Computer Code for Beginners
Week 6

Matt Luckcuck

26th October 2017

Outline

Previously...

- Dictionaries

This Time...

- Exceptions
- File Handling
  - JSON Format

Exceptions

Exceptions

Exceptions Intro

- Errors that occur during execution are not always fatal
  - E.g `TypeError` and `NameError`
  - We can 'handle' them
- Code is said to 'raise' or 'throw' an exception

Exceptions

Exceptions We've Already Seen

```
1 total = 10
2 print("Total = " + total)
3 TypeError: Can't convert 'int' object to str
                              implicitly
```

```
1 var = x + 3
2 NameError: name 'x' is not defined
```

Exceptions

Exceptions in For Loops

```
1 for item in aList:
2   print(item)
```

- When there are no more items in `aList`...
- An `IndexError` is raised

Exception Handling

How?

- Be aware of code that might throw an exception
- Python provides some extra keywords:
  - `try`
  - `except`
  - `finally`

Exception Handling

How?

- Problematic code goes inside a `try:` block
  - *Try* and run this code
- Code to handle an exception goes in an `except:` block
  - If there's an exception in the `try` block, run this code
  - Any exception or a specific exception
    - E.g. `except TypeError:`
    - E.g. `except NameError as e:`
  - May use more than one per `try`

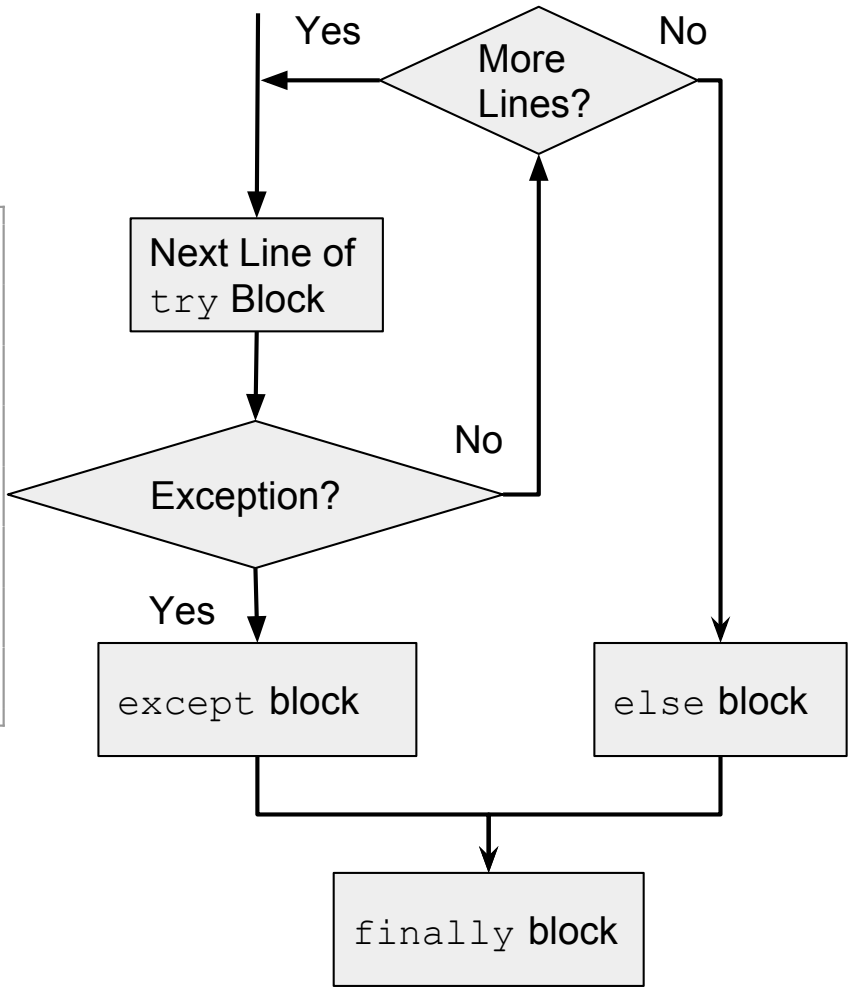
Exception Handling

Optional Extras

- Use an `else:` block to execute code only if there it no exception
- A `finally:` block is
  - Always executed as the last part of a `try` statement
  - Executes whether there's an exception or not

Exception Handling

```
1 try:
2   <problematic code>
3 except:
4   <handle it>
5 else:
6   <no exception>
7 finally:
8   <code to cleanup>
```



Exception Handling

Exception Example

```
1 result = int(input("Input a Number"))
2 print(result)
```

- What if the user *doesn't* input a number...

Exception Handling

Exception Example

```
1 result = int(input("Input a Number"))
2 print(result)
```

- What if the user *doesn't* input a number...
- 'Octopus'

Exception Handling

Exception Example

```
1 result = int(input("Input a Number"))
2 print(result)
```

- What if the user *doesn't* input a number...
- 'Octopus'
- ValueError: invalid literal for int()with base 10 :'Octopus'

Exception Handling

Exception Example

```
1 try:
2   result = int(input("Input a Number"))
3   print(result)
4 except ValueError:
5   print("Not a number")
```

- Run the code in the `try` block...
- If there is a `ValueError` then
  - Jump to the `except` block

Exception Handling

Exception Example

```
1 try:
2   result = int(input("Input a Number"))
3 except ValueError:
4   print("Not a number")
5 else:
6   print(result)
```

- Run the code in the `try` block...
- If there is a `ValueError` then
  - Jump to the `except` block
- If there was no exception
  - Run the `else` block, after the `try`

Exception Handling

When?

- Handling exceptions is similar to using a `if: ... else:`
  - Checking for what could cause an exception
  - But checks line-by-line
- Important to make sure you don't cause more exceptions
  - E.g Handle an exception so that a function returns what is expected of it
- Design Decision...
  - Slight performance overhead, compared to `if: ... else:`
  - The 'Python way' is to try then recover if you fail
  - Use them if exceptions would make your code cleaner or safer

Files

File Handling

Files

- Another useful thing we can write programs to do
- Python allows us to do this very simply:
  - `f = open("theFile.txt","r")` – Read mode
  - `f = open("theFile.txt","w")` – Write mode
  - Now `f` points at the file we opened

File Handling

File Reading

- After opening a file we can read from it...
- As a string:
  - `f.read()`
    - Reads the whole file (careful of file size)
  - `f.readline()`
    - Reads one line, terminated by newline (`'\\n'`)
- As a list:
  - `f.readlines()`
    - Reads all the lines into a list (keeping newline characters)
  - `f.read().splitlines()`
    - Nice way of getting a list of 'clean' strings (no newline)

File Handling

File Reading

- We can also use a loop...

```
1 f = open("theFile.txt","r")
2
3 for line in f:
4     print(line)
```

File Handling

File Writing

- After opening a file we can write to it...
- `f.write('This is a test \\n')`
  - Note the new line character
  - Without it, the next write would be on the same line

File Handling

File Reading

- Once we're done with the file...
- `f.close()`
  - Close the file (always)

File Handling Exceptions

File Exceptions

- File handling is a very good place to use exceptions...
  - We *can* check if a file exists before opening it...
  - But the file could be deleted before we open it
- A `FileNotFoundError` is raised if we try to open a non-existent file
  - Catching this exception is *very* good practice
  - Very specific problem, could catch `OSError` instead...
- Remember that the file needs closing once we're done
  - If it was successfully opened
  - Use `else` block

File Handling Exceptions

Example

```
1 f = open("theFile.txt","r")
2
3 for line in f:
4     print(line)
5
6 f.close()
```

File Handling Exceptions

Example

```
1 try:
2     f = open("theFile.txt","r")
3 except FileNotFoundError as err:
4     print(err)
5 else:
6     for line in f:
7         print(line)
8     f.close()
```

File Handling

File Handling

- Formatting of the data in a file is important
- Strings and numbers as easy enough. . .
- How do we represent:
  - Lists?
  - Dictionaries?
- We can use a standard data format to help

File Handling

JSON

- JavaScript Object Notation
  - Language-independent data exchange format
  - Based on JavaScript
- Built on:
  - **Object:** Unordered collection of key, value pairs (Dictionary)
  - **Array:** Ordered sequence of values (List)
- Also has:
  - Boolean
  - String
  - Numbers

File Handling

JSON in Detail

- **Object:** `{ "key":value, "key":value }`
  - Keys must be strings
- **Array:** `[ value, value, value ]`
- Values:
  - String
  - Number
  - Object
  - Array
  - True
  - False
  - Null (like `None`)
- More detail online: `json.org`

File Handling

JSON in Python

- Python has a json library
  - `import json`
  - Open the file...
- Decode (read) json
  - `json.load(jsonFile)`
- Encode (write) json
  - `json.dump(data, file)`
- Both have string equivalents:
  - Decode from a string `json.loads(jsonString)`
  - Encode to a string `json.dumps(data)`

File Handling

JSON in Python

```
#file.json
{"Octopus":42,"cat": 5}
```

```
import json

f = open("file.json")
d = json.load(f)
print(d) #{"Octopus":42,"cat": 5}
```

File Handling

JSON in Python

- Be careful:
  - `json.decoder.JSONDecodeError` while decoding
    - Good practice to catch this exception
  - Python tuples are encoded as lists

Summary

Summary

Summary

- Exceptions
  - `try:` and `except:`
  - Good practice to catch errors that you anticipate
  - Be careful not to cause other exceptions from handling
- File Handling
  - Reading and writing
  - Potential for `FileNotFoundError`
- JSON Format
  - Simple text-based file format
  - Really useful for data persistence
  - Potential for `json.decoder.JSONDecodeError`

Summary

Exercises

- Longest Word (in a File)
- Letters in a File
- Mine Detector (But with files!)

Summary

Final Week

- ~~More Computer Code course~~
- Top of Course Website for:
  - Learner Plans (post-course)
  - Feedback Forms

Summary

More. . .

- Online Programming Courses. . .
- Raspberry Pi Projects
- Code Club
- Hacking Your Own Project Together
  - `projects.raspberrypi.org`
- Computerphile
  - `www.youtube.com/user/Computerphile`
  - General computer science topics from University of Nottingham
- The Digital Human
  - `http://www.bbc.co.uk/programmes/b01n7094`
  - Digital Technology and its interaction with society