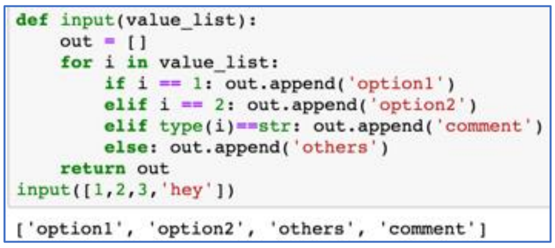
CSCI3180 Assignment 2 Report 1155110208 / Chung Tsz Ting

1)
Advantages:

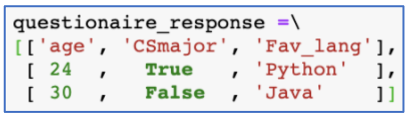


```python
def input(value_list):
    out = []
    for i in value_list:
        if i == 1: out.append('option1')
        elif i == 2: out.append('option2')
        elif type(i)==str: out.append('comment')
        else: out.append('others')
    return out
input([1,2,3,'hey'])
```

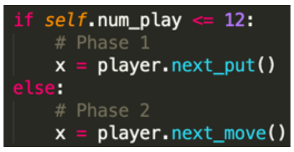
```
['option1', 'option2', 'others', 'comment']
```

(1) From this code segment, functions can be defined to apply on arguments of different types in which the code is more generic for use. For example, you want to get the type of the data, function in python allows input of different types, like integer, string and so on, and therefore allow the classification of the type of data.

(2) From this code segment, we can see it is possible to have a mixed type collection data structure. For example, in questionnaires' response of asking for the age, whether is CS major, favorite programming language, the result can be put within the same collection unlike array and other data structure in language like C and Java.



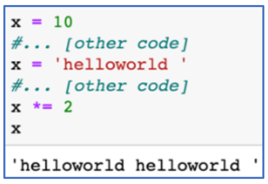
```python
questionaire_response =\
[['age', 'CSmajor', 'Fav_lang'],
 [ 24   ,   True    , 'Python'   ],
 [ 30   ,   False   , 'Java'     ]]
```



```python
if self.num_play <= 12:
    # Phase 1
    x = player.next_put()
else:
    # Phase 2
    x = player.next_move()
```

(2) From this code segment in task1, the player can be of computer class of human class. The 'next_put' and 'next_move' function can be applied on different classes of object (i.e. both 'Human' and 'Computer' class) without the need of type casting before use. This can reduce the code length and enhance readability as well as writability.

Disadvantages:

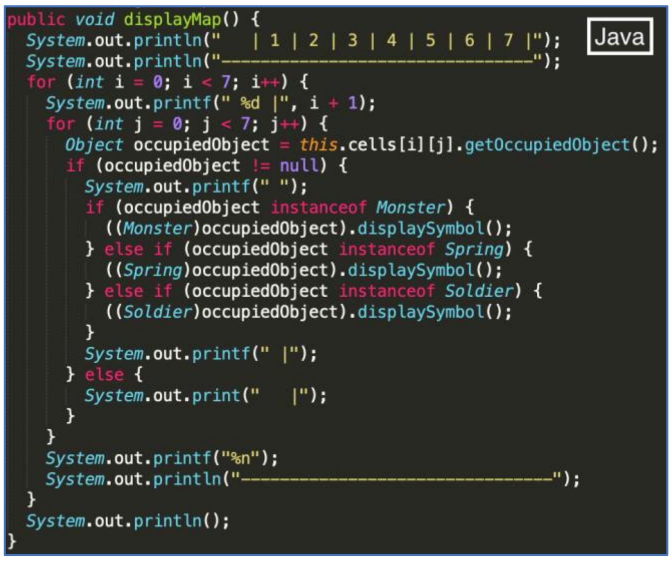


```python
x = 10
#... [other code]
x = 'helloworld '
#... [other code]
x *= 2
x
```

```
'helloworld helloworld '
```

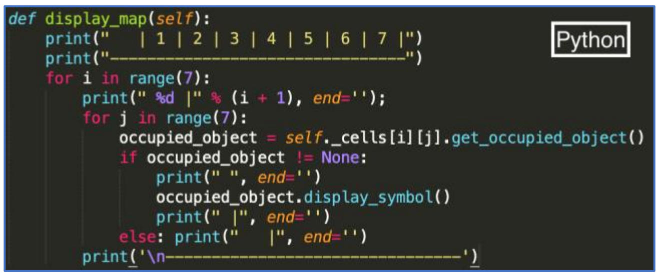
(1) For this code segment, we can see there is loss of type checking at compile time for dynamic typing. In static typing (like C), error will be raised for the mismatch of type in the code during compile. For case that there are lines of code, and you create x to be integer type and planning to multiply it by 2 later. If some collaborators for this coding project accidently write a string type of x in the middle, the programme can run successfully without getting any error but going against your original intention.

2)
Scenario 1:



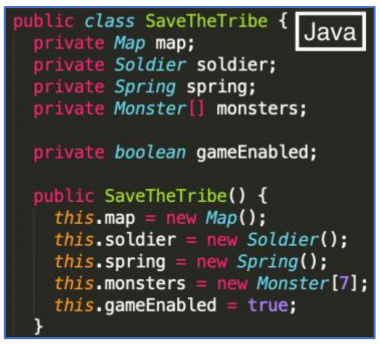
```java
public void displayMap() {
  System.out.println("   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |");
  System.out.println("-------------------------------");
  for (int i = 0; i < 7; i++) {
    System.out.printf(" %d |", i + 1);
    for (int j = 0; j < 7; j++) {
      Object occupiedObject = this.cells[i][j].getOccupiedObject();
      if (occupiedObject != null) {
        System.out.printf(" ");
        if (occupiedObject instanceof Monster) {
          ((Monster)occupiedObject).displaySymbol();
        } else if (occupiedObject instanceof Spring) {
          ((Spring)occupiedObject).displaySymbol();
        } else if (occupiedObject instanceof Soldier) {
          ((Soldier)occupiedObject).displaySymbol();
        }
        System.out.printf(" |");
      } else {
        System.out.print("   |");
      }
    }
    System.out.printf("%n");
    System.out.println("-------------------------------");
  }
  System.out.println();
}
```

I do not need to cast and write the if-else part that much in python which is more convenience in writing and the readability of the code is also better. From the function 'display_map', we can see a significant different in length between the codes of two programming language providing the same functional use. For the code in java, we cannot use the same code for different classes even though the class have the same function name (e.g. 'displaySymbol'). On the other hand, because of duck typing, python doesn't require you to check the class of the object and do the type casting, the function of a class can be called when the class object have that function been declared inside the class. Thus, python implementation is better than java under this scenario because of the availability of polymorphism without inheritance of python.



```python
def display_map(self):
    print("   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |")
    print("-------------------------------")
    for i in range(7):
        print(" %d |" % (i + 1), end='');
        for j in range(7):
            occupied_object = self._cells[i][j].get_occupied_object()
            if occupied_object != None:
                print(" ", end='')
                occupied_object.display_symbol()
                print(" |", end='')
            else: print("   |", end='')
        print('\n-------------------------------')
```

Scenario 2:

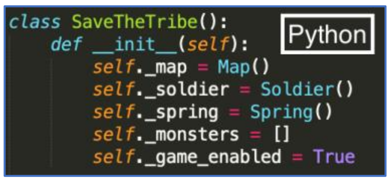


```java
public class SaveTheTribe {
  private Map map;
  private Soldier soldier;
  private Spring spring;
  private Monster[] monsters;

  private boolean gameEnabled;

  public SaveTheTribe() {
    this.map = new Map();
    this.soldier = new Soldier();
    this.spring = new Spring();
    this.monsters = new Monster[7];
    this.gameEnabled = true;
  }
```

In Java, we need to declare attributes without initialization before using and perform initialization afterwards. In python, both declaration and initialization are done at the same time on the _init_(). Therefore, the number of lines in python is shorter (enhance readability) and it is more convenient for the look up of the attributes and their value in python.

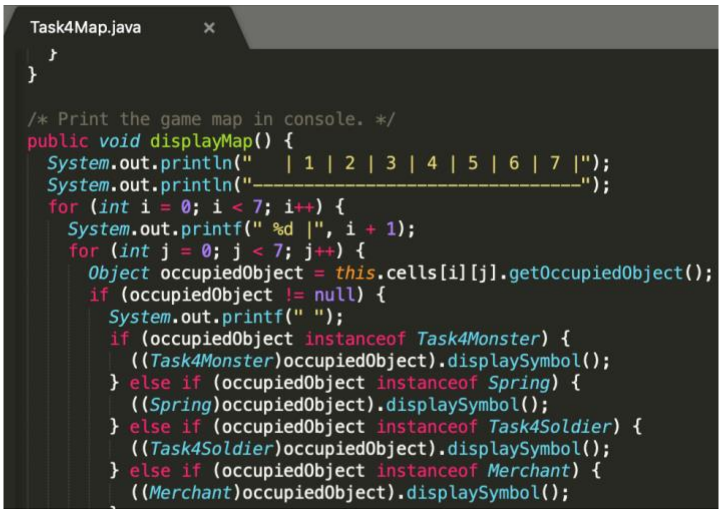


```python
class SaveTheTribe():
    def __init__(self):
        self._map = Map()
        self._soldier = Soldier()
        self._spring = Spring()
        self._monsters = []
        self._game_enabled = True
```

3)
For task4, we need to implement one more class ‘Merchant’ and adding functionality (reaction) between soldiers and merchant and monsters. In python, because of the convenience of duck typing, the class Map doesn’t need to be amended. When the object has the function of a duck and then it is a duck. Therefore, when merchant, soldier, monster and spring class consist of function ‘get_pos’ and ‘display_symbol’ and then it is the occupied_object. However, in java, because of the addition of one more class object Merchant, the function like ‘getPos’ and ‘displaySymbol’ in class Map need to be casted to the corresponding object.

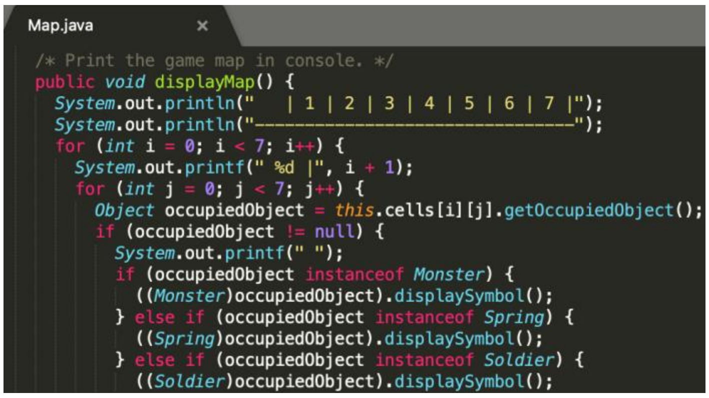
This makes the coding become more convenience and reduce the length of your code as you no longer need to cast the object to its corresponding type (which take you long time to write code especially when the one object actually can represent quite a number of classes in java), and enhance the writability.

In the meantime, you also don’t need to read highly repetitive code with the only different in the if condition with the utilization of duck typing. Thus, the readability is also enhanced.


Task4Map.java

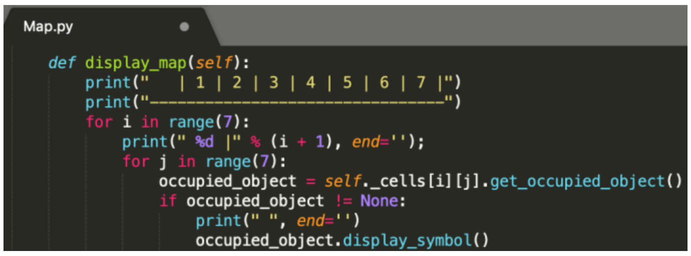
```java
    }
}

/* Print the game map in console. */
public void displayMap() {
  System.out.println("   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |");
  System.out.println("--------------------------------");
  for (int i = 0; i < 7; i++) {
    System.out.printf(" %d |", i + 1);
    for (int j = 0; j < 7; j++) {
      Object occupiedObject = this.cells[i][j].getOccupiedObject();
      if (occupiedObject != null) {
        System.out.printf(" ");
        if (occupiedObject instanceof Task4Monster) {
          ((Task4Monster)occupiedObject).displaySymbol();
        } else if (occupiedObject instanceof Spring) {
          ((Spring)occupiedObject).displaySymbol();
        } else if (occupiedObject instanceof Task4Soldier) {
          ((Task4Soldier)occupiedObject).displaySymbol();
        } else if (occupiedObject instanceof Merchant) {
          ((Merchant)occupiedObject).displaySymbol();
```


Map.java

```java
/* Print the game map in console. */
public void displayMap() {
  System.out.println("   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |");
  System.out.println("--------------------------------");
  for (int i = 0; i < 7; i++) {
    System.out.printf(" %d |", i + 1);
    for (int j = 0; j < 7; j++) {
      Object occupiedObject = this.cells[i][j].getOccupiedObject();
      if (occupiedObject != null) {
        System.out.printf(" ");
        if (occupiedObject instanceof Monster) {
          ((Monster)occupiedObject).displaySymbol();
        } else if (occupiedObject instanceof Spring) {
          ((Spring)occupiedObject).displaySymbol();
        } else if (occupiedObject instanceof Soldier) {
          ((Soldier)occupiedObject).displaySymbol();
```

Remark: adding code of type casting of ‘Merchant’ and displaySymbol() for task4 in java


Map.py

```python
    def display_map(self):
        print("   | 1 | 2 | 3 | 4 | 5 | 6 | 7 |")
        print("--------------------------------")
        for i in range(7):
            print(" %d |" % (i + 1), end='');
            for j in range(7):
                occupied_object = self._cells[i][j].get_occupied_object()
                if occupied_object != None:
                    print(" ", end='')
                    occupied_object.display_symbol()
```

Remark: not need adjustment in python for function of display_symbol()