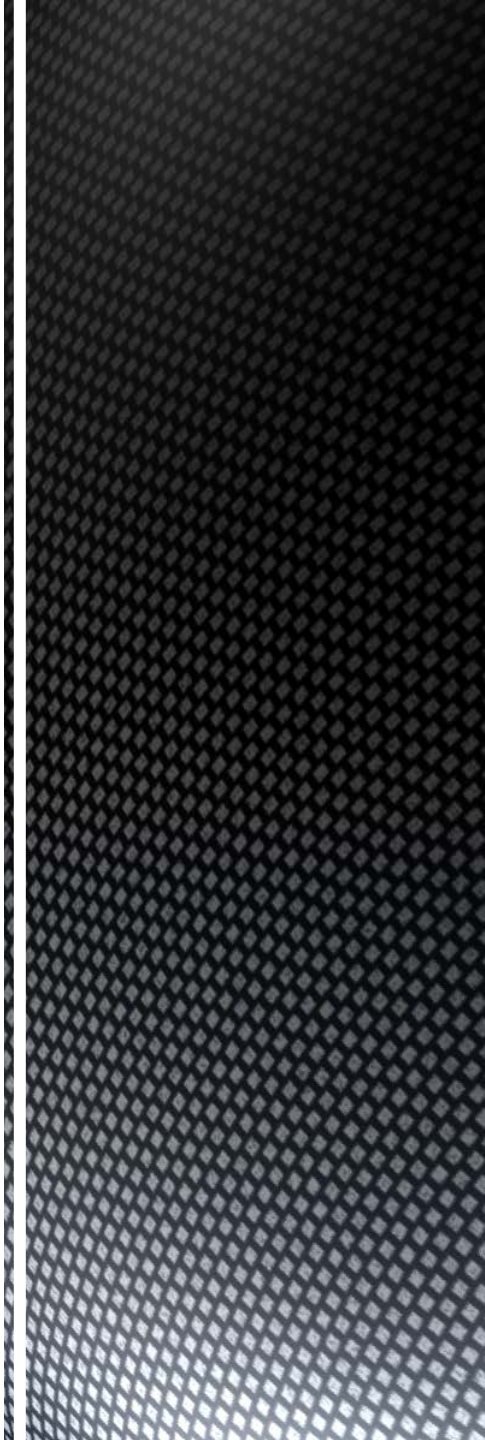
COMPSCI 107
Computer Science Fundamentals

Lecture 14 - JSON



JavaScript Object Notation

- Text-based notation for data interchange
  - Human readable
- Object
  - Unordered set of name-value pairs
  - { name1 : value1, name2 : value2, …, nameN : valueN }
- Array
  - Ordered list of values
  - [value1, value2, … valueN]

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

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

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

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

Writing JSON using pretty printing

- json.dumps(data)

```
{'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"
    ]
}
```

What about user-defined classes?

- Point class

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

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

```
def generate_json(p):
    out = {'_Point' : True, 'x' : p.x, 'y' : p.y}
    return json.dumps(out, sort_keys=True)
```

- Can use json to read dictionary and extract the state information

```
def generate_point(txt):
    inp = json.loads(txt)
    result = Point( inp['x'], inp['y'] )
    return result
```

Program Development

- Start by thinking of the different kinds of input and the output
- Test Cases
- Work on the solution, keeping the test cases in mind
- Test your code after each development advance

Debugging

- Debugging and tracing code are closely linked skills

- To debug your code, you need to know:
  - what your code *should* produce
  - what your code *does* produce
  - why is there is a difference

- Use text output to determine data
  - Test functions at entry and exit points
  - Test loops at entry and exit points

- If data is large or complex, save output to a file
  - JSON may help

Converting from infix to postfix

- A stack can be used in the algorithm to convert infix to postfix
  - Divide expression into tokens
  - Operators: +. -, *, /
  - Operands: single digits
  - Other tokens: brackets

Algorithm for converting infix to postfix

- Create a stack to store operators and a list for the output tokens
- Scan the tokens from left to right
- If the token is an operand, add it to the output list
- If the token is a left parenthesis, push it to the operator stack
- If the token is a right parenthesis, pop the operator stack until the left parenthesis is removed. Append each operator to the output list
- If the token is an operator, push it onto the operator stack. But first, remove any operators that have higher or equal precedence and append them to the output list
- When there are no more tokens, remove operators on the stack and append to the output list

Exercise

- Show the operator stack and the output list at every step as the following infix expression is converted to postfix

$$12 / (3 + 4) * 2 + 4$$

Evaluating postfix expressions

- Create an empty stack
- Scan the list of tokens from left to right
- If the token is an operand, push it to the operand stack
- If the token is an operator, pop the stack twice
  - The first element popped is the right operand
  - The second element popped is the left operand
- Apply the operator to the operands and push the result onto the stack
- When there are no more tokens, the stack should contain the result.

Exercise

- Following the algorithm to evaluate postfix expressions, show the operand stack, and the token being processed (at each step) as the following postfix expression is evaluated:

7 12 8 9 - * 3 / +

Exercise

- How does a user know if the circular_queue is full? What should happen when the circular_queue is full? Discuss

```
class circular_queue:
    def __init__(self, capacity):
        #creates empty list, count, front, back

    def is_empty(self):

    def enqueue(self, item):

    def dequeue(self):

    def size():
```

ADT Deque

- A Double-Ended Queue or Deque (pronounced ‘Deck’)
  - An ordered collection of items where items are added and removed from either end, either front or back

- add_front()
- add_rear()
- remove_front()
- remove_rear()
- is_empty()
- size()

Exercise

- Use a double ended queue to write a function that determines if a string is a palindrome.

- A palindrome is a sentence in which the letters appear in the same order forwards and reverse. Punctuation is ignored.

```
>>> is_palindrome('bob')
True
```

Bob – Weird Al Yankovic

I, man, am regal - a German am I
Never odd or even
If I had a hi-fi
Madam, I'm Adam
Too hot to hoot
No lemons, no melon
Too bad I hid a boot
Lisa Bonet ate no basil
Warsaw was raw
Was it a car or a cat I saw?

Rise to vote, sir
Do geese see god?
"Do nine men interpret?" "Nine men," I nod
Rats live on no evil star
Won't lovers revolt now?
Race fast, safe car
Pa's a sap
Ma is as selfless as I am
May a moody baby doom a yam?

Ah, Satan sees Natasha
No devil lived on
Lonely Tylenol
Not a banana baton
No "x" in "Nixon"
O, stone, be not so
O Geronimo, no minor ego
"Naomi," I moan
"A Toyota's a Toyota"
A dog, a panic in a pagoda

Oh no! Don Ho!
Nurse, I spy gypsies - run!
Senile felines
Now I see bees I won
UFO tofu
We panic in a pew
Oozy rat in a sanitary zoo
God! A red nugget! A fat egg under a dog!
Go hang a salami, I'm a lasagna hog

Exercise

- What is the big-O running time for the following function?

```
def exampleA(n):
    s = "PULL FACES"

    for i in range(n):
        print("I must not ", s)

    for j in range(n, 0, -1):
        print("I must not ", s)
```

Exercise

- What is the big-O running time for the following function?

```
def exampleB(n):
        s = "JUMP ON THE BED"

        for i in range(n):
                for j in range(i):
                        print("I must not ", s)
```

Exercise

- What is the big-O running time for the following function?

```
def exampleC(n):
        s = "WHINGE"
        i = 1
        while i < n:
                for j in range(n):
                        print("I must not ", s)

                i = i * 2
```

Exercise

- What is the big-O running time for the following function?

```
def exampleD(n):
        s = "PROCRASTINATE"

        for i in range(n):
                for j in range(n, 0, -1):
                        outD(s, n / 2)

def outD(s, b):
        number_of_times = int(b % 10)
        for i in range(number_of_times):
                print(i, "I must not ", s)
```

Exercise

- What is the big-O running time for the following function?

```
def exampleF(n):
    s = "FORGET MY MOTHER'S BIRTHDAY"
    i = n
    while i > 0:
        outF(s)
        i = i // 2

def outF(s):
    for i in range(25, 0, -1):
        print(i, "I must not ", s)
```

Challenge Question

- If a particular quadratic time algorithm uses 300 elementary operations to process an input of size 10, what is the most likely number of elementary operations it will use if given an input of size 1000.

- (a) 300 000 000
- (b) 3 000 000
- (c) 300 000
- (d) 30 000
- (e) 3 000

Challenge Question

- You know that a given algorithm runs in $O(2^n)$ time. If your computer can process input of size 10000 in one year using an implementation of this algorithm, approximately what size input could you solve in one year with a computer 1000 times faster?

A. 10 100
B. 320 000
C. 10 010
D. 15 000
E. 10 000 000

ChallengeQuestion

The running time for the following code fragment is $\Theta(f(n))$.

```
for (int i = 0; i < n; i++)
    for (int j = i - 10; j < i; j++)
        for (int k = 1; k < n; k = 4 * k)
            System.out.println(i);
        end for
    end for
end for
```

A. $n \log n$
B. $n^2 \log n$
C. n^2
D. n^3
E. $n \log \log n$