Loading Data from a Pickle File

Now switch to your second Python Shell — i.e. not the one where you created the `entry` dictionary.

```
shell = 2
print (shell)                                          #①
#2

print (entry)                                          #②
#Traceback (most recent call last):
#   File "/usercode/__ed_file.py", line 5, in <module>
# print (entry) #\u2461
#NameError: name 'entry' is not defined
```

```
import pickle
with open('entry.pickle', 'rb') as f:              #③
     entry = pickle.load(f)                        #④


print (entry)                                          #⑤
#{'comments_link': None,
# 'internal_id': b'\xDE\xD5\xB4\xF8',
# 'title': 'Dive into history, 2009 edition',
# 'tags': ('diveintopython', 'docbook', 'html'),
# 'article_link':
# 'http://diveintomark.org/archives/2009/03/27/dive-into-history-2009-edition',
# 'published_date': time.struct_time(tm_year=2009, tm_mon=3, tm_mday=27, tm_hour=22, tm_min=2
# 'published': True}
```

① This is Python Shell #2.

② There is no `entry` variable defined here. You defined an `entry` variable in Python Shell #1, but that's a completely different environment with its own state.

③ Open the `entry.pickle` file you created in Python Shell #1. The `pickle` module uses a binary data format, so you should always open pickle files in binary mode.

④ The `pickle.load()` function takes a stream object, reads the serialized data from the stream, creates a new Python object, recreates the serialized data in the new Python object, and returns the new Python object.

⑤ Now the `entry` variable is a dictionary with familiar-looking keys and values.

The `pickle.dump() / pickle.load()` cycle results in a new data structure that is equal to the original data structure.

```
import pickle
shell = 1
print (shell )                                  #①
#1

with open('entry.pickle', 'rb') as f:
     entry = pickle.load(f)


with open('entry.pickle', 'rb') as f:           #②
    entry2 = pickle.load(f)                     #③

print (entry2 == entry)                         #④
#True

print (entry2 is entry)                         #⑤
#False

print (entry2['tags'])                          #⑥
#('diveintopython', 'docbook', 'html')

print (entry2['internal_id'])
#b'\xDE\xD5\xB4\xF8'
```

① Switch back to Python Shell #1.

② Open the `entry.pickle` file.

③ Load the serialized data into a new variable, `entry2`.

④ Python confirms that the two dictionaries, `entry` and `entry2`, are equal. In this shell, you built `entry` from the ground up, starting with an empty

dictionary and manually assigning values to specific keys. You serialized this dictionary and stored it in the `entry.pickle` file. Now you've read the serialized data from that file and created a perfect replica of the original data structure.

⑤ Equality is not the same as identity. I said you've created a *perfect replica* of the original data structure, which is true. But it's still a copy.

⑥ For reasons that will become clear later in this chapter, I want to point out that the value of the `'tags'` key is a tuple, and the value of the `'internal_id'` key is a `bytes` object.