Practical – 5 : Regular Expression

Write a program to generate data of 500 randomly generated printable characters[a-z,A-Z, Special Chars, 0-9, etc, spaces/blanks] and perform the following operations

1. **Search the instance of patterns**
2. **matches a string that has an a followed by zero or more b's.**
3. **matches a string that has an a followed by one or more b's**
4. **matches a string that has an a followed by zero or one 'b'**
5. **matches a string that has an a followed by two 'b'**
6. **a string that has an a followed by two to three 'b'.**
7. **find sequences of lowercase letters joined with a underscore.**
8. **find the sequences of one upper case letter followed by lower case letters.**
9. **matches a string that has an 'a' followed by anything, ending in 'b'.**
10. **matches a word at the beginning of a string.**
11. **matches a word containing 'z', not at the start or end of the word.**
12. **match a string that contains only upper and lowercase letters, numbers, and underscores.**
13. **string will start with a specific number**
14. **to check for a number at the end of a string.**
15. **to search the numbers (0-9) of length between 1 to 3 in a given string**
16. **to convert a date of yyyy-mm-dd format to dd-mm-yyyy format. For Specific Data Entered**
17. **to find all words starting with 'a' or 'e' in a given string.**
18. **program to separate and print the numbers and their position of a given string.**
19. **To check Valid email address**

Valid PAN CARD Details

CODE:

```
import string
import random

s = 500
ran = ''.join(random.choices(string.ascii_lowercase + 
string.ascii_uppercase + string.digits + string.punctuation, k=s))
```

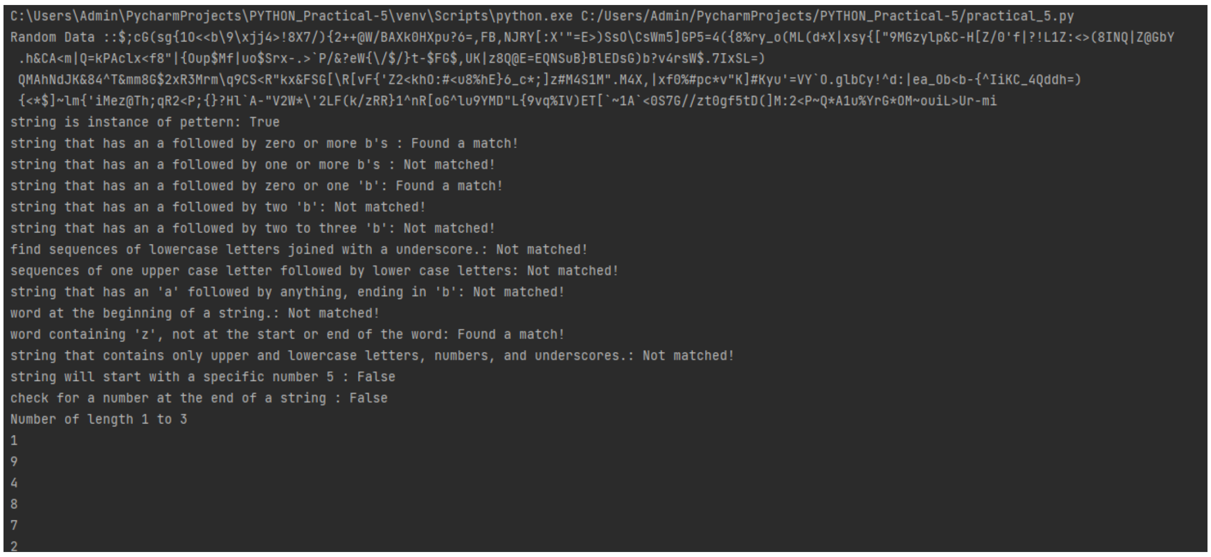
```
print("Random Data :" + str(ran))

# 1.Search the instance of patterns
import re


defis_allowed_character(string):
    char = re.compile(r'[a-x-zA-Z0-9!()-[]{};:''"\,<>./?@#$%^&*_~]')
    string = char.search(string)
    return not bool(string)


print("string is instance of pettern: " + str(is_allowed_character(ran)))

# 2.matches a string that has an a followed by zero or more b's.

import re


deftext_match(text):
    patterns = 'ab*?'
ifre.search(patterns, text):
        return ('Found a match!')
else:
        return ('Not matched!')


print("string that has an a followed by zero or more b's : " +
str(text_match(ran)))

# 3.matches a string that has an a followed by one or more b's.
import re


deftext_match(text):
    patterns = 'ab+?'
ifre.search(patterns, text):
        return 'Found a match!'
else:
        return ('Not matched!')


print("string that has an a followed by one or more b's : " +
str(text_match(ran)))

# 4.matches a string that has an a followed by zero or one 'b'.
import re


deftext_match(text):
    patterns = 'ab?'
ifre.search(patterns, text):
        return 'Found a match!'
else:
        return ('Not matched!')
```

```
print("string that has an a followed by zero or one 'b': " +
str(text_match(ran)))

# 5.matches a string that has an a followed by two 'b'.
import re


deftext_match(text):
    patterns = 'ab{2}?'
ifre.search(patterns, text):
        return 'Found a match!'
else:
        return ('Not matched!')


print("string that has an a followed by two 'b': " + str(text_match(ran)))

# 6.a string that has an a followed by two to three 'b'.

import re


deftext_match(text):
    patterns = 'ab{2,3}'
    if re.search(patterns, text):
        return 'Found a match!'
else:
        return ('Not matched!')


print("string that has an a followed by two to three 'b': " +
str(text_match(ran)))

# 7.find sequences of lowercase letters joined with a underscore.
import re


deftext_match(text):
    patterns = '^[a-z]+_[a-z]+$'
    if re.search(patterns, text):
        return 'Found a match!'
else:
        return ('Not matched!')


print("find sequences of lowercase letters joined with a underscore.: " +
str(text_match(ran)))

# 8. find the sequences of one upper case letter followed by lower case
letters.
import re


deftext_match(text):
    patterns = '[A-Z]+[a-z]+$'
    if re.search(patterns, text):
        return 'Found a match!'
```

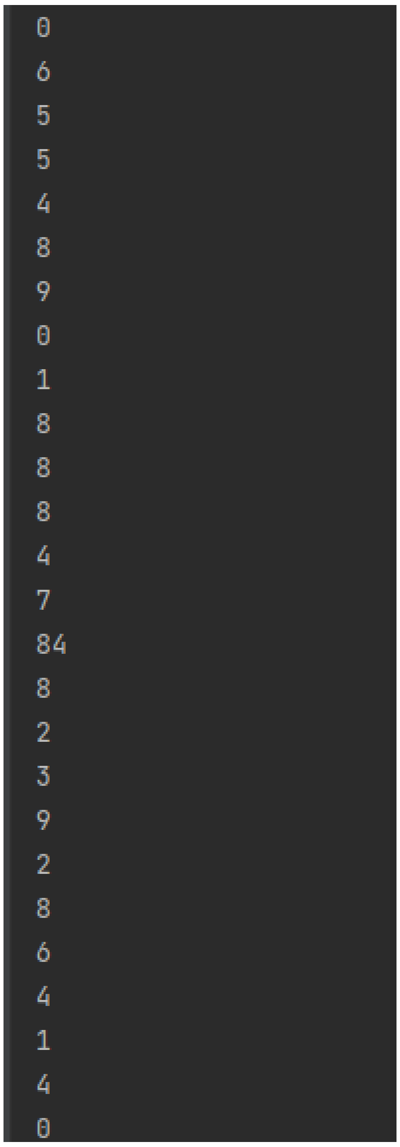
```
else:
        return ('Not matched!')


print("sequences of one upper case letter followed by lower case letters: "
+ str(text_match(ran)))

# 9. matches a string that has an 'a' followed by anything, ending in 'b'.
import re


deftext_match(text):
    patterns = 'a.*?b$'
    if re.search(patterns, text):
        return 'Found a match!'
else:
        return ('Not matched!')


print("string that has an 'a' followed by anything, ending in 'b': " +
str(text_match(ran)))

# 10.matches a word at the beginning of a string.
import re


deftext_match(text):
    patterns = '^\w+'
    if re.search(patterns, text):
        return 'Found a match!'
else:
        return ('Not matched!')


print("word at the beginning of a string.: " + str(text_match(ran)))

# 11. matches a word containing 'z', not at the start or end of the word.
import re


deftext_match(text):
    patterns = '\Bz\B'
    if re.search(patterns, text):
        return 'Found a match!'
else:
        return ('Not matched!')


print("word containing 'z', not at the start or end of the word: " +
str(text_match(ran)))

# 12.match a string that contains only upper and lowercase letters,
numbers, and underscores.
import re


deftext_match(text):
    patterns = '^[a-zA-Z0-9_]*$'
    if re.search(patterns, text):
        return 'Found a match!'
else:
        return ('Not matched!')


print("string that contains only upper and lowercase letters, numbers, and
underscores.: " + str(text_match(ran)))

# 13.string will start with a specific number.
import re


defmatch_num(string):
    text = re.compile(r"^5")
    if text.match(string):
        return True
    else:
        return False


print("string will start with a specific number 5 : " +
str(match_num(ran)))

# 14.to check for a number at the end of a string.
import re


defend_num(string):
    text = re.compile(r".*[0-9]$")
    if text.match(string):
        return True
    else:
        return False


print("check for a number at the end of a string : " + str(end_num(ran)))

# 15.to search the numbers (0-9) of length between 1 to 3 in a given
string.
import re

results = re.finditer(r"([0-9]{1,3})", ran)
print("Number of length 1 to 3")
for n in results:
    print(n.group(0))

# 16.to convert a date of yyyy-mm-dd format to dd-mm-yyyy format. For
Specific Data Entered.

import re


defchange_date_format(dt):
    return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', dt)
```

```
dt1 = "1998-02-15"
print("Original date in YYY-MM-DD Format: ", dt1)
print("New date in DD-MM-YYYY Format: ", change_date_format(dt1))

# 17.to find all words starting with 'a' or 'e' in a given string.

import re

list = re.findall("[ae]\w+", ran)
# Print result.
print(list)

# 18.program to separate and print the numbers and their position of a
given string.
import re

for m in re.finditer("\d+", ran):
    print(m.group(0))
    print("Index position:", m.start())

# 19.To check Valid email address.

import re

regex = '^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}$'


def check(email):
    if (re.search(regex, email)):
        print(str(email) + " is " + " Valid Email")
    else:
        print(str(email) + " is " + "Invalid Email")


if __name__ == '__main__':
    email = "parthkukadiya010@gmail.com"
    (check(email))

# 20.Valid PAN CARD Details.
import re


defisValid(Z):
    Result = re.compile("[A-Za-z]{5}\d{4}[A-Za-z]{1}")
    return Result.match(Z)


Z = "AOWPL6574P"
if (isValid(Z)):
    print(str(Z) + " Valid PAN Number!")
else:
    print(str(Z) + "Invalid PAN Number entered.")
```

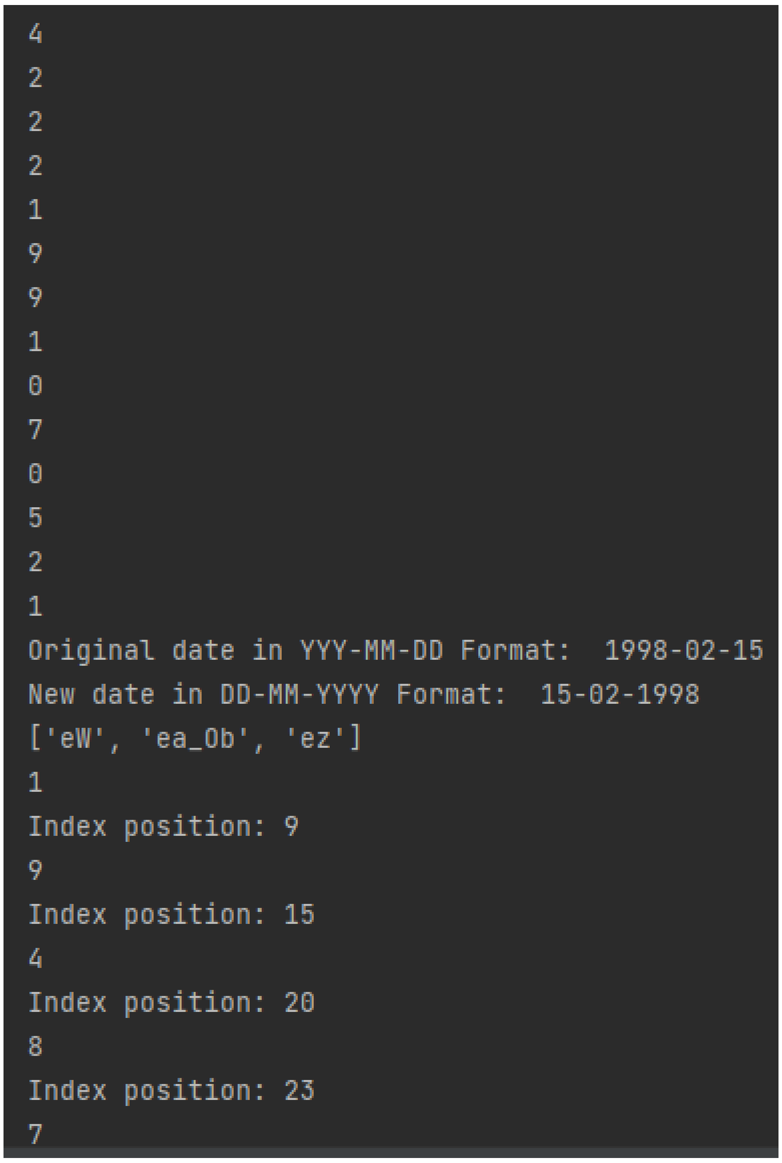
OUTPUT:



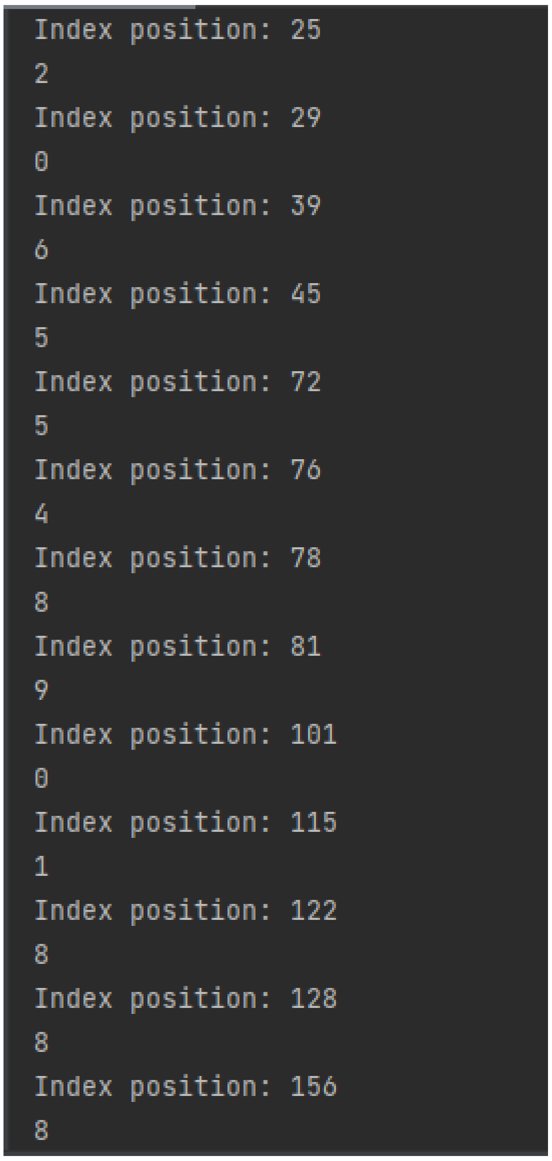
```
C:\Users\Admin\PycharmProjects\PYTHON_Practical-5\venv\Scripts\python.exe C:/Users/Admin/PycharmProjects/PYTHON_Practical-5/practical_5.py
Random Data ::$;cG(sg{10<<b\9\xjj4>!8X7/){2++@W/BAXk0HXpu?6=,FB,NJRY[:X'"=E>)SsO\CsWm5]GP5=4({8%ry_o(ML(d*X|xsy{["9MGzylp&C-H[Z/0'f|?!L1Z:<>(8INQ|Z@GbY
 .h&CA<m|Q=kPAclx<f8"|{Oup$Mf|uo$Srx-.>`P/&?eW{\/$/}t-$FG$,UK|z8Q@E=EQNSuB}BlEDsG)b?v4rsW$.7IxSL=)
 QMAhNdJK&84^T&mm8G$2xR3Mrm\q9CS<R"kx&FSG[\R[vF{'Z2<khO:#<u8%hE}6_c*;]z#M4S1M".M4X,|xf0%#pc*v"K]#Kyu'=VY`O.glbCy!^d:|ea_Ob<b-{^IiKC_4Qddh=)
 {<*$]~lm{'iMez@Th;qR2<P;{}?Hl`A-"V2W*\'2LF(k/zRR}1^nR[oG^lu9YMD"L{9vq%IV)ET[`~1A`<0S7G//zt0gf5tD(]M:2<P~Q*A1u%YrG*OM~ouiL>Ur-mi
string is instance of pettern: True
string that has an a followed by zero or more b's : Found a match!
string that has an a followed by one or more b's : Not matched!
string that has an a followed by zero or one 'b': Found a match!
string that has an a followed by two 'b': Not matched!
string that has an a followed by two to three 'b': Not matched!
find sequences of lowercase letters joined with a underscore.: Not matched!
sequences of one upper case letter followed by lower case letters: Not matched!
string that has an 'a' followed by anything, ending in 'b': Not matched!
word at the beginning of a string.: Not matched!
word containing 'z', not at the start or end of the word: Found a match!
string that contains only upper and lowercase letters, numbers, and underscores.: Not matched!
string will start with a specific number 5 : False
check for a number at the end of a string : False
Number of length 1 to 3
1
9
4
8
7
2
```



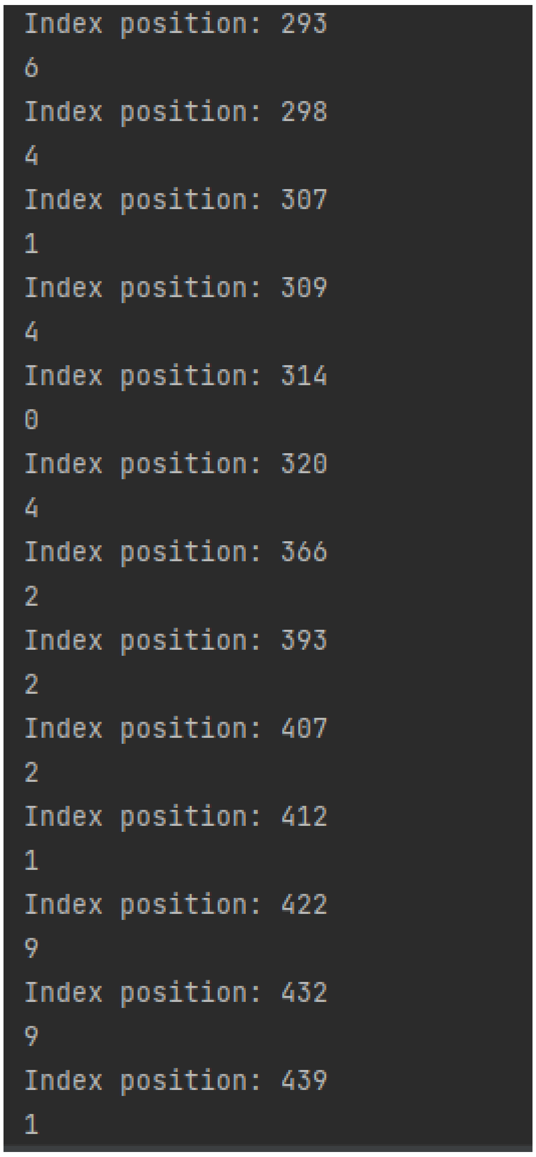
```
0
6
5
5
4
8
9
0
1
8
8
8
4
7
84
8
2
3
9
2
8
6
4
1
4
0
```

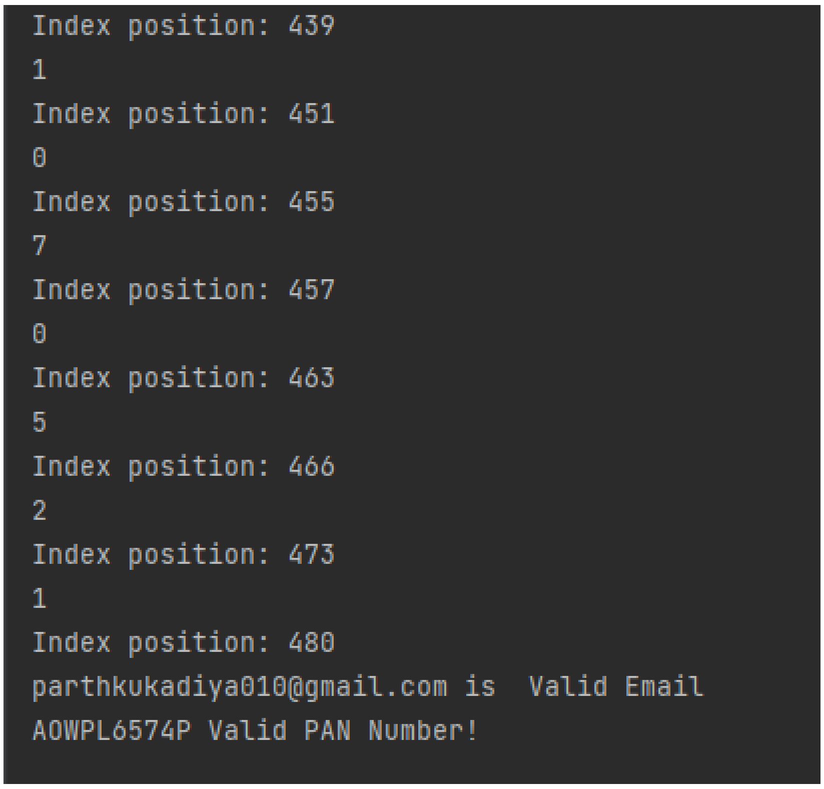


```
4
2
2
2
1
9
9
1
0
7
0
5
2
1
Original date in YYY-MM-DD Format:  1998-02-15
New date in DD-MM-YYYY Format:  15-02-1998
['eW', 'ea_Ob', 'ez']
1
Index position: 9
9
Index position: 15
4
Index position: 20
8
Index position: 23
7
```



```
Index position: 25
2
Index position: 29
0
Index position: 39
6
Index position: 45
5
Index position: 72
5
Index position: 76
4
Index position: 78
8
Index position: 81
9
Index position: 101
0
Index position: 115
1
Index position: 122
8
Index position: 128
8
Index position: 156
8
```


```
Index position: 293
6
Index position: 298
4
Index position: 307
1
Index position: 309
4
Index position: 314
0
Index position: 320
4
Index position: 366
2
Index position: 393
2
Index position: 407
2
Index position: 412
1
Index position: 422
9
Index position: 432
9
Index position: 439
1
```



```
Index position: 439
1
Index position: 451
0
Index position: 455
7
Index position: 457
0
Index position: 463
5
Index position: 466
2
Index position: 473
1
Index position: 480
parthkukadiya010@gmail.com is  Valid Email
AOWPL6574P Valid PAN Number!
```