

COMP7035

Python for Data Analytics and Artificial Intelligence

File IO

Renjie Wan, Jun Qi

24/09/2024

How to interact with PC

- The `input([prompt])` function accepts from the user

```
str = input("Enter your input: ")  
print("Received input is : ", str)
```



This is a string value

```
x = input('Enter your name:')  
print('Hello, ' + x)
```

How to interact with your PC

- How to play with input

```
base = int(input("Enter the base:"))  
height = int(input("Enter the height:"))  
if base < 0 or height < 0:  
    print("Base and height must be non-negative")  
else:  
    area = 0.5*base*height  
    print("The area is:", area)
```

How to transfer value types

- From str to int
- **int()** : Used to **convert string into an integer.**
- **str():** **Convert integer to string**

```
print(type("1"))  
print(type(int("1")))
```

- From int to float
- **float()**: This function is used to convert **any data type to a floating-point number.**

```
print(type(1))  
print(type(float(1)))
```

How to transfer value types

- `tuple()` : This function is used to convert to a tuple.
- `set()` : This function returns the type after converting to set.
- `list()` : This function is used to convert any data type to a list type.

```
s = 'geeks'

# printing string converting to tuple
c = tuple(s)
print ("After converting string to tuple : ",end="")
print (c)

# printing string converting to set
c = set(s)
print ("After converting string to set : ",end="")
print (c)

# printing string converting to list
c = list(s)
print ("After converting string to list : ",end="")
print (c)
```

Opening a file

- Python has a built-in `open()` function to open a file. This function returns a file object, also called a handle, as it is used to read or modify the file accordingly.

```
f = open(filename, option)
```

Filename: path and filename


4-6

Option:

"r" - Default value. Opens a file for reading

"a" - Opens a file for appending at the end of file, creates the file if it does not exist

"w" - Opens a file for writing, creates the file if it does not exist



```
f = open("demofile.txt", "r")  
# open file in current directory  
f = open("demofile.txt")
```

We need to close a file after we are done: `f.close()`

How to write something into a file

- The *write()* method writes any string to an open file. It is important to note that Python strings can have binary data and not just text.
- The *close()* method of a file object flushes any unwritten information and closes the file object, after which no more writing can be done.
- Remember to close file everytime.

4-7

```
#!/usr/bin/python # Open a file
fo = open("test.txt", "w")
fo.write( "Python is a great language.\nYeah its great!!\n")

fo.close()
```

How to write something into a file

- Always close a file object when you are done with it
- If you do not want to use close, you can also try with open:
- with open(filename) as var
 - opens the file <filename>
 - assigns the corresponding file object to <var>
 - automatically closes the file when we leave the with-block

```
with open("test.txt", "w") as fo:  
    fo.write( "Python is a great language.\nYeah its great!!\n")
```

```
f = open('abc.txt', 'r')  
print(f.closed)  
f.close()  
print(f.closed)
```

You can use `f.closed` to check whether the file is closed or not.

How to create a directory

- You can use the `mkdir()` method of the `os` module to create directories in the current directory. You need to supply an argument to this method which contains the name of the directory to be created.

```
#!/usr/bin/python
import os

# Create a directory "test"
os.mkdir("test")
```

getcwd()

- Python method **getcwd()** returns current working directory of a process.

```
import os  
os.getcwd()
```

Try above codes yourself to see what happens

- Python method **chdir()** changes the current working directory to the given path. It returns None in all the cases.

```
os.chdir(path)
```

How to delete a directory

- *os.rmdir()* method in Python is used to remove or delete a empty directory. *OSError* will be raised if the specified path is not an empty directory.

```
#!/usr/bin/python
import os

# Create a directory "test"
os.rmdir("test")
```

Create multiple directories

- *os.makedirs()* method in Python is used to create a directory recursively. That means while making leaf directory if any intermediate-level directory is missing, *os.makedirs()* method will create them all.

```
#!/usr/bin/python  
import os  
os.makedirs("test/test2")
```

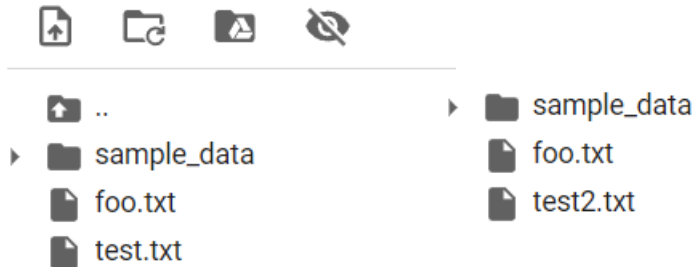
How to delete and rename files

- Rename a file

```
import os  
os.rename( "test.txt", "test2.txt" )
```

- Delete a file

```
import os  
os.remove("test2.txt")
```



Differences between r and r+

- r: Open text file for reading. The stream is positioned at the beginning of the file.
- r+: Open for reading and writing. The stream is positioned at the beginning of the file.
- w: Create text file for writing. The stream is positioned at the beginning of the file.
- w+: Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
- rb, wb use for reading and writing binary format (usually for images)

How to read the opened file

- The *read(size)* method reads a string from an open file. Size here denotes the number of characters to be read

```
#!/usr/bin/python # Open a file
fo = open("test.txt", "r+")
str = fo.read(10);
print("Read String is : ", str)

# Close opened file
fo.close()
```

↓
**Python is a great language.
Yeah its great!!**

If size = 1:

Read String is : P

If size = 2:

Read String is : Py

If we do not set the size number:

Read String is : Python is a great
language. Yeah its great!!

The current position of File Handle

- *tell()* method returns current position of file object

```
#!/usr/bin/python # Open a file
fo = open("test.txt", "r+")
print("Initial Position:", fo.tell())

str2 = fo.read(10);
print("Read String is : ", str2)

print("Current Position:", fo.tell())
# Close opened file
fo.close()
```


readline()

- The *readline()* method returns one line from the file.

```
# open the file
fo = open("test.txt", "r")
print("The file name is: ", fo.name)

print(f.readline()) #read the first line
print(f.readline()) #read the second line

# close the file
fo.close()
```

readlines()

- The *readlines()* method returns all lines from the file.

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

# open the file
fo = open("test.txt", "r")
print("The file name is: ", fo.name)

for line in fo.readlines():
    print("The read file is: %s" % (line))

# close the file
fo.close()
```

writelines

- *writelines()* method to write items in a list object to a file. The newline characters ("\n") should be the part of the string.

```
lines=[" Beautiful is better than ugly.\n", "Explicit  
is better than implicit.\n"]
```

```
lines=[" Beautiful is better than ugly.", "Explicit i  
s better than implicit."]
```

```
f=open("python.txt", "w")  
f.writelines(lines)  
f.close()
```

File IO Functions

- The `rename()` method takes two arguments, the current filename and the new filename.

```
#!/usr/bin/python
import os

# Rename a file from test1.txt to test2.txt
os.rename( "test1.txt", "test2.txt" )
```

File IO Functions

- `remove()` method is to delete files by supplying the name of the file to be deleted as the argument.

```
#!/usr/bin/python
import os

# Delete file test2.txt
os.remove("text2.txt")
```