ZHANG Xinyu 1155091989

1.

```
def add(x, y):
    sum = x + y
    return sum


# add two integers
print(add(1, 2))

# add two floats
print(add(3.313, 9.6))

# add two strings
print(add("add", "strings"))
```

Advantage 1: more generic code can be written. The example code demonstrate a general adding function which is applicable to int, floating points and strings.

```
# tuple
tuple1 = (1, 3.313, "tuple")

# reference
print(tuple1[1])
```

```
# list
list = [1, 3.313]

# catenation
list.append("list")
print(list)
# [1, 3.313, 'list']

# reference
print(list[1])
```

Advantage 2: possibilities of mixed type collection data structures. Both list (which is also serves as Python's arrays) and tuple are able to collected mixed data types' elements or object. The difference is that tuples are immutable but lists are mutable.

```
def add_three(a):
    return a + 3

a = "1"
print(a + "string")

# convert type
a = int(a)
b = add_three(a)
```

Advantage 3: a variable's type can change any number of times during execution. Because the dynamic typing in Python is bounded temporarily, we can change the type of variable a to achieve adding function with a string and integer respectively.

```
# add three
def add_three(x):
    return x + 3

y = add_three("1")
# TypeError:
# must be str,
# not int
```

Disadvantage: lose type checking at compile time. Type checking is done at runtime. Compiling above function doesn't raise error until it add a string to int, and raise TypeError.

2.

a) Nesting Selecters.

When selection statement is nested in the if clause of a selection statement, it is not clear to which if and else clause should be associated in Java code. Without brackets, the following statement can be intepreted in two differently ways, depending on whether the else clause is matched with the first if clause or the second.

```
if (choice.equalsIgnoreCase("1")) {
  if (this.loseHealth()) {
    System.out.printf("=> You defeated Monster%d.%n%n", this.monsterID);
    this.dropItems(soldier);
    fightEnabled = false;
  } else {
    if (soldier.loseHealth()) {
      this.recover(this.healthCapacity);
      fightEnabled = false;
    }
  }
```

However, in Python, the line else is indented to begin in the same column as the nested if, the else clause would be matched with the inner if.

```
if choice is "1":
    if self.loseHealth():
        print("=> You defeated Monster%d.\n" % self._monsterID)
        self.dropItems(soldier)
        fightEnabled = False
    else:
        if soldier.loseHealth():
            self.recover(self._healthCapacity)
            fightEnabled = False
```

b) For statement

The expressions in a for statement in Java requires one to specify those assignment statements. The first expression is for initialization, the second one is the loop control and the third one is executed after each execution of loop body. However, for simple couting loops in Python, the range function cab be used for simplification.

`for (int i = 0; i < 7; i++)`	`for i in range(7):`
Java	Python

3.

Duck typing enhance writablity. Unlike static typed languages like Java, we don't need to specify the variable's type when calling a same function of objects of different classes with duck typing.

`if (occupiedObject instanceof Monster) {`
`  ((Monster)occupiedObject).displaySymbol();`
`} else if (occupiedObject instanceof Spring) {`
`  ((Spring)occupiedObject).displaySymbol();`
`} else if (occupiedObject instanceof Soldier) {`
`  ((Soldier)occupiedObject).displaySymbol();`
`}`
	`occupiedObject.displaySymbol()`

Java	Python

That is, we can cast variables of different types and make code cleaner and simplier.

In the above code segments, we have to declare the variable occupiedObject's type with Monster, Spring and Soldier respectively when calling function displaySymbol(). However, in Python, we don't need to modify the Map.py in Task4 because the Python does not do type checking in compile time and we can cast object of different variables.