Hyperiondev

IO Operations - Input

Visit our website

Introduction

Until now, the Python code you've been writing has only received input in one manner and has only displayed output in one way — you type input using the keyboard and its results are displayed on the console. But what if you want to read information from a file on your computer and write that information to another file? This process is called file I/O (the "I/O" stands for "input/output") and Python has some built-in functions that handle this for you.

A note from the
Hyperion Team

The Python community is alive and growing at a staggering rate. An active community has many benefits. New people bring new ideas, fresh perspectives and different levels of experience.

One such benefit is the proliferation of packages. If you need functionality, there is a good chance that someone else needed it before you. The beauty of the Python community is that most of these functions become packages.

> "There should be one — and preferably only one — obvious way to do it."
> -The Zen of Python

However, this is not always the case. Sometimes you have many options! Feel free to explore them. **Here** are five more packages you should know about.

WORKING WITH FILES IN PYTHON

Files are an important source of information in Python. Before moving on, we want to make sure you know how to read the most simple type of files, text files (.txt), using Python.

Python includes a built-in file type. This is a bit like a string data structure, but much more complex.

There are two methods of opening a file in Python. Firstly, the line below creates a file object named *f* that is linked to the **example.txt** file in the same folder.

```
f = open('example.txt', 'r+')
```

The second example below opens the file using a *with/as* block. This is the more 'pythonic' way of opening a file, and is quite handy because *the file automatically closes* again once the program reaches the end of that block of code.

```
with open('input.txt', 'r+') as f:
    # work with open file
```

In both methods above we are using a built-in function called *open* and passing it the arguments `example.txt` and `r+`. The code in the example above means that `f` is open for reading. The first argument (`example.txt`) is the name of the file that is to be read. The second argument is the mode in which the file will be read. Some of the possible values for the mode are:

Mode	Description
r	Opens the file for reading only. Throws I/O error if the file does not exist.
r+	Opens the file for both reading and writing. Throws I/O error if the file does not exist.
w	Opens the file for writing only. It will create a file if the file doesn't exist. For the existing files, the data is overwritten.
w+	Opens the file for reading and writing. It will create a file if the file doesn't exist. For the existing files, the data is overwritten.
a	Opens the file for writing only, creating the file if it doesn't exist. Any data being written will be inserted at the end (appended, thus 'a'), after the existing data.
a+	Opens the file for reading and writing, creating the file if it isn't there. Any data being written will be inserted at the end (appended, thus 'a'), after the existing data.

Here we intend to read from and write to a text file named **example.txt**, which is already in the same folder as this file. Python will look in this directory for `example.txt`, and try to read its contents.

FILE DIRECTORY

We **never** hard code file locations. The correct way is to specify the location in relation to the file that you are working in.

Wrong	"C:/Users/MYNAME/Documents/Python/1- Python for Data Science/example.txt"
If the file is in the same folder	
Correct	"example.txt" OR "./example.txt"
If the file is 1 folder back	
Correct	"../example.txt"

READING FILES IN PYTHON

There are two common ways of reading files in Python:

1. The most common way to read from a file is to **loop over the lines of the file**. We can directly loop over the variable **f**, which is stored in Python as a list of lines — each line is one line of the file.

```python
for line in f:
    print("The first character of this line is: " + line[0] + "\n")
    print("The entire line is: " + line)
```

When you're not using the with/as structure, remember to always close files when done with them by using *f.close()* to free up the resources the open file was using. Notice this is a function that takes zero arguments.

We could build up all the lines of the text file into one large string called *contents* as follows:

Method 1:

```python
contents = ""
f = open('example.txt', 'r+') # Open the file again

for line in f:                  # Iterate through the lines
    contents = contents + line  # Add the content of each line
print(contents)
```

```
f.close() # Always close files when done with them.
```

Note that you **must** close the file when you're finished if you are going to use this method.

Method 2:

```
contents = ""

with open('example.txt', 'r+') as f:     # Open the file again

    for line in f:                # Iterate through the lines
        contents += line          # Add the content of each line
    print(contents)
 # The with/as structure ensures the file is closed so you don't have
to do it explicitly
```

We now have the contents of an external resource (a text file) stored inside our program in a variable called *contents*. That's pretty powerful! We can then print the contents to a screen with `print(contents):`

2. Another way of reading a file is using the `read()` method. The syntax for this method is as follows:

```
new_contents = f.read()
```

The code in the example above reads the entire file (`example.txt`). You could also pass an integer argument to the `read()` method. This argument would specify the number of characters to read from the file.

FILE ENCODING IN PYTHON

An extra optional argument can be passed to the `open()` function. This argument specifies the encoding of the file. The **official documentation for Python** explains that "serialising a string into a sequence of bytes is known as encoding, and recreating the string from the sequence of bytes is known as decoding".

There are several different methods used to decode or encode files. The optional argument that is passed to the `open()` function specifies which of these methods to use. This should only be used in text mode. The default encoding is

platform-dependent (whatever `locale.getpreferredencoding()` returns). Using the default encoding could result in some strange characters being displayed when Python reads a file.

For example:

```
['ï»¿This is an example file!\n', "You've read from it! Congrats!"]
```

We can correct this by specifying an encoding method, as shown below:

```
f = open('example.txt','r+', encoding='utf-8')
```

See the **codecs module** for the list of supported encodings.

A note from our coding mentor
Joseph

I'd like to tell you about how Apple introduced the Macintosh to the world. They did it with the biggest and most expensive advertising platform available: the 1984 Super Bowl. The advert played on the theme of totalitarianism in George Orwell's book 1984. Apple made a sneaky reference to overcoming IBM by conveying the power of personal computing found in a Macintosh by a depiction of the destruction of "Big Brother".

The Macintosh was the first successful mouse-driven computer with a graphical user interface and was based on the Motorola 68000 microprocessor. Its price was $2 500. The Macintosh came with various applications as part of the package. These included MacPaint, which made use of the mouse and MacWrite, which demonstrated WYSIWYG (What You See Is What You Get) word processing.

Instructions

- First, read and run the **example files** provided. Feel free to write and run your own example code before doing the compulsory task to become more comfortable with the concepts covered in this task.

Compulsory Task

- Create a new Python file in the Dropbox folder for this task, and call it **dob_task.py**.
- In your Python file, write a program that reads the data from the text file provided (**DOB.txt**) and prints it out in two different sections in the format displayed below:

Name
Orville Wright
Rogelio Holloway
Marjorie Figueroa
...etc

Birthdate
21 July 1988
13 September 1988
9 October 1988
...etc

Note: Remember to ensure that the text folder is in the appropriate file directory or Python won't be able to find it when running your program. Get this right first by running the example files, and then do the task.

Rate us
Share your thoughts

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think that the content of this task, or this course as a whole, can be improved, or think we've done a good job?

Click here to share your thoughts anonymously.