

## Lab 9 – Files

### FileI/O

In Python, you don't need to import any library to read and write files.

The first step is to get a file object.

The way to do this is to use the open function.

### File Types:

A file is usually categorized as either text or binary.

A text file is often structured as a sequence of lines and a line is a sequence of characters.

The line is terminated by a EOL (End Of Line) character.

The most common line terminator is the `\n`, or the newline character.

The backslash character indicates that the next character will be treated as a newline.

A binary file is basically any file that is not a text file.

Binary files can only be processed by application that knows about the file's structure.

### Open ( )

To open a file for writing use the built-in `open()` function.

`open()` returns a file object, and is most commonly used with two arguments.

The syntax is:

```
file_object = open(filename, mode)
```

where `file_object` is the variable file object is assigned to.

The second argument describes the way in which the file will be used.

## Mode

The mode argument is optional;

'r' will be assumed if it's omitted.

### **The modes can be:**

'r' when the file will only be read

'w' for only writing (an existing file with the same name will be erased)

'a' opens the file for appending; any data written to the file is automatically added to the end.

'r+' opens the file for both reading and writing.

```
>>> f = open('example.txt', 'w')
>>> print f
<open file 'example.txt', mode 'w' at 0x105c99780>
```

## Create a text file

Let's first create a new text file.

You can name it anything you like, in this example we will name it "example.txt".

```
file = open("example.txt", "w")
file.write("Python for beginners is fun and easy\n")
file.write("we are well on our way to learning File I/O\n")
file.close()
```

## How to read a text file

To read a file, we can use different methods.

### **file.read( )**

If you want to return a string containing all characters in the file, you can use file.read().

```
>>> file = open('example.txt', 'r')
>>> print file.read()
Python for beginners is fun and easy
we are well on our way to learning File I/O
```

We can also specify how many characters the string should return, by using `file.read(n)`, where "n" determines number of characters.

This reads the first 6 characters of data and returns it as a string.

```
>>> file = open('example.txt', 'r')
>>> print file.read(6)
Python
```

### **file.readline( )**

The `readline()` function will read from a file line by line (rather than pulling the entire file in at once).

Use `readline()` when you want to get the first line of the file, subsequent calls to `readline()` will return successive lines.

Basically, it will read a single line from the file and return a string containing characters up to `\n`.

```
>>> file = open('example.txt', 'r')
>>> print file.readline()
Python for beginners is fun and easy
```

Donald Keidel, Ph.D. 2016

### **file.readlines( )**

`readlines()` returns the complete file as a list of strings each separated by `\n`

```
>>> file = open('example.txt', 'r')
>>> print file.readlines()
['Python for beginners is fun and easy\n', 'we are well on our way to learning File I/O\n']
```

### Looping over a file object

For reading lines from a file, you can loop over the file object. This is memory efficient, fast, and leads to simple code.

```
>>> file = open('example.txt', 'r')
>>> for line in file:
...     print line
...
Python for beginners is fun and easy
```

we are well on our way to learning File I/O

file.write()

The write method takes one parameter, which is the string to be written. To start a new line after writing the data, add a `\n` character to the end.

```
>>> file = open('example.txt', 'w')
>>> file.write('Python for beginners is fun and easy\n')
>>> file.write('we are well on our way to learning File I/O\n')
>>> file.close()
>>>
>>> file = open('example.txt', 'r')
>>> for line in file:
...     print line
...
Python for beginners is fun and easy
```

we are well on our way to learning File I/O

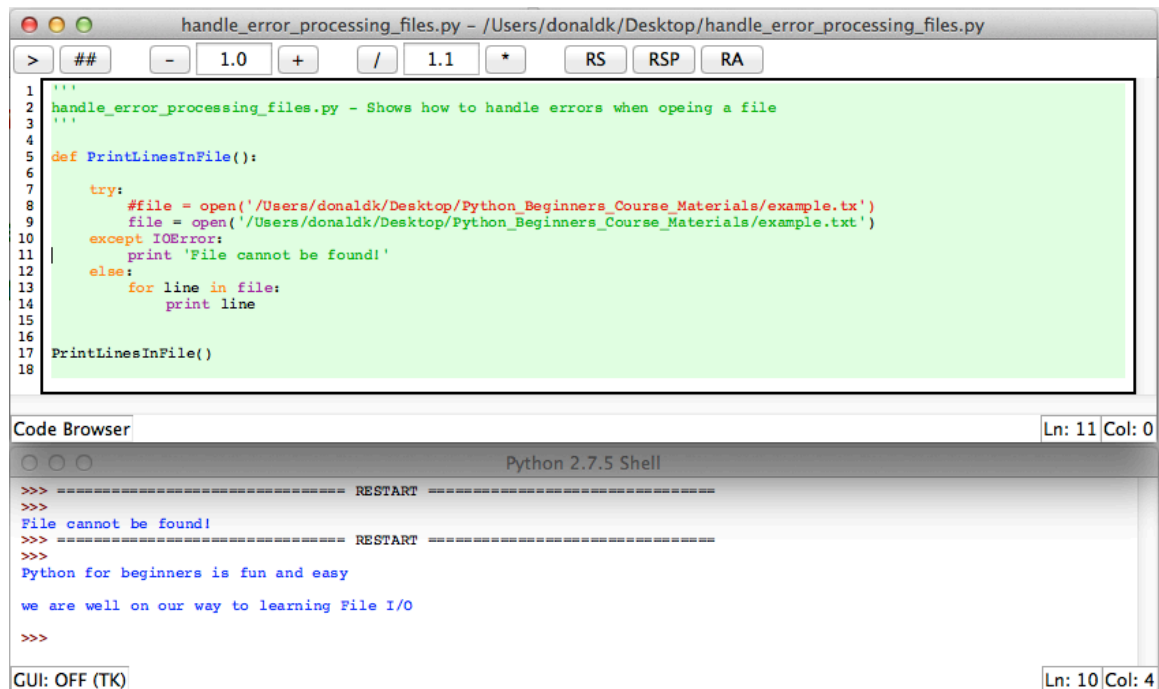
Close()

When you're done with a file, call `f.close()` to close it and free up any system resources taken up by the open file.

After calling `f.close()`, attempts to use the file object will automatically fail.

Donald Keidel, Ph.D. 2016

Example that shows how to handle errors when opening a file:



The screenshot shows a Python IDE window titled 'handle\_error\_processing\_files.py - /Users/donaldk/Desktop/handle\_error\_processing\_files.py'. The code in the editor is as follows:

```
1  """
2  handle_error_processing_files.py - Shows how to handle errors when opening a file
3  """
4
5  def PrintLinesInFile():
6
7      try:
8          #file = open('/Users/donaldk/Desktop/Python_Beginners_Course_Materials/example.tx')
9          file = open('/Users/donaldk/Desktop/Python_Beginners_Course_Materials/example.txt')
10         except IOError:
11             print 'File cannot be found!'
12         else:
13             for line in file:
14                 print line
15
16
17     PrintLinesInFile()
18
```

Below the code editor is a 'Code Browser' panel showing the execution output:

```
>>> ===== RESTART =====
>>>
>>> File cannot be found!
>>> ===== RESTART =====
>>>
>>> Python for beginners is fun and easy
>>> we are well on our way to learning File I/O
>>>
```

The bottom of the window shows 'GUI: OFF (TK)' and line/col indicators: 'Ln: 10 Col: 4'.

## Lab 9 – Exercises

1. Write a program that will open a file for writing (function) and write 4 lines to this file(function). The choice of content is up to you. You will use the append mode to write each line to the file. Write this program using functions.
2. In the same program, write another function that will read the file created in 1. And print its contents.
3. Write another function that will read the file created in 1. and returns the number of lines in the file.
4. Write another function that will read the file created in 1. and add line numbers each line. The function will then write each line to a new file named the same name you named the file in 1. but with the file name appended with \_numbered.txt.

Python 2.7.5 (default, Aug 25 2013, 00:04:04)

[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.0.68)] on darwin

Type "help", "copyright", "credits" or "license" for more information.

```
>>> file = ('example.txt')
```

```
>>> dir(file)
```

```
['_add_', '_class_', '_contains_', '_delattr_', '_doc_', '_eq_', '_format_',  
'_ge_', '_getattribute_', '_getitem_', '_getnewargs_', '_getslice_', '_gt_',  
'_hash_', '_init_', '_le_', '_len_', '_lt_', '_mod_', '_mul_', '_ne_',  
'_new_', '_reduce_', '_reduce_ex_', '_repr_', '_rmod_', '_rmul_',  
'_setattr_', '_sizeof_', '_str_', '_subclasshook_',  
'_formatter_field_name_split', '_formatter_parser', 'capitalize', 'center', 'count',  
'decode', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'index', 'isalnum',  
'isalpha', 'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower',  
'lstrip', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip',  
'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper',  
'zfill']
```

Donald Keidel, Ph.D. 2016

## Lab 10 – Re-Use

### Importing Your Own Module

A module allows you to logically organize your Python code.

Grouping related code into a module makes the code easier to understand and use.

A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code.

A module can define functions, classes and variables. A module can also include runnable code.

### **Writing a module:**

Let's use Homework 2 where we determine if user input is a vowel or not. We have modified this program already when we incorporated lists into the code.

Donald Keidel, Ph.D. 2016

```
howework_2_solution_with_lists_module.py - /Users/donaldkeidel/Desktop/Python_...
1  """
2  homework_2_solution_with_lists_module.py - usage of lists instead of if with many 'or' operators
3  """
4  """
5
6  LOWERCASE_VOWEL_LIST = ['a', 'e', 'i', 'o', 'u']
7  UPPERCASE_VOWEL_LIST = ['A', 'E', 'I', 'O', 'U']
8
9  def AskForLetter():
10     """Will ask the user for a letter and verify that the input is one character
11     in length. If user types 'quit', the program will exit.
12     """
13
14     ask = True
15
16     while ask != False:
17         input = raw_input('Please input a single letter:
18         Type 'quit' to end.')
19         if input == 'quit':
20             ask = False
21
22         if input != 'quit' and len(input) == 1:
23             vowel = IsVowel(input)
24
25             if vowel == True:
26                 print input, 'is a vowel'
27                 ask = False
28
29 |
30 def IsVowel(letter):
31     """Function will call IsLowercaseVowel and IsUppercaseVowel and return True if
32     the single character is uppercase or lowercase
33     """
34
35     # determine if it is lowercase vowel
36     lowercase = IsLowercaseVowel(letter)
37
38     # determine if it is uppercase vowel
39     uppercase = IsUppercaseVowel(letter)
40
41     # return True if either of the above functions
42     # return True
43     if lowercase == True or uppercase == True:
44         return True
45     else:
46         return False
47
48
49
50
51 def IsLowercaseVowel(letter):
52     """Will take as input a single character string and return True if character string
53     is lowercase vowel.
54     """
55
56     #if letter == 'a' or letter == 'e' or letter == 'i' or letter == 'o' or letter == 'u':
57     if letter in LOWERCASE_VOWEL_LIST:
58         return True
59     else:
60         return False
61
62
63 def IsUppercaseVowel(letter):
64     """Will take as input a single character string and return True if character string
65     is uppercase vowel.
66     """
67
68     #if letter == 'A' or letter == 'E' or letter == 'I' or letter == 'O' or letter == 'U':
69     if letter in UPPERCASE_VOWEL_LIST:
70         return True
71     else:
72         return False
73
74 def Run():
75     """Function that calls the necessary functions to run the program as a standalone program"""
76     AskForLetter()
77
78     # needed if file being called from commandline or run in IDLE
79     if __name__ == '__main__':
80         Run()
81
```

The last two lines are kind of cryptic.

In Python, a *file* is an *object* and its name is in the object's `__name__` *attribute*. When we run the file from the console:

```
$ python howework_2_solution_with_lists_module.py
Please input a single letter:
Type 'quit' to end.a
a is a vowel
```

the file's name is `'__main__'`. Thus, Python will evaluate the if statement, find its condition true, run the program by calling `AskForLetter()`.

However, if the file is being imported instead of being called from the console, the name of the file is not `'__main__'` but `'howework_2_solution_with_lists_module'`.

In this case, Python will import the functions but the condition of the if will fail, the function `AskForLetter()` is not called, and the functions are imported without a side effect.

This is the standard way to allow a file to do something useful when called from the console without interfering with its behavior as a module, when its being imported from other files.

Donald Keidel, Ph.D. 2016

We will find this construct at the end of a lot of Python files.

### **Importing the File:**

The easiest (and convenient) way to import our function is to place the file `'howework_2_solution_with_lists_module.py'` in the same directory in which we placed the file that will import it. In this case, the calling file can be something like:

```
>>> import howework_2_solution_with_lists_module
>>> return_value = howework_2_solution_with_lists_module.IsLowercaseVowel('a')
>>> print return_value
True
```

Because of the way in which we imported the module, we are required to specify that `IsLowercaseVowel()` is to be found in `'howework_2_solution_with_lists_module'` by calling it as:  
`return_value = howework_2_solution_with_lists_module.IsLowercaseVowel('a')`

However, we could have called it as:

```
>>> from howework_2_solution_with_lists_module import IsLowercaseVowel
>>> return_value = IsLowercaseVowel('a')
```



```
>>> print return_value
True
```

Here we have to be very careful not to import in this way functions whose name might collide with functions defined elsewhere, i.e., the functions imported this way must have a name different from the names of every other function that is being used.

Above we only made `IsLowercaseVowel` available. If we want to make all the functions available we would do the following:

```
>>> from howework_2_solution_with_lists_module import *
>>> return_value = IsLowercaseVowel('a')
>>> print return_value
True
```

Let's go again over the name collision problem: if we import two modules, each of which has a function called `foo()`, the version imported last will replace the one imported first and Python will direct all calls to the function to it.

Now, all we can hope is that the type and/or number of the arguments and returned values of the two functions differ, which would eventually lead to a runtime error; in the unlikely case in which the number and types of their arguments and return values is the same we will be facing a potentially massively difficult bug to track.

So again, we need to be careful and call our functions with unique names; otherwise, we will be polluting the *namespace*.

An alternative to preserve the ownership of the called functions without incurring in the burden of writing the complete name of the imported file every time that we use its functions is to change the name of the imported file to some short acronym at the time of importing it:

```
>>> import howework_2_solution_with_lists_module as h2swlm
>>> return_value = h2swlm.IsLowercaseVowel('a')
>>> print return_value
True
```

Finally, you can see that `howework_2_solution_with_lists_module` module is available for introspection just like the built-in modules:

```
>>> dir(howework_2_solution_with_lists_module)
['AskForLetter', 'IsLowercaseVowel', 'IsUppercaseVowel', 'IsVowel', '__builtins__',
 '__doc__', '__file__', '__name__', '__package__', 'lowercase_vowel_list',
 'uppercase_vowel_list']
```

```
>>> help(howework_2_solution_with_lists_module)
```

Help on module howework\_2\_solution\_with\_lists\_module:

#### NAME

howework\_2\_solution\_with\_lists\_module -  
homework\_2\_solution\_with\_lists\_module.py - useage of lists instead of if with many  
'or' operators

#### FILE

/Users/donaldk/Desktop/Python\_Beginners\_Course\_Materials/howework\_2\_soluti  
on\_with\_lists\_module.py

#### FUNCTIONS

##### AskForLetter()

Will ask the user for a letter and verify that the input is one character  
in length. If user types 'quit', the program will exit.

##### IsLowercaseVowel(letter)

Will take as input a single charater string and return True is character string  
is lowercase vowel.

##### IsUppercaseVowel(letter)

Will take as input a single charater string and return True is character string  
is uppercase vowel.

##### IsVowel(letter)

Function will call IsLowercaseVowle and IsUppercaseVowel and return True if  
the single character is uppercase or lowercase

#### DATA

lowercase\_vowel\_list = ['a', 'e', 'i', 'o', 'u']

uppercase\_vowel\_list = ['A', 'E', 'I', 'O', 'U']

Donald Keidel, Ph.D. 2016

## Lab 10 – Exercises

1. You will need to open the previous lab example `random_module_randrange_function.py` by downloading it from the course website in Resources -> Solutions to Exercises. To this file, add all that is needed to import this file as a module. Save this file as `random_module_randrange_function_as_module.py`. Remember to all functions add doc strings so that `help(random_module_randrange_function_as_module)` will show a nice help output for the functions and other variables in the module.
2. Now take the module you just created (`random_module_randrange_function_as_module.py`) and import it into a new program for a new different game. This game will have two players. The program will output a toss of coins for the first player. If the coins are the same the player will get 2 points. If they are different, player 1 will get one point. This new program will have function that will ask the user the number of turns they want for the game. After the user defined number of turns, output the scores for each player and report the winner (or tie if there is one).

Donald Keidel, Ph.D. 2016

## Homework 5:

In this homework you will be writing a module that will perform file parsing and writing. Make sure your module has good doc strings for all functions so that when you perform a `help(homework_5_solution_module)` in the python interpreter that you will have very informative output. You will also be writing a program called `homework_5_solution_program.py` that will determine the differences between two files after first writing a new file that is exactly the same as the input file. This program will be importing `homework_5_solution_module.py`.

### Part 1:

Write a module that has the following functions. Make sure you read carefully the requirements for these functions and follow them EXACTLY. Name this `homework_5_solution_module.py`.

1. Function called `AskForFileName( )` that will ask the user for the name of the input file. Make sure you put all files in the same directory/folder as this python program. You can download the input file from the course website under Resources -> Homework 5 input -> 1JKB.pdb. Open this file in a text editor since you will have to refer to it when you write the functions below.
2. Function called `ReadFileContents(file_name )` that will open the file and read all the lines in the file into memory. Name this variable `all_file_contents`.
3. Function called `BuildHeadList(all_file_contents )`. This function will loop over the variable populated in `ReadFileContents` and append to another list called `head_list` the header information from the file. These are the lines from the top of the file to the lines that start with the word ATOM.
4. Function called `BuildAtomList(all_file_contents )`. This function will loop over the variable populated in `ReadFileContents` and append to another list called `atom_list` all the lines that begin with ATOM and ONLY these lines.
5. Function called `BuildTailList(all_file_contents )`. This function will loop over the variable populated in `ReadFileContents` and append to another list called `tail_list` all the lines that are below those that begin with ATOM (the lines left over).
6. Function called `WriteNewFile( )`. This function will take the three lists created above (`head_list`, `atom_list`, `tail_list`) as arguments and will write these lists to an output file called `output.txt` that should look exactly like 1JKB.pdb when finished writing.

HINT: Look at all the built-in function for the string type (i.e. `dir(str)`). One of these functions will be important when looping through `all_file_contents`.

## Part 2:

Write another program that will import the module you created above so that it can be used by this new program. Name this new program `homework_5_solution_program.py`. This program must have the following functions. Make sure you read carefully the requirements for these functions and following them EXACTLY.

1. A function called `Run( )` that will call `AskForFileName( )` in the module you imported. `Run( )` will then call the rest of the functions defined above in the same order they are defined above. `Run( )` will finally call `DifferenceTwoFiles( file_1, file_2)`.
2. Function called `DifferenceTwoFiles( file_1, file_2)` that will take in the name of two files as parameters. The order passed does not matter. Make sure the files are saved in the same directory as this python program. This function will open both files reading in all lines into two separate variables and compare the two to determine if the files are different. This function will accomplish this by looping over the elements of the first variable and checking if the second variable contains the same line in the same order. If there is a difference, then the element that is different should be appended to a list call `diff_list`. This list should then be counted for the number of differences and that number output.

Donald Keidel, Ph.D. 2016

Print out a copy of your homework and bring it to class. Make sure to include the output.