Files

Files

Let’s say I have the following .txt file saved on my computer.

Open file:

```
f= open("myFile.txt", "r")
```

Files

Let's say I have the following .txt file saved on my computer.

Open file:

```
f= open("myFile.txt", "r")
```

File path

- "r": read
- "w": write
- "a": append

Files

Let's say I have the following .txt file saved on my computer.

Open file:

```
f= open("myFile.txt", "r")
```

File path

- "r": read
- "w": write
- "a": append

Reading from Files

Read the entire file into a string all at once

read() method:

```
f= open("myFile.txt", "r")
```

Everything will be read in as a string

```
entire_file = f.read()
entire_file
```

```
'hello text file \ngoodbye text file \n'
```

```
entire_file = f.read()
entire_file
```

```
''
```

Reading from Files

Read the entire file into a string all at once

read() method:

```
f= open("myFile.txt", "r")
```

```
entire_file = f.read()
entire_file
```

```
'hello text file \ngoodbye text file \n'
```

```
entire_file = f.read()
entire_file
```

```
''
```

Reading from Files

Read the entire file into a string all at once

read() method:

```
f= open("myFile.txt", "r")
```

```
entire_file = f.read()
entire_file
```

```
'hello text file \ngoodbye text file \n'
```

```
entire_file = f.read()
entire_file
```

```
''
```

Reading from Files

Read one line at a time

readline() method:

```
f= open("myFile.txt", "r")
```

```
f.readline()
```

```
'hello text file \n'
```

```
f.readline()
```

```
'goodbye text file \n'
```

```
f.readline()
```

```
''
```

Reading from Files

Read one line at a time

readline() method:

```
f= open("myFile.txt", "r")
```

```
f.readline()
```

```
'hello text file \n'
```

```
f.readline()
```

```
'goodbye text file \n'
```

```
f.readline()
```

```
''
```

Reading from Files

Read one line at a time

readline() method:

```
f= open("myFile.txt", "r")
```

```
f.readline()
```

```
'hello text file \n'
```

```
f.readline()
```

```
'goodbye text file \n'
```

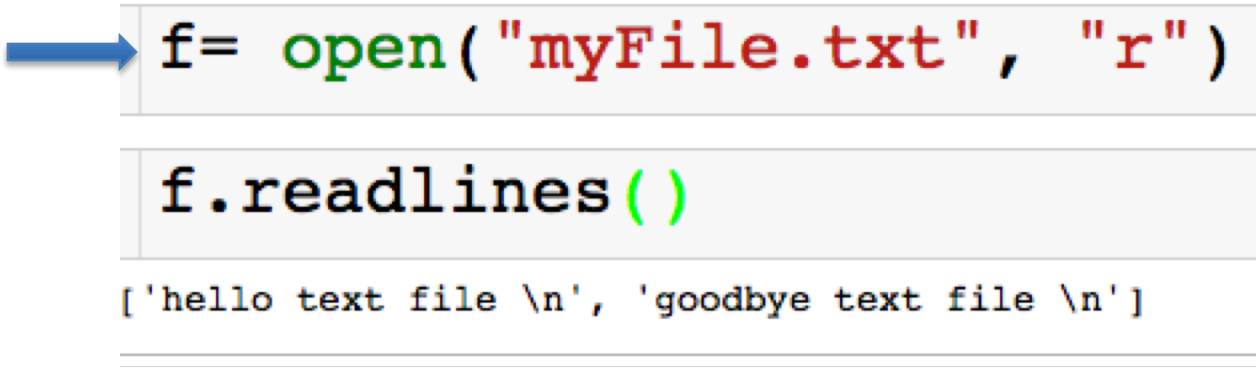
```
f.readline()
```

```
''
```

Reading from Files

Read all the lines at one into a list

readlines() method:



Reading from Files

Read all the lines at one into a list

readlines() method:

```
f= open("myFile.txt", "r")
f.readlines()
['hello text file \n', 'goodbye text file \n']
```

Reading from Files

Read all the lines at one into a list

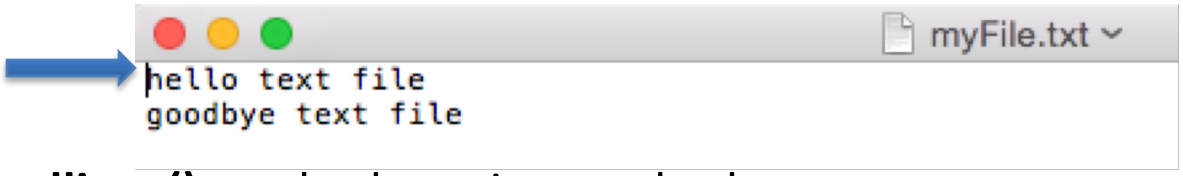
readlines() method:

```
f= open("myFile.txt", "r")
f.readlines()
['hello text file \n', 'goodbye text file \n']
```

We get a list of strings where each element in the list is a line

Reading from Files

Read all the lines at one into a list



readlines() method + string methods:

```
f= open("myFile.txt", "r")
lines = f.readlines()
for line in lines:
    print(line.strip(' \n').split(' '))
```

```
['hello', 'text', 'file']
['goodbye', 'text', 'file']
```

```
lines
```

```
['hello text file \n', 'goodbye text file \n']
```

Reading from Files

Read all the lines at one into a list

readlines() method + string methods:

```
f= open("myFile.txt", "r")
lines = f.readlines()
for line in lines:
    print(line.strip(' \n').split(' '))
```

```
['hello', 'text', 'file']
['goodbye', 'text', 'file']
```

```
lines
```

```
['hello text file \n', 'goodbye text file \n']
```

Reading from Files

Read all the lines at one into a list

readlines() method + string methods:

```
f= open("myFile.txt", "r")
lines = f.readlines()
for line in lines:
    print(line.strip(' \n').split(' '))
```

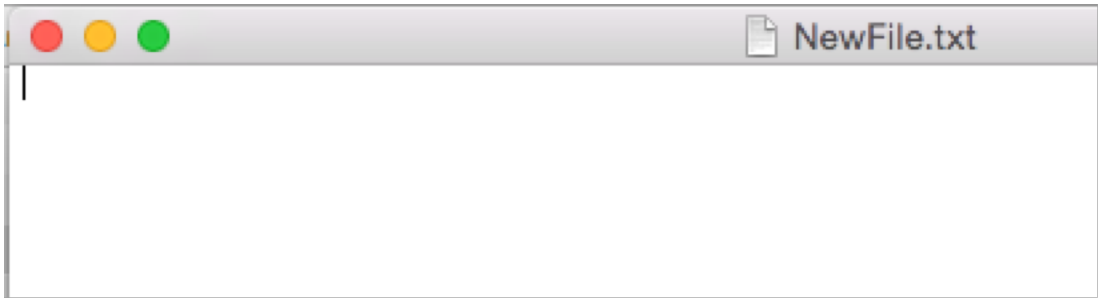
```
['hello', 'text', 'file']
['goodbye', 'text', 'file']
```

```
lines
```

```
['hello text file \n', 'goodbye text file \n']
```

Writing to Files

Writing to files



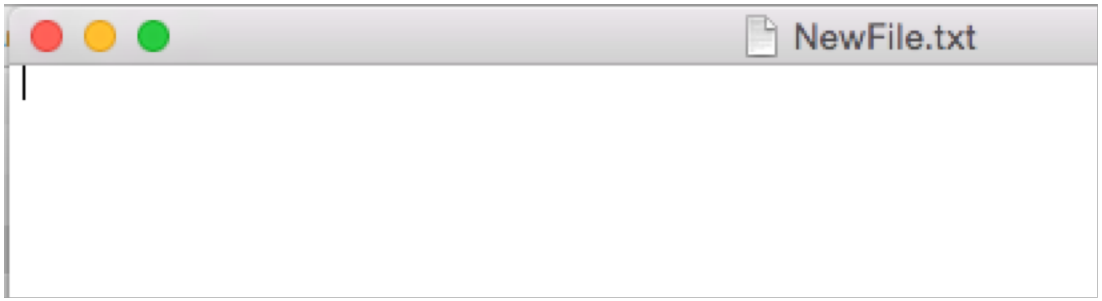
Open file:

```
f= open("NewFile.txt", "w")
```

- Creates new blank .txt file called “NewFile.txt” that we can then write too.
- If this file exists, then it will overwrite it with a blank file…so be careful.

Writing to Files

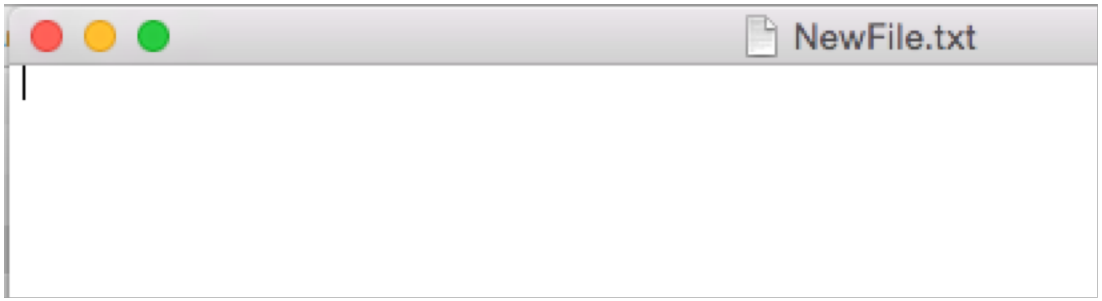
Writing to files



write() method:

Writing to Files

Writing to files



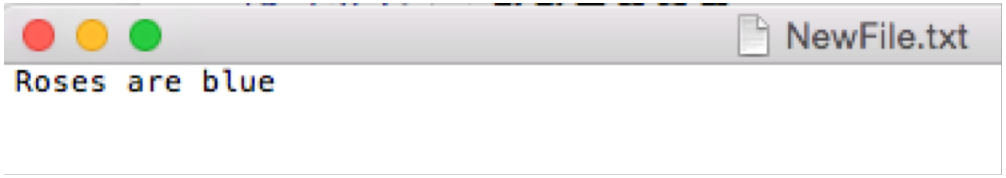
write() method:

```
f= open("NewFile.txt", "w")
f.write("Roses are blue")
```

We can only write strings to files

Files

Writing to files

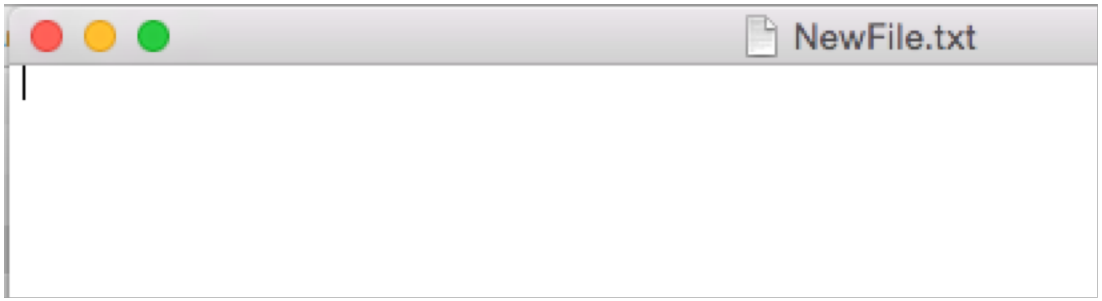


write() method:

```
f= open("NewFile.txt", "w")
f.write("Roses are blue")
```

Writing to Files

Writing to files

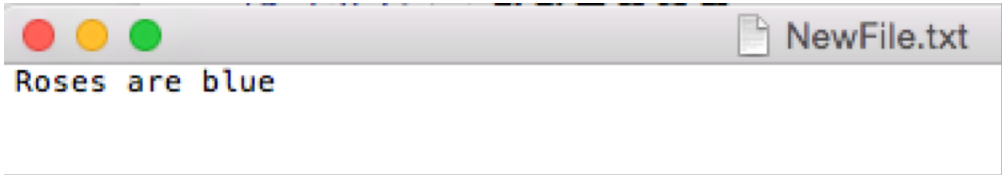


write() method:

```
f= open("NewFile.txt", "w")
f.write("Roses are blue")
f.write("Violets are red")
f.close()
```

Writing to Files

Writing to files

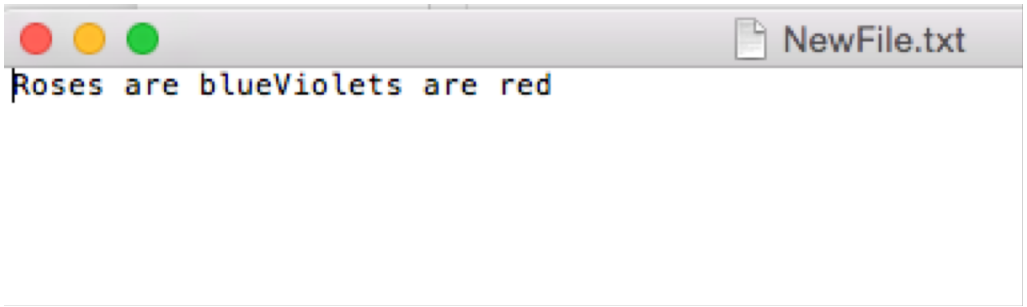


write() method:

```
f= open("NewFile.txt", "w")
f.write("Roses are blue")
f.write("Violets are red")
f.close()
```

Writing to Files

Writing to files

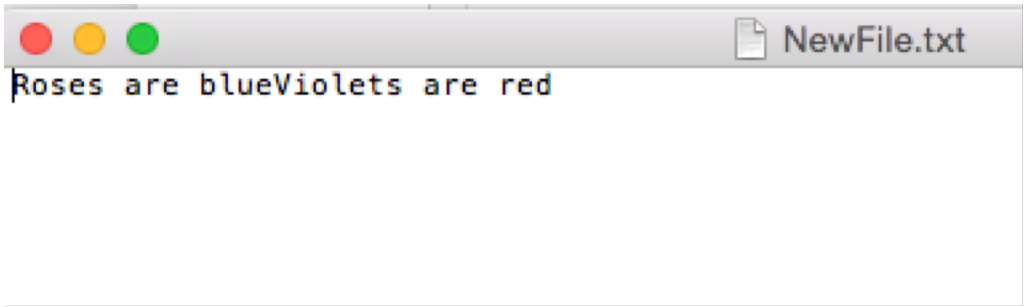


write() method:

```
f= open("NewFile.txt", "w")
f.write("Roses are blue")
f.write("Violets are red")
f.close()
```

Writing to Files

Writing to files



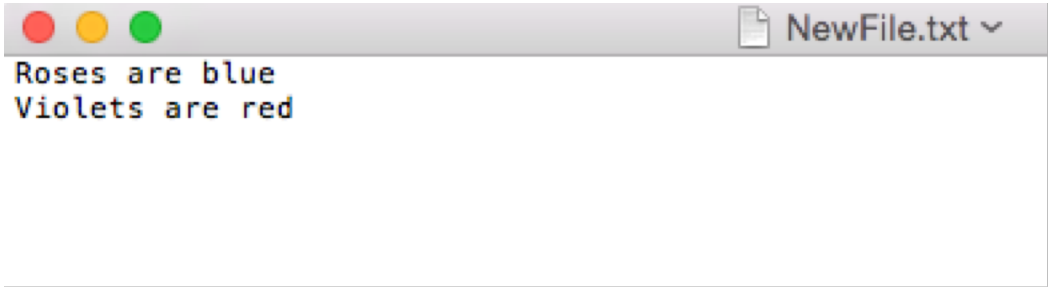
write() method:

```
f= open("NewFile.txt", "w")
f.write("Roses are blue")
f.write("Violets are red")
f.close()
```

We wanted to print the second line on a new line!

Writing to Files

Writing to files



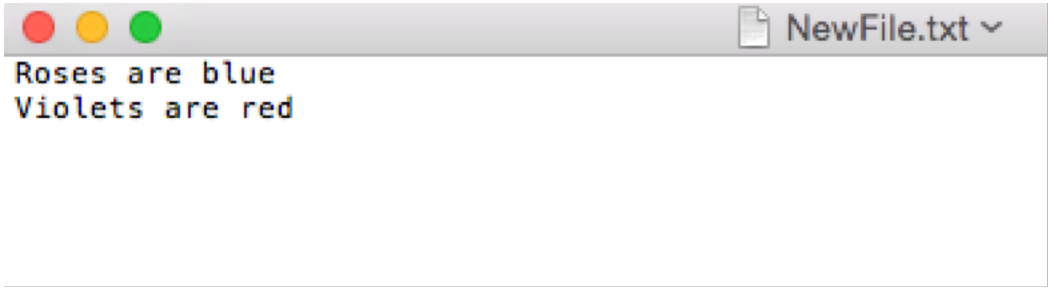
write() method:

```
f= open("NewFile.txt", "w")
f.write("Roses are blue\n")
f.write("Violets are red")
f.close()
```

\n = new line

Appending to Files

Appending to a file



Will not overwrite like the 'w' will.

write() method:

```
f= open("NewFile.txt", "a")
f.write("\nOops...I messed up")
f.close()
```

Appending to Files

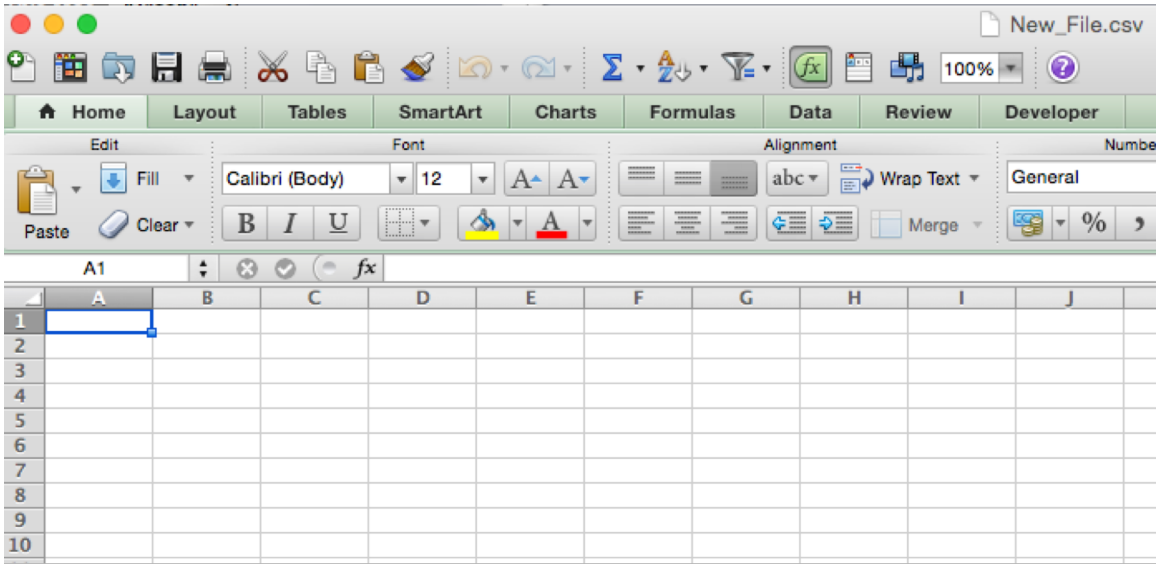
Appending to a file

Will not overwrite like the 'w' will.

write() method:

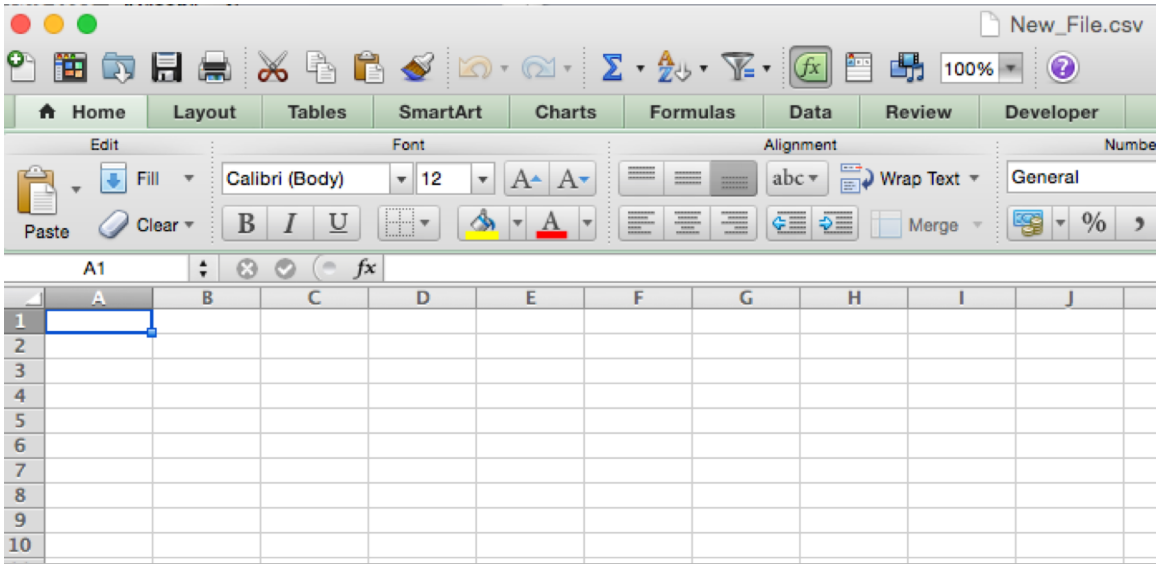
```
f= open("NewFile.txt", "a")
f.write("\nOops...I messed up")
f.close()
```

Writing Variables




```
#Create blank csv
f = open("New_File.csv", "w")
```

Writing Variables

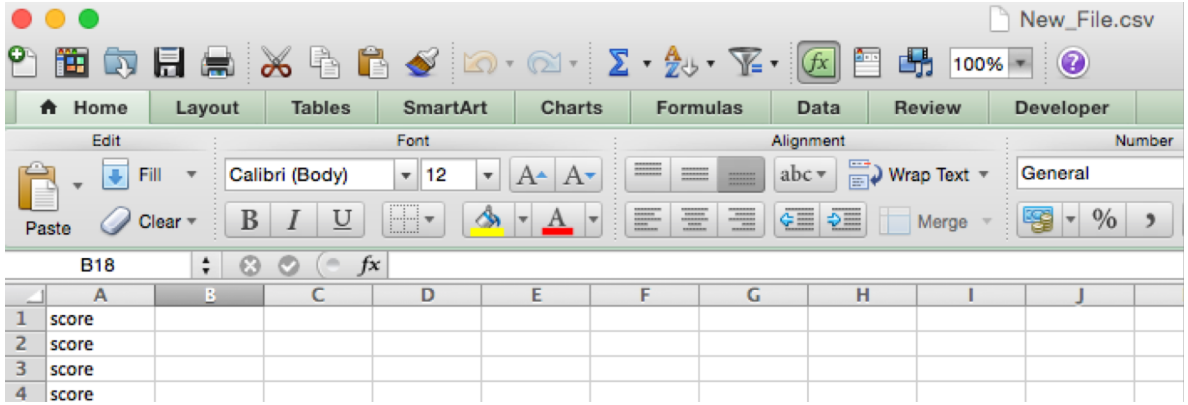



```
#Create blank csv
f = open("New_File.csv", "w")
```

```
#Write column name
score_list = [93,100, 74,50]
for score in score_list:
    f.write("score \n")
```

Always write a string

Writing Variables




```
#Create blank csv
f = open("New_File.csv", "w")
```

```
#Write column name
score_list = [93,100, 74,50]
for score in score_list:
    f.write("score \n")
```

We write the word score instead of each of the scores!

Formatting String

This leads us to a more general question: **How do I place the value of variable (string/float/int) inside of a string?**

Formatting String

This leads us to a more general question: **How do I place the value of variable (string/float/int) inside of a string?**

```
num_cookies = 10
sentence = "I ate num_cookies cookies"

sentence
```

```
'I ate num_cookies cookies'
```

Wanted sentence to be equal to the string “I ate 10 cookies”

Formatting String - Integers

To format strings:

1. On the left of the % operator, provide a format string containing one or more embedded conversion targets, each of which starts with a %.

2. On the right side of the % operator, provide the object (or objects, embedded in a tuple) that you want Python to insert into the format string on the left in place of the conversion targets.

```
num_cookies = 10
sentence = "I ate %d cookies" %num_cookies
```

Conversion Targets:

- %d = replace with integer
- %s = replace with string
- %f = replace with float

Variable value to insert

Formatting String - Integers

```
num_cookies = 10
sentence = "I ate %d cookies" %num_cookies

sentence
```

```
'I ate 10 cookies'
```

```
day = 2
sentence = "His birthday is 4/%d/1996" %day

sentence
```

```
'His birthday is 4/2/1996'
```

```
frac_cookies = 10.5
sentence = "I ate %d cookies" %frac_cookies

sentence
```

```
'I ate 10 cookies'
```

Formatting String - Floats

```
frac_cookies = 10.546
sentence = "I ate %f cookies" %frac_cookies

sentence
```

```
'I ate 10.546000 cookies'
```

%f for inserting float (decimal)

Formatting String - Floats

```
frac_cookies = 10.546
sentence = "I ate %f cookies" %frac_cookies

sentence
```

```
'I ate 10.546000 cookies'
```

%f for inserting float (decimal)

How do I control number of decimal points?

Formatting String - Floats

```
frac_cookies = 10.546
#No extra formatting
sentence = "I ate %f cookies" %frac_cookies

sentence
```

```
'I ate 10.546000 cookies'
```

```
#One decimal point
sentence = "I ate %0.1f cookies" %frac_cookies

sentence
```

```
'I ate 10.5 cookies'
```

```
#Thee decimal points
sentence = "I ate %0.3f cookies" %frac_cookies

sentence
```

```
'I ate 10.546 cookies'
```

Formatting String - Strings

```
name = "Charlie"

greeting = "Hello there, %s!" %name
greeting
```

```
'Hello there, Charlie!'
```

```
num_str = '6'
sentence = "He ate %s slices" %num_str
sentence
```

```
'He ate 6 slices'
```

Formatting String - Strings

```
name = "Charlie"

greeting = "Hello there, %s!" %name
greeting
```

```
'Hello there, Charlie!'
```

```
num_str = '6'
sentence = "He ate %s slices" %num_str
sentence
```

```
'He ate 6 slices'
```

Formatting String- Multiple Replacements

```
goals = 8
assists = 10
#Two conversions, both integers
sentence = "He has %d goals and %d assists" %(goals, assists)

sentence
```

```
'He has 8 goals and 10 assists'
```

- Give variable names as a tuple
- Order matters

Formatting String- Multiple Replacements

```
name = "Joe"
GPA = 3.96
sentence= "%s has a GPA of %0.2f" %(name, GPA)

sentence
```

```
'Joe has a GPA of 3.96'
```

```
name = "Joe"
GPA = 3.96
num_years = 2
sentence= "%s has a GPA of %0.2f after %d years of school" %(name, GPA, num_years)

sentence
```

```
'Joe has a GPA of 3.96 after 2 years of school'
```

Back to Writing to Files

```
#Write column name
score_list = [93,100, 74,50]
for score in score_list:
    f.write("%d \n" %score)

f.close()
```