COMPSCI 105 S1 2017
Principles of Computer Science

JSON

Quizzes

- What is the output of the following program when x is 1, 0 and '0'?

```
def testing(x):
    try:
        print('Trying some code')
        2 / x
    except ZeroDivisionError:
        print('ZeroDivisionError raised here')
    except:
        print('Error raised here')
    else:
        print('Else clause')
    finally:
        print('Finally')
```

Exercise

- MCQ:
  - Which of the following statements is/are true?
    a) A try block is preceded by at least one finally block
    b) For each try block there must be at least one except block defined.
    c) A try block may be followed by any number of finally blocks
    d) If both except and finally blocks are defined, except block must precede the finally block.

Learning outcomes

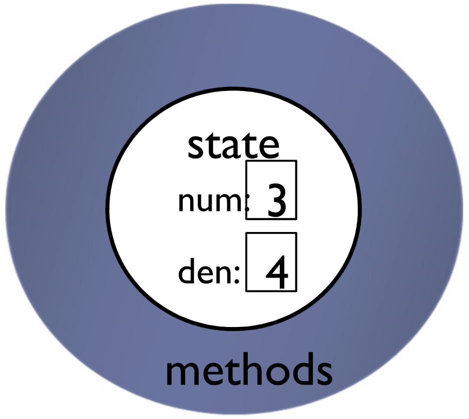
- At the end of this lecture, students should be able to:
  - understand what JSON is used for
  - recognise information in JSON format
  - use the Python JSON library to read and write standard Python data types

- Resources:
  - Tutorials Point: JSON with Python
    - http://www.tutorialspoint.com/json/json_python_example.htm
  - Python Documentation
    - https://docs.python.org/3.3/library/json.html

Question?

- Given a particular set of data, how do you store it permanently?
  - What do you store on disk?
  - What format?
  - Can you easily transmit over the web?
  - Will it be readable by other languages?
  - Can humans read the data?

- Examples:
  - A square
  - A dictionary



Storage using plain text

- Advantages
  - Human readable (good for debugging / manual editing)
  - Portable to different platforms
  - Easy to transmit using web
- Disadvantages
  - Takes more memory than necessary
- Use a standardized system -- JSON
  - Makes the information more portable

JavaScript Object Notation

- Text-based notation for data interchange
  - Human readable

- Object
  - Unordered set of name-value pairs
  - names must be strings
  - { name1 : value1, name2 : value2, …, nameN : valueN }

- Array
  - Ordered list of values
  - [value1, value2, … valueN]

Example01.py

Writing JSON using Python

- json.dumps(data)
  - Accepts Python object as an argument
  - Returns a string containing the information in JSON format
  - Typically write this string to a file

```
def write(data, filename):
    file = open(filename, 'w')
    str_out = json.dumps(data)
    file.write(str_out)
    file.close()
```

`Example01.py`

Reading JSON using Python

- json.loads(data)
  - Accepts string as an argument
  - The string should be in JSON format
  - Returns a Python object corresponding to the data

Double quotes

```
"Hello World"
```

'hello.txt'

```
def read(filename):
    file = open(filename)
    str_in = file.read()
    file.close()
    data = json.loads(str_in)
    return data
```

```
write('Hello World', 'hello.txt')
print(read('hello.txt'))
```

Example02.py

Example 2: Writing a dictionary

- Create a dictionary

```
my_dict = {'Angela':'86620','adriana':'87113, 'ann':'84947'}
file_name = 'test_dict.txt'
write(my_dict, file_name)
```

```
{"ann": "84947", "adriana": "87113", "Angela": "86620"}
```

```
print(read(file_name))
```

`Example03.py`

Writing JSON using pretty printing

- json.dumps(data)

A dictionary

```
{'b': ['HELLO', 'WORLD'], 'a': ['hello', 'world']}
```

- json.dumps(data, indent=4, sort_keys=**True**)
  - Formats the output over multiple lines

```
{
    "a": [
        "hello",
        "world"
    ],
    "b": [
        "HELLO",
        "WORLD"
    ]
}
```

Double quotes

What about user-defined classes?

- Point class

```
class Point:
    def __init__(self, loc_x, loc_y):
        self.x = loc_x
        self.y = loc_y

    def __str__(self):
        return str(self.x) + ',' + str(self.y)
```

- Can create a dictionary to store state information then use JSON

value of x

value of y

```
p = Point(2, 3)
my_dict = {'__class__': 'Point', 'x' : p.x, 'y' : p.y}
```

What about user-defined classes?

- Can use json to read and extract the state information

```
file_name = 'test_point.txt'
write(my_dict, file_name)
```

```
{
    "__class__": "Point",
    "x": 2,
    "y": 3
}
```

- Example:

```
data = read(file_name)
result = Point( data['x'], data['y'] )
print (result)
```

Exercise

- Given a Square class, write methods that dump and read JSON

```
import json
import io

class Square:
    def __init__(self, len):
        self.side_length = len

    def __str__(self):
        #write your code here
```

Summary

- JSON is a standard way to exchange data
  - Easily parsed by machines
  - Human readable form
- JSON uses dictionaries and lists
  - Dictionaries are unordered
  - Lists are ordered
- Symbols used in JSON are the same as Python
  - Double quotes used for strings