1. Write a Python program to read an entire text file

In [1]:
```
import os
```

In [2]:
```
data='''1. Simplest language amongst all languages.
2. Simple to learn, simple to debug.
3. Free & Open source.
4. Globally available.
5. It has a vast library support.'''

file = open('first_file.txt','w+')
file.write(data)
file.seek(0)
print(file.read())
file.close()
```

```
1. Simplest language amongst all languages.
2. Simple to learn, simple to debug.
3. Free & Open source.
4. Globally available.
5. It has a vast library support.
```

2. Write a Python program to read the first n lines of a file

In [3]:
```
file=open('first_file.txt','r')
n=int(input())
x=file.readlines()
print('\n'.join(x[:n]))
```

```
3
1. Simplest language amongst all languages.

2. Simple to learn, simple to debug.

3. Free & Open source.
```

3. Write a Python program to append text to a file and display the text

In [5]:
```
file =open('second_file.txt','a+')
file.write('Data Science ')
file.seek(0)
print(file.read())
file.close()
```

```
Data Science Data Science
```

4. Write a Python program to read the last n lines of a file

In [6]:
```
file = open('first_file.txt','r')
n=int(input())
x=file.readlines()
print('\n'.join(x[-n:]))
```

```
3
3. Free & Open source.

4. Globally available.

5. It has a vast library support.
```

5. Write a Python program to read a file line by line and store it into a list

In [7]:
```
file = open('first_file.txt','r')
x=file.readlines()
file.close()
[i for i in x]
```

Out[7]:
```
['1. Simplest language amongst all languages.\n',
 '2. Simple to learn, simple to debug.\n',
 '3. Free & Open source.\n',
 '4. Globally available.\n',
 '5. It has a vast library support.']
```

6. Write a Python program to count the number of lines in a text file

In [8]:
```
file =open('first_file.txt','r')
x=file.readlines()
file.close()

a=0
for i in x:
    a+=1
print('count of lines:',a)
```

```
count of lines: 5
```

7. Write a Python program to count the frequency of words in a file

In [9]:
```
import re
from collections import Counter
file=open('first_file.txt','r')
x=file.read()
file.close()
xx=re.findall('\S+',x)
print('count of words:',Counter(xx))
```

```
count of words: Counter({'to': 2, '1.': 1, 'Simplest': 1, 'language': 1, 'amongst': 1, 'all': 1, 'languages.': 1, '2.': 1, 'Simple': 1, 'learn,': 1,
'simple': 1, 'debug.': 1, '3.': 1, 'Free': 1, '&': 1, 'Open': 1, 'source.': 1, '4.': 1, 'Globally': 1, 'available.': 1, '5.': 1, 'It': 1, 'has': 1,
'a': 1, 'vast': 1, 'library': 1, 'support.': 1})
```

8. Write a Python program to copy the contents of a file to another file

In [ ]:
```
file =open('1st_file_.txt','w')
file.write('''1.String
2.List
3.Tuple
4.Set
5.Dictonary''')
file.close()

file =open('1st_file_.txt','r')
x=file.readlines()
file1=open('2nd_file.txt','w')
for i in range(0,len(x)):
    file1.write(x[i])
file1.close()
file.close()
```

9. Write a Python script to display the various Date Time formats - Go to the editor

a) Current date and time b) Current year c) Month of year d) Week number of the year e) Weekday of the week f) Day of year g) Day of the month h) Day of week

In [10]:
```
import datetime
```

In [11]:
```
current_date = datetime.datetime.now()

print('Current date and time:',current_date)
print('Current year:',current_date.strftime('%Y'))
print('Month of year:',current_date.strftime('%B'))
print('Week number of the year:',current_date.strftime('%U'))
print('Weekday of the week:',current_date.strftime('%u'))
print('Day of year:',current_date.strftime('%j'))
print('Day of the month:',current_date.strftime('%d'))
print('Day of week:',current_date.strftime('%w'))
```

```
Current date and time: 2022-08-29 21:28:04.858651
Current year: 2022
Month of year: August
Week number of the year: 35
Weekday of the week: 1
Day of year: 241
Day of the month: 29
Day of week: 1
```

10. Write a Python program to determine whether a given year is a leap year

In [12]:
```
n=int(input('given year is : '))
if (n%400==0) and (n%4==0 and n%100!=0):
    print("given year {} is leap year".format(n))
else:
    print("given year {} is leap year".format(n))
```

```
given year is : 2020
given year 2020 is leap year
```

11. Write a Python program to convert a string to datetime. Go to the editor ○ Sample String : Jan 1 2014 2:43PM ○ Expected Output : 2014-07-01 14:43:00

In [13]:
```
date='Jan 1 2014 2:43PM'
changed_date=datetime.datetime.strptime(date,'%b %d %Y %H:%M%p')
print(changed_date)
```

```
2014-01-01 02:43:00
```

12. Write a Python program to subtract five days from current date

In [14]:
```
current_date=datetime.datetime.now()
print('Current Date:',current_date)
that_day=current_date-datetime.timedelta(days=5)
print('That Day:',that_day)
```

```
Current Date: 2022-08-29 21:28:20.254338
That Day: 2022-08-24 21:28:20.254338
```

13. Write a Python program to print yesterday, today, tomorrow

In [15]:
```
today=datetime.datetime.now()
yest=today-datetime.timedelta(days=1)
tomo=today+datetime.timedelta(days=1)
print('Yesterday:',yest)
print('Today:',today)
print('Tomorrow:',tomo)
```

```
Yesterday: 2022-08-28 21:28:22.290525
Today: 2022-08-29 21:28:22.290525
Tomorrow: 2022-08-30 21:28:22.290525
```

14. Write a Python program to print next 5 days starting from today

In [16]:
```
today=datetime.datetime.now()
for i in range(6):
    print(today+datetime.timedelta(days=i))
```

```
2022-08-29 21:28:25.022005
2022-08-30 21:28:25.022005
2022-08-31 21:28:25.022005
2022-09-01 21:28:25.022005
2022-09-02 21:28:25.022005
2022-09-03 21:28:25.022005
```

15. Write a Python program to drop microseconds from DateTime

In [17]:
```
today=datetime.datetime(2022,8,16,16,30,49)
print(today)
```

```
2022-08-16 16:30:49
```

In [18]:
```
today=datetime.datetime.today().replace(microsecond=0)
print(today)
```

```
2022-08-29 21:28:29
```

16. Write a Python program to find the date of the first Monday of a given week

In [19]:
```
current_date=datetime.datetime.today()
current_day=current_date.strftime('%w')
if current_day==0:
    print('date of the first monday is ',(current_date+datetime.timedelta(days=1)).date())
elif current_day==1:
    print('date of the first monday is ',current_date.date())
else:
    print('date of the first monday is ',(current_date-datetime.timedelta(days=(int(current_day)-1))).date())
```

```
date of the first monday is  2022-08-29
```

17. Write a Python program to select all the Sundays of a specified year

In [21]:
```
initial_date=datetime.datetime(2022,1,1)
for i in range(365):
    if int((initial_date+datetime.timedelta(days=i)).strftime('%w'))==0:
        print(initial_date+datetime.timedelta(days=i))
```

```
2022-01-02 00:00:00
2022-01-09 00:00:00
2022-01-16 00:00:00
2022-01-23 00:00:00
2022-01-30 00:00:00
2022-02-06 00:00:00
2022-02-13 00:00:00
2022-02-20 00:00:00
2022-02-27 00:00:00
2022-03-06 00:00:00
2022-03-13 00:00:00
2022-03-20 00:00:00
2022-03-27 00:00:00
2022-04-03 00:00:00
2022-04-10 00:00:00
2022-04-17 00:00:00
2022-04-24 00:00:00
2022-05-01 00:00:00
2022-05-08 00:00:00
2022-05-15 00:00:00
2022-05-22 00:00:00
2022-05-29 00:00:00
2022-06-05 00:00:00
2022-06-12 00:00:00
2022-06-19 00:00:00
2022-06-26 00:00:00
2022-07-03 00:00:00
2022-07-10 00:00:00
2022-07-17 00:00:00
2022-07-24 00:00:00
2022-07-31 00:00:00
2022-08-07 00:00:00
2022-08-14 00:00:00
2022-08-21 00:00:00
2022-08-28 00:00:00
2022-09-04 00:00:00
2022-09-11 00:00:00
2022-09-18 00:00:00
```

```
2022-09-25 00:00:00
2022-10-02 00:00:00
2022-10-09 00:00:00
2022-10-16 00:00:00
2022-10-23 00:00:00
2022-10-30 00:00:00
2022-11-06 00:00:00
2022-11-13 00:00:00
2022-11-20 00:00:00
2022-11-27 00:00:00
2022-12-04 00:00:00
2022-12-11 00:00:00
2022-12-18 00:00:00
2022-12-25 00:00:00
```

18. Write a Python program to create a file and write some text and rename the file name

In [ ]:
```
import os
file=open('18Q_file.txt','w')
file.write('Python for Data Science')
file.close()

os.rename('18Q_file.txt','Q_18_file.txt')
```

19. What are different methods available in the OS module for creating a directory?

```
we can create a directory by using os module by using os.mkdir.
it will create a empty directory/folder.
syntax:
    os.mkdir(folder_name)
we can also create empty folder by using path & folder name
```

20. Explain os.listdir() method.

```
os.listdir:
```

```
    os.listdir gives the list of all the files & folders in a particular given folder.
    In os.listdir method we give path of particular folder then it give the outcome of all the folders and files are in that folder in list format.
    syntax:
        os.listdir(path)
        os.listdir() - for current location
```

21. What are different methods for removing directories and files in Python?

```
we can remove a file and directories using os module.
we need to import os.
we can use os.remove for removing the file and directories.
syntax:
    os.remove(file_name or folder_name or path)

we can also remove a file and directories using os.rmdir.
But it only works on the empty folders.
syntax:
    os.rmdir(folder_name or path)

By using shutil.rmtree we can remove a folder.
we need to import shutil.
we can remove empty as well as having files in it.
syntax:
    shutil.rmtree(folder_name or path)
```

22. What are exceptions in Python?

```
An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the
program's instructions.An exception is a Python object that represents an error.
Some Exceptions in python:
SyntaxError
IndentationError
FileNotFoundError
KeyError
NameError
valueError
ZeroDivisionErroro
```

```
TypeError
IndexError
FileExistError
EOFError
ImportError
KeyboardInterrupt
```

23. What are Built-in exceptions?

```
        Exception : Description
  ArithmeticError : Raised when an error occurs in numeric calculations
   AssertionError : Raised when an assert statement fails
   AttributeError : Raised when attribute reference or assignment fails
        Exception : Base class for all exceptions
         EOFError : Raised when the input() method hits an "end of file" condition (EOF)
FloatingPointError: Raised when a floating point calculation fails
    GeneratorExit : Raised when a generator is closed (with the close() method)
ImportError Raised: when an imported module does not exist
 IndentationError : Raised when indendation is not correct
        IndexError: Raised when an index of a sequence does not exist
        KeyError  : Raised when a key does not exist in a dictionary
KeyboardInterrupt : Raised when the user presses Ctrl+c, Ctrl+z or Delete
       LookupError: Raised when errors raised cant be found
       MemoryError: Raised when a program runs out of memory
        NameError : Raised when a variable does not exist
NotImplementedError:Raised when an abstract method requires an inherited class to override the method
          OSError : Raised when a system related operation causes an error
    OverflowError : Raised when the result of a numeric calculation is too large
   ReferenceError : Raised when a weak reference object does not exist
    RuntimeError  : Raised when an error occurs that do not belong to any specific expections
    StopIteration : Raised when the next() method of an iterator has no further values
      SyntaxError : Raised when a syntax error occurs
        TabError  : Raised when indentation consists of tabs or spaces
      SystemError : Raised when a system error occurs
       SystemExit : Raised when the sys.exit() function is called
        TypeError : Raised when two different types are combined
       ValueError : Raised when there is a wrong value in a specified data type
ZeroDivisionError : Raised when the second operator in a division is zero
UnboundLocalError : Raised when a local variable is referenced before assignment
    UnicodeError  : Raised when a unicode problem occurs
UnicodeEncodeError: Raised when a unicode encoding problem occurs
```

```
UnicodeDecodeError: Raised when a unicode decoding problem occurs
UnicodeTranslateError : Raised when a unicode translation problem occurs
```

24. What are User-defined Exceptions?

```
User defined Exceptions means user can create a error in the code.
By using raise user can create a error.
We give a codition in the code something which we dont want or know if this is availble, if the codition is ture then we can use raise function
after that condition and create and an error in error we can write anything eq.don't want
```

25. When would you not use try-except?

```
A try block allows you to handle an expected error. The except block should only catch exceptions you are
prepared to handle. If you handle an unexpected error, your code may do the wrong thing and hide bugs.
```

26. Can try-except catch the error if a file can’t be opened?

```
You can't open a file with try-except.to open the file like you usually would
and catch exceptions if they occur using try-except.
```