Lab: Operators (and optional Regular Expressions)

This lab is about defining Operators.

Part 1: Create a Score class.

The Score class should have attributes for:

- The score being represented
- The current date
- The current time

The class could be used within your number guess game application to represent Scores (although we won't do that just yet). The current date and time can be generated within the initialiser.

To get the current date you can use:

```
from datetime import date

today = date.today()
```

To get the current time use

```
from datetime import datetime

now = datetime.now()

current_time = now.strftime("%H:%M:%S")
print("Current Time =", current_time)
```

However, at this point it is not possible to add two scores together or to order scores.

To do this we must implement the __add__ method as well as the logical operators such as __lt__.

You should now provide the methods to support the add and subtract operators.

You should then implement the <,>, <=, >=, == and != operators.

You should now be able to run the following test code:

```
from player import Score

s1 = Score(5)
s2 = Score(6)

def test_score_equals():
    s3 = Score(5)
    assert s1 == s3
```

```
def test_score_not_equals():
    assert s1 != s2

def test_score_add():
    assert s1 + s2 == Score(11), "Adding two scores (5 and 6) should equal
11"

def test_score_sub():
    assert s2 - s1 == Score(1), "Subtracting two scores (5 and 6) should
equal 1"

def test_score_lt():
    assert s1 < s2, "S1 should be less than S2"
```

Extension Parts (Regular Expressions)

Step 1: Write a Python function to verify that a given string only contains letters (upper case or lower case) and numbers. Thus spaces and underbars ('_') are not allowed. An example of the use of this function might be:

```
print(contains_only_characters_and_numbers('John')) # True
print(contains_only_characters_and_numbers('John_Hunt')) # False
print(contains_only_characters_and_numbers('42')) # True
print(contains_only_characters_and_numbers('John42')) # True
print(contains_only_characters_and_numbers('John 42')) # False
```

Step 2: Write a function that will extract the value held between two strings or characters such as '<' and '>'. The function should take three parameters, the start character, the end character and the string to process. For example, the following code snippet:

```
print(extract_values('<', '>', '<John>'))
print(extract_values('<', '>', '<42>'))
print(extract_values('<', '>', '<John 42>'))
print(extract_values('<', '>', 'The <town> was in the <valley>'))
```

Should generate output such as:

```
['John']
['42']
['John 42']
['town', 'valley']
```