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*

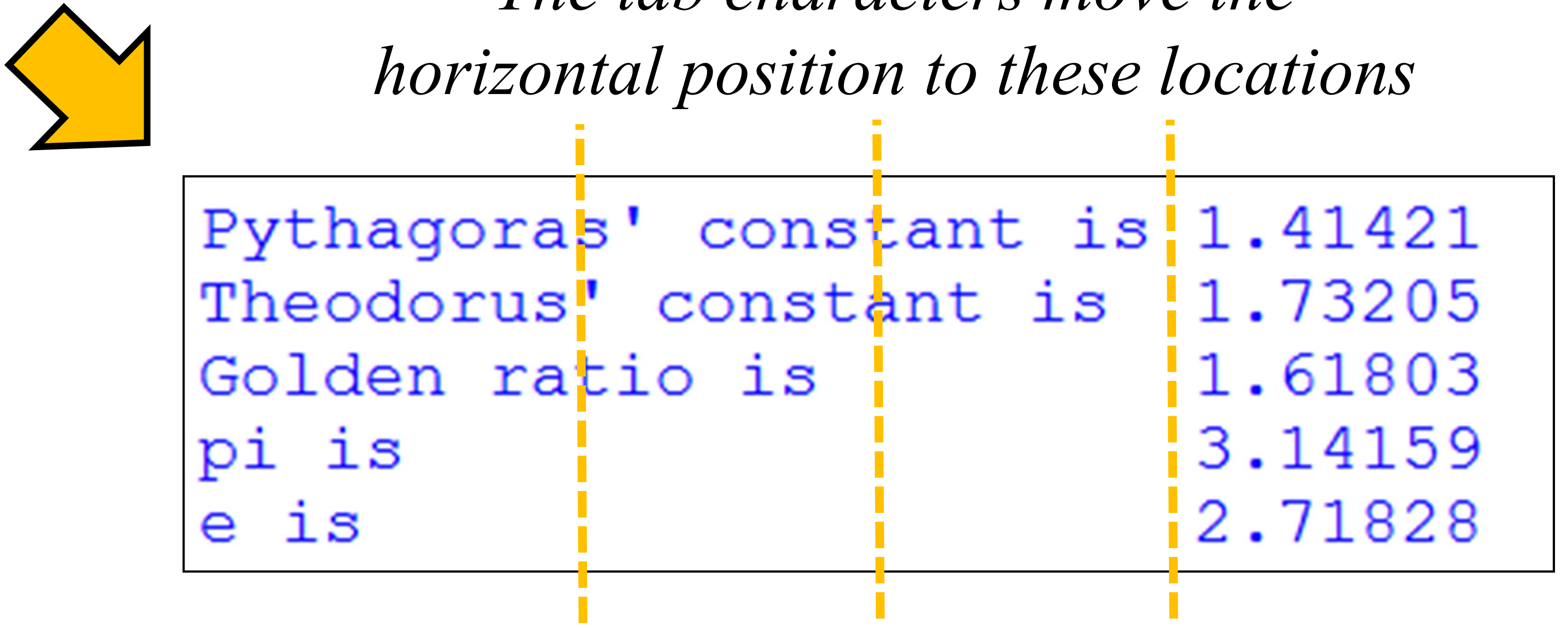
The Tab Character

- 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



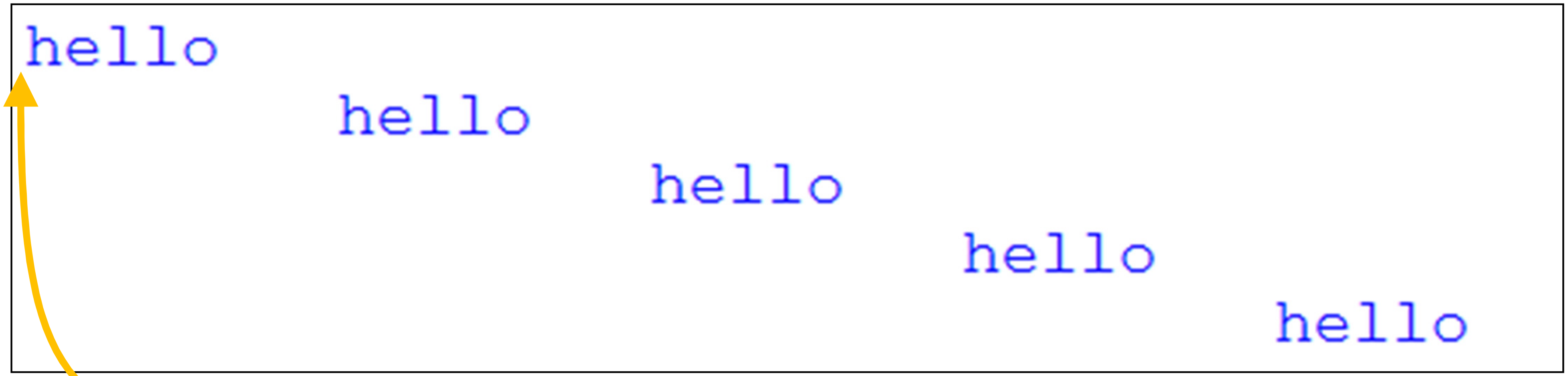
```
Pythagoras' constant is 1.41421
Theodorus' constant is  1.73205
Golden ratio is         1.61803
pi is                   3.14159
e is                    2.71828
```

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



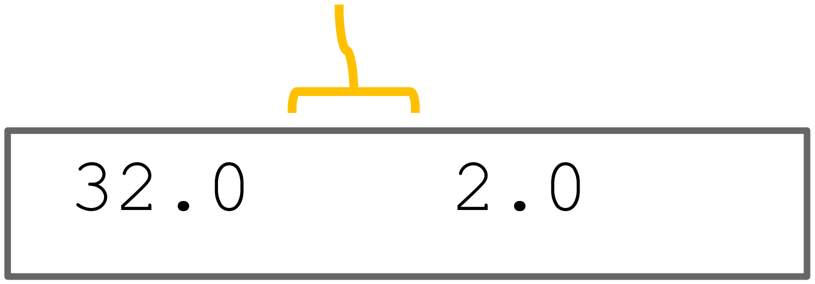
```
hello
	hello
		hello
			hello
				hello
```

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 (we could use other characters if we wanted to e.g. a space)

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



```
32.0	2.0
```

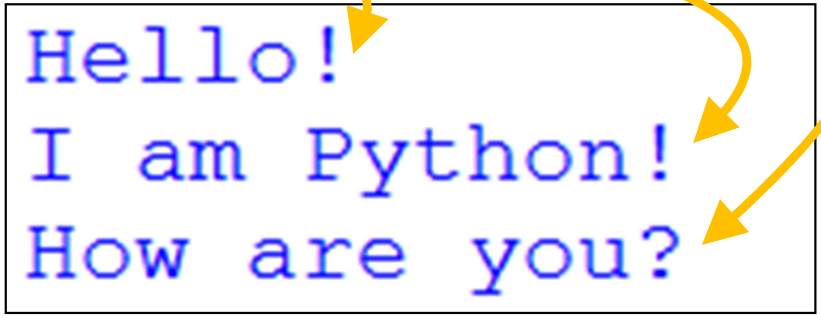
The Newline Character

- 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

An Example of Using the Newline Character

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

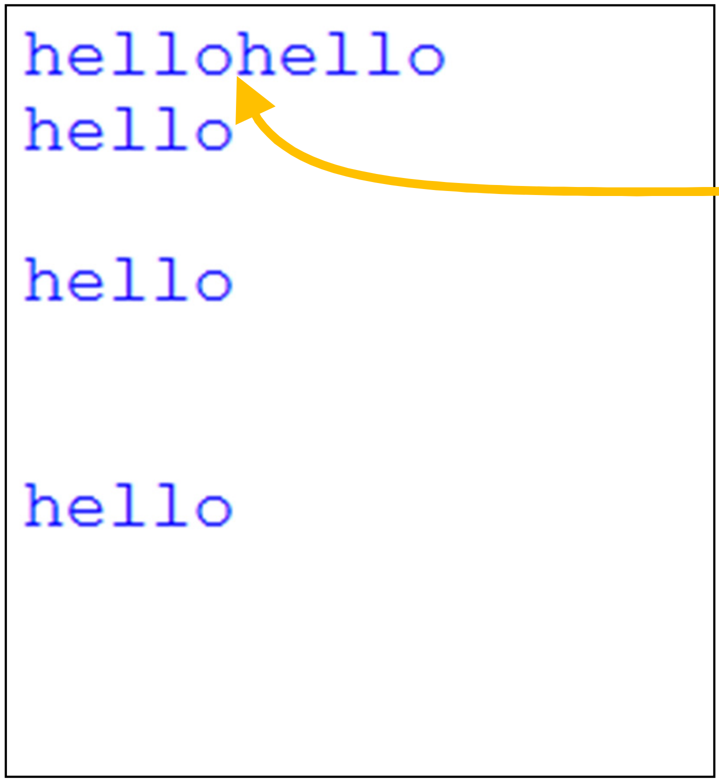
```
print("Hello!\nI am Python!\nHow are you?")
```



```
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
```

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

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

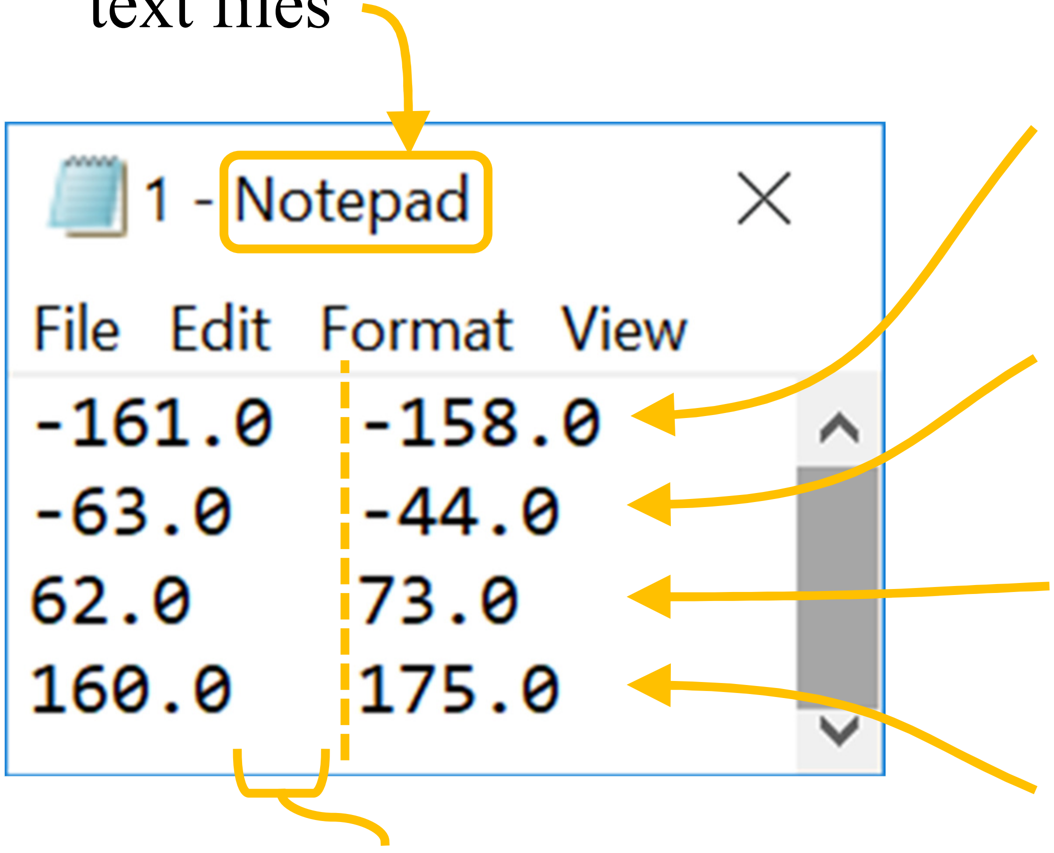
Another Example

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
- For example, imagine you have been trying to solve a difficult jigsaw with many 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

The File We Will Make

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

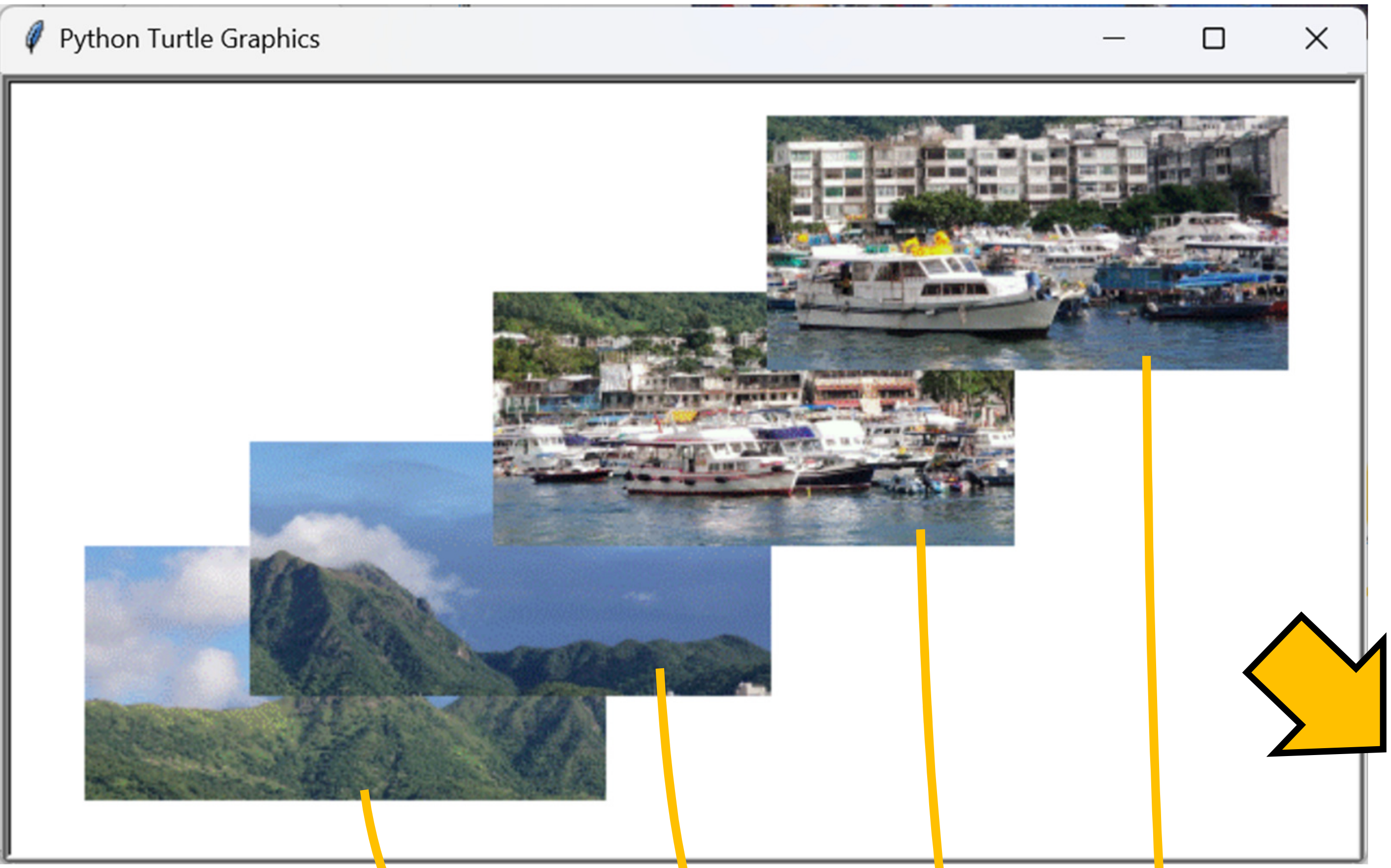


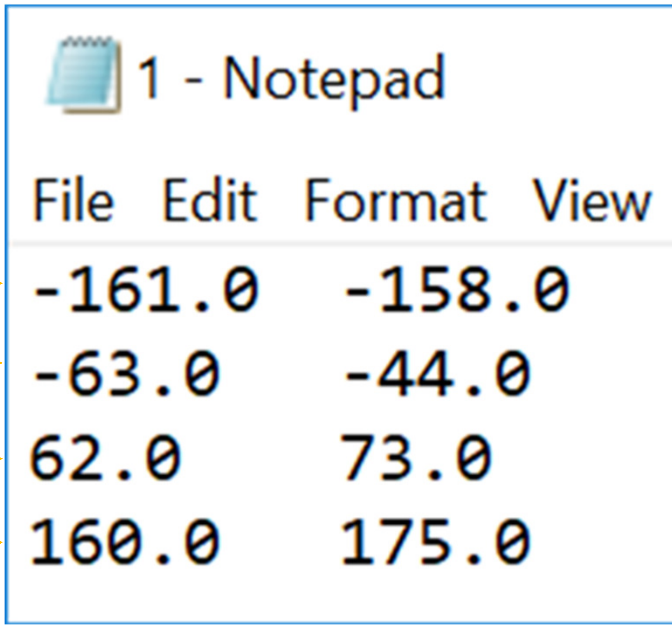
```
-161.0	-158.0
-63.0	-44.0
62.0	73.0
160.0	175.0
```

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

- 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



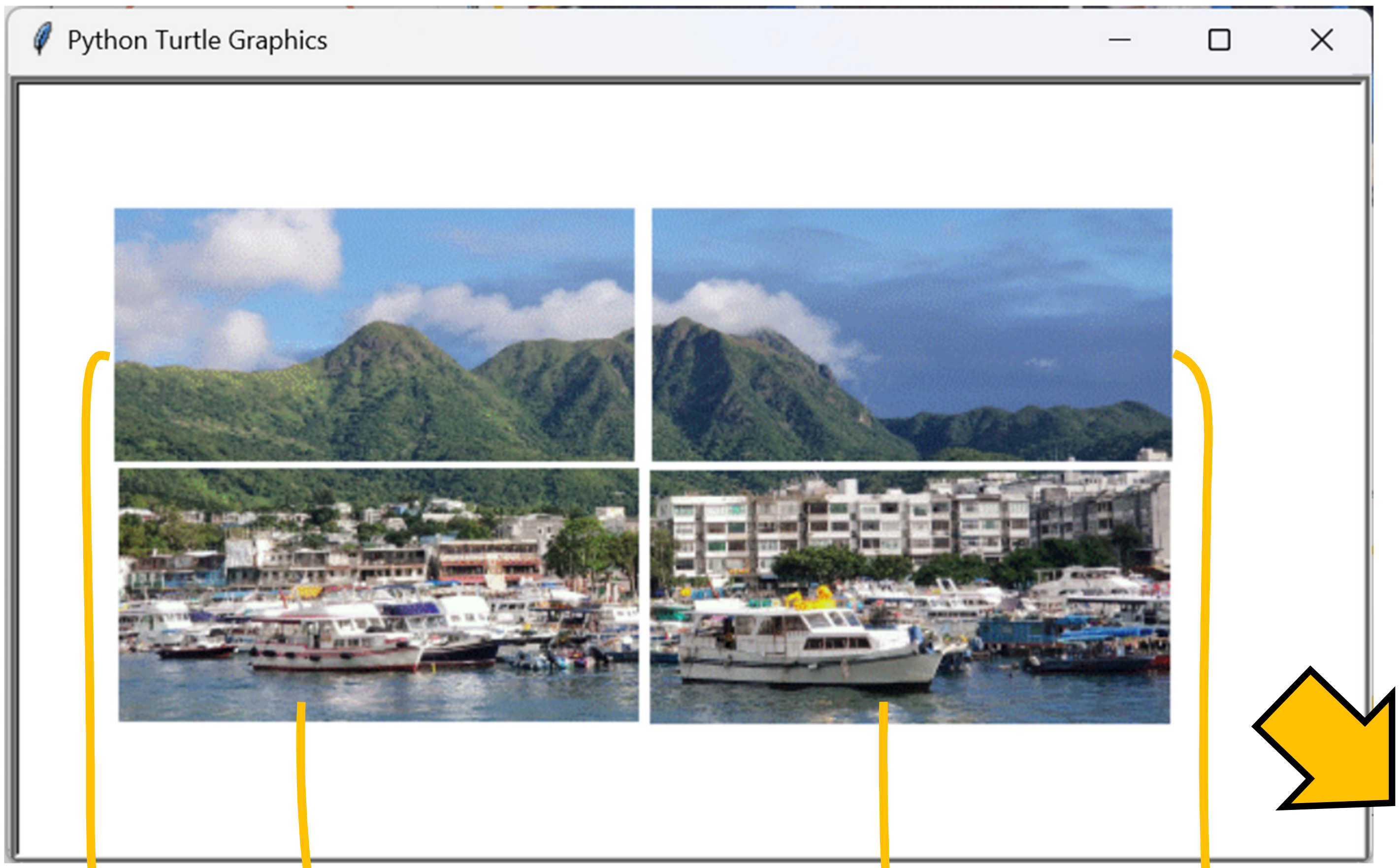


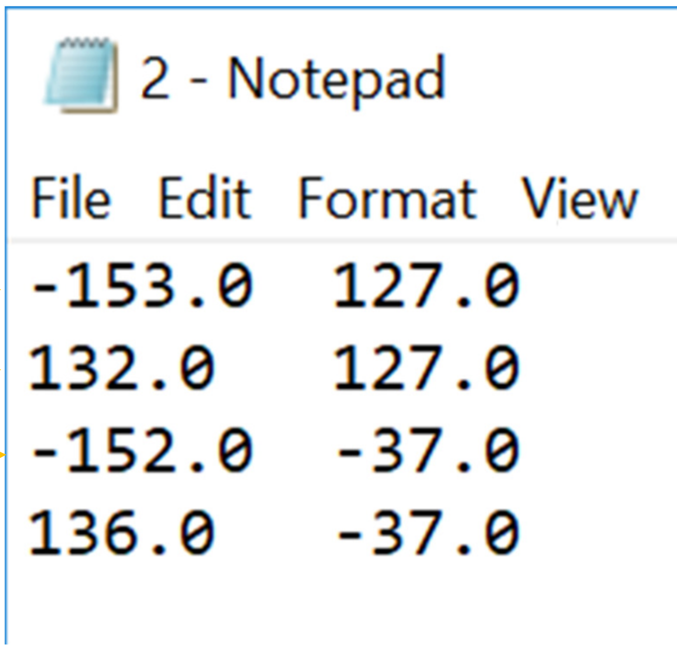
```
-161.0    -158.0
-63.0     -44.0
62.0      73.0
160.0     175.0
```

The first turtle in the list of turtles

The last turtle in the list of turtles

Example 2





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

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

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

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

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

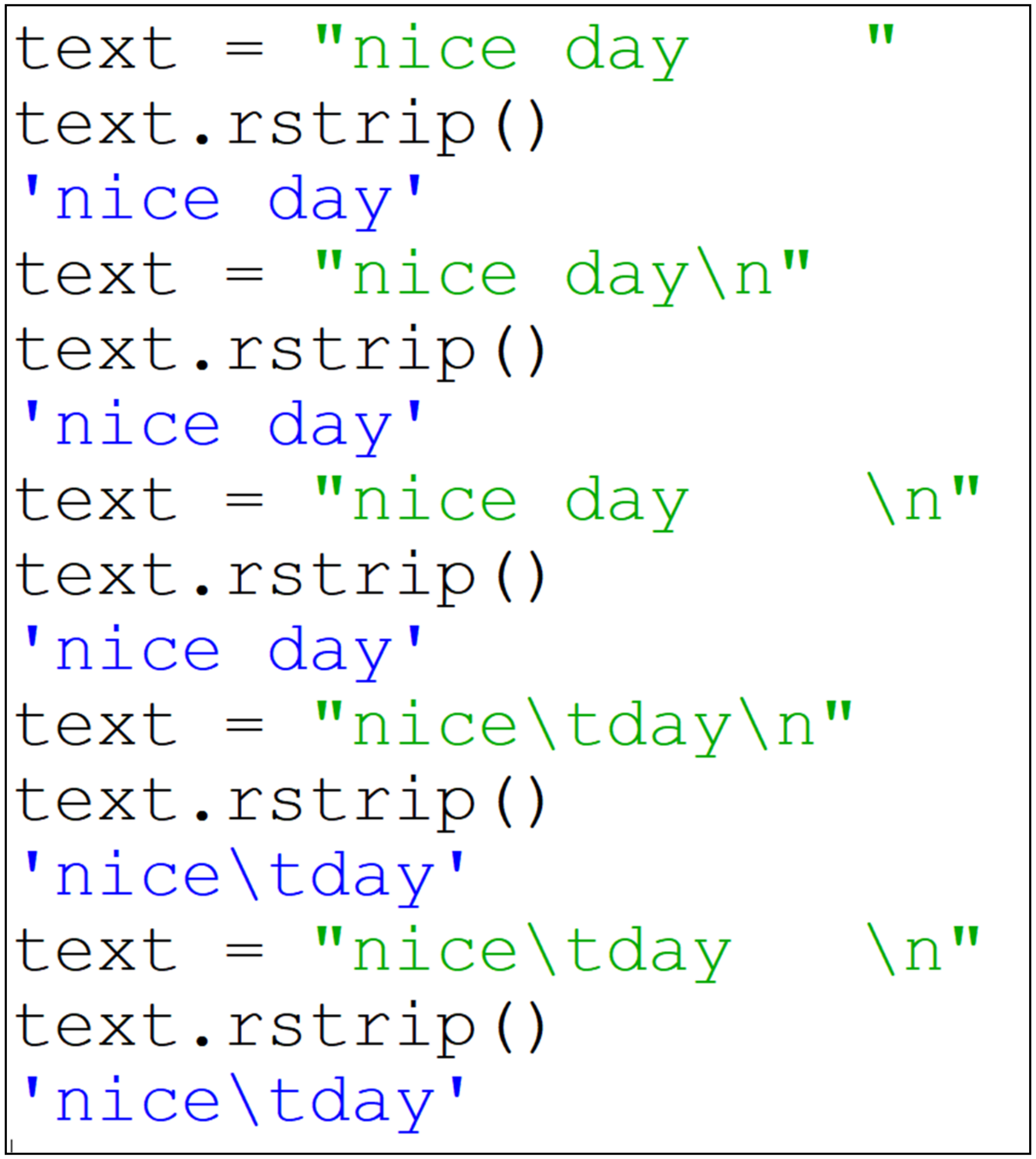
It's possible to have several files open at the same time, so you need to say which file you are referring to

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’
- That includes spaces and end-of line characters
- We use `rstrip()` to remove whitespace
- `rstrip()` means ‘strip (=remove) anything you can’t see on the right side’



```
text = "nice day    "
text.rstrip()
'nice day'
text = "nice day\n"
text.rstrip()
'nice day'
text = "nice day    \n"
text.rstrip()
'nice day'
text = "nice\tday\n"
text.rstrip()
'nice\tday'
text = "nice\tday  \n"
text.rstrip()
'nice\tday'
```

Handling One Line of the Text File

- If one line of the text file is like this:

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

- To handle each line we do this:
  - Read the line
  - Dump the `\n` at the end of the line
  - Extract the two numbers, by dividing 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 `line` contains this:

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

- We do this: `line = line.rstrip()`
- Then this: `items = line.split("\t")`
- Now `items` will contain this list:

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

- So now we can extract the x value from the list and convert it to a float:

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

- Then we can do the same for the y value

Reading 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])
print(x)
-153.0
y=float(items[1])
print(y)
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 to 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