

Files

Opening Files

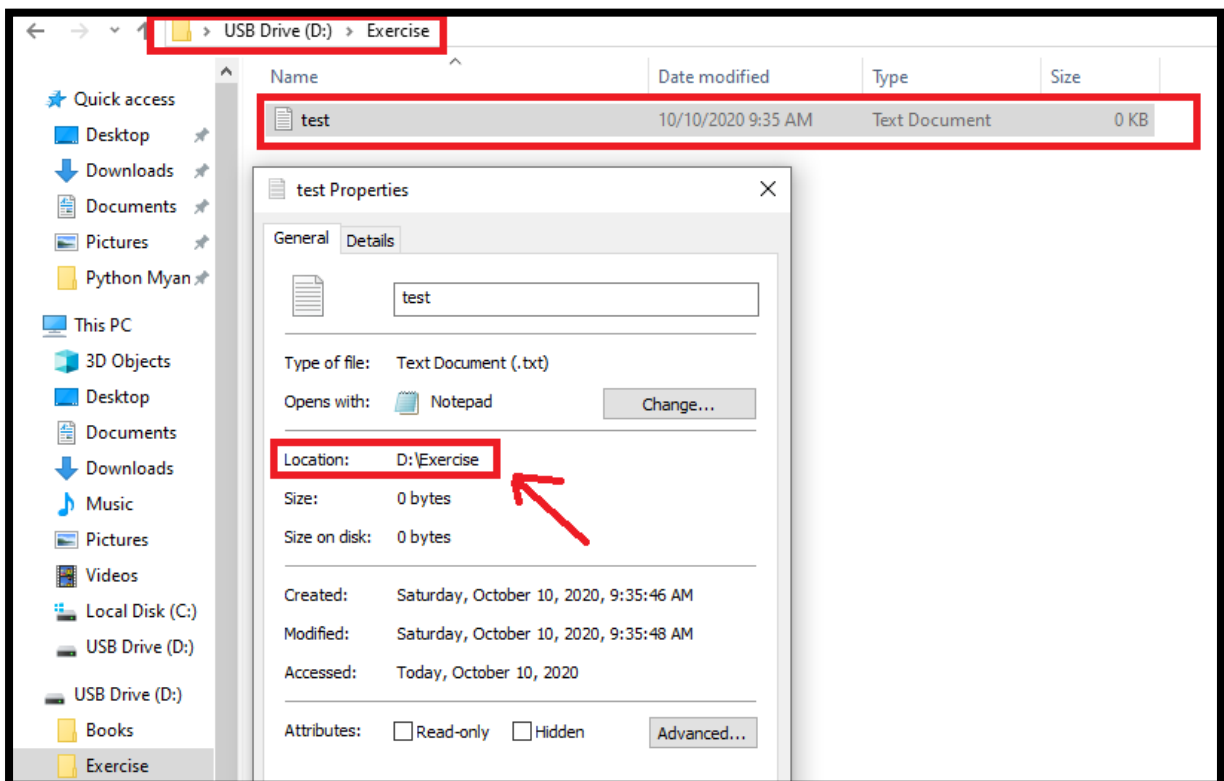
files တွေထဲမှာပါဝင်တဲ့အရာ (content) တွေကို ဖတ်ဖို့နဲ့ ရေးသားဖို့အတွက် Python ကို အသုံးပြုနိုင်ပါတယ်။ text files တွေက ဖတ်ရှုခြင်း၊ ရေးသားခြင်း (read, write) လုပ်ဖို့အတွက် အလွယ်ဆုံးဖြစ်ပါတယ်။ file တစ်ခုကို တည်းဖြတ်ပြင်ဆင်မှု edit မလုပ်ခင် file ကို မဖြစ်မနေ ဖွင့်ရပါမယ်။ **open** function ကို အသုံးပြုပါတယ်။

```
test_1.py * x
1 myfile = open("filename.txt")
2
```

open function ကိုအသုံးပြုတဲ့အခါမှာ **file ရှိတဲ့နေရာကိုပြတဲ့ လမ်းကြောင်း (path)** နဲ့ **file အမည်တို့ကို** ထည့်သွင်းအသုံးပြုရတာဖြစ်ပါတယ်။

မိမိဖွင့်လိုတဲ့ file က ပရိုဂရမ်နဲ့ folder တစ်ခုထဲမှာ အတူတူရှိနေတယ်ဆိုရင် အပေါ်ကပုံလို လမ်းကြောင်း path ကို ထည့်ပေးစရာမလိုဘဲ file နာမည်နဲ့ပဲ အသုံးပြုရတာ ဖြစ်ပါတယ်။

file လမ်းကြောင်းကို မိမိဖွင့်ချင်တဲ့ file ရဲ့ properties ကို နှိပ်ပြီး ကြည့်နိုင်ပါတယ်။



ဒီ text ဖိုင်ကို ဖွင့်ပါမယ်။

```
test_1.py ×
1 myfile = open("D:\Exercise/test.txt")

Shell ×
>>> %Run test_1.py
>>> |
```

Mode

file တစ်ခုကို ဖွင့်တဲ့အခါမှာ မိမိဖွင့်ချင်တဲ့အနေအထားပုံစံ (mode) ကို **open** function ထဲမှာ ဒုတိယ **argument** အနေနဲ့ ထည့်သွင်းပြီး ဖွင့်နိုင်ပါတယ်။

“r” ဆိုတာက **read mode** ဖြစ်ပြီး သူက **default mode** လည်း ဖြစ်ပါတယ်။

“w” ဆိုတာက **write mode** ဖြစ်ပြီး file ထဲမှာရှိတဲ့ contents တွေကို အသစ်ပြန်ရေးဖို့အတွက် အသုံးပြုပါတယ်။

“a” ဆိုတာက **append mode** ဖြစ်ပြီး file ထဲမှာရှိတဲ့ content ရဲ့နောက်မှာ နောက်ထပ်အသစ်တွေ ထပ်မံရေးသားထည့်သွင်းဖို့အတွက် အသုံးပြုပါတယ်။

“b” ဆိုတာက **binary mode** ဖြစ်ပြီး text file မဟုတ်တဲ့ ဖိုင်တွေ ဥပမာ ဓာတ်ပုံနဲ့ အသံဖိုင်တွေအတွက် အသုံးပြုပါတယ်။

```
test_1.py * ×
1 # write mode
2 open("filename.txt", "w")
3
4 # read mode
5 open("filename.txt", "r")
6 open("filename.txt")
7
8 # binary write mode
9 open("filename.txt", "wb")
10
11
12
```

+ ကိုအသုံးပြုပြီး ဖိုင်တွေကို နောက်ထပ် mode တွေနဲ့အသုံးပြုလို့ရအောင် ပြုလုပ်နိုင်ပါတယ်။

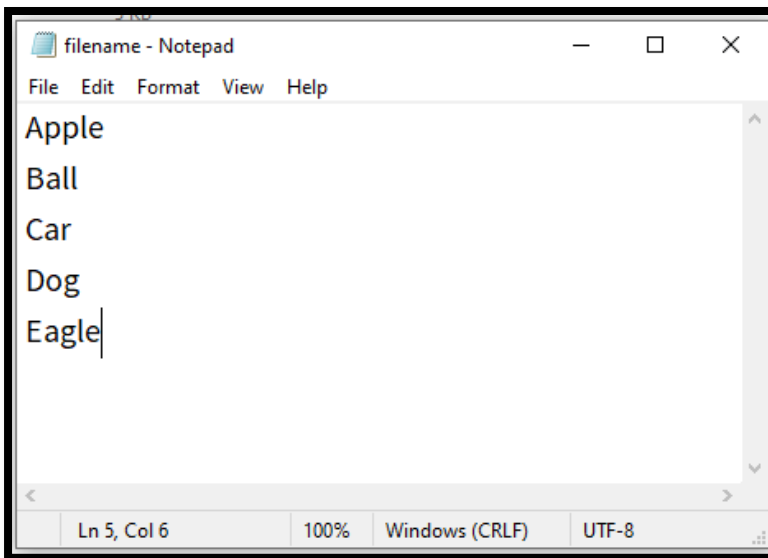
ဥပမာ - r+ ဆိုလို့ရှိရင် ဖိုင်ကို read အတွက်ရော write အတွက်ပါ ဖွင့်လိုက်တာ ဖြစ်ပါတယ်။

ဖိုင်တစ်ခုကိုဖွင့်ပြီး မိမိလုပ်စရာရှိတာတွေ လုပ်ပြီးတာနဲ့ ပြန်ပိတ်ဖို့ လိုပါတယ်။ `close()` method ကို အသုံးပြုပါတယ်။

```
test_1.py * x
1 file = open("filename.txt", "w")
2 # do stuff to the file
3 file.close()
4
```

Reading Files

text ဖိုင်ထဲက content တွေကို ဖတ်ဖို့အတွက် `read` method ကို အသုံးပြုပါတယ်။ (*method* ဆိုတာက *function* နဲ့သဘောတရားချင်းတူပါတယ်။ သို့ပေမဲ့ *method* က *object* အပေါ်မှာ အလုပ်လုပ်တာ ဖြစ်ပါတယ်။ *object oriented programming* အကြောင်းကို သင်တဲ့အခါကျရင် *method* နဲ့ *object* ကို ပိုပြီး နားလည်သွားမှာပါ။)



ဒီအပေါ်ပုံက filename ဆိုတဲ့ text file ထဲမှာရှိနေတဲ့ စာသား content တွေ ဖြစ်ပါတယ်။ ဒီဖိုင်ကို `read` နဲ့ ဖတ်ပါမယ်။

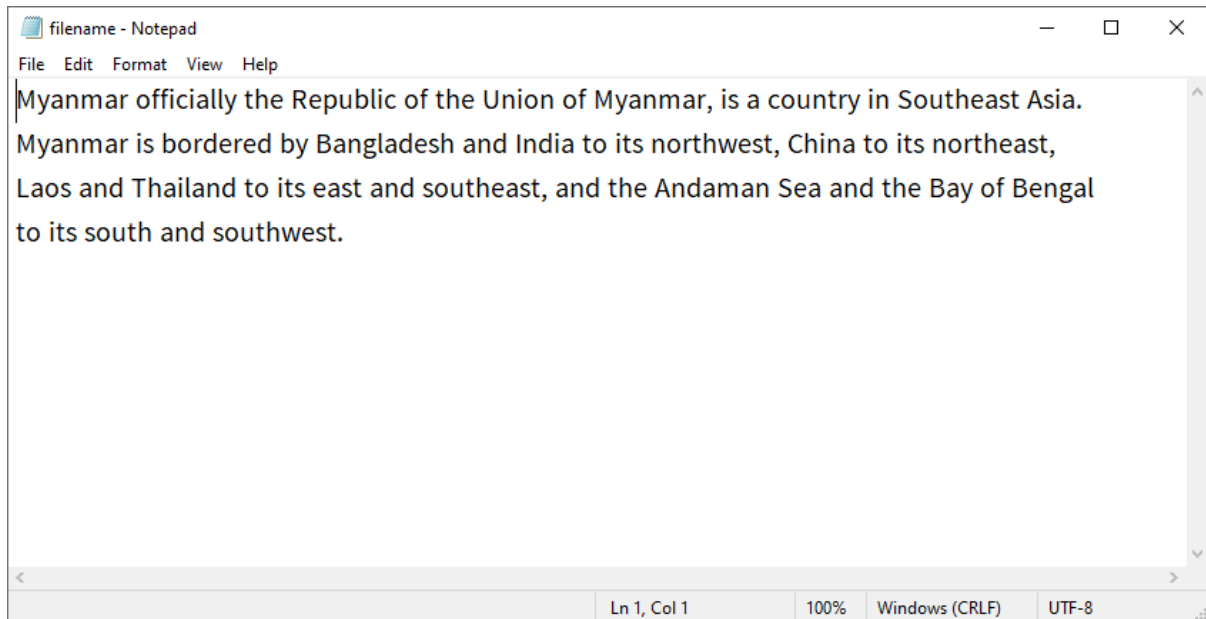
```

test_1.py ×
1 file = open("filename.txt", "r")
2 cont = file.read()
3 print(cont)
4 file.close()
5

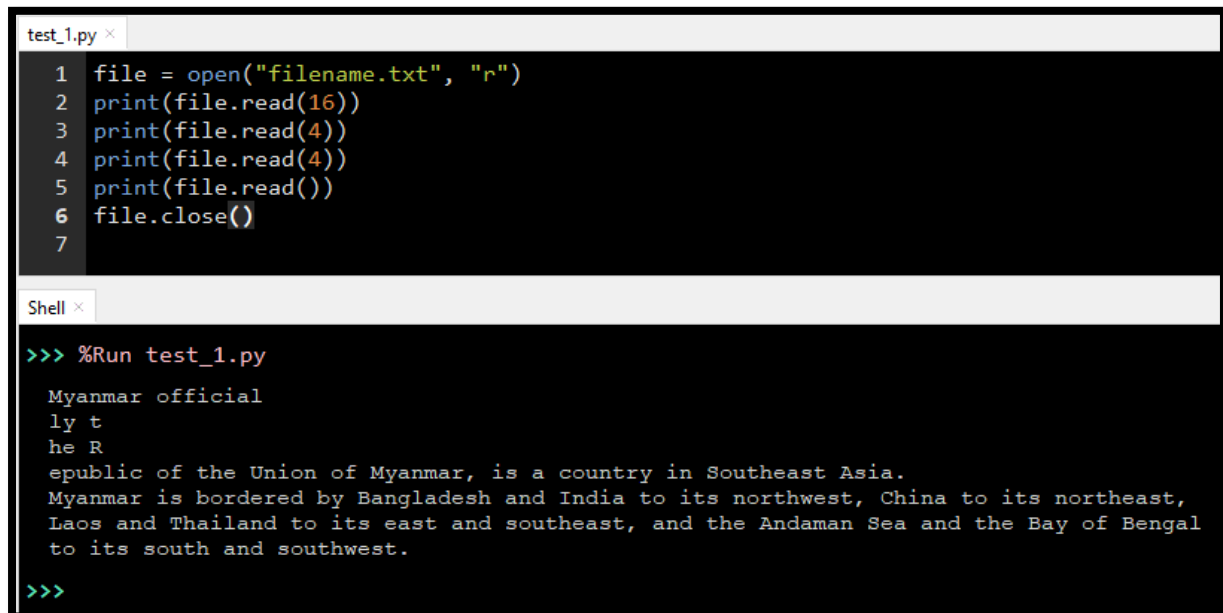
Shell ×
>>> %Run test_1.py
Apple
Ball
Car
Dog
Eagle
>>>

```

file ထဲက content တွေကို ဖတ်တဲ့အခါမှာ အကုန်လုံးကိုမဖတ်ဘဲ မိမိဖတ်စေချင်တဲ့ ပမာဏလောက်ပဲ ဖတ်ပေးစေချင်တယ်ဆိုရင် **read** function ထဲမှာ **argument** အဖြစ် နံပါတ်တွေကို ထည့်သွင်းပြီး ဖတ်ခိုင်းနိုင်ပါတယ်။ argument မပါတော့တဲ့အချိန်မှာတော့ file ထဲမှာ မဖတ်ရသေးဘဲ ကျန်သမျှကို ဖတ်ပေးမှာ ဖြစ်ပါတယ်။



```
filename - Notepad
File Edit Format View Help
Myanmar officially the Republic of the Union of Myanmar, is a country in Southeast Asia.
Myanmar is bordered by Bangladesh and India to its northwest, China to its northeast,
Laos and Thailand to its east and southeast, and the Andaman Sea and the Bay of Bengal
to its south and southwest.
Ln 1, Col 1 100% Windows (CRLF) UTF-8
```



```
test_1.py x
1 file = open("filename.txt", "r")
2 print(file.read(16))
3 print(file.read(4))
4 print(file.read(4))
5 print(file.read())
6 file.close()
7

Shell x
>>> %Run test_1.py
Myanmar official
ly t
he R
epublic of the Union of Myanmar, is a country in Southeast Asia.
Myanmar is bordered by Bangladesh and India to its northwest, China to its northeast,
Laos and Thailand to its east and southeast, and the Andaman Sea and the Bay of Bengal
to its south and southwest.
>>>
```

file ထဲမှာရှိသမျှ content တွေကို ဖတ်ပြီးပြီဆိုရင် နောက်တစ်ခါ ထပ်ဖတ်ဖို့ကြိုးစားတဲ့အခါမှာ ဘာမှမပါဘဲ string ရလဒ်ပဲ ပြန်ရမှာပါ။ ဘာကြောင့်လဲဆိုတော့ ဖတ်ထားတဲ့ ဖိုင်ရဲ့အဆုံးသတ်နေရာကနေ စတင်ပြီးဖတ်ဖို့ကြိုးစားတဲ့အတွက် ဖြစ်ပါတယ်။

```
test_1.py x
1 file = open("filename.txt", "r")
2 print("Reading")
3 print(file.read())
4
5 print("Re-reading")
6 print(file.read())
7
8 print("Finished")
9 file.close()

Shell x
>>> %Run test_1.py
Reading
Apple
Ball
Car
Dog
Eagle
Re-reading
Finished
>>> |
```

file ထဲမှာရှိတဲ့ စာသားတစ်ကြောင်းချင်းစီကို ဖတ်ဖို့အတွက် **readlines** method ကို အသုံးပြုပါတယ်။ file ထဲမှာရှိတဲ့စာကြောင်းတစ်ကြောင်းချင်းစီကို list ထဲမှာ element တစ်ခုချင်းစီအဖြစ် ထည့်သွင်းထားမှာဖြစ်ပါတယ်။

```

filename - Notepad
File Edit Format View Help
I have an iPhone.
I have a BMW car.
I have a house.
Ln 3, Col 16 100% Windows (CRLF) UTF-8

```

```

test_1.py x
1 file = open("filename.txt", "r")
2 print(file.readlines())
3 file.close()

Shell x
>>> %Run test_1.py
['I have an iPhone.\n', 'I have a BMW car.\n', 'I have a house.']
>>> |

```

for loop ကို အသုံးပြုပြီးတော့လည်း text file ထဲကနေ တစ်ကြောင်းချင်းစီ ထုတ်လို့ရပါတယ်။

```

test_1.py x
1 file = open("filename.txt", "r")
2 for line in file:
3     print(line)
4 file.close()

Shell x
>>> %Run test_1.py
I have an iPhone.

I have a BMW car.

I have a house.
>>> |

```

Writing Files

file ထဲမှာ string စာကြောင်းရေးသားဖို့အတွက် **write** method ကို အသုံးပြုပါတယ်။

```

test_1.py ×
1 file = open("newfile.txt", "w")
2 file.write("This has been written to a file")
3 file.close()
4
5 file = open("newfile.txt", "r")
6 print(file.read())
7 file.close()

Shell ×
>>> %Run test_1.py
    This has been written to a file
>>> |
  
```

write mode နဲ့ရေးသားတဲ့အခါမှာ text file အလွတ်တစ်ခုကို မဖန်တီးထားရသေးရင်လည်း အလိုအလျောက်ဖန်တီးပေးပြီး ရေးသားပေးမှာ ဖြစ်ပါတယ်။

file တစ်ခုကို write mode နဲ့ ဖွင့်ပြီးရေးသားတဲ့အခါ မူလရှိတဲ့ content စာသားတွေကို အလိုအလျောက်ဖျက်လိုက်မှာ ဖြစ်ပါတယ်။


```

test_1.py ×
1 file = open("newfile.txt", "w")
2 file.write("Some initial content")
3 file.close()
4
5 file = open("newfile.txt", "r")
6 print(file.read())
7 print("Finished!")
8 file.close()
9
10 file = open("newfile.txt", "w")
11 file.write("Some new text")
12 file.close()
13
14 file = open("newfile.txt", "r")
15 print("Reading new contents")
16 print(file.read())
17 print("Finished!")
18 file.close()

Shell ×
>>> %Run test_1.py

Some initial content
Finished!
Reading new contents
Some new text
Finished!

>>> |

```

ပုံမှာပြထားသလိုပဲ မူလကနဦးရေးထားတဲ့ “Some initial content” ဆိုတဲ့ စာသားပေါ်မှာ “Some new text” ဆိုတဲ့စာသားကို ထပ်ရေးလိုက်တာ တွေ့ရမှာပါ။

write method က အဆင်ပြေပြေ ရေးလိုက်နိုင်တယ်ဆိုရင် ရေးလိုက်တဲ့ စာလုံးအရေအတွက် (number of bytes) ကို ပြန်ပြီးထုတ်ပေး (return) ပါတယ်။

```

test_1.py x
1 msg = "Hello world!"
2 file = open("newfile.txt", "w")
3 amount_written = file.write(msg)
4 print(amount_written)
5 file.close()

Shell x
>>> %Run test_1.py
12
>>> |

```

Working with Files

file တွေ့နေ့အလုပ်လုပ်တဲ့အခါမှာ မိမိရဲ့ file တွေ ပျက်စီးပျောက်ဆုံးမသွားဖို့အတွက် အသုံးပြုပြီးတာနဲ့ ဖိတ်တွေကို ပိတ်ပေးဖို့လိုပါတယ်။ ဖိတ်ကို `close()` လုပ်ပေးဖို့ မေ့သွားတတ်တဲ့အတွက် `close()` လုပ်တဲ့အကျင့်ကို မွေးဖို့လိုပါတယ်။ **try: finally:** ကို အသုံးပြုနိုင်ပါတယ်။ ဒီလိုလုပ်ခြင်းအားဖြင့် error တက်ခဲ့ရင်တောင်မှ file ကို မဖြစ်မနေပြန်ပိတ်ပေးမှာ ဖြစ်ပါတယ်။

```

test_1.py x
1 try:
2     f = open("filename.txt")
3     print(f.read())
4 finally:
5     f.close()

```

ဒီလိုပြုလုပ်လို့ရတဲ့နောက်ထပ်နည်းတစ်ခုက **with** statement ဖြစ်ပါတယ်။ အောက်ကပုံမှာ ဖော်ပြထားသလို file ကို ယာယီ variable “f” အဖြစ် ဖွင့်ပေးမှာဖြစ်ပြီး **with** statement ရဲ့ ကုန် block အတွင်းမှာပဲ အသုံးပြုလို့ရစေမှာဖြစ်ပါတယ်။ **with** statement အပြင်ကို ရောက်သွားတာနဲ့ file ကို အလိုအလျောက်ပိတ်လိုက်မှာဖြစ်ပါတယ်။ **with** statement အတွင်းမှာ ဖွင့်ဖတ်အလုပ်လုပ်ရင်း exception တွေလည်း ပိတ်ပေးမှာဖြစ်ပါတယ်။

```
test_1.py ×  
1 with open("filename.txt") as f:  
2     print(f.read())
```