

# 14 Files

A common programming task is to read or write information from/to a file.

# Files: reading from files

A common programming task is to read or write information from/to a file.

Let's first see how to read from a file:

1) Download the file `source.txt` from our web-page and place it into `Documents` folder

2) type in the following command in the `Python Shell`:

```
>>> mySource = open("source.txt")
```

`source.txt`

1

1.4

0

1.6

-4

# Files: reading from files

A common programming task is to read or write information from/to a file.

Let's first see how to read from a file:

1) Download the file `source.txt` from our web-page and place it into `Documents` folder

2) type in the following command in the `Python Shell`:

```
>>> mySource = open("source.txt")
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    mySource=open("source.txt")
FileNotFoundError: [Errno 2] No such file or
directory: 'source.txt'
```

# Files: reading from files

A common programming task is to read or write information from/to a file.

3) Create a new file in [Python Editor](#), type in the following commands and save the file in the [Documents](#) folder

```
mySource = open("source.txt")
for line in mySource:
    print(line)
mySource.close()
```

4) Run it!

source.txt :

```
1
1.4
0
1.6
-4
```

# Files: reading from files

## Methods for reading from a file:

method	Description
<code>open(&lt;filename&gt;)</code>	opens a file named <filename>
<code>close()</code>	closes an opened file
<code>read()</code>	reads everything from a file and returns it as string
<code>readlines()</code>	returns a list of strings
<code>readline()</code>	reads one line from a file and returns it as a string
<code>for i in f</code>	Iterates over the lines in a file (i is a line from the file)

# Files: reading from files

We can replace the previous program:

```
mySource = open("source.txt")
for line in mySource:
    print(line)
mySource.close()
```

with:

*workWithFiles1.py*

```
mySource = open("source.txt")
text = mySource.read()
print(text)
mySource.close()
```

*workWithFiles2.py*

source.txt :

```
1
1.4
0
1.6
-4
```

Python Shell:

```
1
1.4
0
1.6
-4
```

# Files: reading from files

What if we want to find the sum of all numbers in a file?

source.txt

1

1.4

0

1.6

-4

# Files: reading from files

What if we want to find the sum of all numbers in a file?

```
mySource = open("source.txt")  
→ s = 0  
for line in mySource:  
→ s += float(line)  
mySource.close()  
→ print("The sum of all values is",s)
```

source.txt

1  
1.4  
0  
1.6  
-4

# Files: reading from files

What if we want to find the sum of all numbers in a file?

```
mySource = open("source.txt")
→ s = 0
for line in mySource:
→     s += float(line)
mySource.close()
→ print("The sum of all values is",s)
```

The sum of all values is 0.0

*You can find this program in the file  
workWithFiles3.py*

source.txt

1  
1.4  
0  
1.6  
-4

# Files: reading from files

What if we want to find the sum of all numbers in a file, but the source file is formatted differently?

- the numbers are separated by a **space** or by a **new line**

```
source2.txt
1 8 9.2 -5
1.4 9 8
0 12 -23 -9 1
1.6 2.3 -9.1
-42 -91 76 23 7
```

# Files: reading from files

What if we want to find the sum of all numbers in a file, but the source file is formatted differently?

- the numbers are separated by a **space** or by a **new line**

```
mySource = open("source2.txt")
s = 0
for line in mySource:
    line = line.rstrip()
    numbers = line.split(" ")
    for item in numbers:
        s += float(item)
mySource.close()
print("The sum of all values is",s)
```

source2.txt

1	8	9.2	-5	
1.4	9	8		
0	12	-23	-9	1
1.6	2.3	-9.1		
-42	-91	76	23	7

You can find this program in the file `workWithFiles4.py`

# Files: reading from files

## In-class activity 1

What if we want to find the sum of all numbers in a file, but the source file is formatted differently?

- the numbers are separated by a **comma** and by a **new line**

Modify the program in the file **workwithFiles4.py** to accommodate the new formatting.

**source3.txt**

```
1,8,9.2,-5  
1.4,9,8,13  
0,12,-23,-9,1  
1.6,2.3,-9.1,-5  
-42,-91,7,23,17
```

# Files: writing to files

Let's first see how to write to a file:

Type in the following in the [Python Shell](#):

```
>>> myOut = open("out.txt", 'w')
>>> myOut.write("Hello, my name is Fran.\n")
>>> myOut.write("I like to dance ")
>>> myOut.write("and sing.")
>>> myOut.close()
```

Let's try to find the file we just created:

Try the folder where all the programs you worked today with are.

# Files: writing to files

Let's first see how to write to a file:

Type in the following in the [Python Shell](#):

```
>>> myOut = open("out.txt", 'w')
>>> myOut.write("Hello, my name is Fran.\n")
>>> myOut.write("I like to dance ")
>>> myOut.write("and sing.")
>>> myOut.close()
```

Let's try to find the file we just created:

Try the folder where all the programs you worked today with are.

Or, do the following:

```
>>> myOut = open("out.txt")
>>> print(myOut.read())
>>> myOut.close()
```

# Files: writing to files

We can add more to the file that already exists:

Type in the following in the **Python Shell**:

```
>>> myOut = open("out.txt", 'a')
>>> myOut.write("I went to the circus today.")
>>> myOut.close()
```

Then find the file and see how it changed. Or, do the following:

```
>>> myOut = open("out.txt")
>>> print(myOut.read())
>>> myOut.close()
```

*You can find this program in the file `workWithFiles5.py`*

# Files: writing to files

## Methods for writing to a file

method	description
<code>open(&lt;filename&gt;, 'w')</code> <code>open(&lt;filename&gt;, 'a')</code>	open the file for writing ('w') or appending ('a')
<code>write(&lt;text&gt;)</code>	writes a string <text> to a file

# Files: writing to files

**Example:** let's ask the user to enter a *positive integer  $n$*  and a *file name*, create a file with the given file name and put the squares of the first  $n$  positive integers into it, separated by space.

## Files: writing to files

**Example:** let's ask the user to enter a *positive integer*  $n$  and a *file name*, create a file with the given file name and put the squares of the first  $n$  positive integers into it, separated by space.

```
n = int(input("Enter a positive integer:"))
if n > 0:
    fname = input("Please enter a file name:")
    output = open(fname + ".txt", 'w')

else:
    print("positive integer is expected!")
```

You can find this program in the file `workWithFiles6.py`

## Files: writing to files

**Example:** let's ask the user to enter a *positive integer*  $n$  and a *file name*, create a file with the given file name and put the squares of the first  $n$  positive integers into it, separated by space.

```
n = int(input("Enter a positive integer:"))
if n > 0:
    fname = input("Please enter a file name:")
    output = open(fname + ".txt", 'w')
    for i in range(1, n+1):
        output.write(str(i*i) + ' ')
else:
    print("positive integer is expected!")
```

You can find this program in the file `workWithFiles6.py`

## Files: writing to files

**Example:** let's ask the user to enter a *positive integer*  $n$  and a *file name*, create a file with the given file name and put the squares of the first  $n$  positive integers into it, separated by space.

```
n = int(input("Enter a positive integer:"))
if n > 0:
    fname = input("Please enter a file name:")
    output = open(fname + ".txt", 'w')
    for i in range(1, n+1):
        output.write(str(i*i) + ' ')
    output.close()
    print("Done! Look for the file", fname + '.txt')
else:
    print("positive integer is expected!")
```

You can find this program in the file `workWithFiles6.py`

# Files: writing to files

## In-class activity 2

Modify the program we just did (see file `workWithFiles6.py`) to output the squares of the first  $n$  positive integers into it, separated by new line character (`'\n'`), i.e. one value per line.

out.txt

```
1  
4  
9  
16  
25  
36
```

# Files: writing to files

In-class activities 3-4