file handling

file modes

r - read

w - write ,create a file when not present or overwrite the file

In [6]:
```
import os
file = open("C:/Users/Administrator/Desktop/test_file.txt","r")
lst = file.read()
print(lst)
file.close()
```

```
hello how are you
```

In [20]:
```
import os
os.getenv("C:/Users/Administrator")
file = open("my_file.txt","w")
file.write("this is first module \n")
file.write("this is second module")
file.close()
```

In [48]:
```
file = open("my_file.txt","r")
#lst = file.read()
print(file.tell())
print(file.read(10)) # pointer will read upto 9 charachters
print(file.tell()) # used to get the current pointer position
print(file.read())
file.seek(5) # used to make pointer to reassign to the specifiesd position
print(file.tell())
print(file.read(10))
file.close()
```

```
0
this is fi
10
rst module
this is second module
5
is first m
```

In [43]:
```
st = "this is my string"
print(st[5:10])
# after reading the substring the pointer come back to zero
```

```
is my
```

In [58]:
```
with open("my_file.txt","r") as file:
    print(file.readline())
    lst = file.readlines()
```

```
    for line in lst:
        print(line)
```

```
this is first module

this is second module
```

In [60]:
```
# renaming and deleting files
import os
os.rename("my_file.txt","new_file.txt")
with open("new_file.txt","r") as file:
    print(file.readlines())
    os.remove("new_file.txt")
    file.read()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Cell In[60], line 3
      1 # renming and deleting files
      2 import os
----> 3 os.rename("my_file.txt","new_file.txt")
      4 with open("new_file.txt","r") as file:
      5     print(file.readlines())

FileNotFoundError: [WinError 2] The system cannot find the file specified: 'my_fil
e.txt' -> 'new_file.txt'
```

In [67]:
```
# directory
import os
print(os.getcwd())# used to get the current working directory
print(os.listdir())
os.chdir("C:/Users/Administrator/Desktop") # used to change the current working d
print(os.getcwd())
```

```
C:\Users\Administrator\Desktop
['desktop.ini', 'Microsoft Edge.lnk', 'MongoDBCompass.lnk', 'Readme - Shortcut.ln
k', 'test_file.txt']
C:\Users\Administrator\Desktop
```

In [65]:
```
import os
os.listdir()
```

Out[65]:
```
['desktop.ini',
 'Microsoft Edge.lnk',
 'MongoDBCompass.lnk',
 'Readme - Shortcut.lnk',
 'test_file.txt']
```

In [71]:
```
# create new directory
import os
os.mkdir("directory_100")
print(os.listdir())
```

```
['desktop.ini', 'directory_1', 'directory_100', 'Microsoft Edge.lnk', 'MongoDBCompa
ss.lnk', 'Readme - Shortcut.lnk', 'test_file.txt']
```

In [2]:
```
# Modules and packages
```

In [1]:
```
from datetime import datetime
datetime.now()
```

Out[1]:
```
datetime.datetime(2026, 2, 20, 13, 55, 59, 216954)
```

In [3]:
```
import math as m # renaming using alias
print(m.pi)
```

```
3.141592653589793
```

In [9]:
```
# importing all avaulable functions from module
from math import *
print(pi)
```

```
3.141592653589793
```

python unit testing

In [3]:
```
print("---------------------------------------USERNAME VALIDATION----------------
print("\n")
def name_validation(name):
    print("---------------------------NAME LENGTH VALIDATION---------------------
    print()
    if len(name) >= 8 and len(name) <= 50:
        return True
def test_name_validation():
    assert name_validation("abcdef") == True
    print("----------------------------TEST PASSED-------------------------------

def name_str(name):
    print("---------------------------NAME STRING VALIDATION---------------------
    print()
    if isinstance(name,str):
        return True
def test_str_validation():
    assert name_str({}) == True
    print("----------------------------TEST PASSED-------------------------------
def check_name():
    try:
       test_name_validation()
    except  AssertionError:
        print("TEST FAILED : USERNAME MUST HAVE ATLEAST 8 CHARACTER AND MAXIMUM MU


def first_char_validation(name):
    print("---------------------------FIRST CHARACTER VALIDATION-----------------
    print()
    if str(name[0]).isupper():
        return True
def test_char_validation():
    assert first_char_validation("abcdefgqwerdf") == True
    print("----------------------------TEST PASSED-------------------------------

def check_char():
    try:
       test_char_validation()


    except  AssertionError:
        print("TEST FAILED : FIRST CHARACTER MUST BE CAPITAL")
def check_str():
```

```
    try:
        test_str_validation()
    except:
        print("TEST FAILED: USERNAME MUST BE STRING")


def main():
     check_str()
     check_name()
     check_char()

main()
```

```
---------------------------------------USERNAME VALIDATION----------------------
-----------------


--------------------------NAME STRING VALIDATION-----------------------------

TEST FAILED: USERNAME MUST BE STRING
--------------------------NAME LENGTH VALIDATION-----------------------------

TEST FAILED : USERNAME MUST HAVE ATLEAST 8 CHARACTER AND MAXIMUM MUST BE 12
---------------------------FIRST CHARACTER VALIDATION--------------------

TEST FAILED : FIRST CHARACTER MUST BE CAPITAL
```

In [ ]:
```
import unittest

class TestUserName(unittest.TestCase):
    def TestNameLength(self):
        self.assertE
```