Software Development - Session 1B

Python programming exercises

1. Write a program to produce the following output using for loop

```
+------+
\      /
/      \
\      /
/      \
\      /
/      \
+----------+
```

```
print("+\t+")
for i in range(3):
    print("\\\t/")
    print("/\t\\")
print("+\t+")
```

2. Write a program to produce the following output using for loop

```
**********
**********
**********
**********
**********
```

```
for i in range(5):
    print("**********")
```

3. Complete the code for the following for loop:

```
for in range(1,7):
      //your code here
```

so that it prints the following numbers, one per line:

a)	1	b)	2	c)	4	d)	30	e)	-7	f)	97	g)	-4
	2		4		19		20		-3		94		14
	3		6		34		10		1		91		32
	4		8		49		0		5		88		50
	5		10		64		-10		9		85		68
	6		12		79		-20		13		82		86

```
for i in range(1, 7):
    print(i, i * 2, i * i + 3 * i - 2, 10 - i * 2, i + 4, 87 - 3 * i, i * 18 - 14)
```

4. Write a program to produce the following output using for loops. Then use a class constant to make it possible to change the number of lines in the figure.

```
1
22
333
4444
55555
666666
7777777
```

```
class NumberTriangle:
    NUM_LINES = 7  # Class constant to control the number of lines

    def print_triangle(self):
        for i in range(1, self.NUM_LINES + 1):
            print(str(i) * i)

# Create an instance of the NumberTriangle class
triangle = NumberTriangle()

# Print the triangle
triangle.print_triangle()
```

5. Write a method named `pay` that accepts two parameters: a real number for a TA's salary, and an integer for the number of hours the TA worked this week. The method should return how much money to pay the TA. For example, the call

```
pay(5.50, 6)
```

should return

`33.0`.

The TA should receive "overtime" pay of 1 ½ normal salary for any hours above 8. For example, the call `pay(4.00, 11)` should return (4.00 * 8) + (6.00 * 3) or `50.0`.

```
def pay(salary, hours_worked):
    normal_hours = 8
    overtime_rate = 1.5

    if hours_worked <= normal_hours:
        return salary * hours_worked
    else:
        normal_pay = salary * normal_hours
        overtime_pay = salary * overtime_rate * (hours_worked - normal_hours)
        total_pay = normal_pay + overtime_pay
        return total_pay

# Example usage:
result = pay(5.50, 6)
print(result)  # Output: 33.0

result = pay(4.00, 11)
print(result)  # Output: 50.0
```

6. Consider the following method for converting milliseconds into days:

```
// converts milliseconds to days
def toDays(millis):
    return millis / 1000.0 / 60.0 / 60.0 / 24.0
```

Write a similar method named `area` that takes as a parameter the radius of a circle and that returns the area of the circle. For example, the call

```
area(2.0);
```

should return

```
12.566370614359172.
```

Recall that area can be computed as π times the radius squared and that Python has a constant called `math.pi`

```
import math

def area(radius):
    return math.pi * radius**2

# Example usage:
result = area(2.0)
print(result)  # Output: 12.566370614359172
```

7. **Copy and paste** the following code into pythons IDLE script environment.

```
low = 1
high = 1001
sum = 0
for i in range(low,high):
   sum += i

print("sum = " , sum)
```

Modify the code to use a `input` to prompt the user for the values of `low` and `high`. Below is a sample execution in which the user asks for the same values as in the original program (1 through 1000):

```
low? 1
high? 1001
sum = 500500
```

Below is an execution with different values for `low` and `high`:

```
low? 300
high? 5298
sum = 13986903
```

You should exactly reproduce this format.

```
low = int(input("low? "))
high = int(input("high? "))
sum_result = 0

for i in range(low, high):
    sum_result += i

print("sum =", sum_result)
```

8. Write a program using while loop that prompts the user for numbers until the user types 0, then outputs their sum.

9. Write a program using while loop that prompts the user for numbers until the user types -1, then outputs their sum.

```
sum_result = 0

while True:
    user_input = int(input("Enter a number (or -1 to stop): "))

    if user_input == -1:
        break  # Exit the loop if -1 is entered
    else:
        sum_result += user_input

print("Sum of entered numbers:", sum_result)
```

10. Write a method named `repl` that accepts a `String` and a number of repetitions as parameters and returns the `String` concatenated that many times. For example, the call `repl("hello", 3)` returns `"hellohellohello"`. If the number of repetitions is 0 or less, an empty string is returned.

```
def repl(input_str, repetitions):
    if repetitions <= 0:
        return ""  # Return an empty string for 0 or negative repetitions
    else:
        return input_str * repetitions

# Example usage:
result = repl("hello", 3)

print(result)  # Output: "hellohellohello"
```

11. Write a method called printRange that accepts two integers as arguments and prints the sequence of numbers between the two arguments, separated by spaces. Print an increasing sequence if the first argument is smaller than the second; otherwise, print a decreasing sequence. If the two numbers are the same, that number should be printed by itself. Here are some sample calls to printRange:

```
printRange(2, 7)

printRange(19, 11)

printRange(5, 5)
```

The output produced should be the following:

```
2 3 4 5 6 7

19 18 17 16 15 14 13 12 11
```

5

```
def printRange(num1, num2):
    if num1 < num2:
        for i in range(num1, num2 + 1):
            print(i, end=" ")
    elif num1 > num2:
        for i in range(num1, num2 - 1, -1):
            print(i, end=" ")
    else:
        print(num1)

# Example calls:
printRange(2, 7)
printRange(19, 11)
printRange(5, 5)
```

12. Write a method named smallestLargest that asks the user to enter numbers, then prints the smallest and largest of all the numbers typed in by the user. You may assume the user enters a valid number greater than 0 for the number of numbers to read. Here is an example dialogue:

```
How many numbers do you want to enter? 4

Number 1: 5

Number 2: 11

Number 3: -2

Number 4: 3

Smallest = -2
```

```
Largest = 11
```

```
def smallestLargest():
    num_of_numbers = int(input("How many numbers do you want to enter? "))

    if num_of_numbers <= 0:
        print("Please enter a valid number greater than 0.")
        return

    smallest = float('inf')
    largest = float('-inf')

    for i in range(1, num_of_numbers + 1):
        current_number = float(input(f"Number {i}: "))
        smallest = min(smallest, current_number)
        largest = max(largest, current_number)

    print(f"Smallest = {smallest}")
    print(f"Largest = {largest}")
```

13. Write a method called `printAverage` that uses a sentinel loop to repeatedly prompt the user for numbers. Once the user types any number less than zero, the method should display the average of all nonnegative numbers typed. Display the average as a `double`. Here is a sample dialogue with the user:

```
Type a number: 7

Type a number: 4

Type a number: 16

Type a number: -4

Average was 9.0
```

If the first number that the user types is negative, do not print an average:

```
Type a number: -2
```

```
def printAverage():
    total = 0
    count = 0

    while True:
        user_input = float(input("Type a number: "))

        if user_input < 0:
            if count > 0:
                average = total / count
                print(f"Average was {average}")
            else:
                print("No nonnegative numbers entered.")
            break
```

```
        else:
            total += user_input
            count += 1

# Example usage:
printAverage()
```

14. Write a method named `numUnique` that takes three integers as parameters and returns the number of unique integers among the three. For example, the call `numUnique(18, 3, 4)` should return `3` because the parameters have three different values. By contrast, the call `numUnique(6, 7, 6)` should return `2` because there are only two unique numbers among the three parameters: `6` and `7`.

```
def numUnique(num1, num2, num3):
    unique_set = set([num1, num2, num3])
    return len(unique_set)

# Example usage:
result1 = numUnique(18, 3, 4)
print(result1)  # Output: 3

result2 = numUnique(6, 7, 6)
print(result2)  # Output: 2
```

15. A Random object generates pseudo-random numbers. Find out how to use the Random class and write a program that simulates rolling of two 6-sided dice until their combined result comes up as 7. One possible output can be seen as below:

```
2 + 4 = 6

3 + 5 = 8

5 + 6 = 11

1 + 1 = 2

4 + 3 = 7

You won after 5 tries!
```

```
import java.util.Random;

public class DiceRollSimulation {
    public static void main(String[] args) {
        Random random = new Random();
        int tries = 0;

        while (true) {
            int dice1 = random.nextInt(6) + 1;
```

```
            int dice2 = random.nextInt(6) + 1;
            int sum = dice1 + dice2;

            System.out.println(dice1 + " + " + dice2 + " = " + sum);

            tries++;

            if (sum == 7) {
                System.out.println("You won after " + tries + " tries!");
                break;
```