Files

Files

A **file** is a semi-permanent, named collection of information.

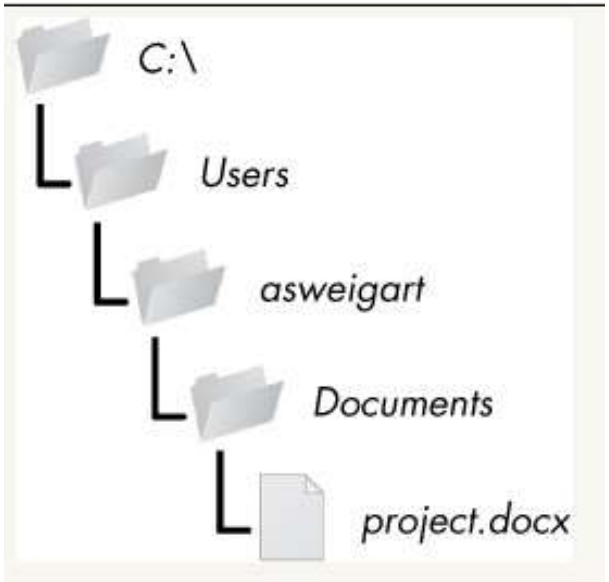
- **semi-permanent:** Files are usually stored on magnetic disk where they will remain for years even when the power is off. Of course, a file can be deleted (sometimes by accident) so they are semi-permanent, not permanent.
- **named:** A file has a name that is used to find it when it is needed. You probably know that MS Windows file names look like:
  - mydata.txt
  - program1.java
  - program1.class
  - doom.exe
  - ... and so on
- **collection of information:** The purpose of a file is to contain a collection of related information, such as a word processing document, a program source file, a spread sheet, a data base, and so on.

This is a conceptual definition of what a file is. It says what a file is and how it is used. A file can be implemented in many ways: as sections of a hard disk, as sections of a DVD, as part of a magnetic tape, and in other ways.

Files and File Paths

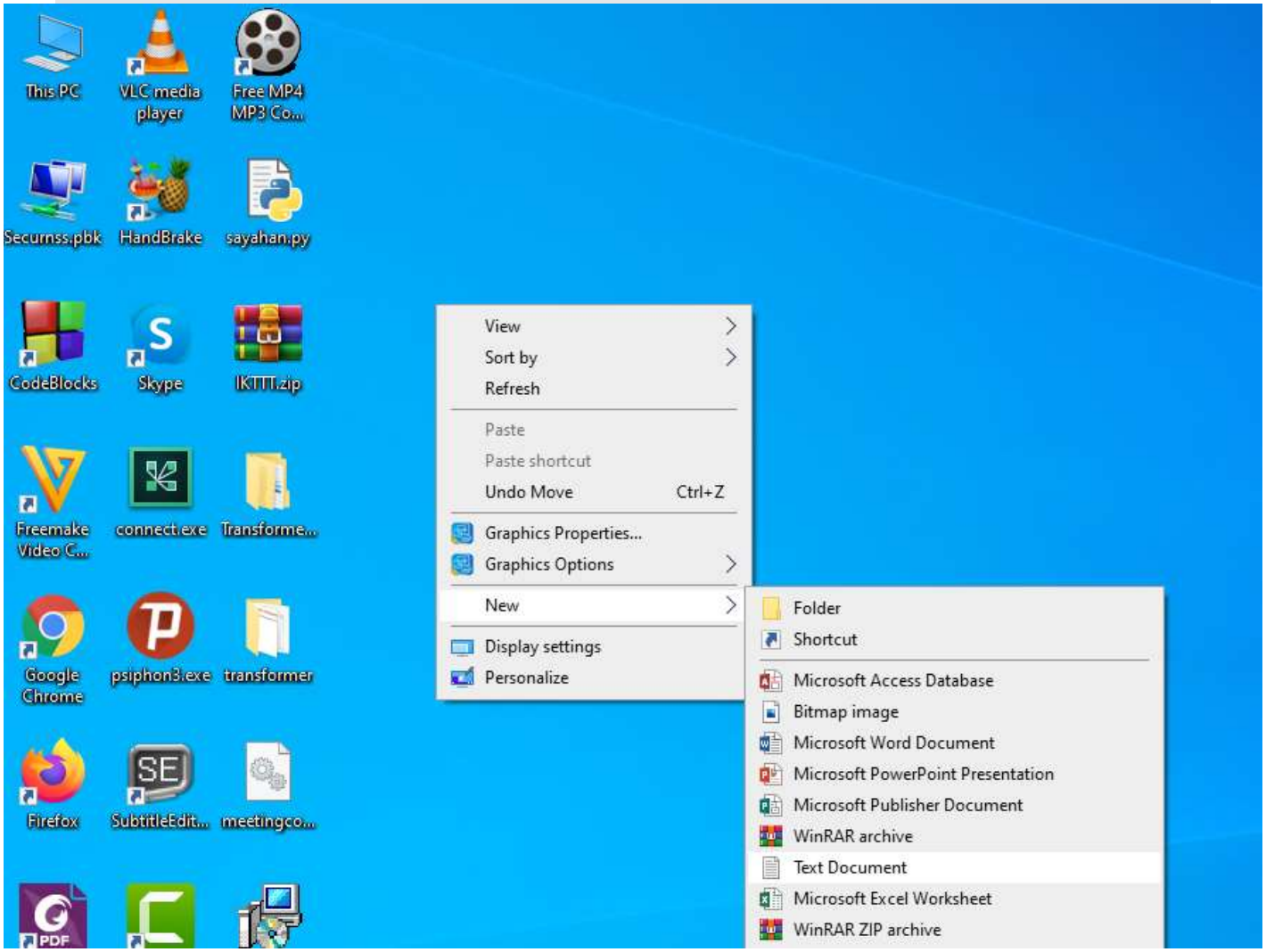
- A file has two key properties: a *filename* (usually written as one word) and a *path*. The path specifies the location of a file on the computer. For example, there is a file on my Windows 7 laptop with the filename *project.docx* in the path *C:\Users\asweigart\Documents*.
- The part of the filename after the last period is called the file's *extension* and tells you a file's type. *project.docx* is a Word document, and *Users*, *asweigart*, and *Documents* all refer to *folders* (also called *directories*).
- Folders can contain files and other folders. For example, *project.docx* is in the *Documents* folder, which is inside the *asweigart* folder, which is inside the *Users* folder.

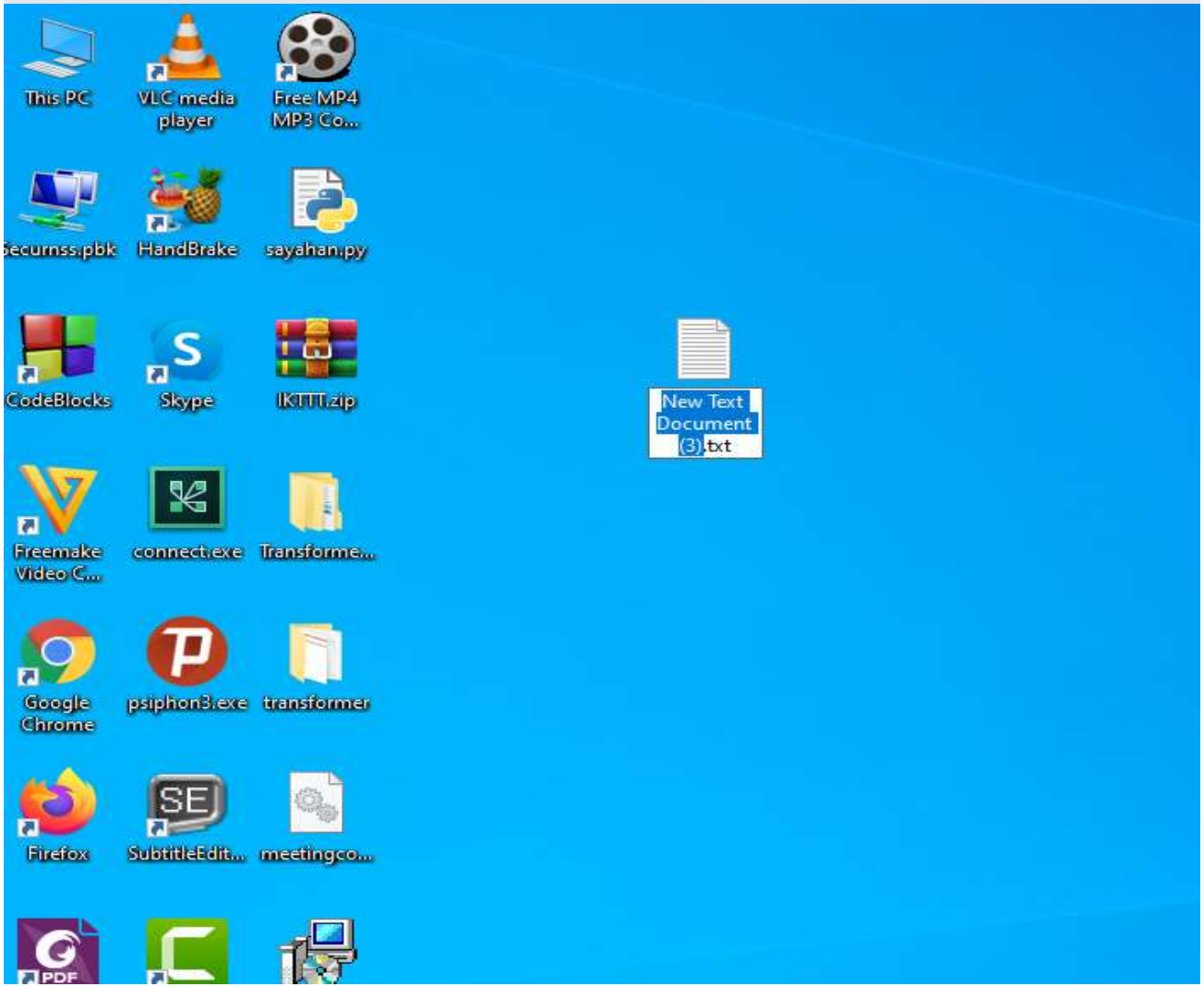
Files and File Paths

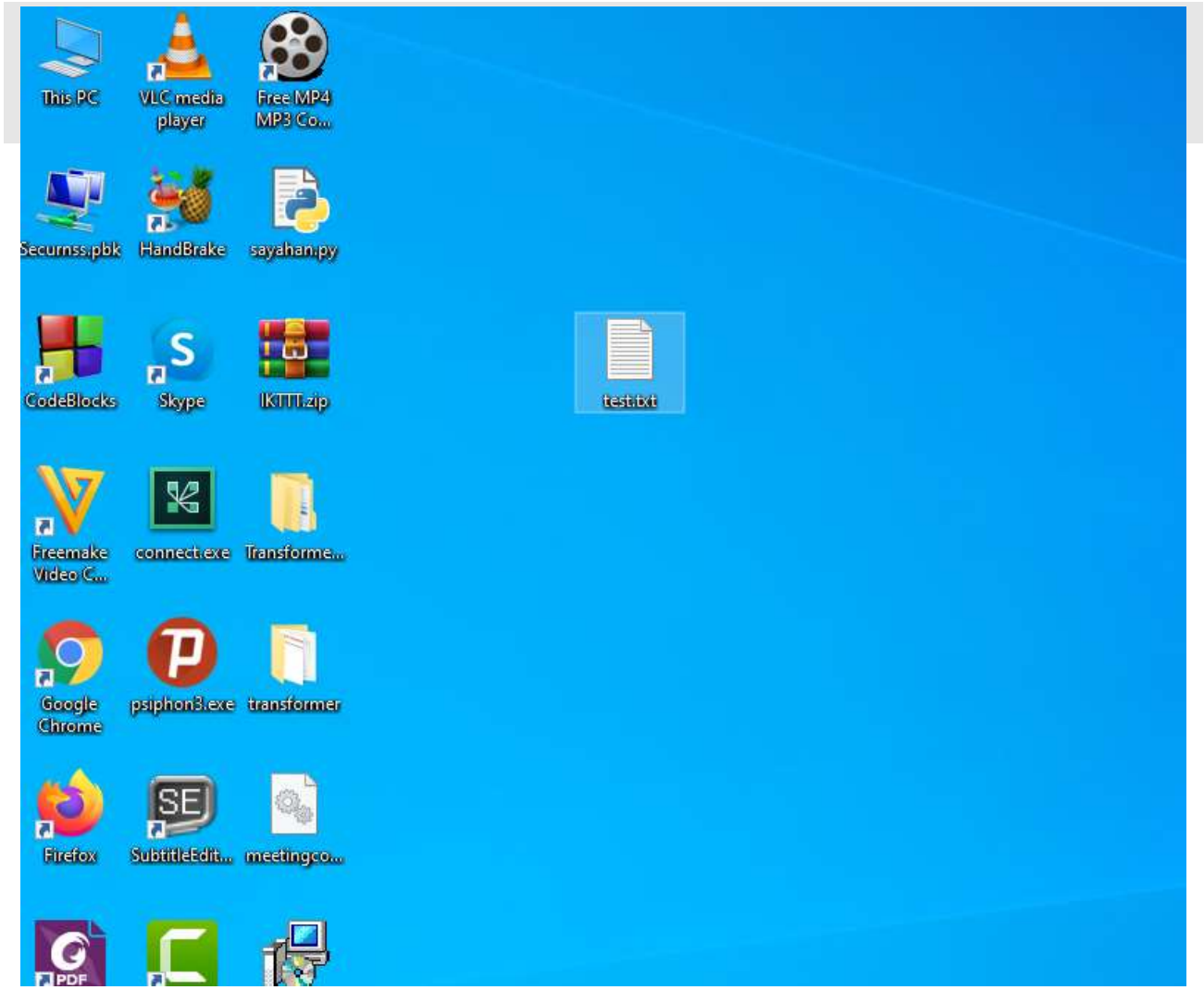


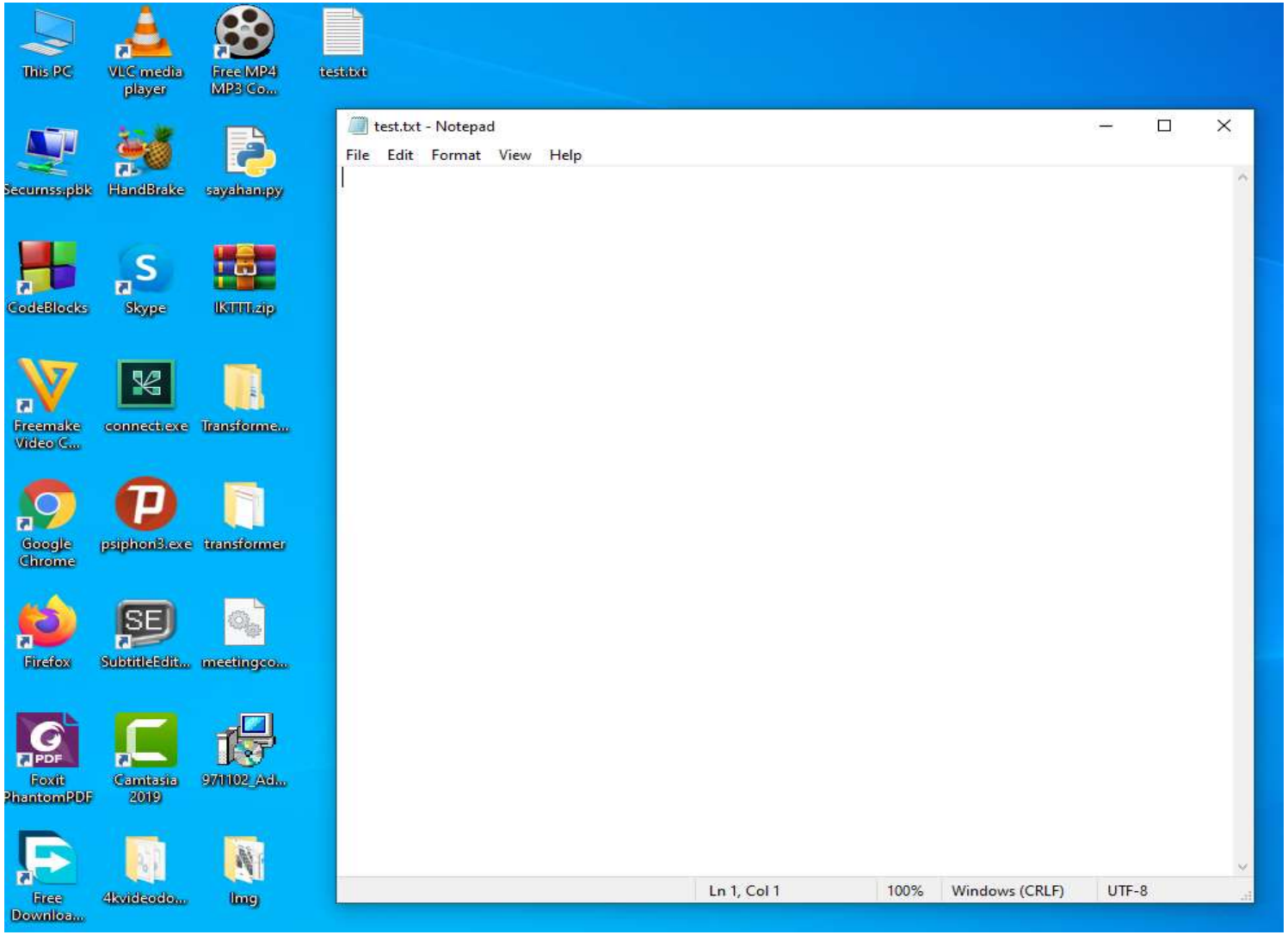
The Current Working Directory

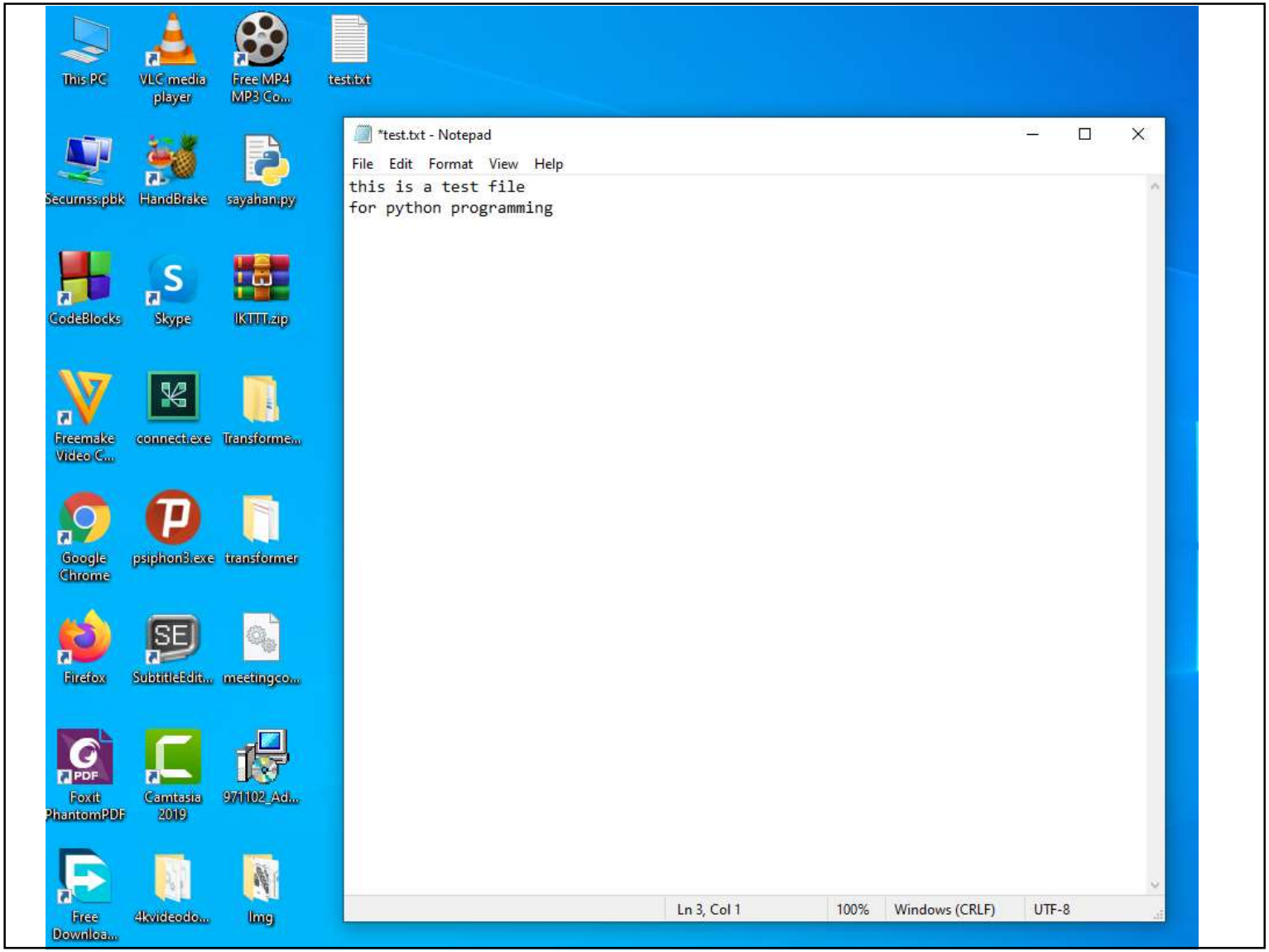
- Every program that runs on your computer has a *current working directory*, or *cwd*.
- Any filenames or paths that do not begin with the root folder are assumed to be under the current working directory.

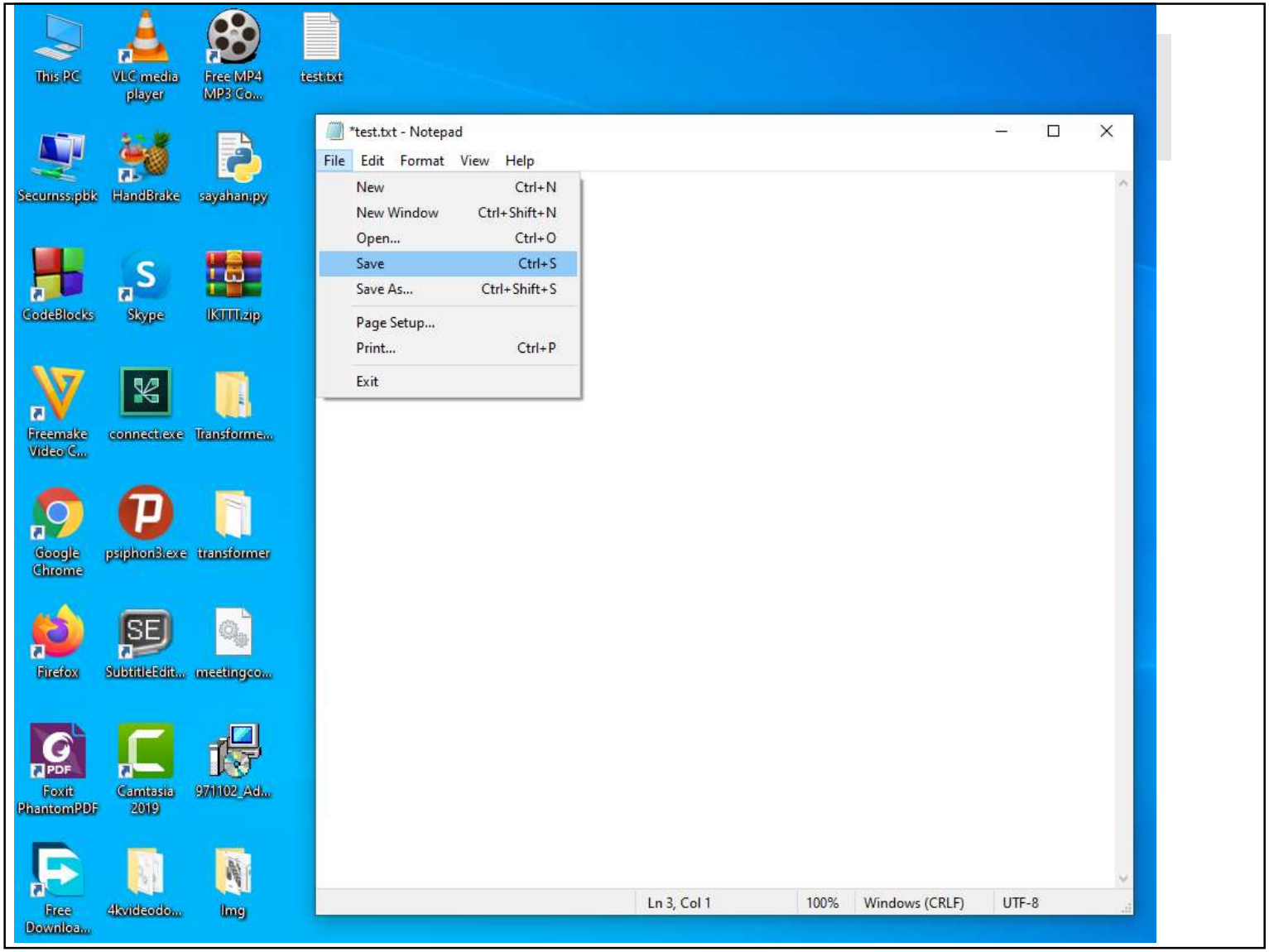

This PC
VLC media player
Free MP4 MP3 Co...
Securnss.pbk
HandBrake
sayahan.py
CodeBlocks
Skype
IKTTT.zip
Freemake Video C...
connect.exe
Transforme...
Google Chrome
psiphon3.exe
transformer
Firefox
SubtitleEdit...
meetingco...
View
Sort by
Refresh
Paste
Paste shortcut
Undo Move
Ctrl+Z
Graphics Properties...
Graphics Options
New
Display settings
Personalize
Folder
Shortcut
Microsoft Access Database
Bitmap image
Microsoft Word Document
Microsoft PowerPoint Presentation
Microsoft Publisher Document
WinRAR archive
Text Document
Microsoft Excel Worksheet
WinRAR ZIP archive

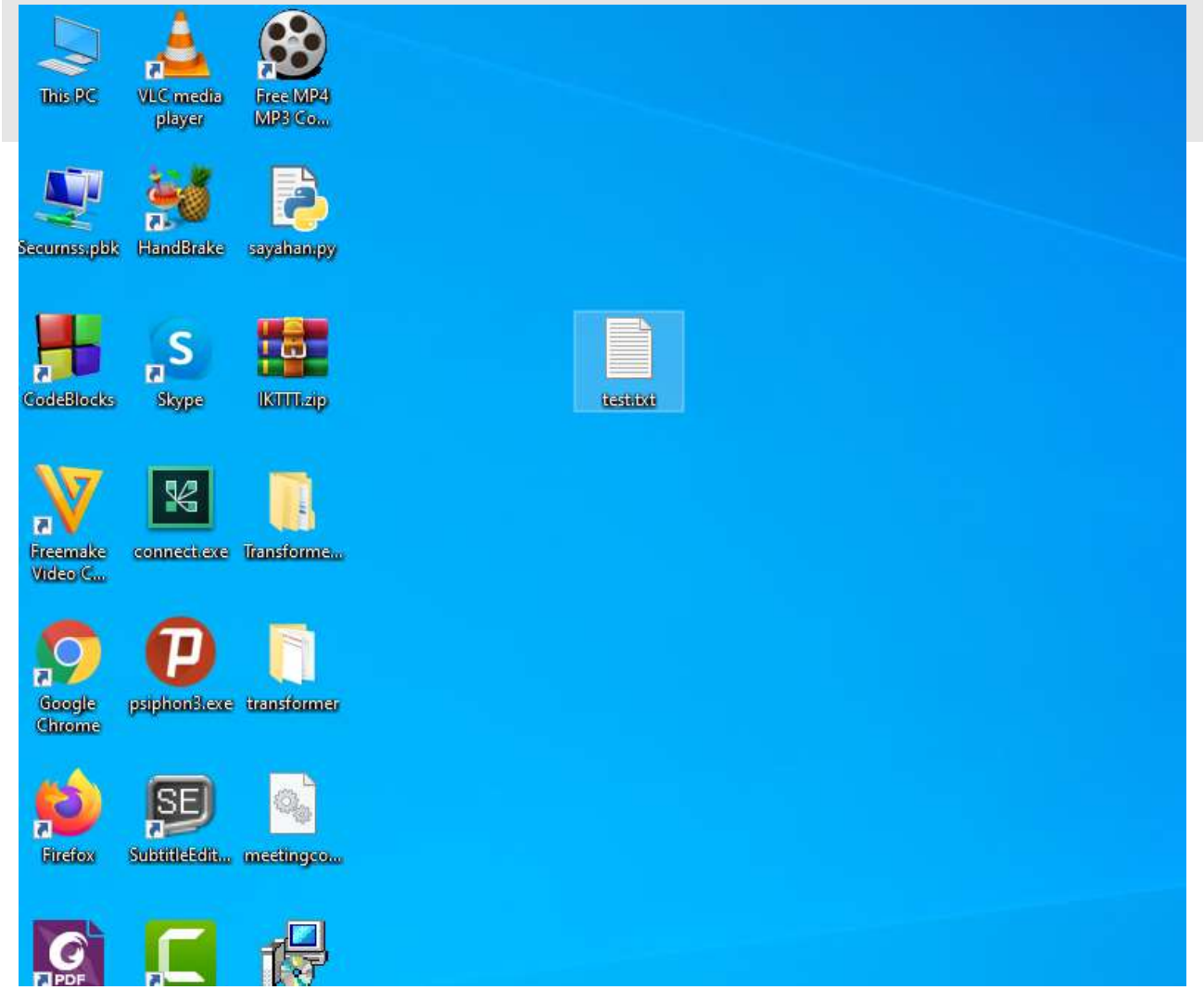

This PC
VLC media player
Free MP4 MP3 Co...
Securnss.pbk
HandBrake
sayahan.py
CodeBlocks
Skype
IKTTT.zip
New Text Document (3).txt
Freemake Video C...
connect.exe
Transforme...
Google Chrome
psiphon3.exe
transformer
Firefox
SubtitleEdit...
meetingco...


This PC
VLC media player
Free MP4 MP3 Co...
Securnss.pbk
HandBrake
sayahan.py
CodeBlocks
Skype
IKTTT.zip
test.txt
Freemake Video C...
connect.exe
Transforme...
Google Chrome
psiphon3.exe
transformer
Firefox
SubtitleEdit...
meetingco...


This PC
VLC media player
Free MP4 MP3 Co...
test.txt
Securnss.pbk
HandBrake
sayahan.py
CodeBlocks
Skype
IKTTT.zip
Freemake Video C...
connect.exe
Transforme...
Google Chrome
psiphon3.exe
transformer
Firefox
SubtitleEdit...
meetingco...
Foxit PhantomPDF
Camtasia 2019
971102_Ad...
Free Downloa...
4kvideodo...
Img
test.txt - Notepad
File Edit Format View Help
Ln 1, Col 1
100%
Windows (CRLF)
UTF-8


This PC
VLC media player
Free MP4 MP3 Co...
test.txt
Securnss.pbk
HandBrake
sayahan.py
CodeBlocks
Skype
IKTTT.zip
Freemake Video C...
connect.exe
Transforme...
Google Chrome
psiphon3.exe
transformer
Firefox
SubtitleEdit...
meetingco...
Foxit PhantomPDF
Camtasia 2019
971102_Ad...
Free Downloa...
4kvideodo...
Img
*test.txt - Notepad
File Edit Format View Help
this is a test file
for python programming
Ln 3, Col 1
100%
Windows (CRLF)
UTF-8


This PC
VLC media player
Free MP4 MP3 Co...
test.txt
Securnss.pbk
HandBrake
sayahan.py
CodeBlocks
Skype
IKTTT.zip
Freemake Video C...
connect.exe
Transforme...
Google Chrome
psiphon3.exe
transformer
Firefox
SubtitleEdit...
meetingco...
Foxit PhantomPDF
Camtasia 2019
971102_Ad...
Free Downloa...
4kvideodo...
Img
*test.txt - Notepad
File Edit Format View Help
New Ctrl+N
New Window Ctrl+Shift+N
Open... Ctrl+O
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup...
Print... Ctrl+P
Exit
Ln 3, Col 1
100%
Windows (CRLF)
UTF-8


This PC
VLC media player
Free MP4 MP3 Co...
Securnss.pbk
HandBrake
sayahan.py
CodeBlocks
Skype
IKTTT.zip
test.txt
Freemake Video C...
connect.exe
Transforme...
Google Chrome
psiphon3.exe
transformer
Firefox
SubtitleEdit...
meetingco...

Text and Binary Files

- **Text file**
- **Binary file**

Text file

- When a file is opened in *text mode, reading its data automatically decodes its content* and returns it as a str; writing takes a str and automatically encodes it before transferring it to the file. Both reads and writes translate per a platform default or a provided ncoding name. Text-mode files also support universal end-of-line translation and additional encoding specification arguments. Depending on the encoding name, text files may also automatically process the byte order mark sequence at the start of a file (more on this momentarily).

A 41

127 31 32 37

Binary file

- When a file is opened in *binary mode by adding a b (lowercase only) to the mode string* argument in the built-in open call, reading its data does not decode it in any way but simply returns its content raw and unchanged, as a bytes object; writing similarly takes a bytes object and transfers it to the file unchanged. Binary-mode files also accept a bytearray object for the content to be written to the file.

Working with Files

- When we want to read from or write to a file we need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed. Hence, in Python, a file operation takes place in the following order.
  - Open a file
  - Read or write (perform operation)
  - Close the file

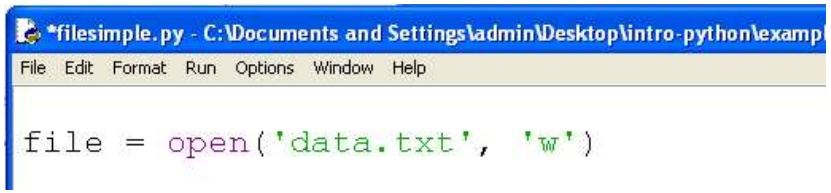
Text Files

- When a file is opened in *text mode, reading its data automatically decodes its content* and returns it as a str; writing takes a str and automatically encodes it before transferring it to the file. Both reads and writes translate per a platform default or a provided ncoding name. Text-mode files also support universal end-of-line translation and additional encoding specification arguments. Depending on the encoding name, text files may also automatically process the byte order mark sequence at the start of a file (more on this momentarily).

The *open* Function

- Before you can read or write a file, you have to open it using Python's built-in *open()* function. This function creates a file object, which would be utilized to call other support methods associated with it.
- **Syntax:**

file object = open(file_name [, access_mode][, encoding])



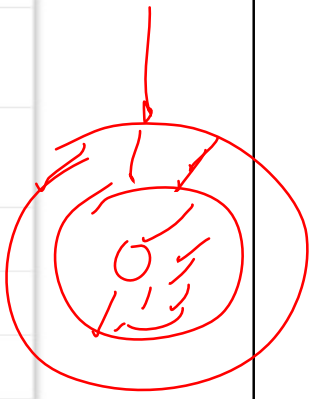
Files mode

- file object = open(file_name [, access_mode][, encoding])
- We can specify the mode while opening a file. In mode, we specify whether we want to read 'r', write 'w' or append 'a' to the file. We also specify if we want to open the file in text mode or binary mode. The default is reading in text mode. In this mode, we get strings when reading from the file. On the other hand, binary mode returns bytes and this is the mode to be used when dealing with non-text files like image or exe files.

Files mode

Python File Modes

Mode	Description
'r'	Open a file for reading. (default)
'w'	Open a file for writing. Creates a new file if it does not exist or truncates the file if it exists.
'x'	Open a file for exclusive creation. If the file already exists, the operation fails.
'a'	Open for appending at the end of the file without truncating it. Creates a new file if it does not exist.
't'	Open in text mode. (default)
'b'	Open in binary mode.
'+'	Open a file for updating (reading and writing)



```
f = open("test.txt")        # equivalent to 'r' or 'rt'
f = open("test.txt",'w')    # write in text mode
f = open("img.bmp",'r+b')   # read and write in binary mode
```

Files encoding

- file object = open(file_name [, access_mode][, encoding])
- Since the version 3.x, Python has made a clear distinction between str (text) and bytes (8-bits). Unlike other languages, the character 'a' does not imply the number 97 until it is encoded using ASCII (or other equivalent encodings). Hence, when working with files in text mode, it is recommended to specify the encoding type. Files are stored in bytes in the disk, we need to decode them into str when we read into Python. Similarly, encoding is performed while writing texts to the file.

Files encoding

- The default encoding is platform dependent. In windows, it is 'cp1252' but 'utf-8' in Linux. Hence, we must not rely on the default encoding otherwise, our code will behave differently in different platforms. Thus, this is the preferred way to open a file for reading in text mode.

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

File Edit Format Run Options Window Help

```
file = open('unidata.txt', 'w', encoding='utf-8')
```

Closing a File

- When we are done with operations to the file, we need to properly close it. Python has a garbage collector to clean up unreferenced objects. But we must not rely on it to close the file. Closing a file will free up the resources that were tied with the file and is done using the close() method.

```
f = open("test.txt",encoding = 'utf-8')
# perform file operations
f.close()
```

Writing to a File

- In order to write into a file we need to open it in write 'w', append 'a' or exclusive creation 'x' mode. We need to be careful with the 'w' mode as it will overwrite into the file if it already exists. All previous data are erased.
- Writing a string or sequence of bytes (for binary files) is done using write() method. This method returns the number of characters written to the file.
- **Syntax:**

  L=fileObject.write(string)

Writing to a File

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

File Edit Format Run Options Window Help

```
f = open('data.txt', 'w')       # Make a new file in output mode ('w' is write)
l=f.write('Hello\n')            # Write strings of characters to it
print(l)                        # Return number of items written
f.write('world\n')
f.close()                       # Close to flush output buffers to disk

#Output
    6
```

Reading From a File

- To read the content of a file, we must open the file in reading mode.
- We can use the read method for reading the file contain.
- **Syntax:**

  text=fileObject.read([count])

Reading From a File

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

File Edit Format Run Options Window Help

```
f = open('data.txt')            # 'r' (read) is the default processing mode
text = f.read()                 # Read entire file into a string
print(text)                     # print interprets control characters
print(text.split())             # File content is always a string
f.close()


#Output

    Hello
    world

    ['Hello', 'world']
```

Examples

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

File Edit Format Run Options Window Help

```
S = 'sp\xc4m'
print(S)
print(type(S))

file = open('unidata.txt', 'w', encoding='utf-8')
file.write(S)
S=' New \n text in file \n'
file.write(S)
S='Hello \n Word'
file.write(S)
file.close()
text = open('unidata.txt', encoding='utf-8').read()
print(text)



#Output
  spÄm
<class 'str'>
spÄm New
 text in file
Hello
 Word
```

Examples

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

File Edit Format Run Options Window Help

```
S = 'sp\xc4m'
file = open('unidata.txt', 'w', encoding='utf-8')
file.write(S)
S=' New \n text in file \n'
file.write(S)
S='Hello \n Word'
file.write(S)
file.close()
file = open('unidata.txt', encoding='utf-8')
for line in file :
    print(line)
file.close()

#Output

spÄm New

 text in file

Hello

 Word
```

Examples

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

```
filename=input('enter the file name: ')
file = open(filename, 'w', encoding='utf-8')
n=int(input('enter the number of data: '))
for i in range(n):
    num=int(input('enter the data: '))
    square=num * num
    file.write(str(square))
    file.write('\n')
file.close()
file = open(filename, encoding='utf-8')
for line in file :
    print(line, end='')
file.close()

#output
    enter the file name: test.txt
    enter the number of data: 3
    enter the data: 4
    enter the data: 6
    enter the data: 5
    16
    36
    25
```

Examples

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

```
filename=input('enter the file name: ')
file = open(filename, 'w', encoding='utf-8')
n=int(input('enter the number of data: '))
for i in range(n):
    num=int(input('enter the data: '))
    square=num * num
    file.write(str(square))
    file.write('\n')
file.close()
file = open(filename, encoding='utf-8')
for i in range(n):
    line=file.readline()
    print(line, end='')
file.close()

#output
    enter the file name: test.txt
    enter the number of data: 3
    enter the data: 4
    enter the data: 6
    enter the data: 5
    16
    36
    25
```

Examples

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

File Edit Format Run Options Window Help

```
X, Y, Z = 43, 44, 45                        # Native Python objects
S = 'Spam'                                  # Must be strings to store in file
D = {'a': 1, 'b': 2}
L = [1, 2, 3]
filename=input('enter the file name: ')
F = open(filename, 'w')
F.write(S + '\n')                           # Terminate lines with \n
F.write('%s,%s,%s\n' % (X, Y, Z))           # Convert numbers to strings
F.write(str(L) + '$' + str(D) + '\n')       # Convert and separate with $
F.close()

chars = open(filename).read()               # Raw string display
print(chars)                                # User-friendly display

#Output
    enter the file name: test.txt
    Spam
    43,44,45
    [1, 2, 3]${'a': 1, 'b': 2}
```

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

File Edit Format Run Options Window Help

```
X, Y, Z = 43, 44, 45                        # Native Python objects
S = 'Spam'                                  # Must be strings to store in file
D = {'a': 1, 'b': 2}
L = [1, 2, 3]
filename=input('enter the file name: ')
F = open(filename, 'w')
F.write(S + '\n')                           # Terminate lines with \n
F.write('%s,%s,%s\n' % (X, Y, Z))           # Convert numbers to strings
F.write(str(L) + '$' + str(D) + '\n')       # Convert and separate with $
F.close()

F = open(filename)
line=F.readline()
print(line)
line=F.readline()
print(line)
line=F.readline()
print(line)
parts = line.split('$')                     # Split (parse) on $
print(parts)
pr=eval(parts[0])                           # Convert to any object type
print(pr)
objects = [eval(P) for P in parts]          # Do same for all in list
print(objects)
```

Examples

```
#Output

enter the file name: test.txt
Spam

43,44,45

[1, 2, 3]${'b': 2, 'a': 1}

['[1, 2, 3]', "{'b': 2, 'a': 1}\n"]
[1, 2, 3]
[[1, 2, 3], {'a': 1, 'b': 2}]
```

With Statement

- Another way of working with file objects is the With statement.

Examples

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

File Edit Format Run Options Window Help

```
filename=input('enter the file name: ')
with open(filename,'w',encoding = 'utf-8') as f:
    f.write("This is my first file\n")
    f.write("This file\n\n")
    f.write("contains three lines\n")

f = open(filename, 'r', encoding = 'utf-8')
print(f.read(4))
print(f.read(4))
print(f.read())
f.close()

#Output

    enter the file name: test.txt
    This
     is
    my first file
    This file

    contains three lines
```

Group of Data

However, sometimes a counting loop must be used with input. The data sometimes consist of several groups where each group starts with a header that says how much data is in that group. For example:

> There are two groups of computer science students. The first group (called group "A") uses on-line lessons. The other group (called group "B") uses traditional printed text. All the students are given the same mid-term examination. Which group has the higher test average?

The file of test scores looks like this (the blue comments are not in the file):

```
3          <-- number of students in group "A"
87
98
95
4          <-- number of students in group "B"
78
82
91
84
```

Group "A" has three students in it and group "B" has four students in it. The program is to compute two averages from the data in this file.

mid.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/file/mid.py (3.4.4)

File Edit Format Run Options Window Help

```
filename=input('enter the file name: ')
F = open(filename, 'r')
sizeA=int(F.readline())
sumA=0
count=0
while(count<sizeA):
    value=int(F.readline())
    sumA=sumA+value
    count=count+1

if (sizeA>0):
    print('Group A average: ', sumA/sizeA)
else:
    print('Group A has no student')

sizeB=int(F.readline())
sumB=0
count=0
while(count<sizeB):
    value=int(F.readline())
    sumB=sumB+value
    count=count+1

if (sizeB>0):
    print('Group B average: ', sumB/sizeB)
else:
    print('Group B has no student')


#Output

enter the file name: data.txt
Group A average:  93.33333333333333
Group B average:  83.75
```

Compare text file

cmp.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/file/cmp.py (3.4.4)

File Edit Format Run Options Window Help

```
import sys

if len(sys.argv) < 3:
    print("Wrong parameter")
    print("./cmp.py filename1 filename2")
    sys.exit(1)

print('Data of the files:')
with open(sys.argv[1]) as f1, open(sys.argv[2]) as f2:
    for pair in zip(f1, f2):
        print(pair)

print()
print('Difference between the files:')
with open(sys.argv[1]) as f1, open(sys.argv[2]) as f2:
    for (linenum, (line1, line2)) in enumerate(zip(f1, f2)):
        if line1 != line2:
            print('%s\t%r\t%r' % (linenum, line1, line2))

#Output

C:\examples\file>cmp script1.txt script2.txt
Data of the files:
('Spam\n', 'Spam\n')
('43,44,45\n', '43,44,45\n')
('student\n', '72\n')
("[1, 2, 3]${'b': 2, 'a': 1}\n", "[1, 2, 3]${'b': 2, 'a': 1}\n")
('final line 1', 'final line 2')

Difference between the files:
2       'student\n'     '72\n'
4       'final line 1'  'final line 2'
```

Copy text file

cpy.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\cpy.py (3.4.4)

File Edit Format Run Options Window Help

```
import sys

if len(sys.argv) < 3:
    print("Wrong parameter")
    print("./cpy.py filename1 filename2")
    sys.exit(1)


print('Copy  file:')
with open(sys.argv[1]) as fin, open(sys.argv[2],'w') as fout:
     for line in fin:
         fout.write(line.upper())
```

Phone Book

- Now here is a new version of the phone numbers program that we made earlier.
- This version uses a text file for loding and saving the phone numbers

```
def print_menu():
    print()
    print('1. Print Phone Numbers')
    print('2. Add a Phone Number')
    print('3. Remove a Phone Number')
    print('4. Lookup a Phone Number')
    print('5. Load numbers')
    print('6. Save numbers')
    print('7. Quit')
    print()

phone_list = {}
menu_choice = 0
while True:
    print_menu()
    menu_choice =int(input("Type in a number (1-7): "))
    if menu_choice == 1:
        print_numbers(phone_list)
    elif menu_choice == 2:
        print("Add Name and Number")
        name = input("Name: ")
        phone =input("Number: ")
        add_number(phone_list, name, phone)
    elif menu_choice == 3:
        print("Remove Name and Number")
        name =input("Name: ")
        remove_number(phone_list, name)
    elif menu_choice == 4:
        print("Lookup Number")
        name = input("Name: ")
        print(lookup_number(phone_list, name))
```

```
phone_list = {}
menu_choice = 0
while True:
    print_menu()
    menu_choice =int(input("Type in a number (1-7): "))
    if menu_choice == 1:
        print_numbers(phone_list)
    elif menu_choice == 2:
        print("Add Name and Number")
        name = input("Name: ")
        phone =input("Number: ")
        add_number(phone_list, name, phone)
    elif menu_choice == 3:
        print("Remove Name and Number")
        name =input("Name: ")
        remove_number(phone_list, name)
    elif menu_choice == 4:
        print("Lookup Number")
        name = input("Name: ")
        print(lookup_number(phone_list, name))
    elif menu_choice == 5:
        filename = input("Filename to load: ")
        load_numbers(phone_list, filename)
    elif menu_choice == 6:
        filename = input("Filename to save: ")
        save_numbers(phone_list, filename)
    elif menu_choice == 7:
        break
    else:
        continue

print("Goodbye")
```

```python
def print_numbers(numbers):
    print("Telephone Numbers:")
    for k, v in numbers.items():
        print("Name:", k, "\tNumber:", v)
    print()

def add_number(numbers, name, number):
    numbers[name] = number

def lookup_number(numbers, name):
    if name in numbers:
        return "The number is " + numbers[name]
    else:
        return name + " was not found"

def remove_number(numbers, name):
    if name in numbers:
        del numbers[name]
    else:
        print(name," was not found")

def load_numbers(numbers, filename):
    in_file = open(filename, "rt")
    while True:
        in_line = in_file.readline()
        if not in_line:
            break
        in_line = in_line[:-1]
        name, number = in_line.split(",")
        numbers[name] = number
    in_file.close()

def save_numbers(numbers, filename):
    out_file = open(filename, "wt")
    for k, v in numbers.items():
        out_file.write(k + "," + v + "\n")
    out_file.close()
```

name , phn

```
1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 2
Add Name and Number
Name: ali
Number: 774493

1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 2
Add Name and Number
Name: hasan
Number: 443356

1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7):
```

```
1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 2
Add Name and Number
Name: kayvan
Number: 354418

1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 1
Telephone Numbers:
Name: kamran    Number: 363131
Name: hasan     Number: 443356
Name: kayvan    Number: 354418
Name: ali       Number: 774493
Name: kazem     Number: 521746
```

Book

```
1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 1
Telephone Numbers:
Name: kamran    Number: 363131
Name: hasan     Number: 443356
Name: kayvan    Number: 354418
Name: ali       Number: 774493
Name: kazem     Number: 521746

Type in a number (1-7): 3
Remove Name and Number
Name: kazem

Type in a number (1-7): 1
Telephone Numbers:
Name: kamran    Number: 363131
Name: hasan     Number: 443356
Name: kayvan    Number: 354418
Name: ali       Number: 774493
```

Book

```
1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 1
Telephone Numbers:
Name: kamran    Number: 363131
Name: hasan     Number: 443356
Name: kayvan    Number: 354418
Name: ali       Number: 774493


1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 4
Lookup Number
Name: ali
The number is 774493
```

Phone Book

```
1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 6
Filename to save: phone.txt

1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit
```

Phone Book

```
1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 5
Filename to load: phone.txt

1. Print Phone Numbers
2. Add a Phone Number
3. Remove a Phone Number
4. Lookup a Phone Number
5. Load numbers
6. Save numbers
7. Quit

Type in a number (1-7): 1
Telephone Numbers:
Name: kamran    Number: 363131
Name: hasan     Number: 443356
Name: kayvan    Number: 354418
Name: ali       Number: 774493
```

Storing Objects in JSON Format

- The prior section's pickle module translates nearly arbitrary Python objects to a proprietary format developed specifically for Python, and honed for performance over many years.
- JSON is a newer and emerging data interchange format, which is both programming-language-neutral and supported by a variety of systems. *MongoDB, for* instance, stores data in a JSON document database (using a binary JSON format).

Storing Objects in JSON Format

- JSON does not support as broad a range of Python object types as pickle, but its portability is an advantage in some contexts, and it represents another way to serialize a specific category of Python objects for storage and transmission. Moreover, because JSON is so close to Python dictionaries and lists in syntax, the translation to and from Python objects is trivial, and is automated by the json standard library module.

Storing Objects in JSON Format

- For example, a Python dictionary with nested structures is very similar to JSON data, though Python's variables and expressions support richer structuring options (any part of the following can be an arbitrary expression in Python code)
- dump for write and load for read in to files
- dumps return json representation of a real data
- loads convert a json to real data

Storing Objects in JSON Format

filesimple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\file\filesimple.py (3.4.4)

File Edit Format Run Options Window Help

```
import json

name = dict(first='Bob', last='Smith')
rec  = dict(name=name, job=['dev', 'mgr'], age=40.5)
print(rec)

s=json.dumps(rec)
print(s)

O = json.loads(s)
print(O)

json.dump(rec, fp=open('testjson.txt', 'w'), indent=4)
print(open('testjson.txt').read())


P = json.load(open('testjson.txt'))
print(P)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32 bit (Int
el)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\intro-python\examples\file\file
simple.py
{'name': {'last': 'Smith', 'first': 'Bob'}, 'age': 40.5, 'job': ['dev', 'mgr']}
{"name": {"last": "Smith", "first": "Bob"}, "age": 40.5, "job": ["dev", "mgr"]}
{'name': {'last': 'Smith', 'first': 'Bob'}, 'age': 40.5, 'job': ['dev', 'mgr']}
{
    "name": {
        "last": "Smith",
        "first": "Bob"
    },
    "age": 40.5,
    "job": [
        "dev",
        "mgr"
    ]
}
{'name': {'last': 'Smith', 'first': 'Bob'}, 'age': 40.5, 'job': ['dev', 'mgr']}
```