CSCI3180 – Principles of Programming Languages – Spring 2019

Assignment 2 — Your First Date with Python and Duck Typing

Deadline: Mar 10, 2019 (Sunday) 23:59

1 Introduction

The purpose of this assignment is to offer you the first experience with Python, which supports the object-oriented programming paradigm. Our main focuses are Dynamic Typing and Duck Typing. Please use **Python 3.6** to finish this assignment.

The assignment consists of 4 tasks.

- You need to implement a famous board game called Othello in Python.
- You are asked to demonstrate the advantages/disadvantages of dynamic typing through some example code.
- We give you the JAVA source code of a game called "The journey". You need to reimplement the game in Python to make the code clean with the help of Duck Typing.
- Additional features and mechanisms are introduced to the "The journey" game. You need to implement the enhanced game in both JAVA and Python.

After completing the 4 tasks, you need to write a report elaborating on Dynamic Typing and Duck Typing.

NOTES: all your codes will be graded on the Linux machines in the Department. You are welcome to develop your codes on any platform, but please remember to test them on Department machines.

2 Task 1: Othello

This is a small programming exercise for you to get familiar with Python, which is a dynamically typed language. In this task, you have to *strictly follow* our proposed OO design and the game rules for "Othello" stated in this section. For the OO design, you have to follow the prototypes of the given functions exactly, but can choose to add new member functions. We provide a simple skeleton for your convenience.

2.1 Description

Othello, also known as Reversi, is a classic board game that involves two players. Two players take turns in adding their own pieces (discs) in order to flip the opponent's discs. The game ends when both players cannot add any more discs onto the board. The player who has more discs on the board wins the game. It is a tie when the disc numbers of both sides are the same.

2.1.1 Game Rules

The game consists of an 8 $\times$ 8 game board with 64 cells. Each cell can hold a disc with two sides: black and white. In our project, we represent the two sides as follows:

- Black side - character O
- White side - character X

Players place their discs alternatively on the game board, one using the black side and the other using the white side. When you place a disc, any opponent's disc in succession in between your new disc and another one of your disc will be flipped over to take your color.

As shown in Figure 1, suppose you are the black-disc player and the board on the left shows the current condition. On the right, the discs with yellow border show the four possible locations for your next disc, since you can then flip some of your opponent's discs.

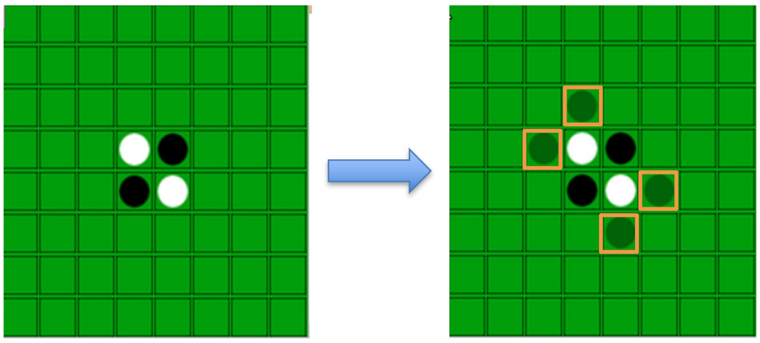
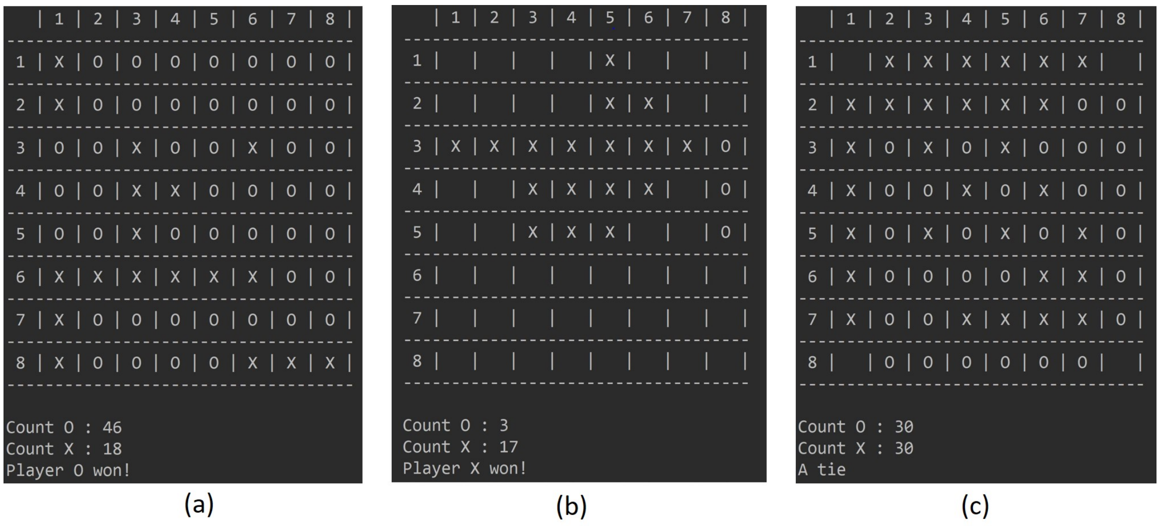


Figure 1: Possible moves for Othello

In each turn, a player must place a disc, so that at least one of the opponent's discs is flipped. If a player cannot find any location to place a disc, the player has to give up the turn without placing any disc. A game ends when:

- either the board is full, or
- no one can make a move, i.e., place a disc.

As shown in Figure 2(a)&(b), the player having more discs of his/her color is the winner. The game is a tie if the number of white discs equals the number of black discs. Here is a web-based online Othello game that you can try: `https://www.coolmath-games.com/0-reversi`.



(a)

```
   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
-------------------------------------
 1 | X | O | O | O | O | O | O | O |
-------------------------------------
 2 | X | O | O | O | O | O | O | O |
-------------------------------------
 3 | O | O | X | O | O | X | O | O |
-------------------------------------
 4 | O | O | X | X | O | O | O | O |
-------------------------------------
 5 | O | O | X | O | O | O | O | O |
-------------------------------------
 6 | X | X | X | X | X | X | O | O |
-------------------------------------
 7 | X | O | O | O | O | O | O | O |
-------------------------------------
 8 | X | O | O | O | O | X | X | X |
-------------------------------------

Count O : 46
Count X : 18
Player O won!
```

(b)

```
   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
-------------------------------------
 1 |   |   |   |   | X |   |   |   |
-------------------------------------
 2 |   |   |   |   | X | X |   |   |
-------------------------------------
 3 | X | X | X | X | X | X | X | O |
-------------------------------------
 4 |   |   | X | X | X | X |   | O |
-------------------------------------
 5 |   |   | X | X | X |   |   | O |
-------------------------------------
 6 |   |   |   |   |   |   |   |   |
-------------------------------------
 7 |   |   |   |   |   |   |   |   |
-------------------------------------
 8 |   |   |   |   |   |   |   |   |
-------------------------------------

Count O : 3
Count X : 17
Player X won!
```

(c)

```
   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
-------------------------------------
 1 |   | X | X | X | X | X | X |   |
-------------------------------------
 2 | X | X | X | X | X | X | O | O |
-------------------------------------
 3 | X | O | X | O | X | O | O | O |
-------------------------------------
 4 | X | O | O | X | O | X | O | O |
-------------------------------------
 5 | X | O | X | O | X | O | X | O |
-------------------------------------
 6 | X | O | O | O | O | X | X | O |
-------------------------------------
 7 | X | O | O | X | X | X | X | O |
-------------------------------------
 8 |   | O | O | O | O | O | O |   |
-------------------------------------

Count O : 30
Count X : 30
A tie
```

Figure 2: (a) Player O has won with a fully filled chessboard; (b) Player X has won with a partially filled chessboard; (c) A tie;

2.2 Game board

The game starts with two black discs and two white discs at the following locations:

- Character O at (4,5) and (5,4)
- Character X at (4,4) and (5,5)

Other cells are just filled with a single SPACE character. The game always starts with the black player ('O').

2.3 Input/Output Specification

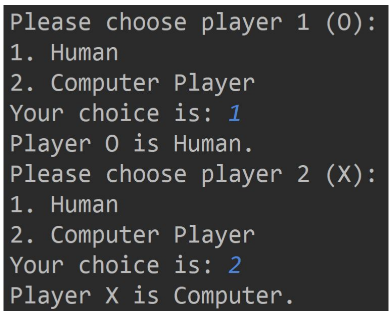
The input/output specification is detailed in this section.

2.3.1 Input Specification

In this exercise, you are required to use the command line to get input information. Since this is a simple program, there are just two operations requiring user-input.

Type of Players

Users can choose Player X and Player O to be either human- or computer-controlled. You have to use the command line to request for the types of the two players before starting the game (as shown in Figure 3).



```
Please choose player 1 (O):
1. Human
2. Computer Player
Your choice is: 1
Player O is Human.
Please choose player 2 (X):
1. Human
2. Computer Player
Your choice is: 2
Player X is Computer.
```

Figure 3: Players' types

Human-Controlled Player's Move

Human-controlled players require user-inputs to determine their next moves. When it is human-controlled player's turn during the game, your program should request for the row and column of the next move. The input should be integers from 1 to 8. During each turn, the system will first check whether there is a legal move for current player. If not, the system will switch to another player's turn automatically, as shown in Figure 4(a). Moreover, any invalid input or moves should be forbidden and the user is requested for input again until a valid move is given. See Figure 4(b) for an example.

Computer-Controlled Player

A computer-controlled player just randomly chooses an unfilled cell to make a *valid* move. During each turn, the system will first check whether there is a legal move for current player. If not, the system will switch to another player's turn automatically.

2.4 Python Classes

Please follow the classes `Othello`, `GameBoard`, `Player`, `Human` and `Computer` defined below in your Python implementation. You are free to add other variables, methods or classes. We would start the program by running the command: `python Othello.py`

1. **Class `Othello`**

   A game object which holds the game board, players' turns and controls game logic. You have to implement it with the following components:

   - **Instance Variable(s)**

     `gameBoard`

     - This is an object variable representing the game board.

     `player1`

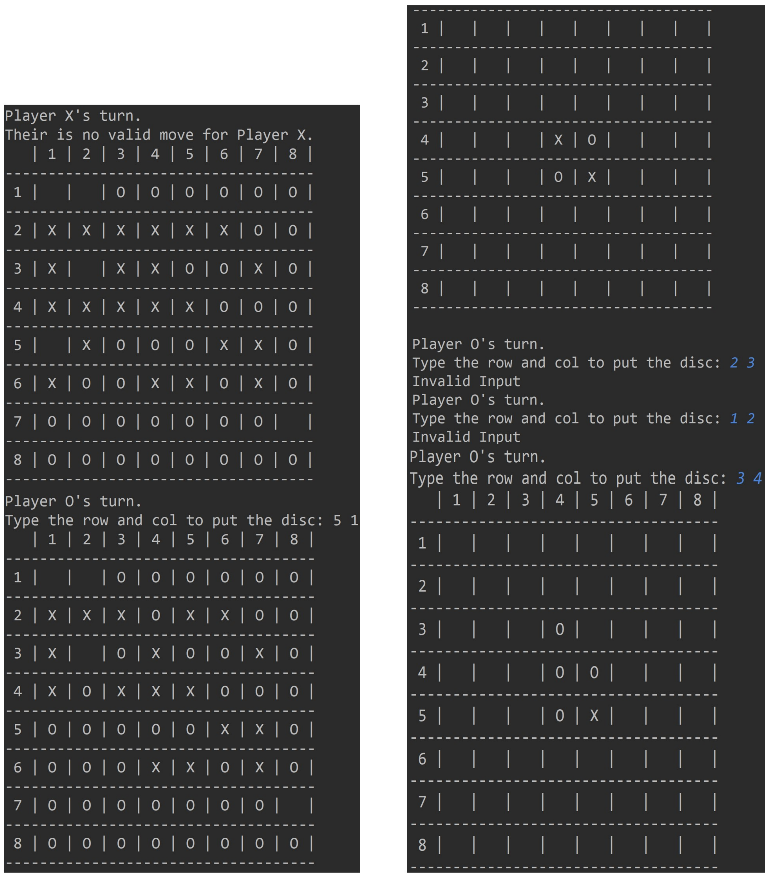
     - This is an object variable representing Player O.

     `player2`

     - This is an object variable representing Player X.

     `turn`

- This is an integer variable (*0 for player1, 1 for player2*) indicating which player should play in the current turn.



```
Player X's turn.
Their is no valid move for Player X.
   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
-----------------------------------
 1 |   |   | O | O | O | O | O | O |
-----------------------------------
 2 | X | X | X | X | X | X | O | O |
-----------------------------------
 3 | X |   | X | X | O | O | X | O |
-----------------------------------
 4 | X | X | X | X | X | O | O | O |
-----------------------------------
 5 |   | X | O | O | O | X | X | O |
-----------------------------------
 6 | X | O | O | X | X | O | X | O |
-----------------------------------
 7 | O | O | O | O | O | O | O |   |
-----------------------------------
 8 | O | O | O | O | O | O | O | O |
-----------------------------------
Player O's turn.
Type the row and col to put the disc: 5 1
   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
-----------------------------------
 1 |   |   | O | O | O | O | O | O |
-----------------------------------
 2 | X | X | X | O | X | X | O | O |
-----------------------------------
 3 | X |   | O | X | O | O | X | O |
-----------------------------------
 4 | X | O | X | X | X | O | O | O |
-----------------------------------
 5 | O | O | O | O | O | X | X | O |
-----------------------------------
 6 | O | O | O | X | X | O | X | O |
-----------------------------------
 7 | O | O | O | O | O | O | O |   |
-----------------------------------
 8 | O | O | O | O | O | O | O | O |
-----------------------------------
```

(a)

```
-----------------------------------
 1 |   |   |   |   |   |   |   |   |
-----------------------------------
 2 |   |   |   |   |   |   |   |   |
-----------------------------------
 3 |   |   |   |   |   |   |   |   |
-----------------------------------
 4 |   |   |   | X | O |   |   |   |
-----------------------------------
 5 |   |   |   | O | X |   |   |   |
-----------------------------------
 6 |   |   |   |   |   |   |   |   |
-----------------------------------
 7 |   |   |   |   |   |   |   |   |
-----------------------------------
 8 |   |   |   |   |   |   |   |   |
-----------------------------------

Player O's turn.
Type the row and col to put the disc: 2 3
Invalid Input
Player O's turn.
Type the row and col to put the disc: 1 2
Invalid Input
Player O's turn.
Type the row and col to put the disc: 3 4
   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
-----------------------------------
 1 |   |   |   |   |   |   |   |   |
-----------------------------------
 2 |   |   |   |   |   |   |   |   |
-----------------------------------
 3 |   |   |   | O |   |   |   |   |
-----------------------------------
 4 |   |   |   | O | O |   |   |   |
-----------------------------------
 5 |   |   |   | O | X |   |   |   |
-----------------------------------
 6 |   |   |   |   |   |   |   |   |
-----------------------------------
 7 |   |   |   |   |   |   |   |   |
-----------------------------------
 8 |   |   |   |   |   |   |   |   |
-----------------------------------
```

(b)

Figure 4: Example gameplay output. (a) No legal movement; (b) Invalid inputs

- **Instance Method(s)**

  `__init__(self)`

  - Create a game board and initialize other instance variables.

  `createPlayer(self,symbol,playerNum)`

  - Create a player (human- or computer-controlled) with the corresponding symbol ('O' or 'X') and player number (1 or 2) by prompting the user.

  `startGame(self)`

  - Create two players, initialize the game board, and then start a new game and play until winning/losing or draw.

2. **Class** `GameBoard`

   You have to implement it with the following components:

   - **Instance Variable(s)**

     `board`

     - This is an 8×8 two-dimensional character array representing the game board.

   - **Instance Method(s)**

     `__init__(self)`

     - Define all instance variables.

`init_gameBoard(self)`
- Initialize the board. Its initial state is referred to Section 2.2.

`execute_flip(self,pos,symbol)`
- Flip over all opponent's symbols in between given position and symbol.

`check_legal_move(self,symbol)`
- Check whether their is a legal move given symbol.

`check_ending(self)`
- Check if the game is over.

`check_winner(self)`
- Count the number for each symbol and return a list, like [17,3]. The first one is the total number of character 'O' and the second one is for character 'X'.

3. **Class** `Player`

   An superclass representing a player object. You have to implement it with the following components:

   - **Instance Variable(s)**

     `playerSymbol`
     - This is a variable indicating the symbol of the player (X or O).

   - **Instance Method(s)**

     `__init__(self, symbol)`
     - Initialize the player with its symbol X or O.

     `nextMove(self, board)`
     - A method to be implemented in subclasses, which returns a list, like [1,2]. The first one is the row and the second one is the column (1-8 for each) of the next valid move. The parameter *board* represents a two-dimensional character array that represents the cell conditions in the game board and your function should not modify the array contents.

4. **Class** `Human` and `Computer`

   Classes extending the superclass `Player` to represent a human- and a computer-controlled player object respectively.

2.5 Tournament

For this bonus part, you can implement a more brilliant AI player to compete with others. We will arrange for an automated tournament environment to test your implementations. If you want to join the tournament, you should submit an additional Python file, named `AI_[your SID].py`, like `AI_1155123456.py`, and the class name is "AI", which is also extended from the class `Player`. The AI strategy should be built in the instance method `nextMove(self,board)`:

- The basic requirements have already been mentioned above.
- The function must produce an output within one second on the lab's computers.
- You need to test carefully to make sure your code could not damage our test system, e.g., the array out of bounds exception!!!

Please note that if your program does not follow any one of the above requirements, you cannot earn extra points for this project. We will run a round robin tournament. A win is worth 3 points and a tie is worth 1 point, while a loss score 0 points. The top five entire will receive additional marks for the assignment: 30, 25, 20, 15, 10, 5 respectively.

3 Task 2: Demonstrating Advantages of Dynamic Typing

There are commonly claimed advantages of Dynamic Typing:

1. More generic code can be written. In other words, functions can be defined to apply on arguments of different types.

2. Possibilities of mixed type collection data structures.

Please provide concise example codes to demonstrate the advantages of Dynamic Typing mentioned above. You are welcome to provide other advantages and disadvantages along with code segment for extra bonus points.

Dynamic Typing makes coding more flexible and convenient, but you should bear in mind that type checking can only be carried out at runtime, incurring time overhead and reliability issues.

4 Task 3: The Journey

This task is also to build a game. A Java program of the game is given. You need to understand its behavior and re-implement it with Python. After finishing this task, you will have experienced a special feature of Python called Duck Typing, which is possible only in a dynamically typed language. And you will see a difference between Java and Python, the former of which does not support Duck Typing.

4.1 Duck Typing

The following synopsis of Duck Typing is summarized from:

`http://en.wikipedia.org/wiki/Ducktyping`
`http://www.sitepoint.com/making-ruby-quack-why-we-love-duck-typing`

In standard statically-typed object-oriented languages, objects' classes (which determine an object characteristic and behavior) are essentially interpreted as the objects' types. Duck Typing is a new typing paradigm in dynamically-typed (late binding) languages that allow us to dissociate typing from objects' characteristics. It can be summarized by the following motto by the late poet James Whitcombe Riley:

> *When I see a bird that walks like a duck and swims like a duck and quacks like a duck,*
> *I call that bird a duck.*

The basic premise of Duck Typing is simple. If an entity looks, walks, and quacks like a duck, for all intents and purposes it is fair to assume that one is dealing with a member of the species anas platyrhynchos. In practical Python terms, this means that it is possible to try calling any method on any object, regardless of its class.

An important advantage of Duck Typing is that we can enjoy *polymorphism without inheritance.* An immediate consequence is that we can write more generic codes, which are cleaner and more precise.

4.2 Background

In the world of Kafustrok, most of the land is covered in dense forest, populated by bizarre monsters and a small number of elves. The only safe haven is a small number of villages where humans live. In order to protect the villages from monster attacks, a selected warrior set foot on the forest to wipe out all monsters. Every monster has its own territory, and the elves also live in some specific areas. The warrior can seek help from the elves to heal himself, but he needs to use a certain amount of magic crystals in exchange.

4.3 Task Description

At the beginning, there exists a warrior, e elves and m monsters. All of them will be distributed in different positions.

- At each iteration, the warrior will try to teleport to a targeted location.
  - If meeting a monster, the monster will talk and ask the warrior whether to fight or not.
    - Yes, if his health is sufficient to kill this monster, with losing certain health, he will occupy this position and get some magic crystals. Otherwise, he will be killed.
    - No, enter next iteration.
  - If meeting an elf, the elf will talk and ask the warrior whether to make a deal or not.
    - Yes, he will pay some magic crystals and his health increases with the elf's magical power, but cannot exceed the health cap.
    - No, enter next iteration.
  - If nothing is encountered, he will end up in that position.
- If all monsters have been killed or the warrior is dead, the game will be over

You should read and execute the given Java program to understand its behavior and design. Please replicate all the behavior of the given Java Program in Python with Duck Typing, following the same class design. You should not introduce extra instance variables or instance methods. Your program should run by calling python `the_journey.py`. You will also be evaluated on your programming style.

5 Task 4: More Warriors!

A warrior falls but more warriors stand up. The warriors from different villages form an alliance and join the journey to destroy the monsters together. Based on the alliance's agreement, a warrior needs to share some crystals if the other has a request. Moreover, some life potions appear in the forest, but their positions always change from time to time, and the warriors can get treatment if they find the potion. The game still ends when all monsters are eliminated or all warriors are dead.

5.1 Task Description

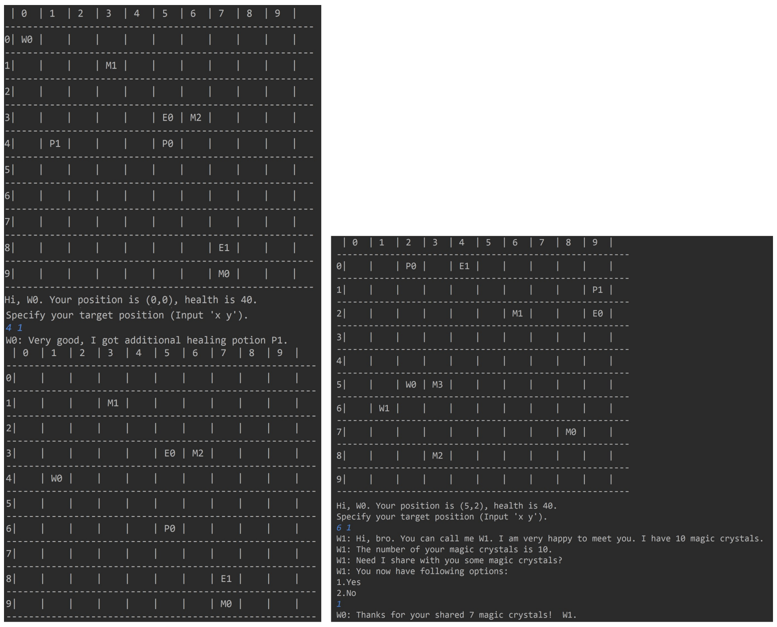
At the beginning, there exists w warriors, e elves, m monsters and p life potions. All of them will be distributed in different positions.

- *At each iteration, all life potions will teleport into different positions randomly.*
- At each iteration, all warriors will try to teleport to a targeted location.
- For each warrior,
  - If meeting a monster, the monster will talk and ask the warrior whether to fight or not.
    - Yes, if his health is sufficient to kill this monster, with losing certain health, he will occupy this position and get some magic crystals. Otherwise, he will be killed.
    - No, enter next iteration.
  - If meeting an elf, the elf will talk and ask the warrior whether to make a deal or not.
    - Yes, he will pay some magic crystals and his health increases with the elf's magical power, but cannot exceed the health cap.
    - No, enter next iteration.
  - *If meeting a life potion, the warrior will occupy the position and have some health increased directly. That life potion will disappear.*
  - *If meeting another warrior, then the other warrior will talk and ask the warrior whether he needs more crystals*

        ∗ *Yes, he will get some magic crystals for free.*
        ∗ *No, enter next iteration.*
    – If nothing is encountered, he will end up in that position.

- If all monsters or warriors have been killed, the game will be over

You are now required to modify/extend both the Java and Python implementation of Task 3. You will also be evaluated on your programming style.



(a)

```
 | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
---------------------------------------------------
0| W0 |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
1|    |    | M1 |    |    |    |    |    |    |    |
---------------------------------------------------
2|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
3|    |    |    |    |    | E0 | M2 |    |    |    |
---------------------------------------------------
4|    | P1 |    |    |    | P0 |    |    |    |    |
---------------------------------------------------
5|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
6|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
7|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
8|    |    |    |    |    |    |    | E1 |    |    |
---------------------------------------------------
9|    |    |    |    |    |    |    | M0 |    |    |
---------------------------------------------------
Hi, W0. Your position is (0,0), health is 40.
Specify your target position (Input 'x y').
4 1
W0: Very good, I got additional healing potion P1.
 | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
---------------------------------------------------
0|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
1|    |    | M1 |    |    |    |    |    |    |    |
---------------------------------------------------
2|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
3|    |    |    |    |    | E0 | M2 |    |    |    |
---------------------------------------------------
4|    | W0 |    |    |    |    |    |    |    |    |
---------------------------------------------------
5|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
6|    |    |    |    |    | P0 |    |    |    |    |
---------------------------------------------------
7|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
8|    |    |    |    |    |    |    | E1 |    |    |
---------------------------------------------------
9|    |    |    |    |    |    |    | M0 |    |    |
---------------------------------------------------
```

(b)

```
 | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
---------------------------------------------------
0|    |    | P0 |    | E1 |    |    |    |    |    |
---------------------------------------------------
1|    |    |    |    |    |    |    |    |    | P1 |
---------------------------------------------------
2|    |    |    |    |    |    | M1 |    |    | E0 |
---------------------------------------------------
3|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
4|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
5|    |    | W0 | M3 |    |    |    |    |    |    |
---------------------------------------------------
6|    | W1 |    |    |    |    |    |    |    |    |
---------------------------------------------------
7|    |    |    |    |    |    |    |    | M0 |    |
---------------------------------------------------
8|    |    |    | M2 |    |    |    |    |    |    |
---------------------------------------------------
9|    |    |    |    |    |    |    |    |    |    |
---------------------------------------------------
Hi, W0. Your position is (5,2), health is 40.
Specify your target position (Input 'x y').
6 1
W1: Hi, bro. You can call me W1. I am very happy to meet you. I have 10 magic crystals.
W1: The number of your magic crystals is 10.
W1: Need I share with you some magic crystals?
W1: You now have following options:
1.Yes
2.No
1
W0: Thanks for your shared 7 magic crystals!  W1.
```

Figure 5: Example gameplay output. (a) when meet a life potion; (b) when meet another warrior

6 Report

Your simple report should answer the following questions within TWO A4 pages.

1. Providing example code and necessary elaborations for demonstrating advantages of Dynamic Typing as specified in Task 2.

2. Using codes for Task 3, give two scenarios in which the Python implementation is better than the Java implementation. Given the reasons.

3. Using codes for Task 4, illustrate further advantages of Dynamic Typing and Duck Typing.

7 Submission Guidelines

Please read the guidelines CAREFULLY. If you fail to meet the deadline because of submission problem on your side, marks will still be deducted. So please start your work early!

1. Your submissions will be accepted latest by 11:59 p.m. on Mar 13, but submissions made after the original deadline would be considered as **LATE** submissions and penalties will be imposed in the following manner:

    - Late submissions before 11:59 p.m. on Mar 12: marks will be deducted by 20%
    - Late submissions before 11:59 p.m. on Mar 13: marks will be deducted by 50%

2. In the following, **SUPPOSE**

your name is *Chan Tai Man*,
your student ID is *1155234567*,
your username is *tmchan*, and
your email address is *tmchan@cse.cuhk.edu.hk*.

3. In your source files, insert the following header. REMEMBER to insert the header according to the comment rule of Python.

```
/*
 * CSCI3180 Principles of Programming Languages
 *
 * --- Declaration ---
 *
 * I declare that the assignment here submitted is original except for source
 * material explicitly acknowledged.  I also acknowledge that I am aware of
 * University policy and regulations on honesty in academic work, and of the
 * disciplinary guidelines and procedures applicable to breaches of such policy
 * and regulations, as contained in the website
 * http://www.cuhk.edu.hk/policy/academichonesty/
 *
 * Assignment 2
 * Name : Chan Tai Man
 * Student ID : 1155234567
 * Email Addr : tmchan@cse.cuhk.edu.hk
 */
```

   The sample file header is available at

   `http://course.cse.cuhk.edu.hk/~csci3180/resource/header.txt`

4. *Task 2 report and the final report should be merged into one report file* and submitted to VeriGuide, which will generate a submission receipt. The report should be named "report.pdf". The VeriGuide receipt of report should be named "receipt.pdf". The report and receipt should be submitted together with codes in the same ZIP archive.

5. `Tar` your source files to `username.tar` by

```
tar cvf tmchan.tar Othello.py task3_python.zip task4_java.zip \
task4_python.zip report.pdf receipt.pdf
```

6. `Gzip` the `tarred` file to `username.tar.gz` by

```
gzip tmchan.tar
```

7. `Uuencode` the `gzipped` file and send it to the course account with the email title "HW2 *studentID yourName*" by

```
uuencode tmchan.tar.gz tmchan.tar.gz \
| mailx -s "HW2 1155234567 Chan Tai Man" csci3180@cse.cuhk.edu.hk
```

8. Please submit your assignment using your Unix accounts.

9. An acknowledgement email will be sent to you if your assignment is received. **DO NOT** delete or modify the acknowledgement email. You should contact your TAs for help if you do not receive the acknowledgement email within 5 minutes after your submission. **DO NOT** re-submit just because you do not receive the acknowledgement email.

10. You can check your submission status at

    `http://course.cse.cuhk.edu.hk/~csci3180/submit/hw2.html`.

11. You can re-submit your assignment, but we will only grade the latest submission.

12. Enjoy your work :>