School Work Summary and Samples

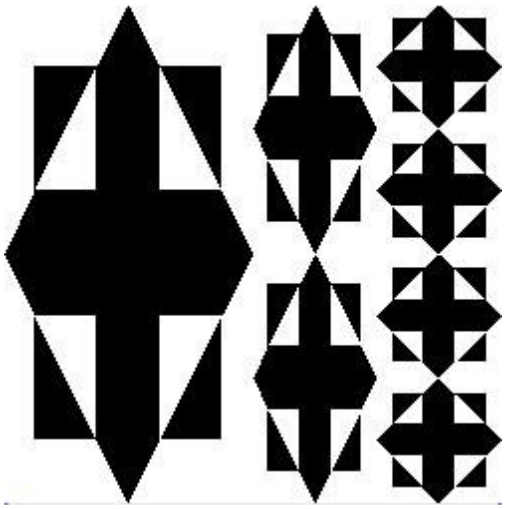
1) NUS' CS1010s: Programming Methodology (Python)

- Learnt basic coding/Python such as:
  - Data Types
  - Strings
  - Numbers & Operators
  - Data Structures
  - If…Else
  - Recursion & Iteration
  - Classes /Objects/Inheritance
  - Functions, Lambda
  - OOP
  - Try…Except

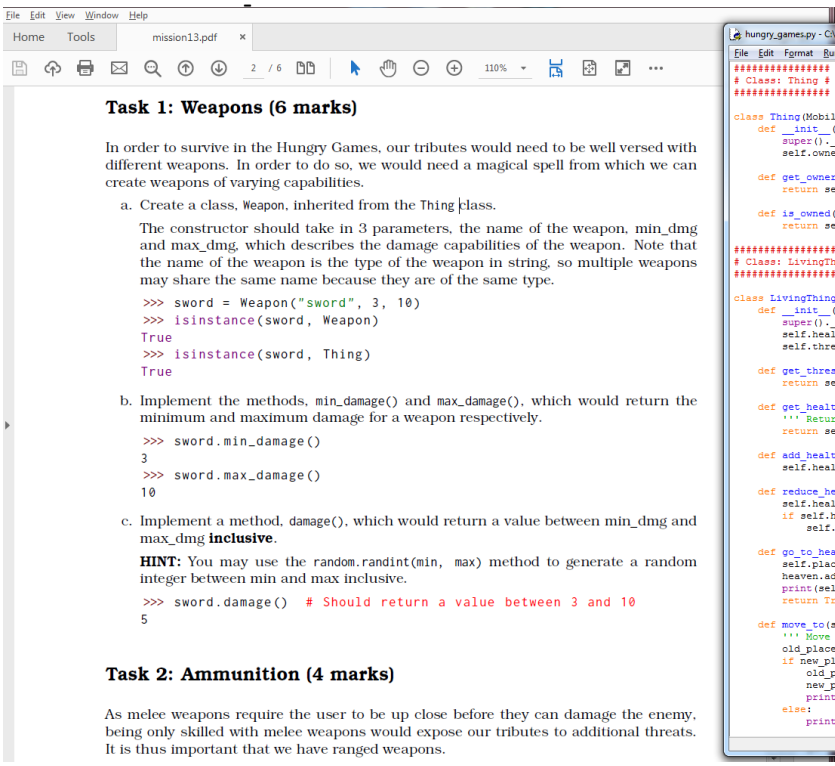
- Some sample school work:

Example 1) Learning Recursion & Iteration with graphics

Write a function fractal that takes as arguments a rune and an integer n > 0. It should generate the rune below by means of a **recursive** process with the following command:
show (fractal (make_cross (rcross_bb), 3))



Example 2) Learning Classes/Objects/Inheritance through game creation



File Edit View Window Help

Home Tools mission13.pdf ×

2 / 6 110%

Task 1: Weapons (6 marks)

In order to survive in the Hungry Games, our tributes would need to be well versed with different weapons. In order to do so, we would need a magical spell from which we can create weapons of varying capabilities.

a. Create a class, `Weapon`, inherited from the `Thing` class.

The constructor should take in 3 parameters, the name of the weapon, min_dmg and max_dmg, which describes the damage capabilities of the weapon. Note that the name of the weapon is the type of the weapon in string, so multiple weapons may share the same name because they are of the same type.

```
>>> sword = Weapon("sword", 3, 10)
>>> isinstance(sword, Weapon)
True
>>> isinstance(sword, Thing)
True
```

b. Implement the methods, `min_damage()` and `max_damage()`, which would return the minimum and maximum damage for a weapon respectively.

```
>>> sword.min_damage()
3
>>> sword.max_damage()
10
```

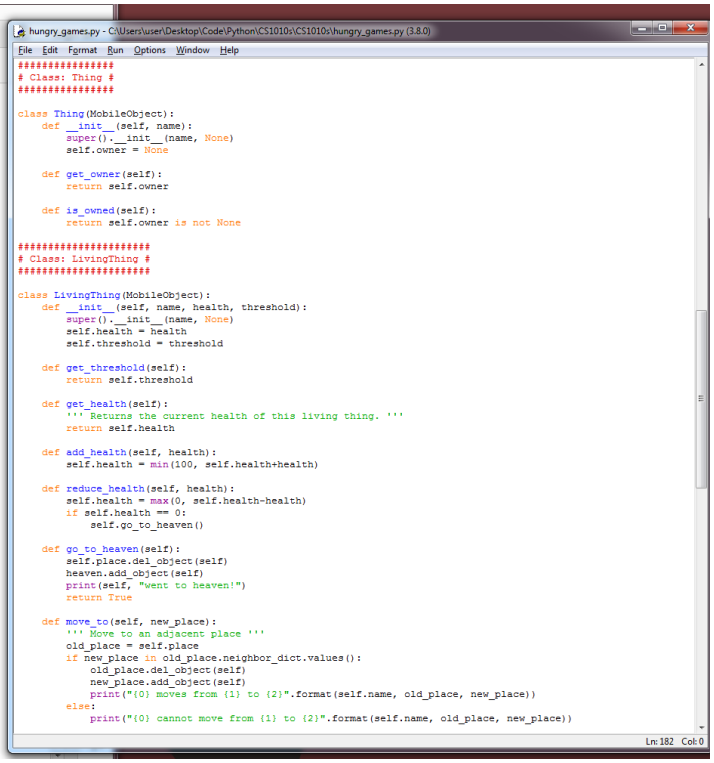
c. Implement a method, `damage()`, which would return a value between min_dmg and max_dmg **inclusive**.

HINT: You may use the `random.randint(min, max)` method to generate a random integer between min and max inclusive.

```
>>> sword.damage()  # Should return a value between 3 and 10
5
```

Task 2: Ammunition (4 marks)

As melee weapons require the user to be up close before they can damage the enemy, being only skilled with melee weapons would expose our tributes to additional threats. It is thus important that we have ranged weapons.



hungry_games.py - C:\Users\user\Desktop\Code\Python\CS1010s\CS1010s\hungry_games.py (3.8.0)

File Edit Format Run Options Window Help

```
################
# Class: Thing #
################

class Thing(MobileObject):
    def __init__(self, name):
        super().__init__(name, None)
        self.owner = None

    def get_owner(self):
        return self.owner

    def is_owned(self):
        return self.owner is not None

#####################
# Class: LivingThing #
#####################

class LivingThing(MobileObject):
    def __init__(self, name, health, threshold):
        super().__init__(name, None)
        self.health = health
        self.threshold = threshold

    def get_threshold(self):
        return self.threshold

    def get_health(self):
        ''' Returns the current health of this living thing. '''
        return self.health

    def add_health(self, health):
        self.health = min(100, self.health+health)

    def reduce_health(self, health):
        self.health = max(0, self.health-health)
        if self.health == 0:
            self.go_to_heaven()

    def go_to_heaven(self):
        self.place.del_object(self)
        heaven.add_object(self)
        print(self, "went to heaven!")
        return True

    def move_to(self, new_place):
        ''' Move to an adjacent place '''
        old_place = self.place
        if new_place in old_place.neighbor_dict.values():
            old_place.del_object(self)
            new_place.add_object(self)
            print("{0} moves from {1} to {2}".format(self.name, old_place, new_place))
        else:
            print("{0} cannot move from {1} to {2}".format(self.name, old_place, new_place))
```

Ln: 182 Col: 0

2) NUS' CS2040: Data Structures & Algorithms (JS)

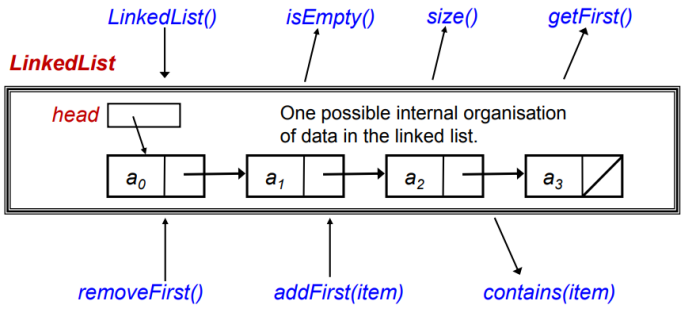
- Learnt:
  - Basic JavaScript
  - Data Structures (Linked lists, stacks, queues, hash tables, binary heaps, trees, graphs)
  - Searching & Sorting Algorithm
  - Basic analysis of algorithms

- Some sample school work:

Example 1) Learning about different Data Structures

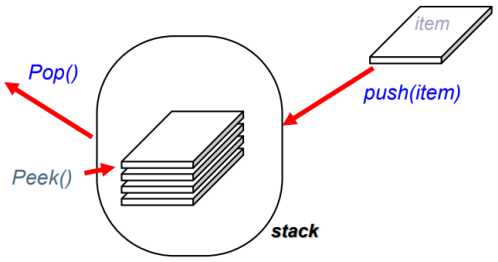
4. ADT of a Linked List (1/3)

- We explore different implementations of Linked List ADT
- For example, if one such implementation is called LinkedList:



1 Stack ADT: Operations

- A Stack is a collection of data that is accessed in a last-in-first-out (LIFO) manner
- Major operations: "push", "pop", and "peek".



Example 2) Quizzes

Hello World!

"Hello World" is a typical phrase used to indicate a beginner program written in a particular language to introduce it to new learners. However, do not let the problem title fool you. This is not a simple "print Hello World" problem.

This problem will introduce you to three different input-parsing methods that will be used during the course of the semester. It is important that you are familiar with the three different input-parsing methods. These methods will be used in the labs including sit-in labs, take-home labs and PE for CS1020 and CS2040.

Of course, there are more than three methods to parse inputs. The three given to you are just simple examples that you will encounter in this module. In this problem, you will encounter three basic techniques of parsing input, which includes the following methods:

1. Given an integer N, you should read N lines, each containing some information.
2. Read until you encounter a special character (e.g. read until -1 or 0).
3. Read until the very end of the file (i.e. read all lines in the input file).

In order to exercise the three different input-parsing methods, we first need to have an actual "problem" to solve. This is a simple logic problem. Given a query consisting of two bits (either "1" or "0") and a logical operator ("AND" or "OR"), output the result of the operation between the two given bits.

Input
The first line of input consists of a single integer, which can be either "1", "2", and "3". This integer represents the method of parsing inputs that you need to use, corresponding to the methods explained in the problem description above. For this problem, we use the special character "0" for method 2. Each query will have a logical operator and two bits, each separated by a single space.

Output
For each query, output the result. Your last line of output must contain a newline character.

Sample Input 1
```
1
2
AND 1 0
OR 0 1
```

Sample Output 1
```
0
1
```

Sample Input 2
```
2
AND 1 1
OR 1 0
AND 1 0
0
```

Sample Output 2
```
1
1
0
```

Sample Input 3
```
3
AND 1 1
OR 1 0
```

Sample Output 3
```
1
1
```

Censorship

Mr. Panda wants to read a short story to his kids. However, he realises that the story contains several bad words that he wants to censor from his kids. Can you help Mr. Panda do so?

Input
The input contains 3 lines.
The first line contains a single integer **N**, the number of bad words that Mr. Panda wants to censor.

The second line contains **N** words that Mr. Panda wants to censor. All the words will only contain English letters and there will be **exactly 1 space** in between these words.

The third line contains the story that Mr. Panda wants to censor. They will consist of a sequence of words that only contain English letters. There will also be **exactly 1 space** in between words.

Output
Print the entire story with the bad words censored.
The censored words should be replaced with '*' characters, with the number of characters equal to the length of the word.
You only need to censor entire words, so for example if "bad" is a bad word, you should not censor "Sinbad".
However, you must censor words regardless of case so for example if "bad" is a bad word, you must censor "BaD".

Limits
- $1 \leq \mathbf{N} \leq 10$
- Each censored word will not be more than 30 characters long.
- The story will contain between 1 and 10000 characters.
- All the words will only contain English letters and there will be **exactly 1 space** in between words.

Sample Input (**censorship1.in**)	Sample Output (**censorship1.out**)
4 code debug cat panda Mr Panda and Rar the Cat like to code	Mr ***** and Rar the *** like to ****
Sample Input (**censorship2.in**)	Sample Output (**censorship2.out**)
4 code debug cat panda but they do not like debugging	but they do not like debugging