

COMP1021
Introduction to Computer Science

Text and File Handling

David Rossiter


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 write content to a text file
 3. Write code to read content from a text file

Handling Files

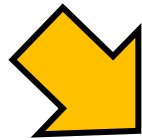
- In this presentation we will look at file handling:
 - How to save data to a file
 - How to load data from a file
- The first things we need to do are to understand:
 - The *tab* character
 - The *end-of-line* character
- Later we will also need to learn about *whitespace*

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 horizontally, to a particular position
 - 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*

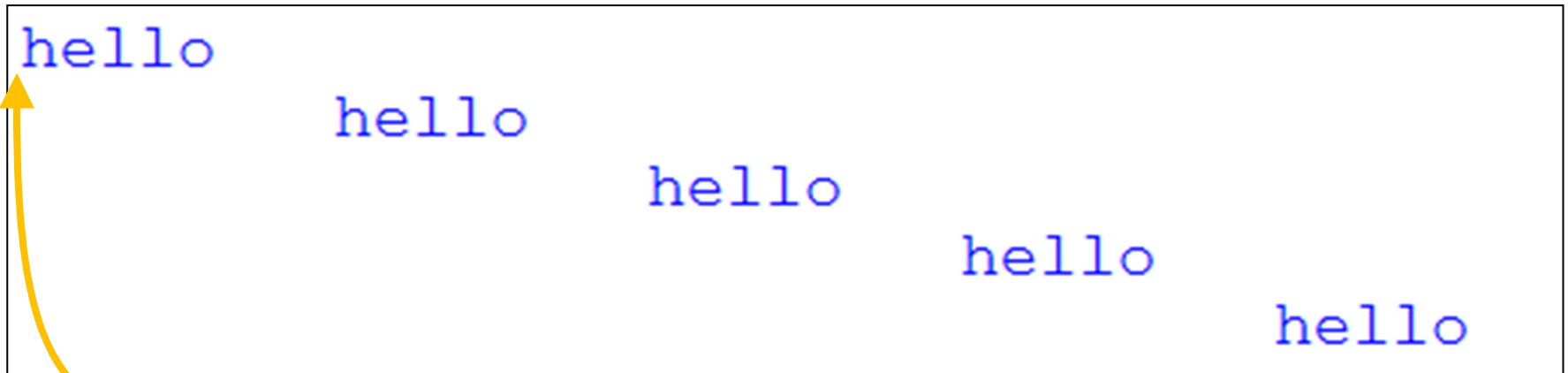
```
Pythagoras' constant is 1.41421
Theodorus' constant is  1.73205
Golden ratio is        1.61803
pi is                   3.14159
e is                    2.71828
>>>
```

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* (discussed elsewhere) than + so it is handled first



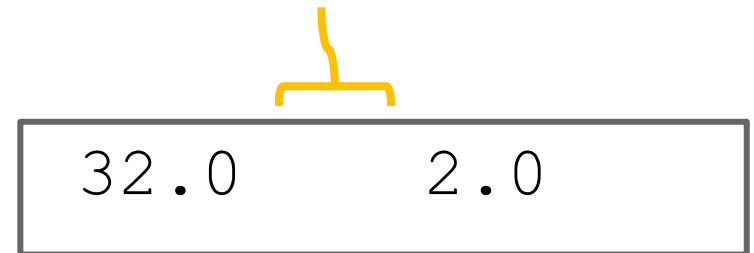
hello
hello
hello
hello
hello

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 the position of some turtles inside a text file
- Each position uses 2 numbers: the x and y values
- We need to separate the two numbers inside the file
- To do that we will use a tab character (although we could use other characters if we wanted to, such as a space)

Here a tab character is used to separate the two numbers in the file



32.0		2.0
------	--	-----

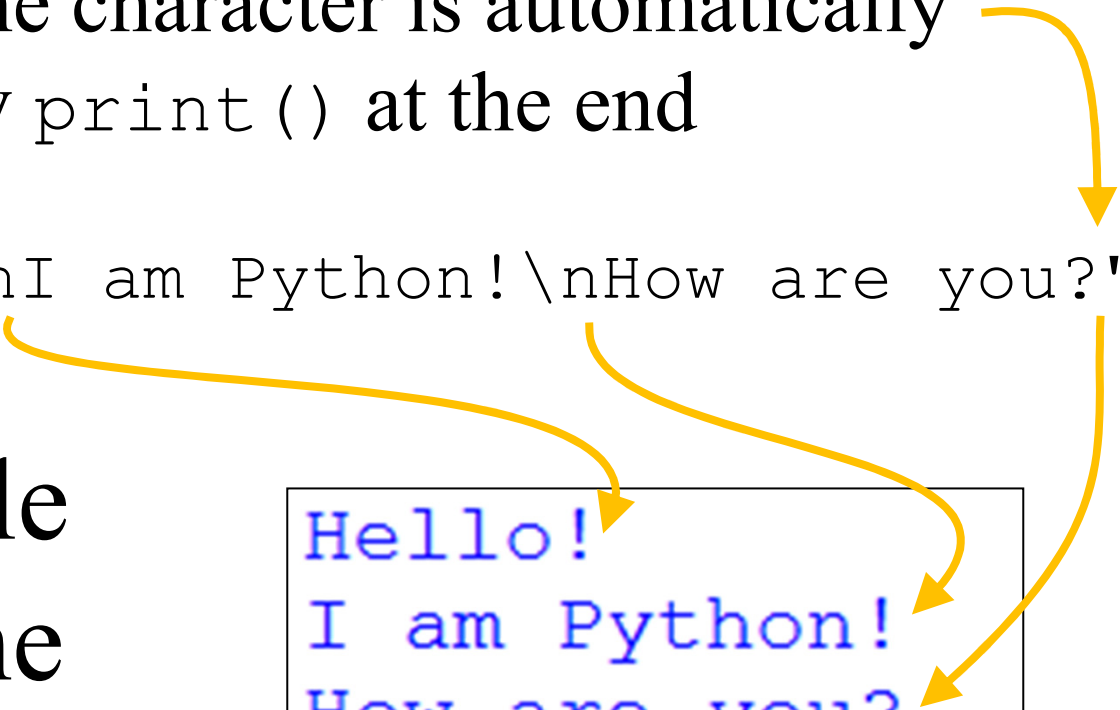
The Newline Character

- 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="")
```

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

Another Example

```
hellohello  
hello
```

```
hello
```

```
hello
```

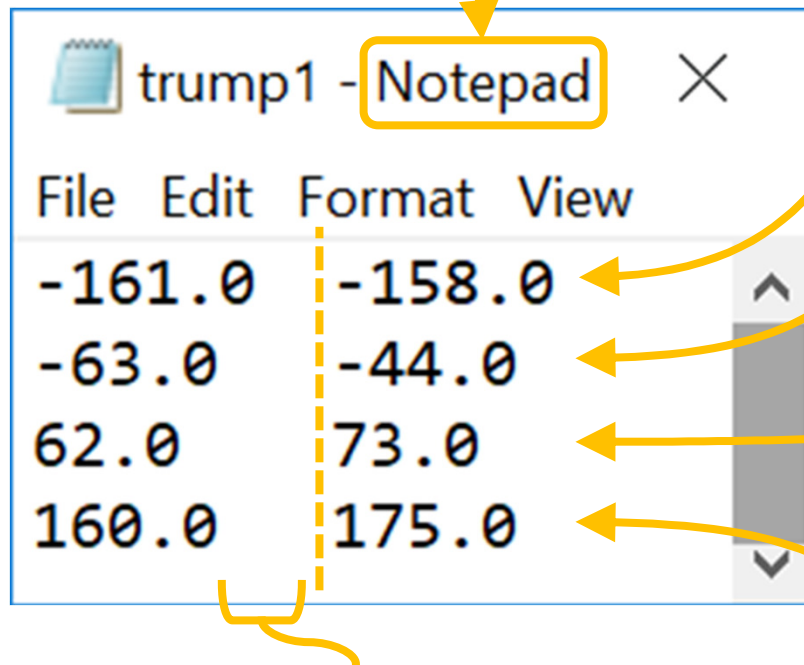
```
>>>
```

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

Reading and Writing Data

- Let's use the jigsaw lab for our example
- We will make code which saves the positions of all the jigsaw pieces (the turtles) into a text file
- And we will make code which loads all the jigsaw position data from the text file, and moves the turtles back to those positions
- This is very helpful: for example, imagine you have been working on a jigsaw with 49 pieces
- Save the jigsaw positions to a file, come back maybe a week later, load the jigsaw positions from the file, then carry on doing the jigsaw

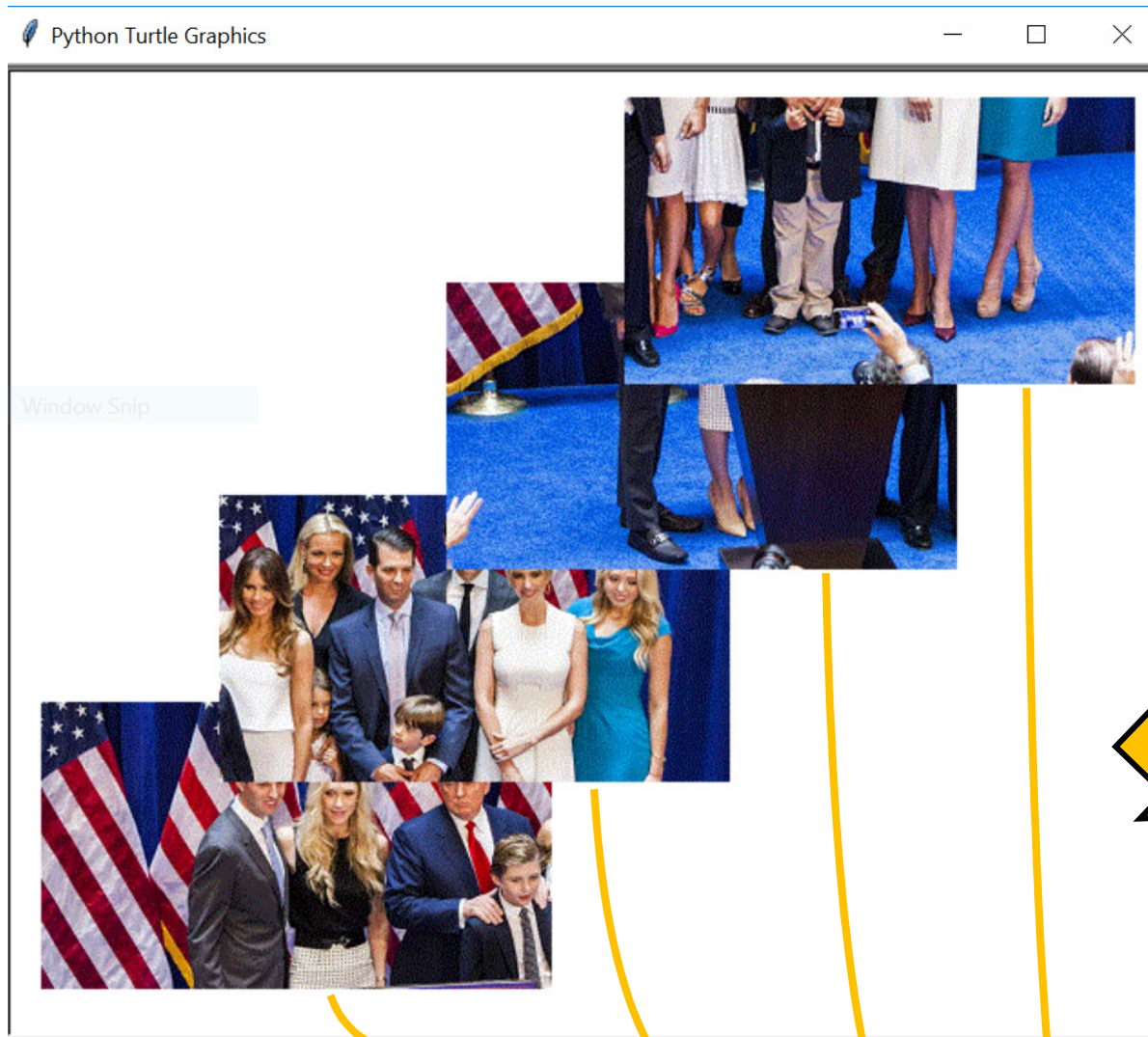
- ‘Notepad’ is a simple program on Windows computers which lets you open and look at text files



*The tab character is
between the two numbers,
on each line*

The File We Will Make

- The position of the first turtle in the list of turtles
- The position of the second turtle in the list
- The position of the third turtle in the list of turtles
- The position of the fourth turtle in the list of turtles

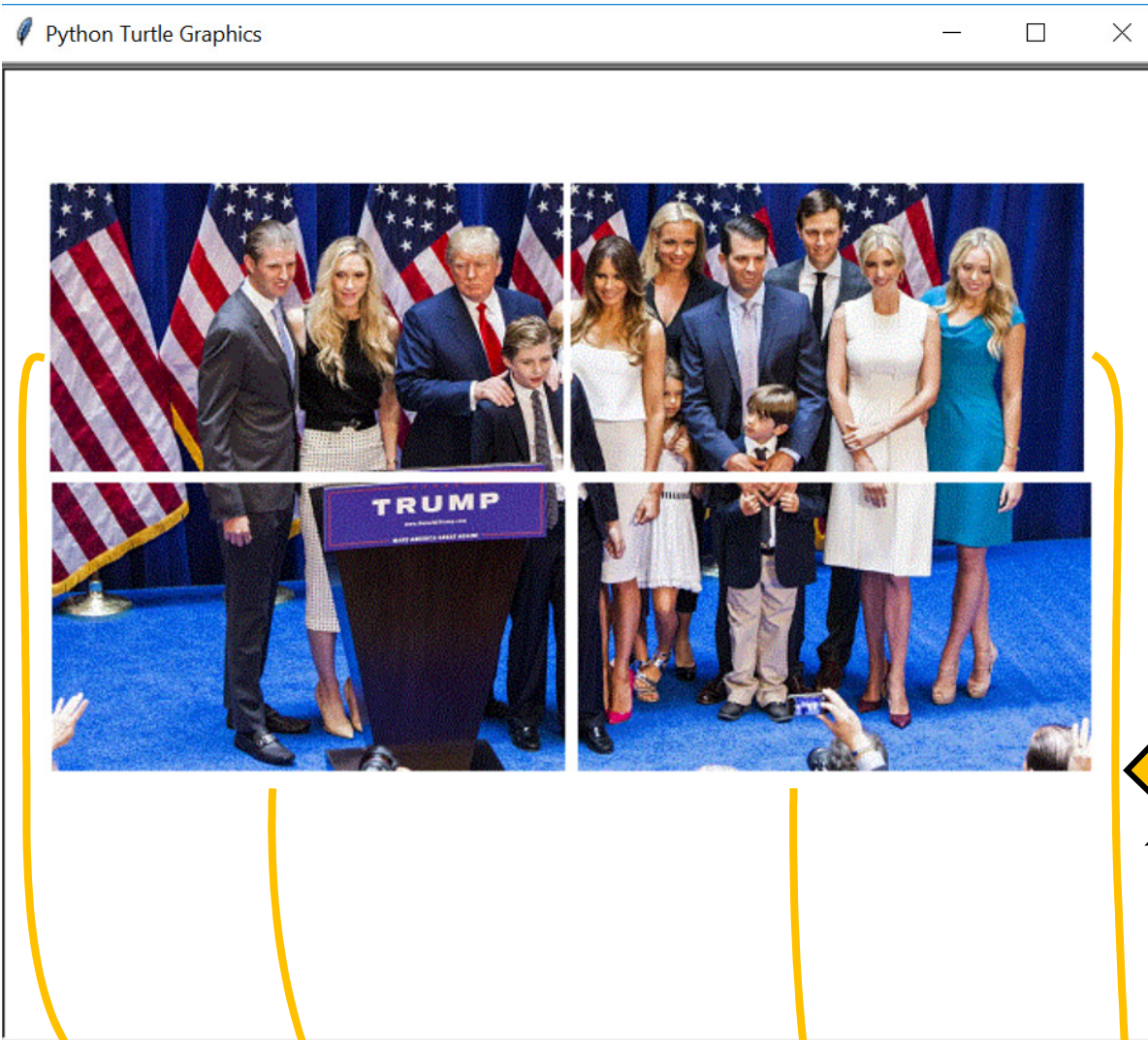


Example 1


*The first turtle
in the list of turtles*

The last turtle in the list of turtles

trump1 - Notepad			
File	Edit	Format	View
-161.0		-158.0	
-63.0		-44.0	
62.0		73.0	
160.0		175.0	



Example 2

 trump2 - Notepad

<u>F</u> ile	<u>E</u> dit	<u>F</u> ormat	<u>V</u> iew
--------------	--------------	----------------	--------------

-153.0	127.0
--------	-------

132.0	127.0
-------	-------

-152.0	-37.0
--------	-------

136.0	-37.0
-------	-------

*The first
turtle
in the list
of turtles*

The last turtle in the list of turtles

Writing the Turtle Positions

- Open the file in ‘write as text’ mode
- For every turtle in the list of turtles:
 - Create one line of text:
 - Convert the turtle x and y into strings
 - Put a tab between the x and y strings
 - Put an end-of-line character at the end
 - Write the line of text to the file
- Close the file

Some Useful Things to Remember

- You can get the x position of a turtle like this:
`turtleName.xcor()`
- You can get the y position of a turtle like this:
`turtleName.ycor()`
- Both of these give you the turtle position
- However, we are creating a text file, so we need to convert the values into strings before we put them in the file, we use `str()` for that

Creating One Line of the Text File

- In the following slide you can see we use this line of code to create the text:

```
one_line = str(thisTurtle.xcor()) + "\t" + \
            str(thisTurtle.ycor()) + "\n"
```

- Then the content of `one_line` will be like this:
-153.0\t127.0\n

```
filename=turtle.textinput("Save jigsaw positions", \
    "What is the jigsaw filename you want to create?")
myfile = open(filename, "wt") #Open the file for writing
```

Use any name to 'point' to the file

```
# Now we go through each turtle in the list of turtles
for thisTurtle in allTurtles:
```

```
    # Make a string for one turtle, in the right format
    one_line = str(thisTurtle.xcor()) + "\t" + \
        str(thisTurtle.ycor()) + "\n"
```

*Put a tab
between the two text*

```
    # Save the string to the file
```

```
    myfile.write(one_line)
```

*Add the end-of-line
character at the end of the line*

```
# Close the file
```

```
myfile.close()
```

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

Reading the File

- We have finished looking at writing the file
- Now let's look at reading the file
- We will read x and y values from each line
- After we read the x and y values, we move the appropriate turtle to that position
- In other words, we are 'restoring' the position of every turtle
- There is one thing which we should learn about first, which is *whitespace*

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 right 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'
>>>
```

Handling One Line of the Text File

- If line of the text file is like this:
 `-153.0\t127.0\n`
- To handle each line we have to do this:
 - Read the line
 - Dump the `\n` at the end of the line
 - Extract the two numbers, by separating the line into separate pieces wherever a `\t` is found
 - Then we can move the turtle to the correct place

Handling One Line of the Text File

- If `one_line` contains this:

```
-153.0\t127.0\n
```

- Then we do:

```
items = one_line.split("\t")
```

- Then `items` will contain this:

```
['-153.0', '127.0']
```

- So now we can e.g. extract the x value and convert it to a float:

```
x=float(items[0])
```

The Sequence For One Line

- This illustrates the series of operations for one line

```
>>> line="-153.0\t127.0\n"
>>> line=line.rstrip()
>>> print(line)
-153.0  127.0
>>> items=line.split("\t")
>>> print(items)
['-153.0', '127.0']
>>> x=float(items[0])
>>> y=float(items[1])
>>> print("The x value is", x)
The x value is -153.0
>>> print("The y value is", y)
The y value is 127.0
```

Reading the File

- 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 `\n` from the end of the string using `rstrip()`
 - Convert the line into a list of two strings using `split("\t")`
 - Convert the x and y values from strings into floats
 - Move the turtle to the x and y values
- Close the file


```
filename=turtle.textinput("Load jigsaw positions", \
    "What is the jigsaw filename you want to load?")
myfile = open(filename, "r") # Open the file for reading
turtleIndex=0
for line in myfile:
    # Handle each line, one by one
    line = line.rstrip()      # Remove the end-of-line

    items = line.split("\t") # Separate the two items

    x=float(items[0])         # Convert x to a float
    y=float(items[1])         # Convert y to a float

    allTurtles[turtleIndex].goto(x, y) # Move turtle
    turtleIndex=turtleIndex+1 # Increase the index,
                                # for the next turtle
myfile.close() # We have finished, now close the file
```

You can use any variable name to 'point to' the file, it doesn't have to be the same one used before