MCQ Practice Questions

In [ ]:
```
if -3 will evaluate to

a. True  #correct
b. False
```

In [ ]:
```
Given the nested if-else below, what will be the value x when the code executed successfully

x = 0
a = 5
b = 5
if a > 0:
    if b < 0:
        x = x + 5
    elif a > 5:
        x = x + 4
    else:
        x = x + 3
else:
    x = x + 2
print(x)


a.0
b.4
c.2
d.3   #correct Answer
```

In [ ]:
```
What is the output of the following nested loop?

for num in range(10, 14):
   for i in range(2, num):
       if num%i == 1:
          print(num)
          break

a.10    #correct
  11
  12
  13



b.11
  13

c.None

d.Error
```

In [ ]:
```
What is the output of the following loop

for l in 'Jhon':
    if l == 'o':
        pass
    print(l, end=", ")

a.J, h, n,
b.J, h, o, n,  #correct
c.None
d.Error
```

In [ ]:
```
What is the output of the following if statement

a, b = 12, 5
if a + b:
    print('True')
else:
  print('False')


a. False
b. True   #correct
```

In [ ]:
```
Given the nested if-else structure below, what will be the value of x after code execution completes

x = 0
a = 0
b = -5
if a > 0:
    if b < 0:
        x = x + 5
    elif a > 5:
        x = x + 4
    else:
        x = x + 3
else:
    x = x + 2
print(x)


a.2  #correct
b.0
c.3
d.4
```

In [ ]:
```
What is the output of the following for loop and  range() function

for num in range(-2,-5,-1):
    print(num, end=", ")

a. -2, -1, -3, -4
b. -2, -1, 0, 1, 2, 3,
c. -2, -1, 0
d. -2, -3, -4,   #Correct
```

In [ ]:
```
What is the output of the following range() function

for num in range(2,-5,-1):
    print(num, end=", ")

a. 2, 1, 0
b. 2, 1, 0, -1, -2, -3, -4, -5
c. 2, 1, 0, -1, -2, -3, -4           #Correct
d. None
```

Subset and Superset

In [ ]:
```
--> A subset is a set that has either some or all of the elements of another set, called the superset.
    If A is a subset of B, then A is contained in B.
    It implies that B contains A, or in other words, B is a superset of A.
    We write A ⊇ B to denote that B is a superset of A.
    For example, if A = {1, 3} and B = {1, 2, 3},
    Since all the elements in A contained in B so B is a superset of A because B contains A
```

Assignment Problem

In [5]:
```
def count_upper_lower(x):
    count_upper=0
    count_lower=0
    for i in range(len(x)):
        if x[i].isupper():
            count_upper+=1
        elif x[i].islower():
            count_lower+=1
    print("No. of Upper case characters : ",count_upper)
    print("No. of Lower case characters : ",count_lower)
    return ""
x = 'The quick Brow Fox'
print(count_upper_lower(x))
```

```
No. of Upper case characters :  3
No. of Lower case characters :  12
```

Regular Expression

In [ ]:
```
Regular Expression: if we want to represent a group of strings , according to a particular
                    pattern then we should use Regular Expression.
```

Examples

In [ ]:
```
Example of RE --> fixed pattern is there for our mobile numbers
                  (0-9 ,10numbers starting_no --> 6,7,8,9)

Example of RE --> Fixed Pattern for PAN Card

Example of RE -->fixed pattern for Vechile number

Example of RE --> Fixed Pattern is for Password --> strong password,weak,medium

Example of RE --> Fixed Pattern for IP Address

Example of RE --> Fixed Pattern for IFSC Code

Example of RE --> Fix Pattern for websites(www.google.com  www.amityuniversity.edu.in)

Example of RE --> Fixed pattern is given to our mail id
```

Mobile Number Pattern

In [ ]:
```
Mobile Number Pattern:
    Mobile Numbers --> 10 --> Starting digit --> 7,8,9,6
    In India There is one fixed pattern for the mobile Number:
        1.only 10 digits are  allowed.
        2.No two Mobile numbers are same
        3.Starting digit of a mobile Number is either 6,7,8,9
```

Email ID Pattern:

In [ ]:
```
Email ID Pattern:
    Email Account --> username@domain_name
    domain_name --> yahoo, hotmail, gmail

    Can we tell that after @ we will get the domain name
```

Uses of Regular Expression

In [ ]:
```
Where we need to use Regular Expression:
    1.Password
    2.Validation of Data
    3.Email
    4.Translator --> TOC , COMPILER DESIGN
    5.Digital Circuit
    6.Protocols like TCP/IP
```

How we can Implement regular Expression in Python

In [ ]:
```
--> In python Programming for implementing regular expression we are having a module
   which is known as re and This module is used for regular expression.

--> re is the module that we need to import for regular expression.
```

Few Important Function related to Regular Expression(Regex)

Finditer() , start(),end() and group() Function

In [ ]:
```
finditer() --> that will take two arguments substring(pattern) and string(input).
           --> return the iterator object which yields match object for every match.

start() --> return the start index of the match

end () --> return end+1 index of the match

group()--> return the matched string
```

Example

In [3]:
```
import re
matcher = re.finditer("abc","abcabcbabc")
for i in matcher:
    print(i.start())
    print(i.end())
    print(i.group())
```

```
0
3
abc
3
6
abc
7
10
abc
```

Python Program to Return the Domain of a Mail

In [14]:
```
#Domain of a Mail
import re
Email_id = input("Enter Your Mail Id : ")
matcher = re.finditer("@",Email_id)
for i in matcher:
    y=i.start()
if Email_id[y+1:]=="hotmail.com":
    print("It is a hotmail Id")

elif Email_id[y+1:]=="yahoo.com":
    print("It is a Yahoo Id")

elif Email_id[y+1:]=="outlook.com":
    print("It is a outlook Id")

elif Email_id[y+1:]=="gmail.com":
    print("It is a Gmail Id")

else:
    print("Enter Valid mail Id")
```

```
Enter Your Mail Id : ashu@gmail.com
It is a Gmail Id
```

Character Classes in RE

In [ ]:
```
--> We can customize all character classes based on our Requirements.
```

In [ ]:
```
Character Classes:
    1.[abc] --> either a or b or c
    2.[^abc] --> except a ,b,c
    3.[a-z] --> any lower case alphabet
    4.[A-Z] --> any upper case alphabet
    5.[A-Za-z] --> any alphabet either upper case or lower case
    6.[0-9] --> Any digit from 0 to 9
    7.[A-Za-z0-9] --> any alphanumeric symbols
    8.[^A-Za-z0-9] --> any special symbol
```

Example of Each Character Classes

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

In [4]:
```
import re
matcher = re.finditer("[abc]","a7b@9xbs")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
                 #0              #1                     a
                 #2              #3                     b
                 #6              #7                     b
```

```
0    1    a
2    3    b
6    7    b
```

[^abc] --> except a ,b,c

In [7]:
```
import re
matcher = re.finditer("[^abc]","a7b@9xbs")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
             #1                     #2                  7
             #3                     #4                  @
             #4                     #5                  9
             #5                     #6                  x
             #7                     #8                  s
```

```
1    2    7
3    4    @
4    5    9
5    6    x
7    8    s
```

[a-z] --> any lower case alphabet

In [6]:
```
import re
matcher = re.finditer("[a-z]","a7b@9xbs")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
            #0                     #1                  a
            #2                     #3                  b
            #5                     #6                  x
            #6                     #7                  b
            #7                     #8                  s
```

```
0    1    a
2    3    b
5    6    x
6    7    b
7    8    s
```

[A-Z] --> any upper case alphabet

In [8]:
```
import re
matcher = re.finditer("[A-Z]","A77#234Bsca")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
           #0                     #1                     #A
           #7                     #8                     #B
```

```
0    1    A
7    8    B
```

[A-Za-z] --> any alphabet either upper case or lower case

In [9]:
```
import re
matcher = re.finditer("[A-Za-z]","A77#ab34B")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
```

```
0    1    A
4    5    a
5    6    b
8    9    B
```

[0-9] --> Any digit from 0 to 9

In [22]:
```
import re
matcher = re.finditer("[0-9]","A77#234Bsca")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
        #1                         #2                    7
        #2                         #3                    7
        #4                         #5                    2
        #5                         #6                    3
        #6                         #7                    4
```

```
1    2    7
2    3    7
4    5    2
5    6    3
6    7    4
```

[A-Za-z0-9] --> any alphanumeric symbols

In [10]:
```
import re
matcher = re.finditer("[A-Za-z0-9]","A77#234Bs")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
        #0                         #1                     A
        #1                         #2                     7
        #2                         #3                     7
        #4                         #5                     2
        #5                         #6                     3
        #6                         #7                     4
        #7                         #8                     B
        #8                         #9                     s
```

```
0    1    A
1    2    7
2    3    7
4    5    2
5    6    3
6    7    4
7    8    B
8    9    s
```

[^A-Za-z0-9] --> any special symbol

In [25]:
```
import re
matcher = re.finditer("[^A-Za-z0-9]","A77#2 34Bs")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
       #3                          #4                      3
       #5                          #6

#Note --> Space is also considered as a Special Characters.
```

```
3    4    #
5    6
```

Customization and Practice Problem Based on Character Classes

In [11]:
```
import re
matcher = re.finditer("[^A-Za-z0-9][A-Za-z0-9]","A77#23%a4Bs")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
```

```
3    5    #2
6    8    %a
```

In [12]:
```
#[klm]
import re
matcher = re.finditer("[klm]","A77#2k%alm")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
```

```
5    6    k
8    9    l
9    10    m
```

In [13]:
```
#[^klm]
import re
matcher = re.finditer("[^klm]","abcklm")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
        #0                         #1                  a
        #1                         #2                  b
        #2                         #3                  c
```

```
0    1    a
1    2    b
2    3    c
```

In [14]:
```
#[a-k]
import re
matcher = re.finditer("[a-k]","abcklm")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
        #0                         #1                    a
        #1                         #2                    b
        #2                         #3                    c
        #3                         #4                    k
```

```
0    1    a
1    2    b
2    3    c
3    4    k
```

In [15]:
```
#[A-K]
import re
matcher = re.finditer("[A-K]","ABCKLM")
for i in matcher:
    print(str(i.start())+"    "+str(i.end())+"    "+str(i.group()) )
        #0                         #1                    a
        #1                         #2                    b
        #2                         #3                    c
        #3                         #4                    k
```

```
0    1    A
1    2    B
2    3    C
3    4    K
```

Important Functions Related to RE Module

re.Match() Function

In [ ]:
```
re.Match --> check the string at the begining of the string
         --> re.match() searches only from the beginning of the string and return match object
             if found. But if a match of substring is found somewhere in the middle of the string,
             it returns none.
```

Example

In [39]:
```
import re
s=input("Enter a String :")
m=re.match(s,"abcabdefg")
if m!=None:
    print("Match found")
else:
    print("Not Found")
```

```
Enter a String :cab
Not Found
```

re.FullMatch Function()

In [ ]:
```
re.fullmatch --> This function is used to check whole string.
             --> re.fullmatch() returns a match object if and only if the entre string matches the
                 pattern. Otherwise, it will return None.
```

Example

In [45]:
```
import re
s=input("Enter a String :")
m=re.fullmatch(s,"abcabdefg")
if m!=None:
    print("Match found")
else:
    print("Not Found")
```

```
Enter a String :abcabdefg
Match found
```

re.search () Function

In [ ]:
```
re.search()--> search the given pattern
           --> re.search() method looks for occurrences of the regex pattern inside the entire
               target string and returns the corresponding Match Object instance where the
               match found
```

Example

In [48]:
```
import re
s=input("Enter a String :")
m=re.search(s,"abcabdefg")
if m!=None:
    print("Match found")
    print(m.start())
else:
    print("Not Found")
```

```
Enter a String :cab
Match found
2
```