Dr Kirill Sidorov

Cardiff University
School of Computer Science and Informatics

CM1101 Computational Thinking

LAB EXERCISE THREE

Introduction to Python

Attempt as many exercises as you can. If you do not manage to finish all the exercises in the lab, please continue doing them at the next lab or at home. If you find the first few exercises too easy — skip to the harder ones. Remember, the lab tutors are here to help. If you get stuck — do not be shy, raise your hand and ask for advice. It is also ok to discuss the solutions with your peers (these labs are not assessed!), however make sure you understand everything by yourself.

Good luck!

1 In the Python REPL, create an empty dictionary and examine it:

```
>>> player = {}
>>> print(player)
```

Add some fields to the dictionary (using strings as keys), and again examine it:

```
>>> player["name"] = "Sheldor The Conqueror"
>>> player["health"] = 79
>>> player["ammo"] = 100
>>> print(player)
>>> print(player["ammo"])
>>> print(player["name"])
```

Delete the `ammo` field from the dictionary using the `del` command, and examine the result:

```
>>> del player["ammo"]
>>> print(player)
```

2 In a text editor, create a new program. In your program, define a dictionary to describe an RPG player (similar to the above), this time using the `{...}` notation. Your definition may look like this:

```
player = {"name": "Sheldor", ... add more fields ...}
```

Add the fields for `health`, `experience`, `mana` (again, using strings as keys) and assign some numeric values to them. Add a Boolean field `alive` to indicate whether the player is alive. Add a field `inventory` which should contain a list of items (a list of strings) which the player is carrying.

3 Write a line of code that would add the `"Sword of Azeroth"` to the list of items carried by your player.

4 Write a loop to iterate over and nicely print all the characteristics of the player. The output should look something like this:

```
Player stats:

health is 79
experience is 117
... etc.
```

Hint: remember, the following construct will iterate through all the keys in the dictionary:

```
for key in player:
    ... your code ...
```

5 Wrap the above loop which prints player attributes in a function `print_player` like so:

```
def print_player(player):
    ... your code ...
```

Test your function by calling it on the `player` dictionary you defined above.

6 Define a function `compute_experience` that takes as an input a number, `damage`, and returns a random number in the range between 0. . . `damage*2`. Again, test your function before continuing.

7 Define a function `take_damage` that takes as arguments: the `player` (a dictionary as defined above), and a number `damage`. The function should decrement the `health` of the player by `damage`, and add a random number of experience points between 0. . . `damage*2` to the player's `experience`. (Use the previously defined function `compute_experience` to accomplish this.) If the above results in the player's `health` becoming less than or equal to zero, set the player's `alive` attribute to `False`. Finally, return the updated `player`. The skeleton of the function may look like this:

```
def take_damage(player, damage):
    ... your code here ...

    return player
```

8 Test the above function: write a `while` loop in which repeatedly inflict damage to the player using `take_damage()` and print the player's stats using `print_player()`, until the player becomes dead.