COMP1021
Introduction to Computer Science

File Handling

David Rossiter and Gibson Lam

Outcomes

- After completing this presentation, you are expected to be able to:
  1. Use the tab character and newline character to output text using the print command
  2. Write code to read content from a text file
  3. Write code to write content to a text file

The Tab Character

- In computer programming, we use `\t` in a string to represent a tab character
  - Remember in programming, a *string* simply mean 'text'
- A tab character moves the text after the tab character to a particular position horizontally
- When you look at it in a text viewing program, it will show things being nicely lined up in columns, to make a nice visual display
- Let's look at some examples of using tabs for nice formatting in columns

Using Tabs for Lining up Columns

```
print("Pythagoras' constant is\t1.41421")
print("Theodorus' constant is\t1.73205")
print("Golden ratio is\t\t1.61803")
print("pi is\t\t\t3.14159")
print("e is\t\t\t2.71828")
```

The tab characters move the horizontal position to these locations

```
Pythagoras' constant is 1.41421
Theodorus' constant is  1.73205
Golden ratio is         1.61803
pi is                   3.14159
e is                    2.71828
>>>
```

Another Example of Using Tabs

- Here's another example of using tab characters

```
for x in range(5):
    print( "\t" * x + "hello")
```

`*` has a higher *precedence* (to be discussed later) than `+` so it is handled first

```
hello
        hello
                hello
                        hello
                                hello
```

The first value generated by `range(5)` *is zero, so there's no tab here*

Using Tabs in a File Format

- When handling files, a tab character is often used to separate things inside the file
- For example, we can put some L-system rules inside a text file
- In this presentation our code will load L-system rules which uses this file format
- Also, we will show code which saves the rules in this file format

Here is the tab character

```
X	X+YF+
Y	-FX-Y
```

The Newline Character

- The other thing we have to understand is the newline character
  (sometimes called the ‘end of line’ character)
- In computer programming, we use `\n` in a string to represent the newline character
- The newline character basically means ‘go to the next line’
- By default, `print()` adds a new line character to whatever you ask it to display

- A newline character is automatically added by `print()` at the end

```
print("Hello!\nI am Python!\nHow are you?")
```

An Example of Using the Newline Character

```
Hello!
I am Python!
How are you?
>>>
```

- Here we turn off the default behaviour of print, to make the example easier to understand

```
for x in range(5):
    print( "hello" + "\n" * x, end="")
```

```
hellohello
hello

hello


hello


>>>
```

The first value generated by `range(5)` *is zero, so there's no end-of-line character here*

** has a higher precedence than + this part is done first*

Another Example

Reading the L-System Rules

- Open the file in ‘read’ mode
- For every line in the file:
  - Read the line as a single string
  - Remove the end-of-line character from the end of the string using `rstrip()`, which means ‘*strip (=remove) whitespace from the right side*’
  - Convert the line into a list of text using `split("\t")`, which separates the line into separate strings wherever a tab character is found
  - Convert the items in the list
  - Add the newly constructed list to the list of lists
- Close the file

What is Whitespace?

- ‘Whitespace’ means ‘anything you can’t see’
- So that means spaces, tabs and also end-of line characters
- We use `rstrip()` to remove whitespace
- `rstrip()` means ‘strip (=remove) anything you can’t see on the **r**ight side’

```
>>> text="hello      "
>>> text
'hello      '
>>> text.rstrip()
'hello'
>>> text="hello\n\n\n"
>>> text
'hello\n\n\n'
>>> text.rstrip()
'hello'
>>> text="hello\t\t\t\t"
>>> text
'hello\t\t\t\t'
>>> text.rstrip()
'hello'
>>> text="hello   \t\n \n\t\t\n"
>>> text
'hello   \t\n \n\t\t\n'
>>> text.rstrip()
'hello'
>>>
```

Reading the Rules from the File

- This sequence shows the operations applied to each line read from the file, and the result after each operation

```
>>> line = "X\tX+YF+\n"
>>> print(line)
X       X+YF+

>>> line = line.rstrip()
>>> print(line)
X       X+YF+
>>> line = line.split("\t")
>>> print(line)
['X', 'X+YF+']
>>>
```

A newline is inserted here

- The complete code for reading the entire file is shown on the following slide

```
rules = []
filename = input("What rule file shall I read? ")
myfile = open(filename, "r") # Open the file for reading

for line in myfile:
    # Read each line as a single rule
    line = line.rstrip()     # remove the end-of-line character
    print("line is", line)

    line = line.split("\t") # separate the two items
    print("line is", line)  # Line is now a list
                            #containing two items
    rules.append(line)
    print()

# Close the file
myfile.close()
print("The data structure is:")
print(rules)
```

Start with an empty list of lists

You can use any variable name to 'point to' the file

Reading the Dragon Curve

```
What rule file shall I read? dragon_curve.txt
line is X        X+YF+
line is ['X', 'X+YF+']

line is Y        -FX-Y
line is ['Y', '-FX-Y']

The data structure is
[['X', 'X+YF+'], ['Y', '-FX-Y']]
>>>
```

```
X        X+YF+
Y        -FX-Y
```

Writing Data To a File

- We have looked at reading a file, now let's look at writing data to a file, in the same file format
- First, the file is opened using 'wt' mode
- 'wt' means 'write to text'
- Then the data in the rule list is converted to text and saved to the file, one line at a time

For the following code we used this data to test the save process:

```
rules =
   [["X", "X+YF+"],
    ["Y", "-FX-Y"]]
```

Writing the Rule File

- Open the file in ‘write as text’ mode
- For every list in the list of lists:
  - Convert the list into the required text format
    - When you write to a text file the list must contain everything so:
      - Put tabs between the two items
      - Put an end-of-line character at the end
  - Write the string to the file
- Close the file

```
filename = input("What rule file shall I create? ")
myfile = open(filename, "wt") # Open the file for writing
```

Use any name, doesn't have to be the same as before

```
# Now we go through each item in the data structure
for rule in rules:

    # Make a string for one line, in the correct format
    one_line = rule[0] + "\t" + rule[1] + "\n"
```

Put a tab between each item

The end-of-line character

```
    # Save the string to the file
    myfile.write(one_line)

# Close the file
myfile.close()
```

It's possible to have several files open at the same time, so you always need to say which file you are referring to