Python Basic- Assignment- 07

1. What is the name of the feature responsible for generating Regex objects?

Answer:

The feature responsible for generating Regex objects in python is called the re.compile() from the "re" module. This module provides a set of functions that allow users to create, manipulate, and search strings using regular expressions (regex).

```
import re
a = re.compile("some_random_pattern")
print(a)
```

2. Why do raw strings often appear in Regex objects?

Answer:

Raw strings are often used in Regex objects because they do not interpret any special characters. This is important for Regex objects because they often use symbols such as '\d' or '\s', which would otherwise be interpreted as special characters. By using raw strings, these symbols can be used without being interpreted as special characters.

3. What is the return value of the search() method?

Answer:

The search() method returns a match object if there is a match found. If no matches are found, the search() method returns None. The match object returned contains information about the match, including the original input string, the regular expression used, and the part of the string where the match was found.

```
import re
a = input("Enter your sentence or word: ")
x = input("which letter you want to search: ")
b = re.search(x, a)
print(f"The first character of {x} is located in position:", b.start())
```

4. From a Match item, how do you get the actual strings that match the pattern?

Answer:

The Match object in Python has a group() method which can be used to retrieve the actual strings that match the pattern. This method takes a single argument, which is an integer

Done by- Md. Abdullah Al Mahmud; Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007

iNeuron

representing the group index. The first group of the match has an index of 1. This method will return the string that was matched by the corresponding part of the regular expression.

5. In the regex which was created from the r'(\d\d\d)-(\d\d\d-\d\d\d\d)', what does group zero cover? Group 2? Group 1?

Answer:

Group 0 covers the entire regex, which is the string '(\d\d\d)-(\d\d\d-\d\d\d\d)'.

Group 1 covers the first set of digits in the regex, which is '\d\d\d'.

Group 2 covers the second set of digits in the regex, which is '\d\d\d-\d\d\d\d'.

6. In standard expression syntax, parentheses and intervals have distinct meanings. How can you tell a regex that you want it to fit real parentheses and periods?

Answer:

In a regular expression, parentheses and periods can be indicated by using the escape character '\', followed by the symbol.

For example, to indicate real parentheses, use '\(' and '\)' and for periods, use '\.'.

7. The findall() method returns a string list or a list of string tuples. What causes it to return one of the two options?

Answer:

The findall() method will return a list of string tuples if the pattern includes capturing groups, otherwise it will return a list of strings. Capturing groups can be defined by parentheses in the pattern and are used to group parts of the string that match the pattern.

8. In standard expressions, what does the '|' character mean?

Answer:

The '|' character is known as a 'pipe' character and it is used to denote a logical OR operation. For example, the expression "cat|dog" would match either "cat" or "dog".

9. In regular expressions, what does the character '?' stand for?

Answer:

The character '?' stands for zero or one occurrence of the preceding element.

10.In regular expressions, what is the difference between the + and * characters?

Answer:

The + symbol is a "one or more" quantifier meaning that the preceding character can be matched one or more times.

The * symbol is a "zero or more" quantifier meaning that the preceding character can be matched zero or more times.

11. What is the difference between {4} and {4,5} in regular expressions?

Answer:

The difference between {4} and {4,5} in regular expressions is that {4} matches exactly 4 occurrences of the preceding expression, while {4,5} matches between 4 and 5 occurrences of the preceding expression. In other words, {4} will only match a string of 4 characters while {4,5} can match a string of 4 or 5 characters.

12. What do you mean by the \d, \w, and \s shorthand character classes signify in regular expressions?

Answer:

- \d is a shorthand character class for any single digit (0-9).
- \w is a shorthand character class for any single alphanumeric character (a-z, A-Z, 0-9, and _).
- \s is a shorthand character class for whitespace characters (space, tab, newline, etc).

13. What do mean by \D, \W, and \S shorthand character classes signify in regular expressions?

Answer:

- \D - matches any non-digit character
- \W - matches any non-alphanumeric character
- \S - matches any non-whitespace character

Done by- Md. Abdullah Al Mahmud; Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007

14. What is the difference between .*? and .*?

Answer:

The difference between .*? and .* in Python is that .*? is a non-greedy operator, meaning it will match as few characters as possible, while .* is a greedy operator, meaning it will match as many characters as possible.

15. What is the syntax for matching both numbers and lowercase letters with a character class?

Answer:

The syntax for matching both numbers and lowercase letters with a character class in Python is [a-z0-9]. This character class will match any lowercase letter from a to z, as well as any number from 0 to 9.

16. What is the procedure for making a normal expression in regex case insensitive?

Answer:

In Python, the easiest way to make a regular expression case insensitive is to add the re.IGNORECASE flag when compiling the regular expression. For example:

```
import re
text = "Hello World!"
regex = re.compile(r"hello", re.IGNORECASE)
match = regex.search(text)
if match:
        print("Match found:", match.group(0))
```

The re.IGNORECASE flag makes the regular expression case-insensitive, so it will match "Hello" as well as "hello".

17. What does this mean? character normally matches? What does it match if re.DOTALL is passed as 2nd argument in re.compile()?

Answer:

The '?' character is normally used to indicate a previous character can optionally be matched. So a string such as "cat?" will match both "cat" and "cats".

If re.DOTALL is passed as a second argument in re.compile(), then the '?' character will match any character including line breaks. This allows the regular expression to match any string that includes a line break character.

18. If numReg = re.compile(r'\d+'), what will numRegex.sub('X', 11 drummers, 10 pipers, five rings, 4 hen') return?

Answer:

This will return the following string: "X drummers, X pipers, five rings, X hen". This is because the numReg regex pattern will match any sequence of digits, so the numReg.sub() method will replace each of these matches with the string "X".

19. What does passing re.VERBOSE as the 2nd argument to re.compile() allow to do?

Answer:

Passing re.VERBOSE as the second argument to re.compile() allows for whitespace and comments in the pattern string which is ignored when the expression is compiled. This makes the pattern easier to read and debug, as well as allowing for multi-line patterns.

For example:

```
pattern = re.compile(r"""
 \d + # the integral part
\. # the decimal point
\d * # some fractional digits
""", re.VERBOSE)
```

20. How would you write a regex that matches a number with a comma for every three digits? It must match the given following:

'42'

'1,234'

'6,368,745'

but not the following:

'12,34,567' (which has only two digits between the commas)

'1234' (which lacks commas)

Answer:

```
import re
pattern = r'^\d{1,3}(,\d{3})*$'
pagex = re.compile(pattern)
l = ['42','1,234', '6,368,745','12,34,567','1234']
for ele in l:
    print('Output:',ele, '->', pagex.search(ele))
```

Done by- Md. Abdullah Al Mahmud;   Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007

21. How would you write a regex that matches the full name of someone whose last name is Watanabe? You can assume that the first name that comes before it will always be one word that begins with a capital letter. The regex must match the following:

'Haruto Watanabe'

'Alice Watanabe'

'RoboCop Watanabe'

but not the following:

'haruto Watanabe' (where the first name is not capitalized)

'Mr. Watanabe' (where the preceding word has a non letter character)

'Watanabe' (which has no first name)

'Haruto watanabe' (where Watanabe is not capitalized)

Answer:

```
import re
pattern = r'[A-Z]{1}[a-z]*\sWatanabe'
namex = re.compile(pattern)
list = ['Haruto Watanabe','Alice Watanabe','RoboCop Watanabe','haruto Watanabe','Mr. Watanabe','Watanabe','Haruto watanabe']
for name in list:
   print('Output: ',name,'->',namex.search(name))
```

22. How would you write a regex that matches a sentence where the first word is either Alice, Bob, or Carol; the second word is either eats, pets, or throws; the third word is apples, cats, or baseballs; and the sentence ends with a period? This regex should be case-insensitive. It must match the following:

'Alice eats apples.'

'Bob pets cats.'

'Carol throws baseballs.'

'Alice throws Apples.'

'BOB EATS CATS.'

but not the following:

'RoboCop eats apples.'

'ALICE THROWS FOOTBALLS.'

'Carol eats 7 cats.'

Done by- Md. Abdullah Al Mahmud; Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007

Answer:

```
import re

pattern = r'(Alice|Bob|Carol)\s(eats|pets|throws)\s(apples|cats|baseballs)\.'

casex = re.compile(pattern,re.IGNORECASE)

for ele in ['Alice eats apples.','Bob pets cats.','Carol throws baseballs.','Alice throws Apples.','BOB EATS CATS.','RoboCop eats apples.'

,'ALICE THROWS FOOTBALLS.','Carol eats 7 cats.']:

    print('Output: ',ele,'->',casex.search(ele))
```

Done by- Md. Abdullah Al Mahmud; Student of Data Science

email- shaowntxt@gmail.com; github link: https://github.com/Almahmud007