Day Objectives:

- Regular Expression
  - Constructing regular expression for various use cases
  - Regular expression module and related in python
  - Improving the contacts application with name and phone number validations
- File Handling
  - Test Files
  - Upgrading the contacts application to store contact information in a text file

```
###### Regular Expressions:
- Pattern Matching
- Symbolic Notation of a pattern
    - Pattern : Format which repeats
    - Pattern (RE) -Represents the set of all values that matches the pattern

- To Denote only Regular Expressions:
- [0-9]  ---> Any Digit
- [a-z] ---> Any lower case alphabet
- [2468] ---> All single digit multiples of 2

- ^[0-9]{1}$ --> Only single digit numbers

- ^[0-9]{3}$ --> Only three digit numbers are accepted

- [0-9]*0$ --> All multiples of 10

- ^[1-9][0-9]*0$ -- > this can highlight 10 but not 0

- ^([1-9][0-9]*[05])$|^([5])$     ---> All multiples of 5

- ^[0-9]{10}$ ---> All 10 digit numbers


- if we want to know how many words of (print) are there in the program we use
   (print)   or [p][r][i][n][t]

 - for both upper and lower cases : ([p][r][i][n][t])|(PRINT)

 - [w][o][r][d] or (word)  ---> searching for a "word"

- ^[6-9][0-9]{9}$|^[0][6-9][0-9]{9}$|^[+][9][1][6-9][0-9]{9}$  ---> Validating Phone numbers(india)(start with 9876 followed by 9 digits) including +91

- ^[0-9a-z][0-9a-z_.]{4,13}[0-9a-z][@][a-z0-9]{3,18}[.][a-z]{2,4}$     ---> Email id validation (username@domain.extension)


   - there will be 3 things/rules:
```

```
        - username
            - Length of username : [6,15] "15 characters in username i.e
length (max) {depends on the domain}"
            - No special characters other than _ and.
            - Should not begin and end with _ and .
            - Character Set : all digits and lower case alphabets it can
also have _ and .
        - domain
            - Length of domain : [3,18]
            - No special characters
            - Character set : all digits and lower case alphabets
        - extension
            - Length of extension : [2,4]
            - No special characters
            - Character set : lower case alphabets
```

Domain :

^[0-9a-z]{2,5}$

Username:

^[0-9a-z][0-9a-z_.]{4,13}[0-9a-z]$

Extension:

^[.][a-z]{2,4}$

###

- Any String of length 5 that starts with 'a' and ends with 'z' in between a and z any alphabet.number and special character can be present
- ^[a]...[z]$
- ^[a].*[z]$ ---> (here in between a and z any number of times any character can be repeated) (any string of any length that starts with 'a' and ends with 'z')

In [2]:

```
#Function to validate a phone number
import re

def phoneNumValidator(num):
    pattern ='^[6-9][0-9]{9}$|^[0][6-9][0-9]{9}$|^[+][9][1][6-9][0-9]{9}$'
    if re.match(pattern,str(num)):
        return True
    return False


phoneNumValidator(9876543210)
```

Out[2]: True

In [4]:
```
import re

def emailValidator(email):
    pattern="^[0-9a-z][0-9a-z_.]{4,13}[0-9a-z][@][a-z0-9]{3,18}[.][a-z]{2,4}
    if re.match(pattern,email):
        return True
    return False

emailValidator("abcdef@gmail.com")
```

Out[4]: True

In [12]:
```
contacts= {"abc":[8765432109, 'abc@domain,ext'],"def" : [7896543210, 'def@do

def addContact(name, phone,email):
    #Verify that the contact doesnot already exist in the dictionary
    if name not in contacts:
        print("Name ALready Exists.")
        return
    else:
        if not phoneNumValidator(phone):
            print("Invalid Phone Number")
            return
        if not emailValidator(email):
            print("Invalid email id")
            return
        newcontact = []
        newcontact.append(phone)
        newcontact.append(email)
        contacts[name]= newcontact
        print(name,"added successfully")
    return


addContact("abc", 9876543210,"abcdefgh@gmail.com")
```

```
abc added successfully
```

In [14]:
```
def searchContacts(name):
    if name in contacts:
        print(name)
        print("Phone :",contacts[name][0])
        print("Email :", contacts[name][1])
    else:
        print("%s doesnot exist" % name)
    return


searchContacts("def")
```

```
def
Phone : 7896543210
Email : def@domain.ext
```

In [18]:
```
#Importing a contact
# newContacts is goven as a dictionary
# Merge new contacts with existing contacts


def importContacts(newContacts):
    contacts.update(newContacts)
    print(len(newContacts.keys()), "Contacts added Successfully")
    return


newContacts={"name":[1234568809,"name3@gmail.com"]}
importContacts(newContacts)
contacts
```

```
1 Contacts added Successfully
```

Out[18]:
```
{'abc': [9876543210, 'abcdefgh@gmail.com'],
 'def': [7896543210, 'def@domain.ext'],
 'name': [1234568809, 'name3@gmail.com'],
 'name2': [1234568809, 'name3@gmail.com'],
 'name3': [1234568809, 'name3@gmail.com']}
```

In [23]:
```
#Function to list all contacts

def listAllContacts():
    for contact,info in contacts.items():     #for key,value from dictionary
        print(contact,"\n","Phone :", info[0],"\n","Email :",info[1])
    return

listAllContacts()


```

```
abc
 Phone : 9876543210
 Email : abcdefgh@gmail.com
def
 Phone : 7896543210
 Email : def@domain.ext
name
 Phone : 1234568809
 Email : name3@gmail.com
name2
 Phone : 1234568809
 Email : name3@gmail.com
name3
 Phone : 1234568809
 Email : name3@gmail.com
```

In [22]:
```
contacts.items()

```

Out[22]:
```
dict_items([('abc', [9876543210, 'abcdefgh@gmail.com']), ('def', [7896543210,
'def@domain.ext']), ('name', [1234568809, 'name3@gmail.com']), ('name2', [12345
68809, 'name3@gmail.com']), ('name3', [1234568809, 'name3@gmail.com'])])
```

In [ ]:
```
#Function to edit contact information

def editContact(name,phone,email):
```

In [ ]:
```

```

File Handling in Python:

- File - Document containing some information residing on the permanent storage
- Types - Text, PDF,CSV etc
- File I/O : Channeling I/O data to files
- Default I/O Channels - Input : Keyboard

  - Output: Screen

- Change I/O Channel to files for reading and writing into files
- Read a file - Input from file
- Write to a file - Output to a file
- Read/ write a file : open(filename,mode)

In [27]:
```
#Function to read a file by using split method

def readFile(filename):
    f = open(filename,'r')
    filedata = f.read()
    f.close()
    return filedata


filename = 'DataFiles/data.txt'
filedata = readFile(filename)
for line in filedata.split('\n'):
    print(line)
```

```
Line 1
Line 2
Line 3
```

In [30]:
```
#To print filedata

def printFileDataLines(filename):
    f=open(filename,'r')
    for line in f:
        print(line, end='')   #If we dont want the lines to be in new line w
    return

printFileDataLines(filename)
#print(readFile(filename))
```

```
Line 1
Line 2
Line 3
```

In [32]:
```
#using with operator to open file (we don't need to close file)


def printFileDataLines(filename):
    with open(filename,'r')as f:
        for line in f:
            print(line, end='')   #If we dont want the lines to be in new li
        return

printFileDataLines(filename)
#print(readFile(filename))
```

```
Line 1
Line 2
Line 3
```

In [38]:
```
#Function to write data into a file:

def writeIntoFile(filename,filedata):
    with open(filename,'w') as f:
        f.write(filedata)
    return
filename = 'DataFiles/data.txt'
writeIntoFile(filename,"new data\n")
```

In [40]:
```
# Function to append data to a file

def appendIntoFile(filename,filedata):
    with open(filename,'a') as f:
        f.write(filedata)
    return
filename = 'DataFiles/data.txt'
appendIntoFile(filename,"haii \n hello how are you")
```

In [43]:
```
#Reading Contacts and Writing to a file
def appendIntoFile(filename,filedata):
    with open(filename,'a') as f:
        for line in filedata:
            f.write('\n'+line)
    return
filedata = ["Line4", "Line 5"]
appendIntoFile(filename,filedata)

```

In [ ]:
```

```