Lab Test 1

General Instructions:

- Time allowed: 120 mins
- Total: 120 marks
- There are 6 questions
- Each Q in this Lab Test 1 is worth 20 marks
- Marks may be deducted for not following Python coding conventions, writing code that is hard to understand (e.g. unclear variable names or poor layout) or writing very inefficient code (e.g. unnecessary lines of code that do not improve clarity)
- You may write on this question paper, and you may keep this question paper
- For all questions, you may assume that if a function is expecting input parameters of a certain data type, then the function will always receive input parameters of that data type. For example, if a function is expecting to receive an integer, then the function will only ever be given integers.
- If you finish early, you are welcome to leave early. However, please raise your hand first, so that the time can be recorded. You are not allowed to resubmit code after you leave. If you submit after you leave the room, you will be given a total score of 0 marks.

Submission Instructions:

- Write your answer for each Q in a separate cell, but all in ONE .ipynb (Jupyter Notebook) file
- Write a comment at the top of each cell, indicating which Q, e.g. `"### Q1"`
- When marking your work, we will assume that all the code for a Q is in only ONE cell, and that no other cell has previously been run
- Submit only ONE .ipynb file
- The name of the zip file should be the same as your ID. For example, if your email address is **jason.chan.2025@msa.smu.edu.sg**, then your file should be called **jason.chan.2025**.ipynb

Q1: Area of circle

Write a function called `calculate_area()` which returns the area of a circle. The function is given a number as parameter, which represent the radius of a circle.

You may assume that your function is always given a positive number as the radius parameter.

The area of a circle is given by this formula:

$$A = \pi r^2$$

where:

- A is the area of the circle
- π is the constant 3.14159…
- r is the radius of the circle

For the value of π, you should use the built-in value in the math package, instead of creating your own variable to store the value. To do this, you will need to first write this code before you call your function:

```
# Import math package so that we can access value of pi
import math
```

and then later, you can access the value of π in the math package (the code to do that is not given here).

For example:

- `calculate_area(5)` should return 78.53981633974483
- `calculate_area(8)` should return 201.06192982974676

Q2: Check birthdays

Write a function called `check_birthdays()` which takes six integers as parameters (not a list, not a tuple), which represents the following (and in this order):

- d1, m1 and y1 are the day, month and year of birthday of Person 1, respectively
- d2, m2 and y2 are the day, month and year of birthday of Person 2, respectively

You may assume that your function is always given days, months and years that are valid and integers.

The function should not return any values. Instead, the function should print ONLY ONE LINE on the screen, depending on the following:

- If both persons have exactly the same birthday, including same year, then print EXACTLY:
  "Both have exactly the same age"
- If both persons have the same birthday (day and month), but Person 1 is born in an earlier year, then print EXACTLY:
  "Person 1 is younger but they both have same birthday"
- If Person 1 is younger and also has their birthday earlier in the year, then print EXACTLY:
  "Person 1 is younger and has their birthday earlier in the year"
- If Person 1 is younger, but Person 2 has their birthday earlier in the year, then print EXACTLY:
  "Person 1 is younger but Person 2 has their birthday is earlier in the year"
- Similar logic above applies if Person 2 is younger instead, then you would print "Person 2" instead of "Person 1", etc.

For example:

- `check_birthdays(23, 9, 2025, 5, 9, 2025)` should print to the screen:

```
Person 1 is younger but Person 2 has their birthday earlier in the year
```

- `check_birthdays(5, 9, 2025, 23, 9, 2025)` should print to the screen:

```
Person 2 is younger but Person 1 has their birthday earlier in the year
```

- `check_birthdays(23, 9, 2025, 5, 2, 2025)` should print to the screen:

```
Person 1 is younger but Person 2 has their birthday earlier in the year
```

- `check_birthdays(23, 9, 1999, 5, 2, 2025)` should print to the screen:

```
Person 2 is younger and has their birthday is earlier in the year
```

- `check_birthdays(23, 9, 1999, 23, 9, 2025)` should print to the screen:

```
Person 2 is younger but they both have same birthday
```

- `check_birthdays(23, 9, 2025, 23, 9, 2025)` should print to the screen:

```
Both have exactly same age
```

Q3: Add digits of string

Write a function called `add_digits()`. The function is given a string as a parameter, and returns an integer, which is the sum of all the digits in the string.

You may assume that your function is always given a valid string.

For example:

- `add_digits("abc621de4f")` should return 13, since 6 + 2 + 1 + 4 = 13.
- `add_digits("abc-621de.4f")` should also return 13, since 6 + 2 + 1 + 4 = 13.
    - Note that the first digit is 6, not -6.
    - Note that the last digit is 4, not 0.4
- `add_digits("abcde")` should return 0, since there are no digits.

Q4: Remove smallest pair

Write a function called `remove_smallest_pair()`. The function takes in one integer as a parameter. The function should return a new integer, where the smallest pair of adjacent digits in the number have been removed.

For example, if the integer parameter is 17812345, then the pairs of adjacent digits are 17, 78, 81, 12, 23, 34 and 45. So the smallest pair is 12, and therefore, the function should return 178345.

If the smallest pair of adjacent digits occurs more than once in the number, then the right-most occurrence should be removed.

If the integer parameter has fewer than 2 digits, then the same integer should be returned, without change.

For example:

- `remove_smallest_pair(17812345)` should return 178345
    - The smallest pair was 12, so it was removed
- `remove_smallest_pair(71234125)` should return 712345
    - The smallest pair was 12, but it occurred more than once, so the right-most occurrence was removed
- `remove_smallest_pair(340915)` should return 3415
    - The smallest pair was 09, so it was removed
- `remove_smallest_pair(9)` should return 9
    - The integer has less than 2 digits, so it remains unchanged
- `remove_smallest_pair(10)` should return 0
    - The smallest pair was 10, so it was removed, leaving 0

You may assume that your function is always given a valid positive integer, and that no integer begins with a 0.

WARNING: Your function should only use the integer data type. If you use any other data type (e.g. strings or arrays), then a maximum of 10 marks (not 20 marks) can be awarded for this question.

Q5: Print pattern

Write a function called `print_pattern()` which prints a pattern on the screen, depending on the value of the integer parameter n, which you may assume is always a positive even integer.

The pattern always contains 3 triangles (of width and height n) in the first n rows, and below the triangles, always contains a rectangle of width 3n and height n, with a "hole" of width n in the middle of the bottom n/2 rows.

Some examples are given below.

If n is 4, your function should print the following:

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

If n is 6, your function should print the following:

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

There are no blank rows or blank columns in the above patterns.

WARNING: Your solution should only contain one pair of nested loops (a loop inside a loop), and only one if-statement inside the inner-most loop, as taught in lesson 4. If you do not follow this requirement, then a maximum of 5 marks (not 20 marks) can be awarded for this question.

Q6: Average list

Write a function called `average_list()` which is given two parameters: a list of numbers and an integer n. You may assume that n is always an odd integer that is at least 3.

The function does not return a value. Instead, the function will change the contents of the list that is given to it, so that each value in the list becomes a float (decimal number), which is the average value of the n adjacent values (if any) in the original list, centred on itself, including itself.

For example, if we run this code:

```
numbers = [9,7,4,5,6]
average_list(numbers, 3)
print(numbers)
```

then only the following should appear on the screen:

```
[8.0, 6.666666666666667, 5.333333333333333, 5.0, 5.5]
```

because:

- First value in numbers was 9, so it must be replaced by the n=3 adjacent numbers (if they exist), centred on the 9. Since there is no number before 9, therefore we only average the 9 and 7, which is 8. Therefore, the first value in the array must become 8.0.
- Second value in numbers was 7, so it must be replaced by the n=3 adjacent numbers (if they exist), centred on the 7. Therefore we average the 9, 7 and 4, which is 6.666666667. Therefore, the second value in the array must become 6.666666667.
- Similarly, third value in list is 5.333333333, because the average of 7, 4 and 5 is 5.333333333.
- Fourth value in list is 5.0, because the average of 4, 5 and 6 is 5.0.
- Fifth value in list is 5.5, because the average of 5 and 6 is 5.5.

Another example: if we run this code:

```
numbers = [9,7,4,5,6,7]
average_list(numbers, 5)
print(numbers)
```

then only the following should appear on the screen:

```
[6.666666666666667, 6.25, 6.2, 5.8, 5.5, 6.0]
```

- First value in numbers was 9, so it must be replaced by the n=5 adjacent numbers (if they exist), centred on the 9. Since there are no numbers before 9, therefore we only average the 9, 7 and 4, which is 6. 66666667. Therefore, the first value in the array must become 6.666667.
- Second value in numbers was 7, so it must be replaced by the n=5 adjacent numbers (if they exist), centred on the 7. Therefore we only average 4 numbers (the 9, 7, 4 and 5), which is 6.25. Therefore, the second value in the array must become 6.25.
- Similarly, third value in list is 6.2, because the average of 9, 7, 4, 5 and 6 is 6.2.
- Fourth value in list is 5.8, because the average of 7, 4, 5, 6 and 7 is 5.8.
- Fifth value in list is 5.5, because the average of 4, 5, 6 and 7 is 5.5.
- Sixth value in list is 6.0, because the average of 5, 6 and 7 is 6.0.

There are no more questions