User-Defined functions

Built-in functions

- len () – requires an object (argument)
  - Length of string – number of characters
    - s = ‘CSC 401’
    - len(s) – returns?
  - Length of list – number of items
    - courses = [‘CSC401’, ‘CSC402’, ‘CSC451’, ‘CSC452’]
    - len(courses) – returns?
- min() – requires 1 or more arguments
  - min(30, 10, 20) – return?
- print() is this a function?

User-Defined functions

DEFINING NEW FUNCTIONS

```
def <function name> (<0 or more arguments>):
        < iterated statements>

# one or more Python statements,
# all with a tab (indent) in front of them
```

functions can print() or return values

```
def f(x) :
    res = x ** 2 + 10
    print('The result is ',  res)
    return res
```

def: function definition keyword
f: name of function
x: variable name for input argument

Exercise: z = 1
what is printed ?
what is returned ?

Immutable Parameter Passing

Called module: def f(x)

```
def f(x):
     x = x ** 2
     res  = x  + 10
     print ( x, '  ', res)
     return res
```

At start of module

Calling module: main()

```
def main():
     a = 2
     y = f(a)
     print (a, '  ', y)
```

User-Defined Function Practice

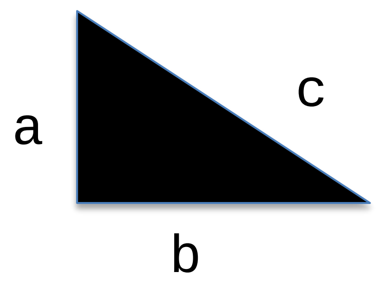
UDF – Example 1

In Python Program:
A function must be defined before it can be called
A function is executed when it is called

Write the function hyp (a, b) that
- Takes two numbers as input
- Returns the length of the hypotenuse c

1. a and b are passed from the calling function
2. Perform calculation
3. Pass back c to the calling program

Problem solving:



UDF – Example 2

Note: Parameters are immutable

Write the function median (a, b, c) that
- Takes three numbers as input
- Returns the median value

1. Called function:
   1. Compute the median value
   2. Return the median value
2. Calling function
   1. Get 3 numbers from the user
   2. Call median function
   3. Output median value

Key Knowledge:

a < b and b < c, median is b
a < b and a < c, median is c
a < b and c < b, median is a
b <= a and a < c, median is a
b <= a and c < b, median is b
b <= a and b < c, median is c

Median is the "middle" of the sequence of 3 numbers.

Example: 12, 3, 10
The median value is 10.

UDF – Example 3

Note: Parameter is mutable

Write the function medianList (nums) that
- Is passed a **list** of numbers via a parameter
- Returns the index of the value in the median position

1. Called function:
   1. Sort the list
   2. Compute the median index
   3. Return index value
2. Calling function
   1. Get list of numbers from user
   2. Call medianList function
   3. Output median value

Example: [12, 3, 10, 5, 6]
The median value is 6.

Key Knowledge: Use list sort method (p. 32)
Divide the length of the list by 2 – want the quotient

Iteration Structures

Change the flow of control

Strings and Lists

Iteration (Repetition)

- A loop is used to repeat a sequence of statements a number of times.
- At each repetition the statements act upon variables whose values are changing.
- Strings and lists are sequences of objects
  - String: sequence of one-character strings
  - List: sequence of object of any types

Iteration structures

List loop

```
for x in lst:
   statement_1
   statement_2
   . . .
statement
```

String loop

```
for x in str:
   statement_1
   statement_2
   . . .
statement
```

Nest decision (if) structures in loop.

Iteration structures

```
for variable in range(n):
    statement_1
    statement_2
    .....
statement
```

Where n is the number of repetitions, begins at 0

```
for variable in range(a,b):
    statement_1
    statement_2
    .....
statement
```

Where a is the start value (inclusive) and b is the stop (exclusive) value

```
for variable in range(a,b,s):
    statement_1
    statement_2
    .....
statement
```

Where a is the start value (inclusive) and b is the stop (exclusive) value, and s is the step value

Nest decision structures in loop.

Iteration Structures Practice

Example 1

- **Problem:** Print all the composes the list [‘Mozart’, ‘Puccini’, ‘Rossini’, ‘Wagner’, ‘Verdi’, ‘Strauss’]
- Describe the algorithm
  1. Define a list of composer names
  2. For every name in the list:
     Print each name on a separate line
- **Key Knowledge:**
  – Use For loop that iterates over the items in the **list**; use for x in y where x is a variable name and y is a **list**.
- **Modify example1 to include only names that end in ‘i’.**

Example 2

- **Problem:** Print all letters in a word
- Describe the algorithm

```
1. Get word from user
2. For each letter in the word:
       Print each letter on a separate line
```

- **Key Knowledge:**
  - Use For loop that iterates over the characters in a **string**; use for x in y where x is a variable name and y is a string.
- **Modify example 2 to print only the consonants in a word.**
- **Consonants are all alpha characters that are not vowels.**

Example 3

- **Problem:** Print all vowels in a word
- Describe the algorithm

```
1. Get word from user
2. For each letter in the word:
       If character is a vowel:
           Print each vowel on a separate line
```

- **Key Knowledge:**
  - Use For loop that iterates over the characters in a **string**; use for x in y where x is a variable name and y is a string.
  - Define a string of vowels; compare each character to vowel string
- Modify example 3 to accommodate a word that has no vowels; print message ‘no vowels in word’

Example 4

- **Problem:** Print all numbers in a list
- Describe the algorithm

```
1. Get list of numbers from user
2. For each number:
       Print the number on a separate line
```

- **Key Knowledge:**
  - Use For loop that iterates over the items in a **List**
    use for x in y where x is a variable name and y is a List.

What if we want to calculate the average of the numbers in the list?

Example 5

- **Problem:** Print the average of a list of numbers
- Describe the algorithm

```
1. Get a list of numbers from user
2. For each number in the list:
       Add number to total
3. Divide total by the number of values in the list
4. Print average
```

- **Key Knowledge:**
  - Use len() function for lists to get the count of numbers
  - Use a variable ‘total’ to accumulate the values
  - Initialize total to 0

Example 6

- **Problem:** Print ‘Hello’ ten times
- Describe the algorithm

```
1. For i from 0 to 10:
       Print the message
```

- **Key Knowledge:**
  - Use range function
  - With range up to 10 knowing that the start value is 0; iterates through the values 0,1,2,3,4,5,6,7,8,9
- Make more general:
  - Ask user for the message
  - Numbers of times to be repeated.

Example 7

- **Problem:** Print the square of even numbers between 0 and 10
- Describe the algorithm

```
1. For each number from 0 to 10
       Divide number by 2
       If remainder is 0:
           Square the number
           Print the number and it’s square
```

- **Key Knowledge:**
  - Even numbers are divisible by 2 with remainder 0
  - Use % division

Alternative method is to retrieve only even numbers

List methods

Lists methods

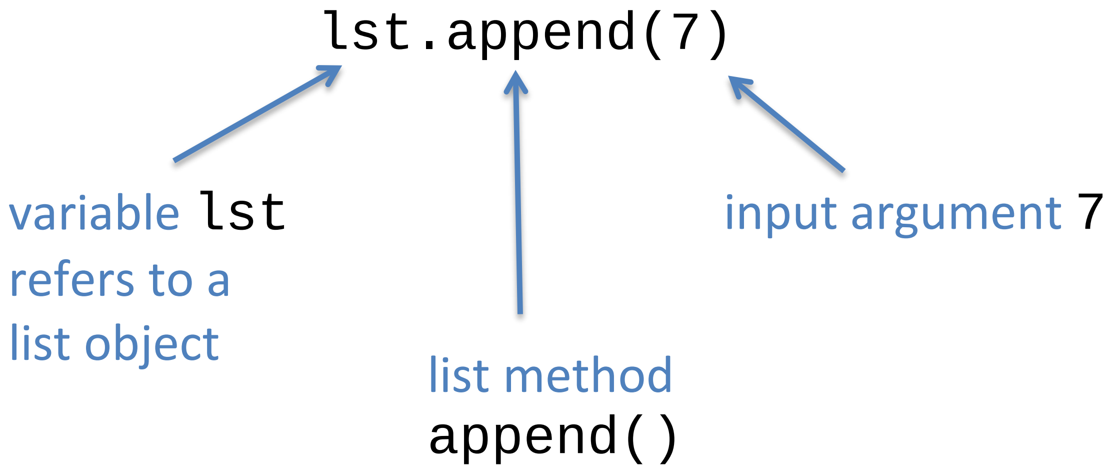
Usage	Explanation
`lst.append(item)`	adds `item` to the end of `lst`
`lst.count(item)`	returns the number of times `item` occurs in `lst`
`lst.index(item)`	Returns index of (first occurrence of) `item` in `lst`
`lst.pop()`	Removes and returns the last item in `lst`
`lst.remove(item)`	Removes (the first occurrence of) `item` from `lst`
`lst.reverse()`	Reverses the order of items in `lst`
`lst.sort()`	Sorts the items of `lst` in increasing order

Methods `append()`, `remove()`, `reverse()`, and `sort()` do not return any value; they, along with method `pop()`, modify list `lst`

```
>>> lst = [1, 2, 3]
>>> lst.append(7)
>>> lst.append(3)
>>> lst
[1, 2, 3, 7, 3]
>>> lst.count(3)
2
>>> lst.remove(2)
>>> lst
[1, 3, 7, 3]
>>> lst.reverse()
>>> lst
[3, 7, 3, 1]
>>> lst.index(3)
0
>>> lst.sort()
>>> lst
[1, 3, 3, 7]
>>> lst.remove(3)
>>> lst
[1, 3, 7]
>>> lst.pop()
7
>>> lst
[1, 3]
```

Lists methods

- List methods are functions that are called on a list.



```
lst.append(7)
```

To create a new list, use lst = [ ], the variable lst refers to a empty list object

To add items to a list use append( ) method.

List methods practice

Example 1

- **Problem:** Write a function that create a list of student names entered by the instructor one at a time
- **Describe the algorithm**

```
1.  Initialize list
2.  Get number of students, assign to num
3.  For each value from 0 to num:
        Get name of student and add to list
4.  Output completed list
```

- **Key Knowledge:**
  - Use For loop that iterates over the length of the list;
  - Use append() to add each name to the list

Example 2

- **Problem:** Write a function that prints all the names in a list that begin with ‘B’ or ‘C’
- **Describe the algorithm**

```
1. Define a list of names
2. Initialize list for names that meet the criteria
3. For each value in names:
       If the name begins with ‘B’ or ‘C’
           Add name to a new list
4. Output new list
```

- **Key Knowledge:**
  – Use For loop that iterates over the length of the list
  – Use index(item) method to get first letter of each name
  – Use append() to add each name to the new list

Example 3

- **Problem:** Write a function that prints the names beginning with the letter specified by the user
- **Describe the algorithm**
  - Define list of names

```
-- UDF – whichLetter(ltr):
     1.   For each name  in list:
           if the name begins with ltr:
              Print  new name on a same line separated by a space
– UDF – main():
     1.   Get letter from user
     2.   Call whichLetter function passing letter
```

- **Key Knowledge:**
  - Use index(item) method to get first letter of each name and compare it to the letter specified by the user
  - Print using argument end = ' '

Formatted output

Built-in function `print()`, revisited

Function `print()` takes 0 or more arguments and prints their string representation in the shell

```
>>> prod = 'morels'
>>> cost = 139
>>> wght = 1/2
>>> total = cost * wght
>>> print(prod, cost, wght, total)
morels 139 0.5 69.5
>>> print(prod, cost, wght, total, sep='; ')
morels; 139; 0.5; 69.5
>>> print(prod, cost, wght, total, sep=':::')
morels:::139:::0.5:::69.5
>>>
```

A blank space separator is printed between the arguments

The `sep` argument allows for customized separators

Built-in function print(), revisited

Function `print()` prints, by default, a newline character after printing its arguments

```
>>> pets = ['boa', 'cat', 'dog']
>>> for pet in pets:
        print(pet)


boa
cat
dog
>>> for pet in pets:
        print(pet, end=', ')


boa, cat, dog,
>>> for pet in pets:
        print(pet, end='!!! ')


boa!!! cat!!! dog!!!
>>>
```

The `end` argument allows for customized end characters

General output formatting

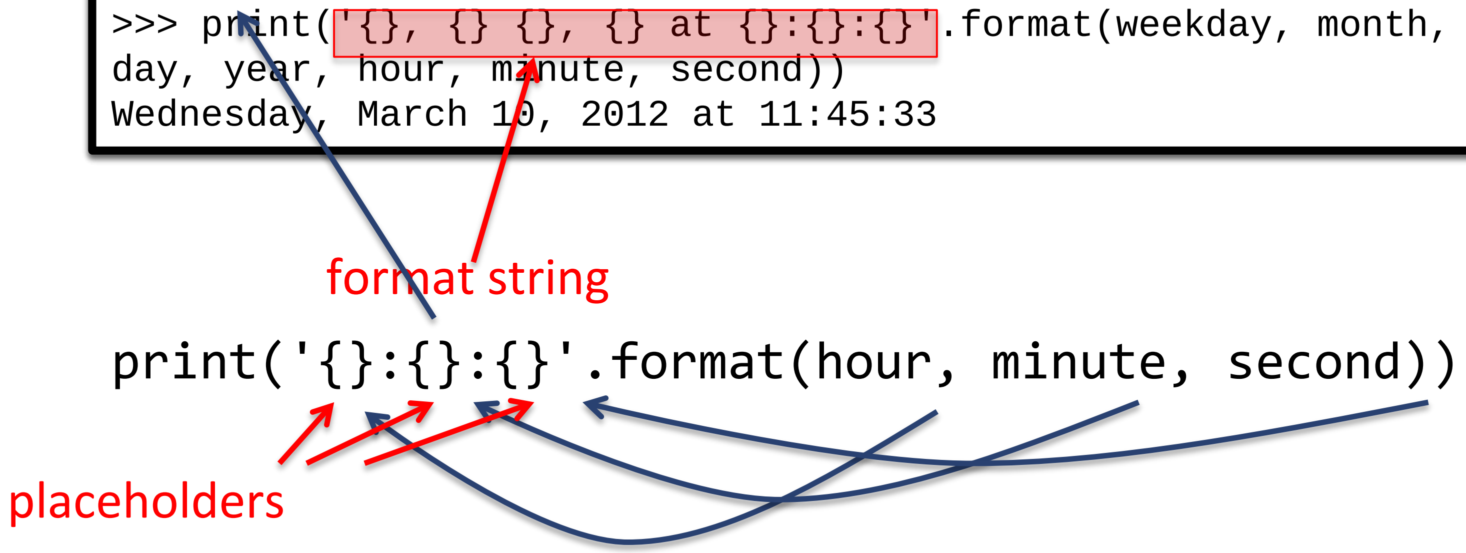
Suppose we have

```
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print(hour+':'+minute+':'+second)
Traceback (most recent call last):
  File "<pyshell#113>", line 1, in <module>
    print(hour+':'+minute+':'+second)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> print(str(hour)+':'+str(minute)+':'+str(second))
11:45:33
>>> print('{}:{}:{}'.format(hour, minute, second))
11:45:33
```

and we want to print `Wednesday, March 10, 2010 at 11:45:33`

Method `format()` of class `str`

```
>>> day = 'Wednesday'
>>> month = 'March'
>>> weekday = 'Wednesday'
>>> month = 'March'
>>> day = 10
>>> year = 2010
>>> year = 2012
>>> hour = 11
>>> minute = 45
>>> second = 33
>>> print('{}:{}:{}'.format(hour, minute, second))
11:45:33
>>> print('{}, {} {}, {} at {}:{}:{}'.format(weekday, month,
day, year, hour, minute, second))
Wednesday, March 10, 2012 at 11:45:33
```



```
print('{}:{}:{}'.format(hour, minute, second))
```

Specifying field width

The `format()` method can be used to line up data in columns

Numbers are aligned to the right

```
>>> for i in range(1,8):
        print(i, i**2, 2**i)


1 1 2
2 4 4
3 9 8
4 16 16
5 25 32
6 36 64
7 49 128
>>> for i in range(1, 8):
        print('{} {:2} {:3}'.format(i, i**2, 2**i))


1  1   2
2  4   4
3  9   8
4 16  16
5 25  32
6 36  64
7 49 128
>>>
```

plus a blank space between the columns

reserves 2 spaces for `i**2`

reserves 3 spaces for `2**i`

Specifying field width

The `format()` method can be used to line up data in columns

Numbers are aligned to the right

Strings are aligned to the left

```
>>> lst = ['Alan Turing', 'Ken Thompson', 'Vint Cerf']
>>> for name in lst:
        fl = name.split()
        print(fl[0], fl[1])


Alan Turing
Ken Thompson
Vint Cerf
>>> for name in lst:
        fl = name.split()
        print('{:5} {:10}'.format(fl[0], fl[1]))


Alan  Turing
Ken   Thompson
Vint  Cerf
>>>
```

Output format type

Inside the curly braces of a placeholder, we can specify the field width, the type of the output, and the decimal precision

Type	Explanation
b	binary
c	character
d	decimal
X	hexadecimal
e	scientific
f	fixed-point

```
>>> n = 10
>>> '{:b}'.format(n)
'1010'
>>> '{:c}'.format(n)
'\n'
>>> '{:d}'.format(n)
'10'
>>> '{:X}'.format(n)
'A'
>>> '{:e}'.format(n)
'1.000000e+01'
>>> '{:7.2f}'.format(n)
'  10.00'
>>>
```

`'{:7.2f}'`

field width

decimal precision

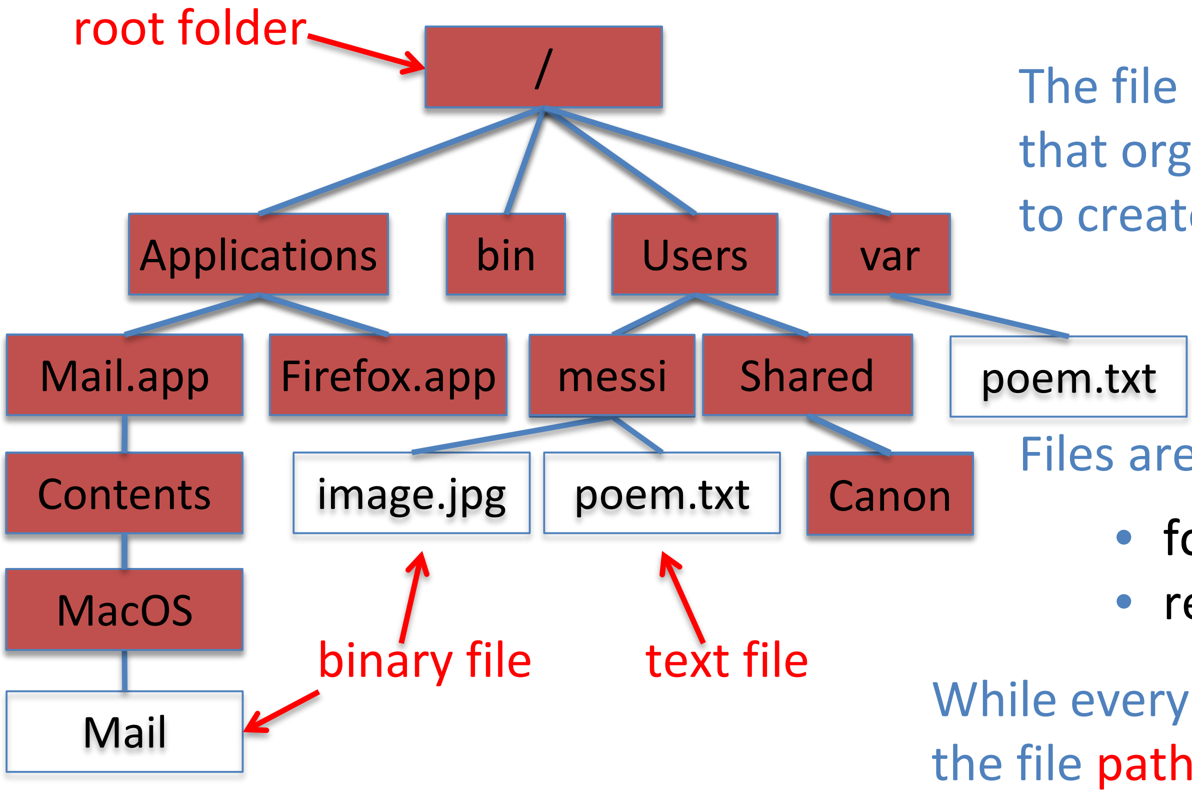
Files

Pathname

File pathname

- Absolute pathname – sequence of folders starting from the root
- Relative pathname – sequence of directories that must be traversed, starting with the current working directory.

Files and the file system - absolute



The file system is the OS component that organizes files and provides a way to create, access, and modify files

Files are organized into a tree structure

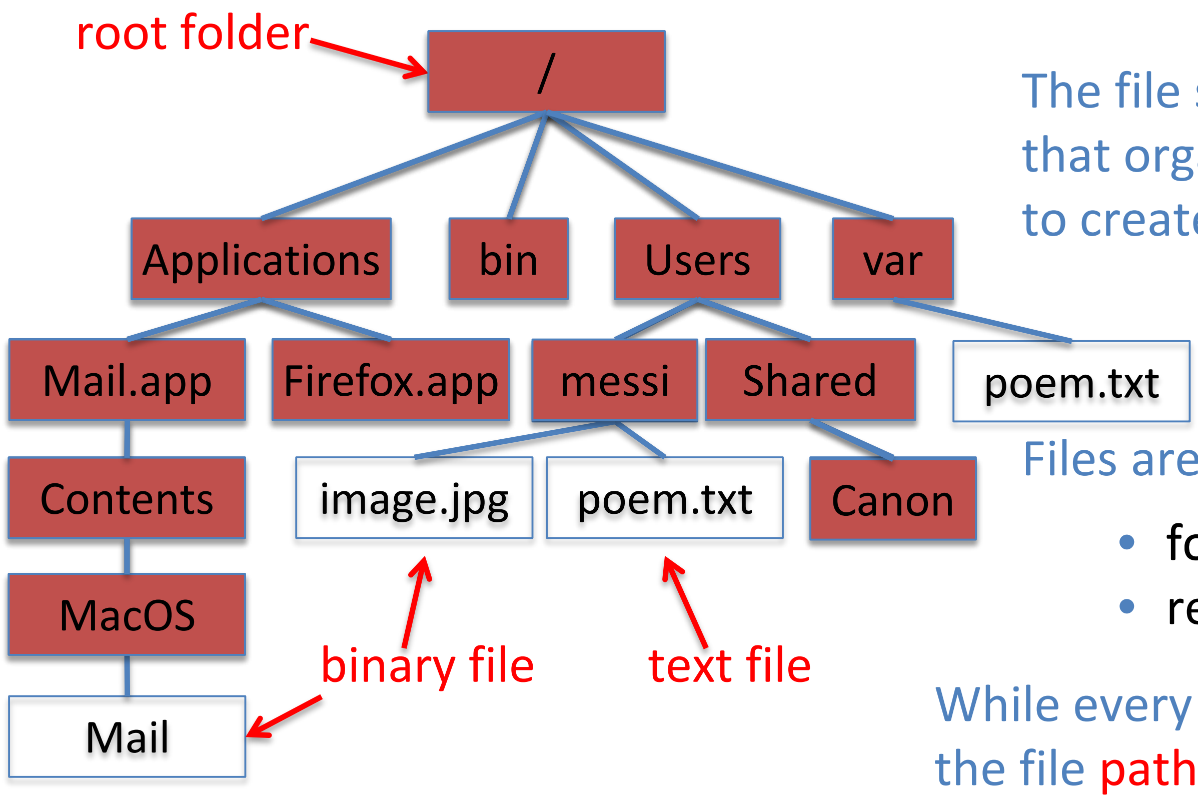
- folders (or directories)
- regular files

While every file and folder has a name, it is the file pathname that identifies the file

Absolute pathnames

- `/var/poem.txt`
- `/Users/messi/poem.txt`
- `/Applications/Mail.app/`

Files and the file system - current



The file system is the OS component that organizes files and provides a way to create, access, and modify files

Files are organized into a tree structure

- folders (or directories)
- regular files

While every file and folder has a name, it is the file pathname that identifies the file

Relative pathnames (relative to current working directory `Users`)

- `messi/poem.txt`
- `messi/image.jpg`
- `Shared`

Reading Files

Reading a file

1. Open the file
2. Read from the file
3. Close the file.

Use built-in function to **open( )**

```
infile = open (filename, 'r')
```

where r is reading mode.

filename is the name of the file enclosed in quotes

Open

infile = open ('example.txt', 'r')

infile is the handle for the file

File Methods	
infile.read(n)	Read n characters from the infile or until the end of the file is reached, and return characters as a **string**
infile.read()	Read characters from the file infile until the end of the file and return characters as **string**
infile.readline()	Read file infile until (and including) the new line character or until end of file, whichever is first, and return characters read as **string**
infile.readlines()	Read file infile until the end of the file and return the characters read as a **list** of line
infile.close()	Close the file infile

String methods

Strings are immutable; none of the string methods modify string `s`

Usage	Explanation
`s.capitalize()`	returns a copy of `s` with first character capitalized
`s.count(target)`	returns the number of occurences of `target` in `s`
`s.find(target)`	returns the index of the first occurrence of `target` in `s`
`s.lower()`	returns lowercase copy of `s`
`s.replace(old, new)`	returns copy of `s` with every occurrence of `old` replaced with `new`
`s.split(sep)`	returns **list** of substrings of `s`, delimited by `sep`
`s.strip()`	returns copy of `s` without leading and trailing whitespace
`s.upper()`	returns lowercase copy of `s`

Example 1

- **Problem:** Write a function that reads a small text file and computes number of characters, words and sentences
- **Describe the algorithm**
  1. Open, read and close text file
     Assign file contents to a string
     Determine number of characters in the string
  2. Convert string to a list of words
     Determine number of words and number of lines
  3. Print the stats
- **Key Knowledge:**
  - Use read() method to get a string of all characters;
  - Use string split() method to get list of words
  - A line ends with a period ('\n')
  - Save file to be read in the same folder as program