Assignment :3

Regular Expression Practice Questions

Question 1- Write a RegEx pattern in python program to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9).

```
import re

def contains_only_valid_characters(s):
    # Define the regular expression pattern
    pattern = r'[a-zA-Z0-9]+'

    # Use re.fullmatch to check if the entire string matches the pattern
    return bool(re.fullmatch(pattern, s))

# Test examples
Sample_strings = [
    "Sanket123",   # Valid
    "Sanket@123",  # Invalid (contains @)
    "123456",     # Valid
    "SanKET",     # Valid
    "san KET",    # Invalid (contains space)
    "SanKET...!"     # Invalid (contains ...)
]

# Check each test string
for SAM in Sample_strings:
    result = contains_only_valid_characters(SAM)
    print(f"'{SAM}' contains only valid characters: {result}")
```

Explanation:

1. **Pattern Definition**:
    - r'[a-zA-Z0-9]+' matches one or more alphanumeric characters (letters and digits).
2. **Function Logic**:
    - re.fullmatch(pattern, s) attempts to match the entire string s against the pattern.
    - bool(re.fullmatch(pattern, s)) converts the result to a boolean value. If the match is successful, it returns True; otherwise, it returns False.
3. **Testing**:
    - The Sample_strings list includes a variety of test cases to validate the function.

Question 2- Write a RegEx pattern that matches a string that has an a followed by zero or more b's?

```
import re
def matches_pattern(s):
    # Define the regular expression pattern
    pattern = r'ab*'

    # Use re.fullmatch to check if the entire string matches the pattern
    return bool(re.fullmatch(pattern, s))

# Test examples
sampple_strings = [
    "s",       # Valid (matches 's' followed by zero b's)
    "sa",      # Valid (matches 's' followed by one b)
    "abb",     # Valid (matches 'a' followed by two b's)
    "abbbbsankkk",    # Valid (matches 'a' followed by four b's)
    "b",       # Invalid (does not start with 'a')
    "ba",      # Invalid (contains 'b' before 'a')
    "bba",     # Invalid (contains 'b' before 'a')
    "abc",     # Invalid (contains 'c' after 'a' and 'b') ]
```

```
# Check each test string
for test in sampple_strings:
    out = matches_pattern(test)
    print(f"'{test}' matches the pattern: {out}")
```

Explanation of the Code:

1. **Pattern Definition**:
    - r'ab*' specifies that the string should start with 'a' followed by zero or more 'b's.
2. **Function Logic**:
    - re.fullmatch(pattern, s) checks if the entire string s matches the pattern.
    - bool(re.fullmatch(pattern, s)) returns True if the match is successful, otherwise False.
3. **Testing**:
    - The test_strings list contains various strings to verify whether they match the defined pattern.

Question 3- Write a RegEx pattern that matches a string that has an a followed by one or more b's

```
import re

def matches_pattern(s):
    # Define the regular expression pattern
    pattern = r'ab+'

    # Use re.fullmatch to check if the entire string matches the pattern
    return bool(re.fullmatch(pattern, s))

# Test examples
test_strings = [
    "ab",      # Valid (matches 'a' followed by one b)
```

```
    "abb",      # Valid (matches 'a' followed by two b's)
    "abbbbb",   # Valid (matches 'a' followed by five b's)
    "a",        # Invalid (no 'b' after 'a')
    "b",        # Invalid (does not start with 'a')
    "ba",       # Invalid (contains 'b' before 'a')
    "abcb",     # Invalid (contains 'c' after 'a' and 'b')
]

# Check each test string
for test in test_strings:
    result = matches_pattern(test)
    print(f"'{test}' matches the pattern: {result}")
```

Explanation of the Code:

1. **Pattern Definition**:
    - r'ab+' specifies that the string should start with 'a' followed by one or more 'b's.
2. **Function Logic**:
    - re.fullmatch(pattern, s) checks if the entire string s matches the pattern.
    - bool(re.fullmatch(pattern, s)) returns True if the match is successful, otherwise False.
3. **Testing**:
    - The test_strings list contains various strings to validate whether they conform to the pattern of 'a' followed by one or more 'b's.

Question 4- Write a RegEx pattern that matches a string that has an a followed by zero or one 'b'.

```
import re

def matches_pattern(s):
```

```
    # Define the regular expression pattern
    pattern = r'ab?'

    # Use re.fullmatch to check if the entire string matches the pattern
    return bool(re.fullmatch(pattern, s))

# Test examples
test_strings = [
    "a",       # Valid (matches 'a' followed by zero b's)
    "ab",      # Valid (matches 'a' followed by one b)
    "abb",     # Invalid (contains more than one b after 'a')
    "b",       # Invalid (does not start with 'a')
    "ba",      # Invalid (contains 'b' before 'a')
    "abc",     # Invalid (contains 'c' after 'a' and 'b') ]
# Check each test string
for test in test_strings:
    result = matches_pattern(test)
    print(f"'{test}' matches the pattern: {result}")
```

Explanation of the Code:

1. **Pattern Definition**:
    - r'ab?' specifies that the string should start with 'a', followed by zero or one 'b'.
2. **Function Logic**:
    - re.fullmatch(pattern, s) checks if the entire string s matches the pattern.
    - bool(re.fullmatch(pattern, s)) converts the result to a boolean value. If the string matches the pattern, it returns True; otherwise, it returns False.
3. **Testing**:
    - The test_strings list contains various strings to check if they match the pattern of 'a' followed by zero or one 'b'.

Question 5- Write a RegEx pattern in python program that matches a string that has an a followed by three 'b'.

```
import re


def matches_pattern(s):
    # Define the regular expression pattern
    pattern = r'ab{3}'

    # Use re.fullmatch to check if the entire string matches the pattern
    return bool(re.fullmatch(pattern, s))

# Test examples
test_strings = [
    "abbbb",    # Invalid (contains four b's, not exactly three)
    "abbbb",    # Invalid (contains four b's, not exactly three)
    "ab",       # Invalid (contains fewer than three b's)
    "abbbb",    # Invalid (contains four b's, not exactly three)
    "abbbb",    # Invalid (contains four b's, not exactly three)
    "abbbb",    # Invalid (contains four b's, not exactly three)
]

# Check each test string
for test in test_strings:
    result = matches_pattern(test)
    print(f"'{test}' matches the pattern: {result}")
```

Explanation of the Code:

1. **Pattern Definition**:
    - r'ab{3}' specifies that the string should start with 'a', followed by exactly three 'b's.
2. **Function Logic**:

- re.fullmatch(pattern, s) checks if the entire string s matches the pattern.
- bool(re.fullmatch(pattern, s)) converts the result to a boolean value. If the string matches the pattern, it returns True; otherwise, it returns False.

3. **Testing**:
   - The test_strings list includes various strings to verify whether they conform to the pattern of 'a' followed by exactly three 'b's.

Question 6- Write a RegEx pattern in python program that matches a string that has an a followed by two to three 'b'.

```
import re


def matches_pattern(s):
    # Define the regular expression pattern
    pattern = r'ab{2,3}'

    # Use re.fullmatch to check if the entire string matches the pattern
    return bool(re.fullmatch(pattern, s))


# Test examples
test_strings = [
    "abb",     # Valid (matches 'a' followed by two b's)
    "abbb",    # Valid (matches 'a' followed by three b's)
    "abbbb",   # Invalid (contains more than three b's)
    "ab",      # Invalid (contains fewer than two b's)
    "a",       # Invalid (no b's after 'a')
    "b",       # Invalid (does not start with 'a')
    "ba",      # Invalid (contains 'b' before 'a')
    "abcb"     # Invalid (contains characters other than 'b' after 'a')
]
```

```
# Check each test string
for test in test_strings:
    result = matches_pattern(test)
    print(f"'{test}' matches the pattern: {result}")
```

Explanation of the Code:

1. **Pattern Definition**:
    - r'ab{2,3}' specifies that the string should start with 'a', followed by 2 to 3 'b's.
2. **Function Logic**:
    - re.fullmatch(pattern, s) checks if the entire string s matches the pattern.
    - bool(re.fullmatch(pattern, s)) converts the result to a boolean value. If the string matches the pattern, it returns True; otherwise, it returns False.
3. **Testing**:
    - The test_strings list contains various strings to verify whether they match the pattern of 'a' followed by between two and three 'b's.

Question 7- Write a Python program that matches a string that has an 'a' followed by anything, ending in 'b'.

```
import re

def matches_pattern(s):
    # Define the regular expression pattern
    pattern = r'a.*b'

    # Use re.fullmatch to check if the entire string matches the pattern
    return bool(re.fullmatch(pattern, s))

# Test examples
test_strings = [
```

```
    "ab",       # Valid (starts with 'a' and ends with 'b')
    "acb",      # Valid (starts with 'a', has characters in between, and ends with 'b')
    "a12345b",  # Valid (starts with 'a', has digits in between, and ends with 'b')
    "a",        # Invalid (ends with 'a', not 'b')
    "b",        # Invalid (does not start with 'a')
    "ac",       # Invalid (does not end with 'b')
    "bca"       # Invalid (does not start with 'a' and does not end with 'b') ]
# Check each test string
for test in test_strings:
    result = matches_pattern(test)
    print(f"'{test}' matches the pattern: {result}")
```

Explanation of the Code:

1. **Pattern Definition**:
    - r'a.*b' specifies that the string should start with 'a', can have any characters in between (or none), and end with 'b'.
2. **Function Logic**:
    - re.fullmatch(pattern, s) checks if the entire string s matches the pattern.
    - bool(re.fullmatch(pattern, s)) converts the result to a boolean value. If the string matches the pattern, it returns True; otherwise, it returns False.
3. **Testing**:
    - The test_strings list contains various strings to check if they match the pattern of starting with 'a' and ending with 'b' with any characters in between.

Question 8- Write a RegEx pattern in python program that matches a word at the beginning of a string.

```
import re
def match_sanket_at_start(s):
    # Define the regular expression pattern
    pattern = r'^SANKET'

    # Use re.match to check if the start of the string matches the pattern
    match = re.match(pattern, s)

    # Return True if the word 'SANKET' is at the start, otherwise False
    return bool(match)

# Test examples
test_strings = [
    "SANKET is a name",     # Valid (starts with 'SANKET')
    "SANKET123",            # Valid (starts with 'SANKET')
    "Sanket is a name",     # Invalid (starts with 'Sanket' with lowercase 's')
    "I am SANKET",          # Invalid (does not start with 'SANKET')
    "SANKET!",              # Valid (starts with 'SANKET')
    "  SANKET",             # Invalid (starts with space before 'SANKET')
    ""                      # Invalid (empty string)
]

# Check each test string
for test in test_strings:
    result = match_sanket_at_start(test)
    print(f"'{test}' starts with 'SANKET': {result}")
```

Explanation of the Code:

1. **Pattern Definition**:
    - r'^SANKET' specifies that the pattern should match the exact string "SANKET" at the beginning of the string.
2. **Function Logic**:
    - re.match(pattern, s) checks if the beginning of the string s matches the pattern.
    - bool(match) converts the match object to a boolean value. It returns True if the string starts with "SANKET", otherwise False.
3. **Testing**:
    - The test_strings list contains various strings to check if they start with "SANKET".

Question 9- Write a RegEx pattern in python program that matches a word at the end of a string.

```
import re

def match_shaks_at_end(s):
    # Define the regular expression pattern
    pattern = r'shaks$'

    # Use re.match to check if the end of the string matches the pattern
    match = re.search(pattern, s)

    # Return True if the word 'shaks' is at the end, otherwise False
    return bool(match)
```

```
# Test examples
test_strings = [
    "This is a test shaks",  # Valid (ends with 'shaks')
    "shaks",                 # Valid (exactly 'shaks')
    "shaks123",              # Invalid (contains characters after 'shaks')
    "The word is shaks",     # Valid (ends with 'shaks')
    "shaksandmore",          # Invalid (contains characters before 'shaks')
    "shaks!"                 # Valid (ends with 'shaks' with an exclamation mark)
]

# Check each test string
for test in test_strings:
    result = match_shaks_at_end(test)
    print(f"'{test}' ends with 'shaks': {result}")
```

Explanation of the Code:

1. **Pattern Definition**:
    - r'shaks$' specifies that the string should end with "shaks".
2. **Function Logic**:
    - re.search(pattern, s) searches the entire string s for a match to the pattern. This method is used here instead of re.fullmatch because re.fullmatch is more strict and checks the entire string, which is not necessary here.
    - bool(match) converts the match object to a boolean value. It returns True if the string ends with "shaks", otherwise False.
3. **Testing**:
    - The test_strings list includes various strings to determine if they end with "shaks".

Question 10- Write a RegEx pattern in python program to find all words that are 4 digits long in a string.

Sample text- '01 0132 231875 1458 301 2725.'

Expected output- ['0132', '1458', '2725']

```
import re

def find_four_digit_words(text):
    # Define the regular expression pattern for 4-digit words
    pattern = r'\b\d{4}\b'

    # Use re.findall to find all matches of the pattern
    matches = re.findall(pattern, text)

    return matches

# Sample text
sample_text = '01 0132 231875 1458 301 2725.'

# Find all 4-digit words
result = find_four_digit_words(sample_text)
print(result)  # Expected output: ['0132', '1458', '2725']
```

Explanation of the Code:

1. **Pattern Definition**:
    - r'\b\d{4}\b' is a regex pattern that matches exactly 4 digits surrounded by word boundaries, ensuring that we match only complete 4-digit words.
2. **Function Logic**:
    - re.findall(pattern, text) searches the entire text for all non-overlapping matches of the pattern and returns them as a list of strings.

3. **Testing**:
    - The sample_text variable contains the input string with various digit sequences.
    - The find_four_digit_words function is called to find all 4-digit sequences in the sample text.

OR

Sample Test with Sanku

If you want to test this code with a different sample text:

```
import re


def find_four_digit_words(text):
    # Define the regular expression pattern for 4-digit words
    pattern = r'\b\d{4}\b'

    # Use re.findall to find all matches of the pattern
    matches = re.findall(pattern, text)

    return matches
# Different sample text
sample_text = 'Sanku 1234 4567 89 1011 5678 xyz'

# Find all 4-digit words
result = find_four_digit_words(sample_text)
print(result)  # Expected output: ['1234', '4567', '1011', '5678']
```

OUTPUT:

```
['1234', '4567', '1011', '5678']
=== Code Execution Successful ===
```
