CSCI3180 – Principles of Programming Languages – Spring 2017

Assignment 2 — Your First Date with Python

Deadline: Mar 12, 2017 (Sunday) 23:59

1 Introduction

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

The assignment consists of two main parts, which are further divided into four tasks. First, you need to implement the rules of a famous game called "Connect-4" in Python. Also, you need to design a simple computer player of the game which can at least generate valid moves. Of course, you can earn bonus points by implementing a sophisticated AI player. Second, a Java implementation of a future hotel scenario is given. You need to understand the Java program and re-implement it in Python with Duck Typing. In the process, you will learn to appreciate the flexibility of Duck Typing. All Python implementations in this assignment should be built on Python 2.7.

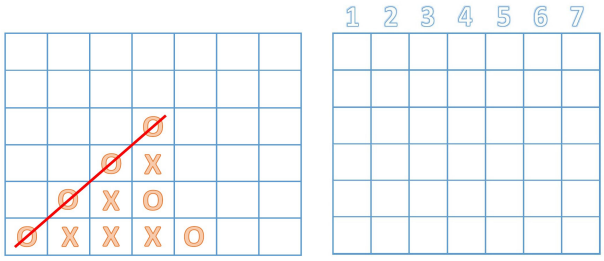
2 Task 1: Connect-4

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 "Connect-4" 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.

Figure 1: Connect-4

2.1 Description

The Connect-4 (Figure 1) game is a two-player game in which each player is associated with a symbol to stand for his/her discs and then takes turns dropping the discs from the top into one of the seven columns, each with six rows. Discs stack on top of others. We mark the two types



(a) Player O wins (b) Indices of columns

Figure 2: Connect-4 game board

of discs by "O" and "X" respectively. The pieces fall straight down, occupying the next available space within a column. The objective of the game is to connect four of one's own discs of the same symbol vertically, horizontally, or diagonally before your opponent. If no one win the game until all cells are filled, the game ends in a tie. Figure 2(a) shows a game won by Player O.

A column is *unfilled* if it contains less than 6 discs. A player's move is *valid* if the specified column exists and is unfilled.

2.2 Types of Players

The two players, Player X and Player O, can be controlled by either human or computer. Users can choose the types (human or computer) of Player X and Player O before the game starts. Player O starts first.

Human-Controlled Player

A human-controlled player needs a human user to provide instruction for each move. In each turn, the symbol of the human-controlled player will be put in an unfilled column as specified by the human player. The input is an integer from 1 to 7 (as shown in Figure 2(b)).

Computer-Controlled Player

A computer-controlled player just randomly chooses an unfilled column to make a valid move.

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 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 command line to request for the types of the two players before starting the game (as shown in Figure 3).

```
Please choose the first player:
1.Computer
2.Human
Your choice is:1
Player O is Computer
```

```
Please choose the second player:
1.Computer
2.Human
Your choice is:2
Player X is Human
```

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 column of the next move. The input should be an integer from 1 to 7. Any invalid input or moves should be forbidden and the user is requested for input again until a valid move is given.

Figure 4: Output shown in the command line window

2.3.2 Output Specification

All the output should be printed to the command line window. Whenever there are changes made in the game board, your program should print out the updated game board. When the game is finished, a message that indicates the result of the game should be printed. The winning connected four pieces should also be highlighted in reverse video. If there are more than one "connect-4", you can show either one of them or all of them.

2.4 Python Classes

Please follow the classes `Connect_Four`, `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 Connect_Four.py`

1. **Class** `Connect_Four`

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

   - **Instance Variable(s)**

     `gameBoard`

     - This is a two-dimensional array representing the game board.

     `player1`

     - This is a variable representing Player O.

     `player2`

     - This is a variable representing Player X.

     `turn`

     - This is a variable indicating the player that should play in the current turn.

   - **Instance Method(s)**

     `__init__(self)`

     - Initialize the Connect_Four game.

     `StartGame(self)`

     - Start a new game and play until winning/losing or draw.

     `PrintGameBoard(self)`

     - Print out the game board in the command line window.

2. **Class** `Player`

   An abstract super class 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, PlayerSymbol)`
     - Initialize the player with its symbol X or O.

     `NextColumn(self, gameBoard)`
     - An abstract method to be implemented in subclass, which returns the column of next move.

3. **Class** `Human` and `Computer`
   Classes extending the super class `Player` to represent a human- and a computer-controlled player object respectively.

2.5 Bonus

Random moves are naive. We encourage you to make clever AI. If your AI player has a 50% odds against the original AI, you will earn an extra 15 points on this assignment. Please indicate on your submission if you have implemented an AI player. You should specify an extra choice (of value 3) for users to choose a clever AI player.

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 a scenario and 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: A Future Hotel

This task is to simulate a future hotel in Python. A Java program of the system 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 characteristics and behavior) are essentially interpreted as the objects' types. Duck Typing is a new typing paradigm in dynamically-typed (late binding) languages that allows 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 future, smart fixtures enhanced with advanced AI techniques become accessible. A future hotel is equipped with traditional "dumb" fixture, such as tables and chairs, and also various kinds of "smart" fixtures that could finish intelligent tasks and also move by itself instead of by people. One example is smart refrigerators that could automatically adjust the temperature.

4.3 Task Description

You are required to simulate several activities in a future hotel. A fixture, depending on its smartness, moves from one room to another either by itself or by the attendant that is assigned for managing the fixture. People also move around rooms and they may be served food in the dining hall. All kinds of moves consume energy, either in terms of calories for a person or in terms of battery usage for smart fixtures. The consumption of energy is proportional to the distance travelled and the weight of the objects being moved.

The requirement is simple. Please replicate all the behavior of the given Java Program in Python with Duck Typing. You program should run by calling `python hotel.py`. You will also be evaluated on your programming style.

5 Task 4: Introducing Robots into the hotel

In the further future, robots are introduced to alleviate the burden of their human staff. The hotel purchases dozens of robots which can also be assigned to manage dumb fixtures. A robot consumes battery power in its daily work and charges the battery if the battery is low. Obviously, a robot can move by itself.

You are now required to enhance both the Java and Python implementation of Task 3 by adding a Robot class. For Java, a new main program will be supplied to you for the new scenario involving robots. You are required to complete the Java implementation and may reuse part of the Java codes provided in Task 3 as necessary. For Python, just modify your program in Task 3 so that it can simulate the same scenario in the given Java main program.

6 Report

You 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, compare polymorphism obtained with Duck Typing (Python) and with inheritance (Java).

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

4. Using codes for Task 4, illustrate further advantages of 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. 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*.

2. 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`

3. Late submission policy: less 20% for 1 day late and less 50% for 2 days late. We shall not accept submissions more than 2 days after the deadline.

4. The report should be 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 Connect_Four.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 :>