Files

When a program needs to save data for later use, it writes the data in a file. The data can be read from the file at a late. The process of retrieving data from a file is known as "reading data from" the file.

When a piece of data is read from a file, it is copied from the file into RAM and referenced by a variable.

1. **Open the file**—Opening a file creates a connection between the file and the program. Opening an output file usually creates the file on the disk and allows the program to write data to it. Opening an input file allows the program to read data from the file.

2. **Process the file**—In this step data is either written to the file (if it is an output file) or read from the file (if it is an input file).

3. **Close the file**—When the program is finished using the file, the file must be closed. Closing a file disconnects the file from the program.

1. Two types of files
Text files
Binary Files

2. File Access methods

3. Sequential Access Methods
Direct Access Methods
In this we only dedl with Sequential Access method

File names and File Objects
Many systems support the use of filename extensions, which are short sequences of characters that appear at the end of a filename preceded by a period (which is known as a "dot"). For example, the files have the extensions .jpg,.txt, and.doc. The extension usually indicates the type of data stored in the file. For example, the .jpg extension usually indicates that the file contains a graphic image that is compressed according to the JPEG image standard. The .txt extension usually indicates that the file contains text. The .doc extension (as well as the .docx extension) usually indicates that the file contains a Microsoft Word document.

File object
A file object is an object that is associated with a specific file and provides a way for the program to work with that file. In the program, a variable references the file object. This variable is used to carry out any operations that are performed on the file.

Opening a File

You use the open function in Python to open a file. The open function creates a file object and associates it with a file on the disk. Here is the general format of how the open function is used:

file_variable = open(filename, mode)

Mode	Description
'r'	Open a file for reading only. The file cannot be changed or written to.
'w'	Open a file for writing. If the file already exists, erase its contents. If it does not exist, create it.
'a'	Open a file to be written to. All data written to the file will be appended to its end. If the file does not exist, create it.

>>>test_file=open('honey_test','w')

>>> import os

>>>os.getcwd()

'C:\\Users\\Administrator\\AppData\\Local\\Programs\\Python\\Python37-32'

os.chdir('C:/Users/Administrator/Desktop/Python Training')

>>>os.getcwd()

'C:\\Users\\Administrator\\Desktop\\Python Training'

To open the file with path

test_file=open('C:/Users/Administrator/Desktop/Python Training/honey_check.txt','w')

test_file = open(r'C:\Users\Blake\temp\test.txt', 'w')

Writing data to file
File_variable.write(string)

>>>test_file=open('C:/Users/Administrator/Desktop/Python Training/honey_check2.txt','w')

>>>test_file.write('honey\t honey')

12

>>>test_file.close()

Reading Data from file

>>>test_file=open('C:/Users/Administrator/Desktop/Python Training/honey_check2.txt','r')

>>>contents=test_file.read()

>>> contents

'hoeny\t honey'

>>>test_file.close()

Readline method

>>> t=test_file.readline()

>>> t

'hoeny\t honey'

>>> t=test_file.readline()

>>> t

"

Problem :

Write names of friends in file.

Reading the String and Stripping the Newline from it

>>>test_file=open('C:\\Users\\Administrator\\Desktop\\Python Training\\honey_check2.txt', 'r')

>>>test_file.readline()

'3rd line\n'

>>>test_file.readline()

'4th line\n'

>>>test_file=open('C:\\Users\\Administrator\\Desktop\\Python Training\\honey_check2.txt', 'r')

>>> h=test_file.readline()

>>> h

'3rd line\n'

>>> h=h.rstrip('\n')

>>> h

'3rd line'

>>>

Appending Data to an existing file

```
myfile = open('friends.txt', 'a')
myfile.write('Matt\n')
myfile.write('Chris\n')
myfile.write('Suze\n')
myfile.close()
```

Writing and Reading the numeric Data

```
def main():
    # Open a file for writing.
    outfile = open('numbers.txt', 'w')

    # Get three numbers from the user.
    num1 = int(input('Enter a number: '))
    num2 = int(input('Enter another number: '))
    num3 = int(input('Enter another number: '))

    # Write the numbers to the file.
    outfile.write(str(num1) + '\n')
    outfile.write(str(num2) + '\n')
    outfile.write(str(num3) + '\n')

    # Close the file.
    outfile.close()
    print('Data written to numbers.txt')

# Call the main function.
main()
```

```
# This program demonstrates how numbers that are
# read from a file must be converted from strings
# before they are used in a math operation.

def main():
    # Open a file for reading.
    infile = open('numbers.txt', 'r')

    # Read three numbers from the file.
    num1 = int(infile.readline())
    num2 = int(infile.readline())
    num3 = int(infile.readline())

    # Close the file.
    infile.close()

    # Add the three numbers.
    total = num1 + num2 + num3

    # Display the numbers and their total.
    print('The numbers are:', num1, num2, num3)
    print('Their total is:', total)

# Call the main function.
main()
```


Appending data to file

Use the mode a instead of w

Loops to Process Files

Files usually hold large amounts of data, and programs typically use aloop to process the data in a file

```
    # Open a new file named sales.txt.
    sales_file = open('sales.txt', 'w')

    # Get the amount of sales for each day and write
    # it to the file.
    for count in range(1, num_days + 1):
      # Get the sales for a day.
      sales = float(input('Enter the sales for day #' + \
                str(count) + ': '))

      # Write the sales amount to the file.
      sales_file.write(str(sales) + '\n')

    # Close the file.
    sales_file.close()
    print('Data written to sales.txt.')

# Call the main function.
main()
```

```
# This program reads all of the values in
# the sales.txt file.

def main():
    # Open the sales.txt file for reading.
    sales_file = open('sales.txt', 'r')

    # Read the first line from the file, but
    # don't convert to a number yet. We still
    # need to test for an empty string.
    line = sales_file.readline()

    # As long as an empty string is not returned
    # from readline, continue processing.
    while line != '':
        # Convert line to a float.
        amount = float(line)

        # Format and display the amount.
        print(format(amount, '.2f'))

        # Read the next line.
        line = sales_file.readline()

    # Close the file.
    sales_file.close()

# Call the main function.
main()
```


When you simply want to read the lines in a file, one after the other, thistechnique is simpler and moreelegant than writing a while loop that explicitly tests for anend of the file condition. Here is the general format of the loop:

for *variable* in *file_object:*

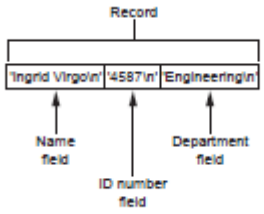
statement

statement

etc.

```
# This program uses the for loop to read
# all of the values in the sales.txt file.

def main():
    # Open the sales.txt file for reading.
    sales_file = open('sales.txt', 'r')

    # Read all the lines from the file.
    for line in sales_file:
        # Convert line to a float.
        amount = float(line)
        # Format and display the amount.
        print(format(amount, '.2f'))

    # Close the file.
    sales_file.close()

# Call the main function.
main()
```

Record in file




```
# This program gets employee data from the user and
# saves it as records in the employee.txt file.

def main():
    # Get the number of employee records to create.
    num_emps = int(input('How many employee records ' + \
            'do you want to create? '))

    # Open a file for writing.
    emp_file = open('employees.txt', 'w')

    # Get each employee's data and write it to
    # the file.
    for count in range(1, num_emps + 1):

        # Get the data for an employee.
        print('Enter data for employee #', count, sep='')
        name = input('Name: ')
        id_num = input('ID number: ')
        dept = input('Department: ')

        # Write the data as a record to the file.
        emp_file.write(name + '\n')
        emp_file.write(id_num + '\n')
        emp_file.write(dept + '\n')

        # Display a blank line.
        print()

    # Close the file.
    emp_file.close()
    print('Employee records written to employees.txt.')

# Call the main function.
main()
```


Reading the record from file

```
# This program displays the records that are
# in the employees.txt file.

def main():
    # Open the employees.txt file.
    emp_file = open('employees.txt', 'r')

    # Read the first line from the file, which is
    # the name field of the first record.
    name = emp_file.readline()

    # If a field was read, continue processing.
    while name != '':
        # Read the ID number field.
        id_num = emp_file.readline()

        # Read the department field.
        dept = emp_file.readline()

        # Strip the newlines from the fields.
        name = name.rstrip('\n')
        id_num = id_num.rstrip('\n')
        dept = dept.rstrip('\n')

        # Display the record.
        print('Name:', name)
        print('ID:', id_num)
        print('Dept:', dept)
        print()

        # Read the name field of the next record.
        name = emp_file.readline()

    # Close the file.
    emp_file.close()

# Call the main function.
main()
```