Python les-materialen

# With Statement Context Managers

When you open a file using f = open('test.txt'), the file stays open until you specifically call f.close(). Should an exception be raised while working with the file, it remains open. This can lead to vulnerabilities in your code, and inefficient use of resources.

A context manager handles the opening and closing of resources, and provides a built-in try/finally block should any exceptions occur.

The best way to demonstrate this is with an example.

### Standard open() procedure, with a raised exception:

p = open('oops.txt','a')  
p.readlines()  
p.close()

---------------------------------------------------------------------------  
  
UnsupportedOperation Traceback (most recent call last)  
  
<ipython-input-1-ad7a2000735b> in <module>()  
 1 p = open('oops.txt','a')  
----> 2 p.readlines()  
 3 p.close()  
  
  
UnsupportedOperation: not readable

Let’s see if we can modify our file:

p.write('add more text')

13

Ouch! I may not have wanted to do that until I traced the exception! Unfortunately, the exception prevented the last line, p.close() from running. Let’s close the file manually:

p.close()

### Protect the file with try/except/finally

A common workaround is to insert a try/except/finally clause to close the file whenever an exception is raised:

p = open('oops.txt','a')  
try:  
 p.readlines()  
except:  
 print('An exception was raised!')  
finally:  
 p.close()

An exception was raised!

Let’s see if we can modify our file this time:

p.write('add more text')

---------------------------------------------------------------------------  
  
ValueError Traceback (most recent call last)  
  
<ipython-input-5-1209a18e617d> in <module>()  
----> 1 p.write('add more text')  
  
  
ValueError: I/O operation on closed file.

Excellent! Our file is safe.

### Save steps with with

Now we’ll employ our context manager. The syntax follows with [resource] as [target]: do something

with open('oops.txt','a') as p:  
 p.readlines()

---------------------------------------------------------------------------  
  
UnsupportedOperation Traceback (most recent call last)  
  
<ipython-input-6-7ccc44e332f9> in <module>()  
 1 with open('oops.txt','a') as p:  
----> 2 p.readlines()  
  
  
UnsupportedOperation: not readable

Can we modify the file?

p.write('add more text')

---------------------------------------------------------------------------  
  
ValueError Traceback (most recent call last)  
  
<ipython-input-7-1209a18e617d> in <module>()  
----> 1 p.write('add more text')  
  
  
ValueError: I/O operation on closed file.

Great! With just one line of code we’ve handled opening the file, enclosing our code in a try/finally block, and closing our file all at the same time.

Now you should have a basic understanding of context managers.