[Solved] AttributeError: __enter__

In this article, we are going to discuss AttributeError: **enter**. We will understand why this error occurs and what are the possible solutions for that. We will also discuss a brief about python **Context Manager** to understand the error more clearly.

So what are we waiting for, Let's get started!

Understanding the AttributeError

So, attributes are the values or function that is associated with any class or a datatype. In simple words, we can say that the properties associated or defined in any class or datatype are called its **attribute**.

When we try to call or access any attribute on a value that is not associated with its class or data type. We get an attribute error. Let's try to understand it more clearly. So when we define any variable or instance for any class or datatypes we have access to its attributes. We can use it for our operations but when we try to call an attribute that is not defined for that particular class we get the attribute error. I hope it is clear to you!

Now, before moving to AttributeError: **enter**, let's understand the concept of **Python Context Manager**. It will help us to understand the **error** clearly and then fix it.

Python Context Manager

So, When we work on any real-time problems, there is the case that we have to handle different **files and databases** for storing and retrieving data. These files or databases are known as resources. When we try to access these files or databases then there is a supply constraint to them. It means we can access only a limited number of files at once. So it is important to free all those resources on which we are not working. If we somehow failed to free those resources it will cause resource leakage and consequently, slow down the system.

Now, think that when we handle more files and we have to write the same code, again and again, to open and then close the file. Therefore, to solve this problem we try to achieve the **DRY (Don't Repeat Yourself)** approach.

Here, python **Context Manager** comes into the picture. It helps us to create different classes to manage those resources. Python uses the "**with**" keyword to evaluate that whether the class or function is context manager or not.

How to create Context Manager

We can create a context manager by using both class and functions. We will discuss it one by one.

Using Class

```
class FileManager():             # defining context manager class named FileManager
    def __init__(self, filename, mode):
        self.filename = filename
        self.mode = mode

    def __enter__(self):
        self.file = open(self.filename, self.mode)
        return self.file

    def __exit__(self, exc_type, exc_value, traceback):
        self.file.close()

# loading the file Sample.txt to work upon.

with FileManager('Sample.txt', 'w') as file:
    file.write('Sample Testing')

print(file.closed)
```

```
Output: True
```

The given snippet of code describes how to create a Context Manager using the python class. Now, let’s try to understand each function in the defined class.

__init__:- This method is used to initialize objects for the given class. once, initialization is complete it invokes the __enter__ function.

__enter__:- The __enter__ function then opens the file in the defined mode and then returns an instance of FileManager class. Then the instance is stored in the “file” variable from this line of code(**with FileManager(‘Sample.txt’, ‘w’) as file**:). After this file.write will get executed. Once the writing operation is over, __exit__ function is called automatically.

__exit__:- This function closes the file without mentioning it explicitly every time.

Now, to check whether the file is closed or not we wrote “print(file.closed)” which returns a **true** value which means that the file is closed.

Now, we will imply the same functionality of context manager using functions.

Using Functions

```
from contextlib import contextmanager

@contextmanager
def file_manager(file, mode):
        # opens the file in given mode and assign it to the variable file.
        file = open(file,mode)
        yield file            #returns the file variable
        f.close()                        # closes the file

```

```
with open_file('sample.txt') as file:
       file.write('Sample Testing')

print(f.closed)              #returns true
```

```
Output: True
```

So in this way, we can also implement context manager using functions. We can also use somewhat the same kind of code for databases too.

Click here to learn more about python Context Manager

Solving AttributeError: __enter__

Talking about AttributeError: __enter__, we can say that the issue lies within the context manager class. Whenever there is a problem with returning an instance of the class, the given error occurs. Following **improper syntax** might be one of the reasons for that.

AttributeError: __enter__ in SQLAalchemy

Let’s refer to the following code snippet from StackOverflow to understand this error.

```
Session = scoped_session(sessionmaker(autoflush=True, autocommit=False, bind=engine))

@contextmanager
def session_scope():
    session = Session()
    try:
        yield session
        session.commit()
    except:
        session.rollback()
        raise
    finally:
        session.close()

class SomeClass:

    def __init__(self):
        self.session_scope = session_scope

    def something_with_session(self):
        with self.session_scope as session:               # <-- error
               . . . . . . . . . . .
```

```
Output: AttributeError:__enter__
```

In the above snippet of code, we are facing the error due to line ” **with self.session_scope as session:”** which should be as follow:
`with self.session_scope() as session:`

In this case, we need to understand that we are calling the function to execute and hence we have to write parenthesis after the function.

AttributeError: enter in Tensorflow

```
def network_run():
with tf.Session as sess:                # <-- error
    sess.run(tf.global_variables_initializer())
    for i in range(200):
        sess.run(opt_D, feed_dict={x_ten: images[np.random.choice(range(len(images)),
batch_size)].reshape(batch_size, x_ten_size),
        z_ten:z_noise(batch_size)})
        sess.run(opt_G, feed_dict={z_ten:z_noise(batch_size)})
        sess.run(opt_G, feed_dict={z_ten:z_noise(batch_size)})

        gen_cost=sess.run(G_img, feed_dict={z_ten:z_noise(batch_size)})
        disc_cost=sess.run(D_img, feed_dict={x_ten:
images[np.random.choice(range(len(images)), batch_size)].reshape(batch_size, x_ten_size),
        z_ten:z_noise(batch_size)})

        image=sess.run(G(z_ten), feed_dict={z_ten:z_noise(batch_size)})
        df=sess.run(tf.sigmoid(D_img_fake), feed_dict={z_ten:z_noise()})
        print (i, gen_cost, disc_cost, image.max(), df[0][0])

    image=sess.run(G(z_ten), feed_dict={z_ten:z_noise(batch_size)})
    image1 = image[0].reshape([28, 28])
    im = Image.fromarray(image1)
    im.show()
  network_run()
```

```
Output:

Traceback (most recent call last):
File "C:\Python Practice\gan.py", line 93, in <module>
n()
File "C:\Python Practice\gan.py", line 73, in nn
with tf.Session as sess:
AttributeError: __enter__
```

In the above snippet of code, we are facing the error due to line ” **with tf.Session as sess:”** which should be as follow:

```
with tf.Session() as sess:
```

Here, we have to take care that we are creating objects for the context manager class from the TensorFlow library. To refer to it as an instance, we have to put parentheses over there.

So it is necessary to follow proper syntax to create and use context manager function and class as explained above.

Conclusion

So, today we discussed, attribute errors, took a brief walkthrough of python **Context Manager**. We discussed how we can create a **context manager class** and **context manager functions**. Then, we also discussed, different functions used in defining the context manager and reason of AttributeError: **enter**. Then we discuss possible solutions for the given error.

Hope this article helped you. Thank You.

If you want to know more about Attribute Errors in Python **Click here**.