Exercises – Operations using Data Types and Operators

Exercise 1

You have been tasked with rewriting the following two string definitions, as they are both producing errors.

```
message = "He said "There are some who call me...Tim""
```

The above line of code produces the following error:

```
    message = "He said "There are some who call me...Tim""
                       ^
SyntaxError: invalid syntax
```

```
message = 'He's not the messiah-he's a very naughty boy!'
```

The above line of code produces the following error:

```
    message = 'He's not the messiah-he's a very naughty boy!'
                  ^
SyntaxError: invalid syntax
```

Please look to correct each of these statements in two different ways.

1. Look to use the escape character (a backslash)
2. Look to switch the string delimiters(from double quotes to single quotes and vice-versa)

Exercise 2

All built-in objects in Python are considered to be either truthy or falsy from a Boolean context. That is, they will evaluate to either True or False, depending on their value.

For each of the following primitive data types (int, float, string, boolean), please determine a value that would be considered truthy (evaluates to True) and a value that would be considered falsy (evaluates to False).

You can use the built-in bool() function to determine if a value is considered truthy or not. The following statement would evaluate to False:

```
bool(False)
```

If you are not in a repl environment such as the python shell, you will need to wrap your call to the bool() function within a call to the print() function.

```
print(bool(False))
```

This is your answer for a boolean value that is considered falsy. The truthy answer for a boolean value is just as straight forward!

Play around with the int, float and string data types so that you can determine which values are considered truthy and which are considered falsy.

Exercise 3

You are writing some code for a banking app, but the following snippet of code is leading to an error:

```
savings = 1050

print("You have €" + savings + " in the bank!")
```

The error is caused because it is not possible to concatenate an int to a string using the + operator.

Please try to solve this issue in two different ways:

1. Use the appropriate conversion function so that the savings variable can be concatenated to the strings in the print statement
2. Pass the strings and the savings variable as comma-separated arguments to the print function

Exercise 4

Let's say that our banking app allows customers to make a deposit. The deposit amount is taken from the user using the input function (which is covered in a later tutorial). The input function always returns the user input as a string.

```
balance = 350.00
deposit = input("Please enter the amount that you wish to deposit")

balance += deposit
```

The above code leads to a runtime error, as you cannot add a float to a string. Let's say that the user inputs 70.50 as a deposit amount. The sum: balance += deposit is trying to add "70.50" to 350.00, which is not possible.

To fix this error, please use the suitable conversion function that will allow the addition to be carried out, ensuring that the cents portion of the deposit remains intact.

Exercise 5

You have created an app called ‘Mr C’s Weather Warnings’ and you wish to simplify the temperature display so that it just displays the temperature as a whole number, with no decimal places. In fact, we do not need to store the decimal portion at all, so the data type of the temp variable should be changed.

Please look to add a conversion function to the following code so that we are only storing a whole number in the variable temp.

```
temp = 19.9

print('The current temp is', temp, 'degrees')
```

```
The current temp is 19.9 degrees
```

Exercise 6

You are developing a game called ‘Bad Ambiance!’.

Mr C, the main character in the game, is an undercover agent (the cover is usually a cosy blanket!). He is highly customizable, as you can change his clothing to suit the weather and terrain.

He also has a set of watches to choose from: ‘Compass Watch’, ‘GPS Watch’, ‘Stun Watch’, ‘Nerve Gas Watch’, ‘Inflatable Dingy Watch’, ‘Android Watch’, ‘Lunchtime Alarm Watch(1:30pm)’.

- Please store the above watches in a list called ‘watches’
- Only the ‘on duty’ watches have been added to the list, so the following item should be added to the end of the list using the appropriate method: ‘Jingly Jangly Watch(for the evening ambiance)’
- There seems to be a glitch in the game, as Mr C does not want to put on the Android Watch. Please look to remove this item from the list using the appropriate method
- The item ‘Apple Watch’ should be inserted at the start of the list instead

Your list should now look like the following:

```
['Apple Watch',
 'Compass Watch',
 'GPS Watch',
 'Stun Watch',
 'Nerve Gas Watch',
 'Inflatable Dingy Watch',
 'Lunchtime Alarm Watch(1:30pm)',
 'Jingly Jangly Watch(for the evening ambiance)']
```

Now that we have our list of watches in place, let's add the functionality to our app to go ahead and select one of the watches to be worn.

- Please use indexing to grab the 'Inflatable Dingy Watch' from the list. This can be stored in a variable called 'current_watch'

Although Mr C is technically 'a machine', he still likes to indulge himself. The following list is defined in the code.

```
rations = ['Mars Bar', 'Powdered Soup', 'Dry Cereal', 'Trifle']
```

- Please look to sort the list elements alphabetically using the appropriate method
- You can then use indexing to take the last element from the list and store it in a variable called 'lunch'
- As you may have noticed, these are magic rations as the items are never removed from the list and are available for the next meal. Mr C likes this ambiance!

The rations list should now look as follows:

```
['Dry Cereal', 'Mars Bar', 'Powdered Soup', 'Trifle']
```

With breakfast being the main meal of the day, Mr C likes to have three items from the rations list to give him the energy for his marathon adventures.

- Please look to use list slicing to grab the first three elements from the list. The returned list should be stored in a variable called 'breakfast_of_champions'

Exercise 7

In this exercise, we will continue working on the 'Bad Ambiance!' app.

Along with being able to play the game in story mode, completing missions, it is also possible to play a couple of silly side games such as 'Hyper Hurdles' and 'We Have Lift-off!'.

Although these may seem like silly side games, a game player can actually unlock weapons and missions by achieving certain milestones in these side games. For instance, if the Hyper Hurdles game is completed in 10 seconds or under, or the Car rises off the ground by more than 11 centimetres when going over a hill at speed, a new weapon is unlocked in story mode. If both of the above criteria has been met (hurdles in 10 seconds or under **and** height is over 11 centimetres), a new mission is unlocked.

You have been given the following sample data:

```
hurdle_time = 9.95
```

```
height_from_road = 10.75
```

You have been tasked with writing two expressions to check if the game player has managed to unlock weapons and/or missions.

- The expression to determine if both criteria has been met, should store the result in a variable called unlock_mission
- The expression to determine if either of the criteria has been met, should be stored in a variable called unlock_weapon

End of Exercises