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

In [39]:

```
file1=open("sample.txt",'w')
file1.write('''good morning welcome to python
            hi
            bye
            hello,hi welcome to python''')
file1.close
file1=open("sample.txt",'r')
file1.read()
```

Out[39]:

```
'good morning welcome to python\n            hi\n            bye\n
hello,hi welcome to python'
```

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

In [9]:

```
with open('sample.txt','r') as file1:
    d=file1.readlines()
    print(d)
```

```
['good morning welcome to python\n', '            hi\n', '            bye
\n', '            hello']
```

In [20]:

```
n=int(input("enter no of n lines you want to read: "))
with open('sample.txt','r') as file1:
    for i in (file1.readlines() [:n]):
        print(i,end=" ")
```

```
enter no of n lines you want to read: 2
good morning welcome to python
            hi
```

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

In [30]:

```
file2=open("sample2.txt",'a')
file2.write("hi")
file2.close()
f2=open("sample2.txt")
d=f2.read()
print(d)
f2.close()
```

```
hi
```

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

In [31]:

```
n=int(input("enter no of last n lines you want to read: "))
with open('sample.txt','r') as file1:
    for i in (file1.readlines() [-n:]):
        print(i,end=" ")
```

```
enter no of last n lines you want to read: 3
            hi
             bye
             hello
```

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

In [34]:

```
list1=[]
with open('sample.txt','r') as file1:
    for i in file1.readlines():
        list1.append(i)
print(list1)
len(list1)
```

```
['good morning welcome to python\n', '            hi\n', '            bye
\n', '            hello']
```

Out[34]:

```
4
```

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

In [37]:

```
count=0
with open('sample.txt','r') as file1:
    for i in file1.readlines():
        count+=1
print("the total no of lines present in text file is ",count)
```

```
the total no of lines present in text file is  4
```

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

In [40]:

```
from collections import Counter
with open("sample.txt") as f1:
    d=f1.read().split()
    counts=Counter(d)
    print(counts)
```

```
Counter({'welcome': 2, 'to': 2, 'python': 2, 'good': 1, 'morning': 1, 'hi':
1, 'bye': 1, 'hello,hi': 1})
```

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

In [41]:

```
with open ("sample.txt",'r') as f1,open("sample3.txt",'w') as f2:
    for i in f1:
        f2.write(i)
f3=open("sample3.txt",'r')
d=f3.read()
print(d)
f3.close()
```

```
good morning welcome to python
            hi
            bye
            hello,hi welcome to python
```

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 [45]:

```
import datetime
current_date_time=datetime.datetime.now()
current_date_time
```

Out[45]:

```
datetime.datetime(2022, 8, 23, 19, 3, 26, 98305)
```

In [50]:

```
current_year=datetime.datetime.today().year
current_year
```

Out[50]:

```
2022
```

In [51]:

```
current_month=datetime.datetime.today().month
current_month
```

Out[51]:

```
8
```

In [54]:

```
current_date_time=datetime.datetime.today()
d=current_date_time.strftime("%V")
print("current week of the year:",d)
```

```
current week of the year: 34
```

In [56]:

```
current_date_time=datetime.datetime.today()
d=current_date_time.strftime("%w")
print("weekday of the week:",d)
```

```
weekday of the week: 2
```

In [57]:

```
current_date_time=datetime.datetime.today()
d=current_date_time.strftime("%j")
print("day of the year:",d)
```

```
day of the year: 235
```

In [58]:

```
current_date_time=datetime.datetime.today()
d=current_date_time.strftime("%d")
print("day of the month:",d)
```

```
day of the month: 23
```

In [59]:

```
current_date_time=datetime.datetime.today()
d=current_date_time.strftime("%a")
print("day of the week:",d)
```

```
day of the week: Tue
```

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

In [60]:

```
year=2022
if(year%4==0):
    print("leap year")
else:
    print("not a leap year")
```

```
not a 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 [65]:

```
str_form='Jan 1 2014 2:43PM'
date_form='%b %d %Y %I:%M%p'
dateformat=datetime.datetime.strptime(str_form,date_form)
print(dateformat)
```

```
2014-01-01 14:43:00
```

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

In [69]:

```
import datetime
current_date_time=datetime.datetime.today()
new_date=current_date_time-datetime.timedelta(days=5)
print(new_date)
```

```
2022-08-18 19:37:58.701266
```

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

In [74]:

```
current_datetime=datetime.datetime.today()
print("current_datetime:",current_datetime)
yesterday_datetime=current_datetime+datetime.timedelta(days=-1)
print('yesterday_datetime:',yesterday_datetime)
tomarrow_datetime=current_datetime+datetime.timedelta(days=1)
print('tomarrow_datetime:',tomarrow_datetime)
```

```
current_datetime: 2022-08-23 19:54:48.859961
yesterday_datetime: 2022-08-22 19:54:48.859961
tomarrow_datetime: 2022-08-24 19:54:48.859961
```

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

In [83]:

```
today=datetime.date.today()
print(current_datetime)
for i in range(1,6):
    next5_datetime=today+datetime.timedelta(days=i)
    print(next5_datetime)
```

```
2022-08-23
2022-08-24
2022-08-25
2022-08-26
2022-08-27
2022-08-28
```

15. Write a Python program to drop microseconds from DateTime

In [85]:

```
dt=datetime.datetime.today().replace(microsecond=0)
print(dt)
```

```
2022-08-23 20:05:09
```

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

In [95]:

```
 import time
print(time.asctime(time.strptime('2022 34 1', '%Y %W %w')))
```

```
Mon Aug 22 00:00:00 2022
```

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

In [97]:

```
from datetime import date, timedelta

def all_sundays(year):
# January 1st of the given year
       dt = date(year, 1, 1)
# First Sunday of the given year
       dt += timedelta(days = 6 - dt.weekday())
       while dt.year == year:
          yield dt
          dt += timedelta(days = 7)

for s in all_sundays(2020):
   print(s)
```

```
2020-01-05
2020-01-12
2020-01-19
2020-01-26
2020-02-02
2020-02-09
2020-02-16
2020-02-23
2020-03-01
2020-03-08
2020-03-15
2020-03-22
2020-03-29
2020-04-05
2020-04-12
2020-04-19
2020-04-26
2020-05-03
2020-05-10
2020-05-17
2020-05-24
2020-05-31
2020-06-07
2020-06-14
2020-06-21
2020-06-28
2020-07-05
2020-07-12
2020-07-19
2020-07-26
2020-08-02
2020-08-09
2020-08-16
2020-08-23
2020-08-30
2020-09-06
2020-09-13
2020-09-20
2020-09-27
2020-10-04
2020-10-11
2020-10-18
2020-10-25
2020-11-01
```

```
2020-11-08
2020-11-15
2020-11-22
2020-11-29
2020-12-06
2020-12-13
2020-12-20
2020-12-27
```

In [ ]:

```
```

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

In [110]:

```
import os
ff1=open("newfile.txt",'w')
ff1.write("good morning")
ff1.close()
os.rename("newfile.txt","kfile.txt")
```

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

There are different methods available in the OS module for creating a director. These are –

os.mkdir()
os.makedirs()
Using os.mkdir()
os.mkdir() method in Python is used to create a directory named path with the specified numeric mode. This method raise FileExistsError if the directory to be created already exists.

Syntax: os.mkdir(path)

Using os.makedirs()
os.makedirs() method in Python is used to create a directory recursively. That means while making leaf directory if any intermediate-level directory is missing, os.makedirs() method will create them all.

20. Explain os.listdir() method.

listdir() method in python is used to get the list of all files and directories in the specified directory.
If we don't specify any directory,
then list of files and directories in the current working directory will be returned.

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

Various methods provided by Python are –

```
Using os.remove()
Using os.rmdir()
Using shutil.rmtree()

1)os.remove() :-
    This method in Python is used to remove or delete a file path.
This method can not remove or delete a directory. If the specified path is a directory
then OSError
will be raised by the method.
Syntax of os.remove(path)

2)os.rmdir():-
This method in Python is used to remove or delete an empty directory. OSError will be
raised if the specified path
is not an empty directory.

Syntax of os.rmdir(path)

3)shutil.rmtree() is used to delete an entire directory tree, a path must point to a
directory
(but not a symbolic link to a directory).

Syntax of shutil.rmtree()
```

```
22. What are exceptions in Python?
```

```
An Exception is an Event, which occurs during the execution of the program. It is also
known as a run time error.
When that error occurs, Python generates an exception during the execution and that can
be handled, which avoids your
program to interrupt.

Exception handling with try, except, else, and finally
Try: This block will test the excepted error to occur
Except:  Here you can handle the error
Else: If there is no exception then this block will be executed
Finally: Finally block always gets executed either exception is generated or not
```

```
23. What are Built-in exceptions?
```

```
Illegal operation can raise exceptions. There ar plenty built-in exceptionsin pyt
when the corresponding error occurs
When can check all the built-in exceptions using built-in local function
```

In [2]:

```
print(dir(locals()['__builtins__']))
```

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'Bl
ockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProc
essError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedEr
ror', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis',
'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundEr
ror', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'Im
portError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedE
rror', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError',
'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryErr
or', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'P
endingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'Recursi
onError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarni
ng', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning',
'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeErro
r', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'Unicod
eError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueErr
or', 'Warning', 'WindowsError', 'ZeroDivisionError', '__IPYTHON__', '__build
_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__',
'__package__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bre
akpoint', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile',
'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'display', 'div
mod', 'enumerate', 'eval', 'exec', 'execfile', 'filter', 'float', 'format',
'frozenset', 'get_ipython', 'getattr', 'globals', 'hasattr', 'hash', 'help',
'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'lic
ense', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'objec
t', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'range', 'repr', 'reve
rsed', 'round', 'runfile', 'set', 'setattr', 'slice', 'sorted', 'staticmetho
d', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
```

24. What are User-defined Exceptions?

In [ ]:

```
 Programmer may name their own execption by creating a new execption class.
Exception needs to derived from the Exception class either directly or indirectly
```

In [4]:

```
try:
 a=int(input('a:'))
 b=int(input('b:'))
 div=a/b
 print(div)
except ZeroDivisionError as msg:
 print(msg)
```

```
a:10
b:5
2.0
```

25. When would you not use try-except?

In [ ]:

```
 The try block lets you test a block of code for errors. The except block lets you handle t
The else block lets you execute code when there is no error.
```

In [ ]:

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

In [ ]:

```
try:
 with open("file.txt","r") as f:
 #operations
except IOError:
 #do what you want if there is an error with the file opening
```