LGT3109
Introduction to Coding for Business with Python
(week 6)

Xiaoyu Wang

Dept. of LMS, The HK PolyU

Summary of Week 5

- Loop: Repeat
  - Iteration, iteration variable
- for loop: definite
  - Iteration for each item of a sequence
- while loop: Indefinite
  - Iteration until condition is violated
- break and continue

- Loop Patterns:
  - Initialize, Iteration over each entry of data, and repeat processing and updates
    - Maximum of data
    - Sum of data
    - Counting of data
  - Filtering of data by conditions, and use of Flag variables
  - Use of None, is, and is not
    - Minimum

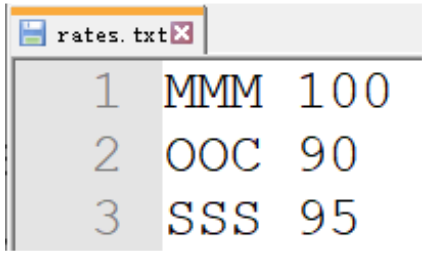
Strings and Files

- Strings
  - Basics
  - Inside structure:
    numbering, length, loop
  - String operations
  - String functions

- Files
  - Opening files
  - Processing files
  - Writing and closing files

Motivation Case

- Foxconn ships cargoes from Hong Kong to Los Angeles.
- Shipping rates stored the in a text file (rates.txt).
- Carrier's ID has 3 letters: AAA, ABC
- Followed by a space and a shipping rate.
- Foxconn wants to know the shipping rate for a specific carrier.




```
Enter carrier name: OOC
Shipping rate of carrier OOC is 90
>>>
```

```
Enter carrier name: AAA
Shipping rate of carrier AAA is not found
```

Motivation Case-Code

- For each line read from the input file:
- If the line starts with the carrier_name:
- Extract the shipping rate from the line.

```
carrier_name = input('Enter carrier name: ')
file = open('rates.txt', 'r')
flag_found = False
for line in file:
    if line.startswith(carrier_name):
        rate = int(line[4:])
        print('Shipping rate of carrier', carrier_name,'is', rate)
        flag_found = True
        break

if not flag_found:
    print('Shipping rate of carrier', carrier_name,'is not found')
file.close()
```

Strings and Files

- Strings
  - Basics
  - Inside structure:
    numbering, length, loop
  - String operations
  - String functions
- Files
  - Opening files
  - Processing files
  - Writing and closing files

String Review

- A string is a sequence of characters.
- A string containing numbers is still a string.
- A string uses quotes ‘Hello’ or “Hello”.

- For strings, “+” combines, and “*” duplicates strings.

➢How about number?

- Convert numbers in a string into a number using int() or float().

Escape Characters

- Some special requirement to write a string.

  - `Error: txt = "We are the "Vikings" from the north."`
  - `Correct: txt = "We are the \"Vikings\" from the north."`

- Common escape characters used in Python:

Code	Result	Code	Result
\”	Double Quote	\’	Single Quote
\\	Backslash	\n	New Line
\t	Tab	\b	Backspace
\r	Carriage Return		

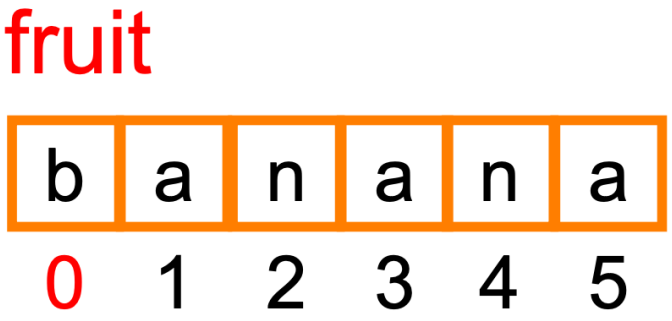
Strings and Files

- Strings
  - Basics
  - Inside structure:
    numbering, length, loop
  - String operations
  - String functions

- Files
  - Opening files
  - Processing files
  - Writing and closing files

Looking inside a String

- We can get at any single character in a string using an index specified in square brackets, e.g., fruit[1].
- The index value must be an integer and starts at zero.
- The index value can be an expression that is computed.

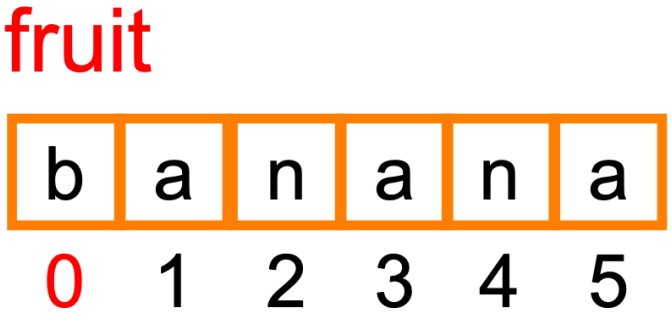



```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> print(letter)
a
>>> x = 3
>>> w = fruit[x - 1]
>>> print(w)
n
```

Error: Index Out of Range

- You will get a python error if you attempt to index beyond the end of a string.

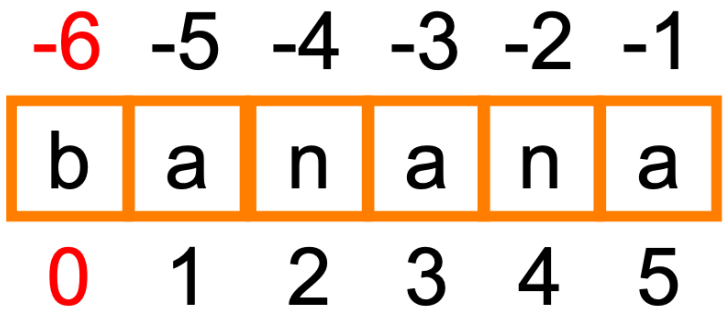
- So be careful when constructing index values and slices.




```
>>> fruit = 'banana'
>>> print(fruit[6])
Traceback (most recent call
last):  File "<stdin>", line
1, in <module>
IndexError: string index out
of range
>>>
```

Negative Index: Backward

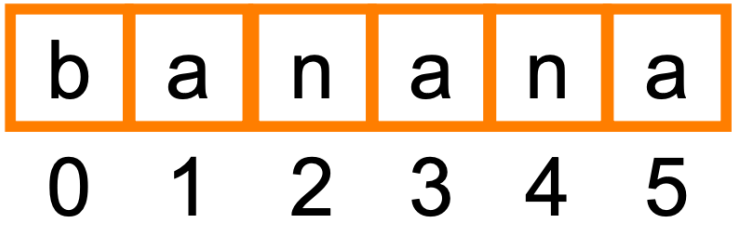
- The index value can be negative.
- Negative indices count backward from the end of the string.




```
>>> fruit = 'banana'
>>> letter = fruit[-1]
>>> print(letter)
a
>>> x = 3
>>> w = fruit[-x]
>>> print(w)
a
```

String Length

- The built-in function **len** gives the length of a string.

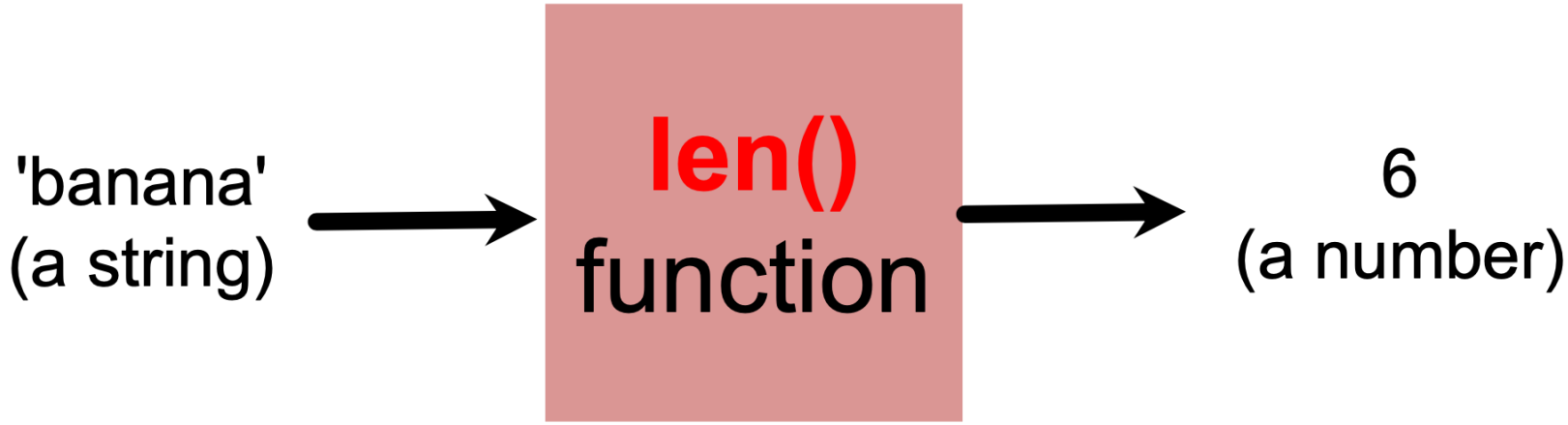



```
>>> fruit = 'banana'
>>> print(len(fruit))
6
```

String Length

```
>>> fruit = 'banana'
>>> x = len(fruit)
>>> print(x)
6
```

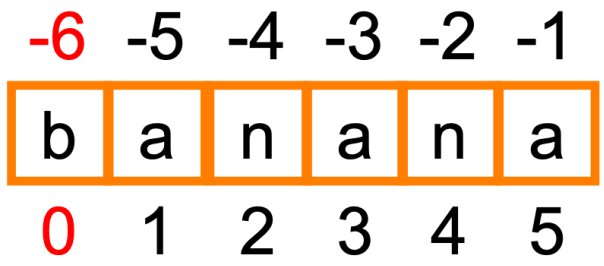
A **function** is some stored code that we use. A function takes some input and produces an output.



How to Get the Last Character of a String

```
>>> fruit = ‘banana’
>>> length = len(fruit)
>>> last = fruit[length]
IndexError: string index out of range
```

- no letter in 'banana' with the index 6




```
>>> fruit = ‘banana’
>>> length = len(fruit)
>>> last = fruit[length-1]
>>> last = fruit[-1]
```

- To get the last character, you have to subtract 1 from *length*
- *Negative Index:* count backward from the end

Looping Through a String

The loop condition is index < len(fruit)
Note that len(fruit) = 6

Using a **while** statement, an **iteration variable**, and the **len** function, we can construct a *indefinite loop* to look at each of the characters in a string individually

```
fruit = 'banana'
index = 0
while index < len(fruit):
    letter = fruit[index]
    print(index, letter)
    index = index + 1
```

```
0 b
1 a
2 n
3 a
4 n
5 a
```

Looping Through a String

A *definite loop* using a **for statement** is much more elegant

The **iteration variable** is completely taken care of by the for loop

```
fruit = 'banana'
for letter in fruit:
    print(letter)
```

```
b
a
n
a
n
a
```

Looping Through a String

A definite loop using a **for statement** is much more elegant

The **iteration variable** is completely taken care of by the for loop

```
fruit = 'banana'
for letter in fruit:
    print(letter)
```

```
fruit = 'banana'
index = 0
while index < len(fruit):
    letter = fruit[index]
    print(index, letter)
    index = index + 1
```

b
a
n
a
n
a

Looping and Counting

E.g., to count how many times ‘a’ appear in ‘banana’

This is a simple loop that loops through each character in a string and counts the number of times the loop encounters the 'a' character

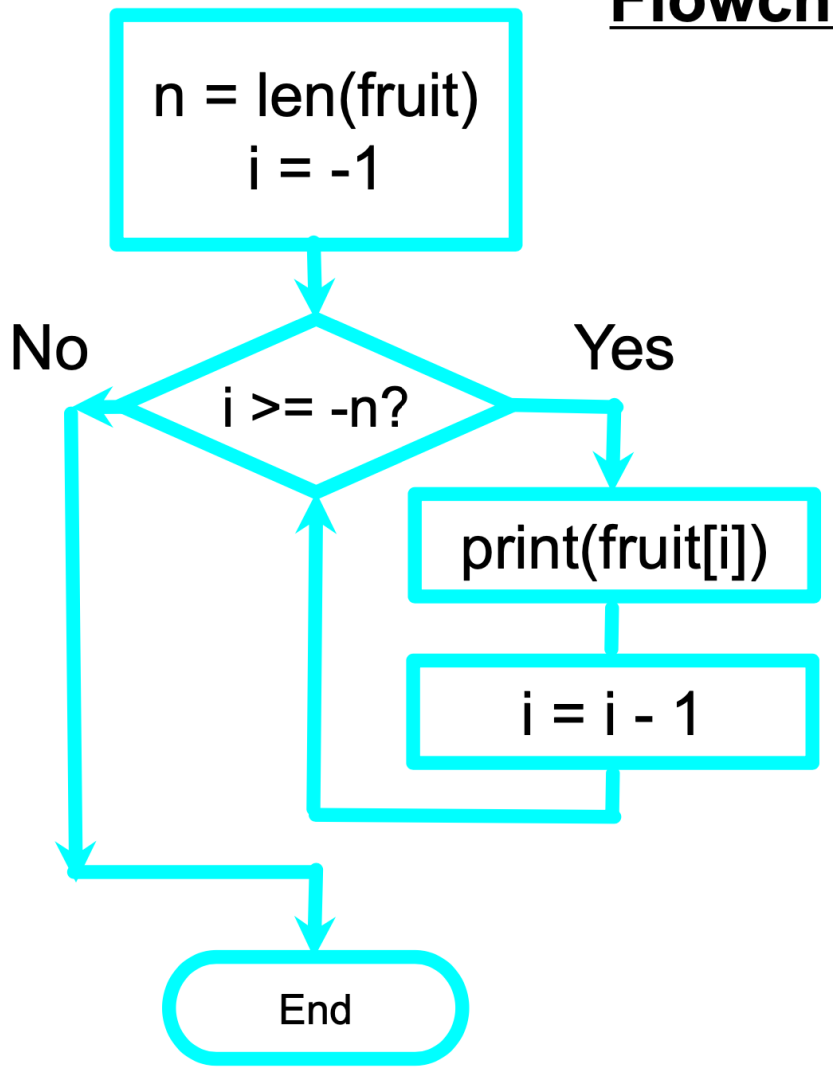
```
word = 'banana'
count = 0
for letter in word :
    if letter == 'a' :
        count = count + 1
print(count)
```

Reverse Loop

- Write a loop that
  - starts at the last character in the string and
  - works its way backwards to the first character in the string

-6	-5	-4	-3	-2	-1
b	a	n	a	n	a
0	1	2	3	4	5

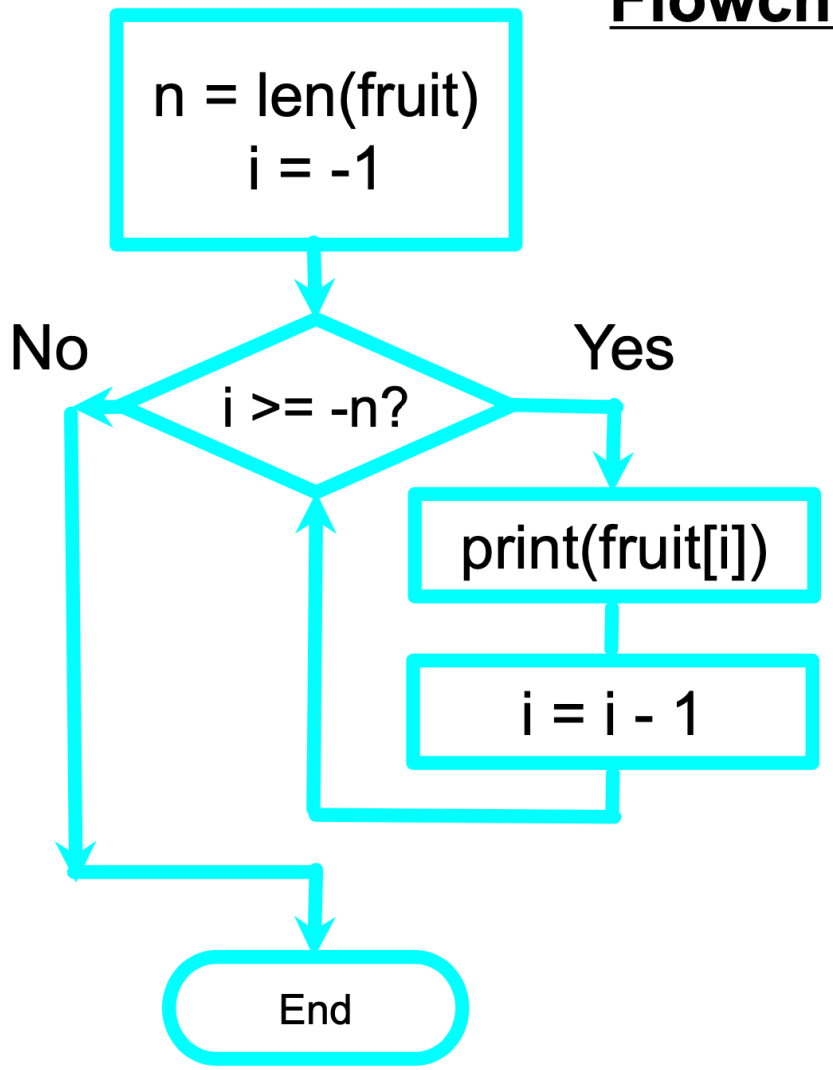
Flowchart



Reverse Loop

```
fruit='banana'
n = len(fruit)
i = -1
while i>=-n:
    print(fruit[i])
    i=i-1
```

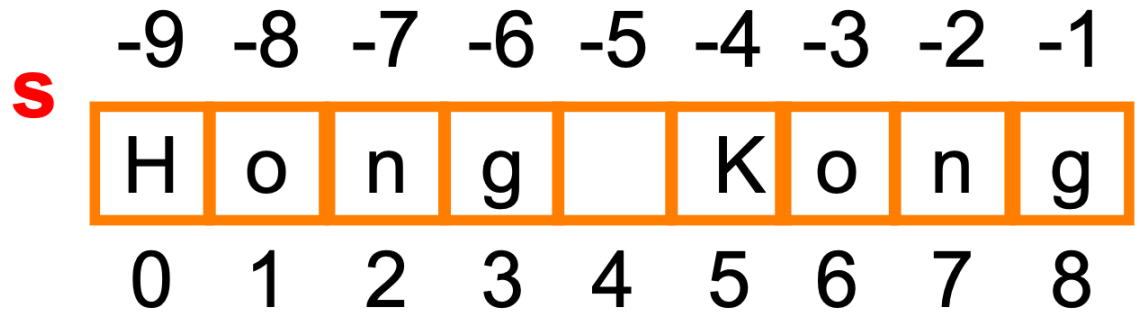
Flowchart



Strings and Files

- Strings
  - Basics
  - Inside structure:
    numbering, length, loop
  - String operations
  - String functions
- Files
  - Opening files
  - Processing files
  - Writing and closing files

Slicing Strings



s[i:j] equals s[i]s[i+1]…s[j-1]

The second number is one beyond the end of the slice - "**up to but not including**"

If the second number is beyond the end of the string, it stops at the end

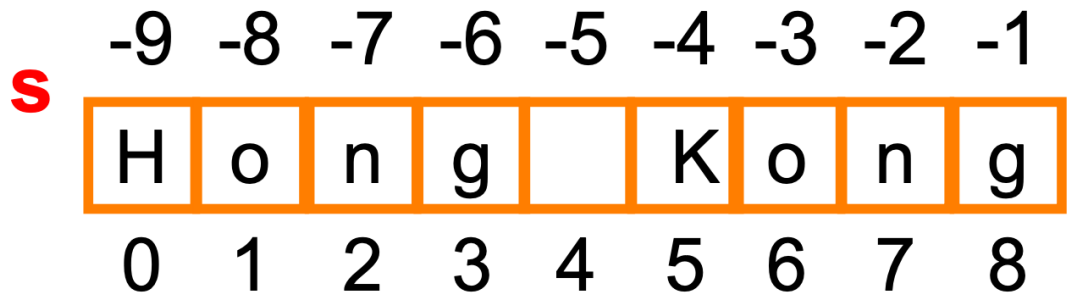
```
>>> s = 'Hong Kong'
>>> print(s[0:4])
Hong
>>> print(s[6:7])
o
>>> print(s[6:20])
ong
>>> print(s[-9:-6])
Hon
```

Slicing Strings

s[:j] or s[i:]

- **s[:j]**: If we leave off the first number, it is assumed to be the beginning of the string,
  - i.e., equivalent to s[0:j]

- **s[i:]**: if we leave off the last number of the slice, it is assumed to be the end of the string,
  - i.e., equivalent to s[i:len(s)]




```
>>> s='Hong Kong'
>>> print(s[:2])
Ho
>>> print(s[0:2])
Ho
>>> print(s[5:])
Kong
>>> print(s[5:len(s)])
Kong
>>> print(s[:])
Hong Kong
>>> print(s[0:len(s)])
Hong Kong
>>> print(s[-8:-3])
ong K
```

String Concatenation

When the + operator is applied to strings, it means "concatenation"

```
>>> a = 'Hello'
>>> b = a + 'There'
>>> print(b)
HelloThere
>>> c = a + ' ' + 'There'
>>> print(c)
Hello There
>>>
```

String Multiplication

When the * operator is applied to strings, it means "repeat" a string for multiple times

```
>>> a = 'Hello'
>>> b = a * 3
>>> print(b)
HelloHlloHello
```

Logical Operators

The ***in* keyword** can also be used to check to see **if one string is "*in*" another string**

The **in expression** is a logical expression that returns True or False and can be used in an if statement

```
>>> fruit = 'banana'
>>> 'n' in fruit
True
>>> 'm' in fruit
False
>>> 'nan' in fruit
True
>>> if 'a' in fruit :
...      print('Found it!')
...
Found it!
>>>
```

String Comparison

Two strings are equal:

```
if word == 'banana':
    print('All right, bananas.')
```

Other comparison operations are based on in **alphabetical order**:

```
if word < 'banana':
    print('Your word,' + word + ', comes before banana.')
elif word > 'banana':
    print('Your word,' + word + ', comes after banana.')
else:
    print('All right, bananas.')
```

All the uppercase letters come before all the lowercase letters:
Your word, Pineapple, comes before banana.

Strings are Immutable

- String variables are immutable
- Values of string variables cannot be changed

```
>>> greeting = 'Hello, Hong Kong!'
>>> greeting[0] = ‘h'
TypeError: 'str' object does not support item assignment
```

How to change the first character of greeting to ‘h’?

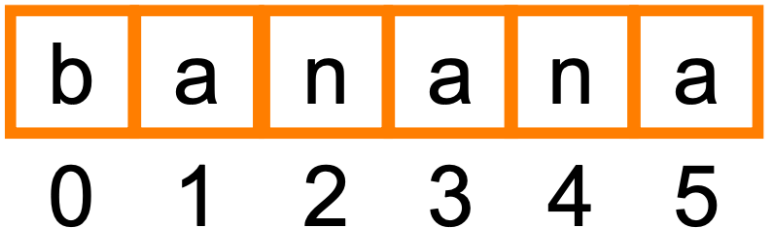
- The best you can do is create a new string that is a variation on the original

```
>>> greeting = 'Hello, Hong Kong!'
>>> new_greeting = ‘h’ + greeting[1:]
>>> print(new_greeting)
hello, Hong Kong!
```

Strings and Files

- Strings
  - Basics
  - Inside structure:
    numbering, length, loop
  - String operations
  - String functions
- Files
  - Opening files
  - Processing files
  - Writing and closing files

Useful String Functions-find



- We use the find() function to search for a substring within another string.

➢find() finds the (non-negative) index of the first occurrence of the substring.

➢If the substring is not found, find() returns -1.

- Remember that string position starts at zero.

```
>>> fruit = 'banana'
>>> pos = fruit.find('na')
>>> print(pos)
2
>>> aa = fruit.find('z')
>>> print(aa)
-1
```

Useful String Functions-upper/lower

- You can make a copy of a string in lower case or upper case
- Case-Insensitive Search: search a string regardless of case
  - Convert the string to lower case
  - Use find() to search

```
>>> greet = 'Hello Bob'
>>> nnn = greet.upper()
>>> print(nnn)
HELLO BOB
>>> www = greet.lower()
>>> print(www)
hello bob
>>> res = greet.lower().find('bob')
>>> print(res)
6
```

Useful String Functions-replace

The replace() function is like a “search and replace” operation in a word processor

It replaces all occurrences of the search string (ss) with the replacement string (rs)

```
>>> greet = 'Hello Bob'
>>> nstr = greet.replace('Bob','Jane')
>>> print(nstr)
Hello Jane
>>> nstr = greet.replace('o','X')
>>> print(nstr)
HellX BXb
>>>
```

Useful String Functions-strip

Sometimes we want to take a string and remove whitespace at the beginning and/or end

- lstrip() and rstrip() remove whitespace at the left or right
- strip() removes both beginning and ending whitespace

```
>>> greet = '   Hello Bob   '
>>> greet.lstrip()
'Hello Bob   '
>>> greet.rstrip()
'   Hello Bob'
>>> greet.strip()
'Hello Bob'
>>>
```

Useful String Functions-startswith/endswith

- We want to know whether a given string starts with a certain prefix or ends with a certain suffix
  - startswith(pf) returns True if the string starts with the value of pf
  - endswith(sf) returns True if the string ends with the value of sf

```
>>> line = 'Hello Bob'
>>> line.startswith('Hello')
True
>>> line.startswith('hello')
False
>>> line.endswith('Bob')
True
```

Useful String Functions-f/F

- Include Python expressions inside a string by prefixing the string with f or F and writing expressions as {expression}.

```
rate = 100.0
quantity = 150
print(f'{rate}*{quantity}={rate*quantity}.')
```

```
100.0*150=15000.0.
```

Strings and Files

- Strings
  - ➢ Basics
  - ➢ Inside structure:
    numbering, length, loop
  - ➢ String operations
  - ➢ String functions
- Files
  - ➢ Opening files
  - ➢ Processing files
  - ➢ Writing and closing files

File Structure

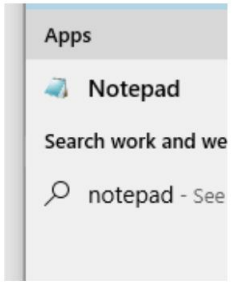
A text file can be thought of as a sequence of lines

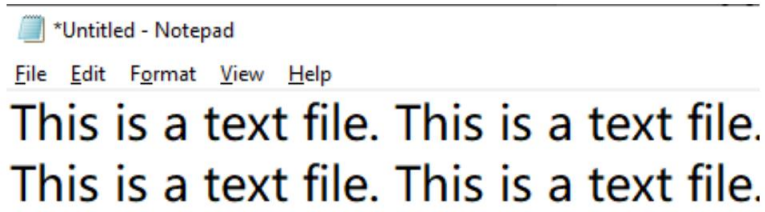
```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Date: Sat, 5 Jan 2008 09:12:18 -0500
To: source@collab.sakaiproject.org
From: stephen.marquard@uct.ac.za
Subject: [sakai] svn commit: r39772 - content/branches/

Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772
```

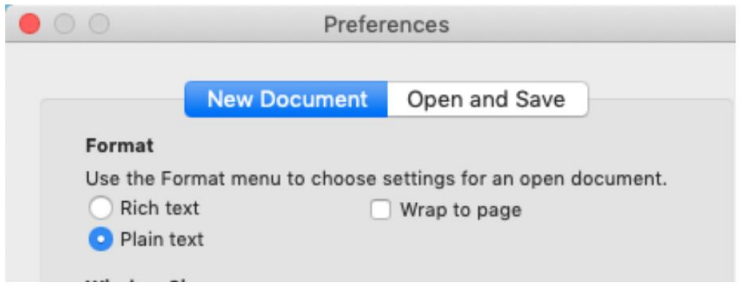
Create a Text File

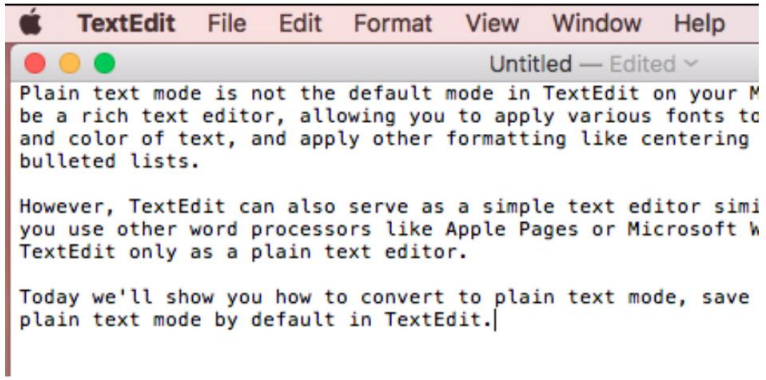
- Use Text Editor
  - Windows: Notepad





  - MacOS: TextEdit
    - Preferences → Plain Text





Opening a File

Before we can read the contents of the file, we must tell Python which file we are going to work with and what we will be doing with the file

This is done with the open() function

- open() returns a “file handle” - a variable used to perform operations on the file

Similar to “File -> Open” in a Word Processor

```
carrier_name = input('Enter carrier name: ')
infile = open('rates.txt', 'r')
outfile = open('rates-new.txt', 'w')
flag_found = False
for line in infile:
    if line.startswith(carrier_name):
        rate = float(line[4:])
        new_rate = rate * 1.1
        flag_found = True
        outfile.write('%s %g\n'%(carrier_name
        print('Shipping rate of carrier', ca
    else:
        outfile.write(line)

if not flag_found:
    print('Shipping rate of carrier', carrie
infile.close()
```

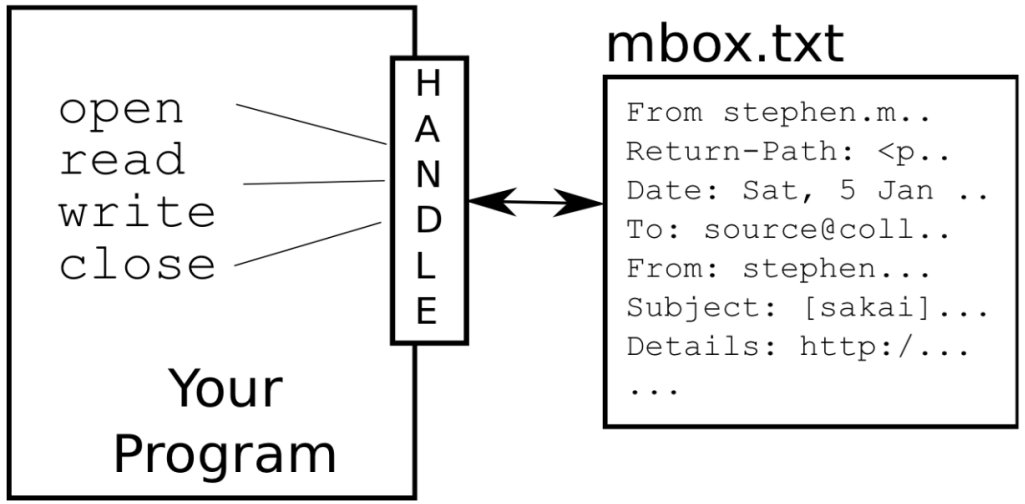
Opening a File

- handle = open(filename, mode)
- open() returns a handle used to manipulate the file
- filename is a string
- mode is optional and should be
  - 'r' if we are planning to read the file and
  - 'w' if we are going to write to the file
- encoding is optional to define how texts are encoded

Handle

- The file handle is not the actual data contained in the file
- It is a "handle" (interface) that we can use to read data.

- You are given a handle if the requested file exists and you have the proper permissions to read the file.



Handle

If the file does not exist, open will fail with a traceback and you will not get a handle to access the contents of the file:

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

```
try:
    fhand = open('stuff.txt')
    print(fhand)
except:
    print('File not found')
```

```
File not found
```

We can use try and except to deal more gracefully with the situation where we attempt to open a file that does not exist

Strings and Files

- Strings
  - Basics
  - Inside structure:
    numbering, length, loop
  - String operations
  - String functions

- Files
  - Opening files
  - Processing files
  - Writing and closing files

Newline Character

We use a special (escape) character called the “newline” to indicate when a line ends

We represent it as \n in strings

- Newline is still one character - not two

```
>>> stuff = 'Hello\nWorld!'
>>> stuff
'Hello\nWorld!'
>>> print(stuff)
Hello
World!
>>> stuff = 'X\nY'
>>> print(stuff)
X
Y
>>> len(stuff)
3
```

File Processing

A text file can be thought of as a sequence of lines

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
Return-Path: <postmaster@collab.sakaiproject.org>
Date: Sat, 5 Jan 2008 09:12:18 -0500
To: source@collab.sakaiproject.org
From: stephen.marquard@uct.ac.za
Subject: [sakai] svn commit: r39772 - content/branches/

Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772
```

File Processing

A text file has newlines at the end of each line

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008\n
Return-Path: <postmaster@collab.sakaiproject.org>\n
Date: Sat, 5 Jan 2008 09:12:18 -0500\n
To: source@collab.sakaiproject.org\n
From: stephen.marquard@uct.ac.za\n
Subject: [sakai] svn commit: r39772 - content/branches/\n
\n
Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772\n
```

Counting Lines in a File

Open a file read-only

Use a for loop to read each line

- Count the lines and print out the number of lines

```
fhand = open('mbox.txt', 'r')
count = 0
for line in fhand:
    count = count + 1
print('Line Count:', count)
```

```
Line Count: 1910
```

Reading the Whole File

We can read the whole file (newlines and all) into a single string

```
>>> fhand = open('mbox.txt')
>>> inp = fhand.read()
>>> print(len(inp))
94626
>>> print(inp[:20])
From stephen.marquar
```

```
>>> infile=open('rates.txt', 'r')
>>> s = infile.read()
>>> print(s)
MMM 100
OOC 90
SSS 95
```

Searching Through a File

Parsing-Extracting:

We can put an **if statement** in our for loop to only print lines that meet some criteria

```
fhand = open('mbox.txt')
for line in fhand:
    if line.startswith('From:') :
        print(line)
```

Searching Through a File

What are all these blank lines doing here?

```
From: stephen.marquard@uct.ac.za

From: louis@media.berkeley.edu

From: zqian@umich.edu

From: rjlowe@iupui.edu
...
```

Searching Through a File

What are all these blank lines doing here?

- Each line from the file has a newline at the end
- The print statement adds a newline to each line

```
From: stephen.marquard@uct.ac.za\n
\n
From: louis@media.berkeley.edu\n
\n
From: zqian@umich.edu\n
\n
From: rjlowe@iupui.edu\n
\n
...
```

Searching Through a File

- We can strip the whitespace (including \n) from the right-hand side of the string using rstrip() from the string library
- The newline \n is considered "white space" and is stripped

```
fhand = open('mbox.txt')
for line in fhand:
    line = line.rstrip()
    if line.startswith('From:') :
        print(line)
```

From: stephen.marquard@uct.ac.za
From: louis@media.berkeley.edu
From: zqian@umich.edu
From: rjlowe@iupui.edu
....

Skipping with Continue

We can conveniently skip a line by using the continue statement

```
fhand = open('mbox.txt')
for line in fhand:
    line = line.rstrip()
    if not line.startswith('From:') :
        continue
    print(line)
```

Skipping with Continue

We can look for a string anywhere in a line as our selection criteria

```
fhand = open('mbox.txt')
for line in fhand:
    line = line.rstrip()
    if not '@uct.ac.za' in line :
        continue
    print(line)
```

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
X-Authentication-Warning: set sender to stephen.marquard@uct.ac.za using -f
From: stephen.marquard@uct.ac.za
Author: stephen.marquard@uct.ac.za
From david.horwitz@uct.ac.za Fri Jan  4 07:02:32 2008
X-Authentication-Warning: set sender to david.horwitz@uct.ac.za using -f...
```

Skipping with Continue

We can use **find** to determine whether the search string is anywhere in the line, using it as our selection criteria

```
fhand = open('mbox.txt')
for line in fhand:
    line = line.rstrip()
    if line.find('@uct.ac.za') == -1:
        continue
    print(line)
```

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
X-Authentication-Warning: set sender to stephen.marquard@uct.ac.za using -f
From: stephen.marquard@uct.ac.za
Author: stephen.marquard@uct.ac.za
From david.horwitz@uct.ac.za Fri Jan  4 07:02:32 2008
X-Authentication-Warning: set sender to david.horwitz@uct.ac.za using -f...
```

Strings and Files

- Strings
  - Basics
  - Inside structure:
    numbering, length, loop
  - String operations
  - String functions
- Files
  - Opening files
  - Processing files
  - Writing and closing files

Writing and Closing File

- To write a file, **open** it with mode **"w"** as a second parameter:
- The **write** method puts data into the file, returning the number of characters written.
  - When *write called* again, it adds new data to the end, **without adding "\n" at the end**.
  - The print statement automatically appends a newline, but the write method does not add the newline automatically.
- When you are done writing and reading a file, you are suggested to **close** the file

```
>>> fout = open('output.txt', 'w')

>>> print(fout)
<_io.TextIOWrapper name='output.txt'
mode='w' encoding='cp1252'>

>>> line1 = "Hello, Hong Kong!\n"
>>> fout.write(line1)
18

>>> fout.close()
```

Motivation Case Revisit

- For each line read from the input file:
- If the line starts with the carrier_name:
- Extract the shipping rate from the line.

```
carrier_name = input('Enter carrier name: ')
file = open('rates.txt', 'r')
flag_found = False
for line in file:
    if line.startswith(carrier_name):
        rate = int(line[4:])
        print('Shipping rate of carrier', carrier_name,'is', rate)
        flag_found = True
        break

if not flag_found:
    print('Shipping rate of carrier', carrier_name,'is not found')
file.close()
```

Acknowledgement

- Acknowledgements / Contributions

- Initial Development: Charles Severance, University of Michigan School of Information
- Further Development: Zhou Xu, Hong Kong Polytechnic University
- Continuous development: Xiaoyu Wang, Hong Kong Polytechnic University