length:",match\_obj.end()-match\_obj.start()) # 7

else:

    print("No match found")

OUTPUT:

python2

12

18

length: 7

MATCH:

text="""This is for

python2 and there are two major vers

python2 and

python3 in future python4"""

pat=r"\bpython[23]?\b"

print(re.match(pat,text)) #only first line  --> None

Match Multiline

text="""python2 is for

python2 and there are two major vers

python2 and

python3 in future python4"""

pat=r"\bpython[23]?\b"

match\_obj=re.match(pat,text)

#print(match\_obj)

# <re.Match object; span=(12, 19), match='python2'>

if match\_obj:

    print(match\_obj.group()) # python2

    print(match\_obj.start()) #12

    print(match\_obj.end()-1) #18

    print("length:",match\_obj.end()-match\_obj.start()) # 7

else:

    print("No match found")

OUTPUT:

python2

0

6

length: 7

FINDALL and FINDITER

text="python and python2 and python3 and python4"

pat=r"\bpython[23]?\b"

print(re.match(pat,text))

OUTPUT:

1. text="This python and python2 and python3 and python4" –None
2. text="python and python2 and python3 and python4" --> <re.Match object; span=(0, 6), match='python'>

Code 2:

text="This is python and python2 and python3 and python4"

pat=r"\bpython[23]?\b"

print(re.findall(pat,text))

#['python', 'python2', 'python3'] --> 3

Code 3:

text="This is python and python2 and python3 and python4"

pat=r"\bpython[23]?\b"

print(re.finditer(pat,text))

#<callable\_iterator object at 0x000001E2967B3BA8

Code 4:

for each\_obj in re.finditer(pat,text):

    print(each\_obj.start(),each\_obj.end(),each\_obj.group())

Output:

pat=r"\bpyxthon[23]?\b"    --> No print

pat=r"\bpython[23]?\b"

    # <re.Match object; span=(8, 14), match='python'>

    #<re.Match object; span=(19, 26), match='python2'>

    #<re.Match object; span=(31, 38), match='python3'>

print(each\_obj.start(),each\_obj.end(),each\_obj.group())

#  8 14 python

# 19 26 python2

# 31 38 python3

SPLIT/SUB/SUBn:

text="This is about python and python is very easy and we having python vers and Python3 vers"

pat=r"python[23]?"

Output:

# print(re.split(pat,text)) --> # ['This is about ', ' and ', ' is very easy and we having ', 'w vers and ', ' vers']

SUB:

print(re.sub(pat,"Jython",text,flags=re.I))

print(re.sub(pat,"Jython",text,count=1,flags=re.I))

# Case 1:

print(re.sub(pat,"Jython",text,flags=re.I))

original "This is about python and python is very easy and we having pythonw vers and Python3 vers"

sub:  This is about Jython and Jython is very easy and we having Jythonw vers and Jython vers

# Case 2

print(re.sub(pat,"Jython",text,count=1,flags=re.I))

#original: text="This is about python and python is very easy and we having pythonw vers and Python3 vers"

#sub: count=1 (only one time)

#This is about Jython and python is very easy and we having pythonw vers and Python3 vers

SUBN:

text="This is about python and python is very easy and we having pythonw vers and Python3 vers"

pat=r"python[23]?"

OUTPUT 1:

print(re.subn(pat,"jython",text,count=2,flags=re.I))

# ('This is about jython and jython is very easy and we having pythonw vers and Python3 vers', 2)

OUTPUT 2:

print(re.subn(pat,"jython",text,flags=re.I))

# ('This is about jython and jython is very easy and we having jythonw vers and jython vers', 4)

COMPILE Operation:

CODE:

pat=r"\bpython[23]?\b"

text="This is about python. Python is easy to learn andwe have two major versions: python2 and python3"

print(re.findall(pat,text,flags=re.I)) # ['python', 'Python', 'python2', 'python3']

print(re.search(pat,text,flags=re.I)) #<re.Match object; span=(14, 20), match='python'>

print(re.split(pat,text))

#['This is about ', '. Python is easy to learn andwe have two major versions: ', ' and ', '']

CODE 2:

Recompile

pat=r"\bpython[23]?\b"

pat\_obj=re.compile(pat,flags=re.I)

print(pat\_obj) # re.compile('\\bpython[23]?\\b', re.IGNORECASE)

text="This is about python. Python is easy to learn andwe have two major versions: python2 and python3"

print(pat\_obj.findall(text)) #['python', 'Python', 'python2', 'python3']

print(pat\_obj.search(text)) #<re.Match object; span=(14, 20), match='python'>

print(pat\_obj.split(text)) #['This is about ', '. ', ' is easy to learn andwe have two major versions: ', ' and ', '']

#re.findall(my\_pat,my\_str) == re.compile(my\_pat).findall(my\_str)