1405 exam review - Lecture notes all

Introduction to Computer Science I (Carleton University)

Program development cycle:

- Design
- Write code
- Correct syntax
- Run
- Correct logic

And then repeat the steps again.

3-step process for programming:

- Input
- Process (of input)
- output

Programs:

- Natural language (human to human)
- Algorithms
- Programming (pseudocode/flowcharts)
- compilation/interpretation (done by python in this case)
- Machine language (binary language- computer to computer)

Ways you CANNOT name variables/functions:

- Can ONLY use numbers and _ or letters
- CAN ONLY START with letters or _

Loops (Always indent statements/body):

- Event/Condition controlled (When you DON'T know how many times to loop)
    - While loop
        - Condition that's tested for true/false and statement that loops as long as condition is true
        - preconditioned : (Condition before body, Body might never be executed) - USE FLAGS
        - Postcondition : (Condition after body, body executed at least once) - USE T/F AND BREAK
- Counter controlled
    - For loop (When you KNOW how many times to loop)
        - For num in list: (If the list was [1,2,3] it makes num a value of 1 then 2 then 3)
        - For num in range(start,stop,step) Or range(start,stop) Or range(stop)
            - **Note**:
                - Stops just before stop value (so stop-1 is actual stop)
                - Step can be -ve which will increment by decreasing value
                - Default start is 0 and default step is 1

Nested loops:

- Loops/loop within loops/loop
- Inner loops go as many times as it's counter plus as many times as outer loop,Innermost loop is the multiplication of all loops

Huffman encoding:

- Write letters with frequency number under it in a line
- Pick two lowest frequency number and draw as one **Note**: smaller number on the right- *lighter* on the *righter*
- Sum of these two numbers are the new number, Repeat steps until only one diagram
- Every left branch gets a 0 and every right gets a 1

Programming paradigms:

- Fundamental practices and approaches to design of computer
    - Imperative/procedural
        - Imperative
            - Incremental changes in program state over time
            - Control structure determines order of execution
            - Computational steps = commands
        - Procedural
            - Same as imperative with addition of
            - Data is organized and combined (after operations are done on it) into *functions* (or procedures)
            - Ex: (Python is imperative/procedural and object oriented)
    - Object-oriented
        - Abstract representation and behaviours
        - Objects (contain data and operators) have attributes and methods
        - Ex:C++, Java, (Python is imperative/procedural and object oriented)
        - Variables with an object are called *Attributes*
        - Functions with an object are called *methods*
        - *Class* species attribute and method of an object (ALWAYS caps, ex: class Ball:

- Functional
  - Evaluation of expression
  - Recursion only, no loops
  - No assignments or global state
  - Ex: Huskell, Lisp
- Logical
  - Set of constraints (instead of commands)
  - Predicate logic and relations
  - Problems viewed as theorems to be proven
  - Ex: Godel,prolog

List (Note: they start from index 0)

- Object that contains multiple data structures
- ex: x=[1,2,'Blah']
- List range
  - x=list(range(5))
- List Index
  - List[0] or list[1] note: will give the value at index 0 or 1 etc
  - List[0] = 'bleh' note: will change the value in 0 position to bleh
- List concatenation
  - x=['hi,'im']
  - y=['michelle']
  - z=x+y
  - print(z)
  - ['hi','im','michelle']
- List slicing
  - Makes a copy of the slice of the list you want
  - List[start:stop] note: Stop would stop just before the number (from 0 doe)
  - List[:stop] note: default start is 0
  - List[start:] note:default stop is end
  - List[:] note:copies entire list

```
list.append(x)     Add item x to the end of the list
list.insert(i, x)  Insert an item at a given position
list.remove(x)     Remove first item with value of x
list.pop(i)        Remove item at i and return it
list.pop()         Remove last item and return it
list.clear()       Remove all items from the list
```

Dictionary:

- Stores collection of data (mapping: use key to locate a value)
- {'chris':'613911','Hazel':'61349'} key is chris, value is 613911 etc
- Keys can be strings ,integer ,floats, tuples etc but not lists
- Values can be anything including lists
- Order doesn't matter/ it's not stored in same order as entered
- Dictionary_name[key] will give you the value stored at that key
- Cannot have duplicate keys
- del dictionary_name[key]
- Case sensitive
- dic_name['Michelle']='female' will create/add to the dictionary if made an empty dictionary first using dic_name={}

Simulation

- Imitating real world process/system over time
- Agent: stimulated entity that uses sensors on the surrounding environment and acts on it with effectors
- Ideal rational agents: expected action (based on current knowledge) =max rewards (from environment)

Codes:

1. ' ' or " " #String
   Use ' ' if " " is inside (ex: print('learn "Hamlet" tmr')
   Use " " if ' ' is inside (ex: print("Don't be late")
2. Print () #Print function
3. # #used to comment
4. Int () #no decimals
5. float() #with decimals
6. Str() #string

7. chr() #converts into ASCII characters
8. type() #gives 'class type' (ex: class type 'int' or class type 'str')
9. + #add
10. + #string concatenation (ex: print("Hi" + "Hello")
11. - #subtract
12. * #multiple
13. ** #exponent
14. / #divide with float as return
15. // #divide with int as return
16. % #divide with remainder as return (also called modulus)
17. end =" " #prints space instead of new line
    print('Hi', end=' ')
    print('Oh',end=' ')
    Hi Oh
18. sep=' ' #separates by item instead of space
    print('Hi','I','Love','You',sep='--')
    Hi--I--Love--You
19. \ #break code
    print('Blah bleh', \
    'bleh')
20. \n #prints each item on a new line
    print('Hakuna\nMatata\n')
    Hakuna
    Matata
21. \t #prints each item after tabbing
    print("Hakuna\tMatata")
    Hakuna Matata
22. format() #Format takes in two parameters, first for value second for format
    format(12345.6789,'.1f')
    12345.7
    Formatted to 1 decimal place and made it a f(float) value
    Using d instead of f returns an int
    Using '%' instead of '.1f' returns a percentage
23. And #Both statements connected are true (returns true or false)
24. Or #one statement connected is true (returns true or false)
25. Not #opposite of statement (returns true or false)
26. == #Equality (returns true or false)
27. = #stores in memory
28. != #not equality (returns true or false)
29. <> #not equality (returns true or false)
30. > #greater than (returns true or false)
31. < #less than (returns true or false)
32. <= #less than or equal to (returns true or false)
33. >= #greater than or equal to (returns true or false)
34. If #Only continues when value is true (always indent)
35. Else #If the if statement isn't true do else
36. Elif #To have if's and else's indented into each other (multiple if's and else's)

Definitions:

- Software requirement: Single task program must perform to satisfy customer
- Algorithm: Set of well defined logical steps to perform a task
  - Finite
  - Ordered set
  - Unambiguous instructions
  - Will eventually terminate
- Pseudocode: means, 'fake' code, used as a model for real code and syntax doesn't count
- Flowcharts: Diagram that depicts steps in a code
  - Ovals: start and end 'terminals'
  - Parallelogram: input/output
  - Rectangle: processes
  - Arrows: connectors

- Programs: algorithms executed by computers
- Command prompt: entry point, allows users to perform tasks without GUI (graphical user interface)
- Functions: Self-contained programs for specific tasks or named collection of statements that perform a task
- Computer: Device that stores and processes data and follows instructions from a computer program
- Charles Babbage: ‘Father of computer’, created mechanical analytic engine
- Ada lovelace: ‘First programmer’, designed program (Bernoulli number sequence) for analytic engine
- Binary: code understood by computers (machine language)
- Binary digit: 0 or 1 (also called bits)
- Byte: eight bits
- ASCII: American standard code for information interchange (converts text into binary)
- Infinite loop: Loops that don't terminate (usually by accident bc forgot to code the termination part)
- Dichotomous key: Expert system of AI used for decision making
- Condition: boolean value to determine if the loop should terminate or continue
- Body: block of code inside the loop to be repeated
- Initialization: Phase before the condition (first code usually)
- Termination: Phase at the end to end/terminate the code
- Input validation: Inspects data to ensure validity using a loop
- Luminance: Brightness calculated by 0.2126R+0.7152G+0.0722B (RGB=red,green,blue)
- Encoding: Changing data from one system of communication to another
- Fixed length coding: Length of code used to represent a symbol is always the same (numeral substitution cipher)
- Variable length coding: Length of code used to rep a symbol is different (Morse code)
- Top-down function: Breaking down an algorithm into many sub functions (top =main)
- Variable scope: part of the code that can see the variable or call it (only after variable)
- Local variable: Variable defined within a function (can only be seen by that function)
- Global variable: Variable defined outside a function (can be seen by every function)
- Argument: ACTUAL code passed into the parameters of a function ex: main(argument): so main(2):
- Parameters: The thing you NAME the code you wanna pass into the function. Ex: function(parameter): so function(number):
- Tuples: exactly like lists but values cannot be changed and uses () brackets, has index, +,* etc but not insert,remove,append
- String: Same like lists, with slicing,min,max,+,len,in and index only
- Mean: sum of all values divided by number of values
- Median: middle value
- Element: each value in a list
- Recursive function: A function calls itself
- Base case: easiest possible case in recursion