**Overview**

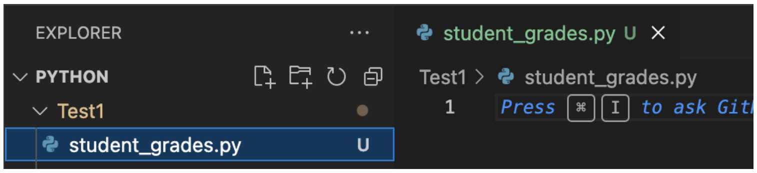
Loops are a fundamental control structure in programming that allow you to execute a block of code repeatedly until a certain condition is met. Nested loops are loops placed inside other loops, creating multiple levels of iteration. In this worksheet, we'll use a scenario of managing student grades to demonstrate these nested loops.

**Pre-requisites:**

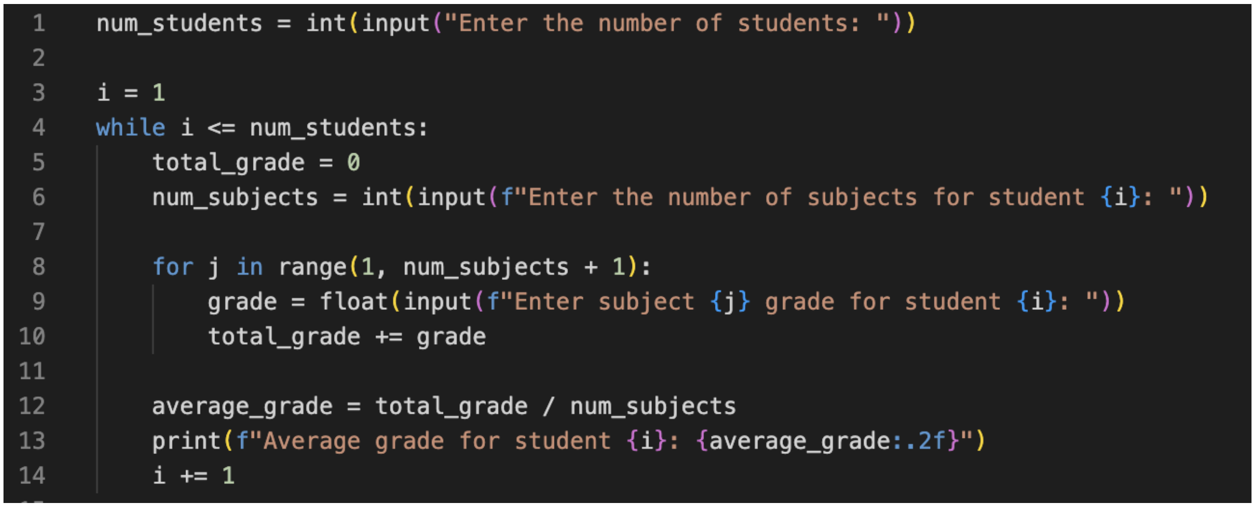
1. Github account
2. Git
3. VSCode
4. Python
5. Basic understanding of Python syntax.
6. Basic understanding of loops
7. Familiarity with git commands

**Instructions:**

1. **Open the cloned folder in VSCode**.
    a. Create a python file called "student_grades.py"



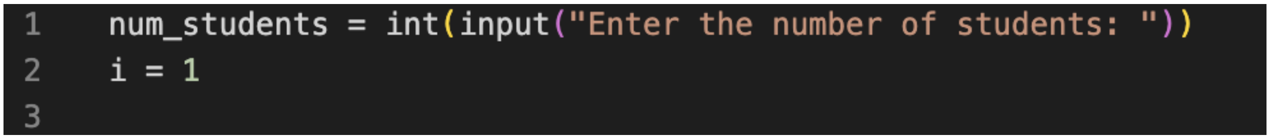
2. **Write a program**



```python
num_students = int(input("Enter the number of students: "))

i = 1
while i <= num_students:
    total_grade = 0
    num_subjects = int(input(f"Enter the number of subjects for student {i}: "))

    for j in range(1, num_subjects + 1):
        grade = float(input(f"Enter subject {j} grade for student {i}: "))
        total_grade += grade

    average_grade = total_grade / num_subjects
    print(f"Average grade for student {i}: {average_grade:.2f}")
    i += 1
```

**Explanation**

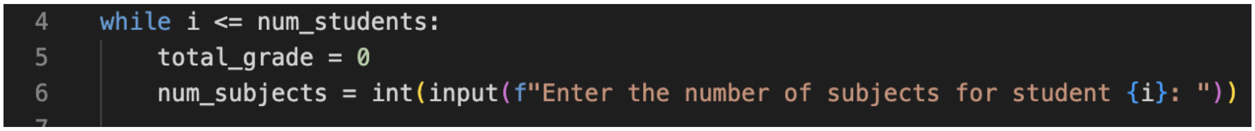
a. Prompts the user to enter the total number of students whose grades will be calculated. Converts the user's input (string) to an integer and stores it in num_students. `i = 1` initializes a variable i with a value of 1. This variable acts as a counter to keep track of the current student being processed.



```python
num_students = int(input("Enter the number of students: "))
i = 1

```

b. The while loop will continue to iterate as long as i (current student number) is less than or equal to the total number of students (num_students).

Inside the loop, initializes a variable total_grade to 0. This variable will accumulate all grades for the current student. After that, prompts the user to enter the number of subjects for student i. Converts the user's input to an integer and stores it in num_grades



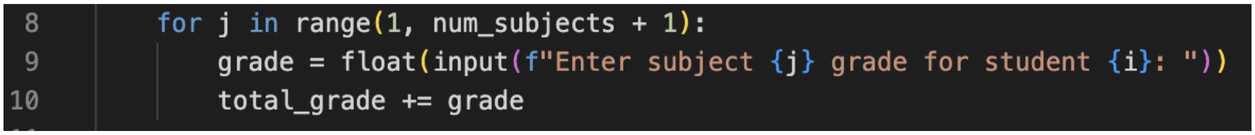
```python
while i <= num_students:
    total_grade = 0
    num_subjects = int(input(f"Enter the number of subjects for student {i}: "))
```

c. This loop will iterate num_subjects times (number of subjects entered for the current student). The range() function creates a sequence of numbers from 1 (inclusive) up to num_subjects.

*Note: we want to iterate over all the students, hence the need to add 1 to num_students to ensure that the loop iterates up to and includes the num_students itself.*

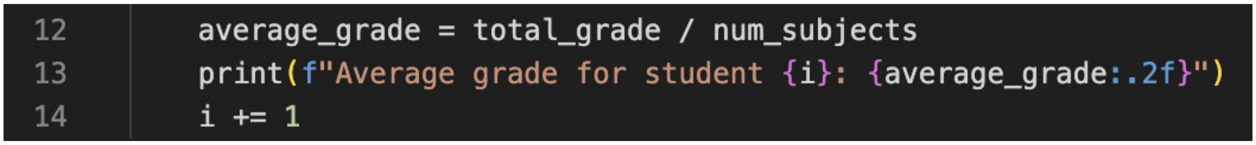
The variable j acts as a counter within this loop, keeping track of the current subject being processed.

Inside the nested loop, prompts the user to enter the grade of the j-th subject for student i. Uses an f-string to include the student number and current grade number in the prompt. total_grade accumulates the entered grade (grade) into the total_grade variable. This calculates the sum of all grades for the current student.



```python
    for j in range(1, num_subjects + 1):
        grade = float(input(f"Enter subject {j} grade for student {i}: "))
        total_grade += grade
```

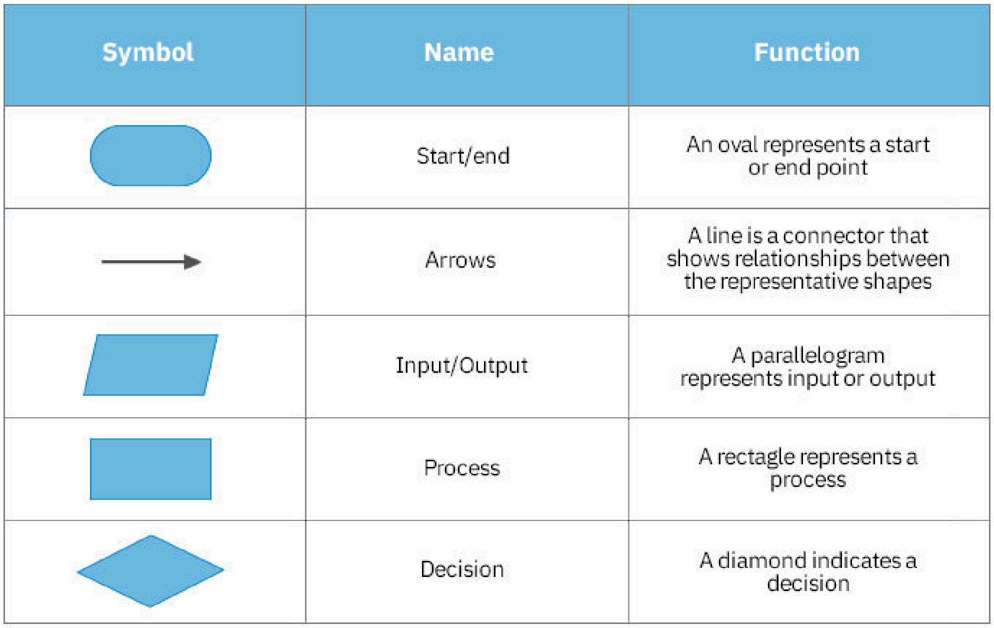
d. After the nested loop finishes, then calculates the average grade for the current student. It divides the total_grade (sum of all grades) by the num_subjects (number of subjects entered). Prints a message displaying the average grade for the current student (i). Uses an f-string to format the output, including the student number and the average grade with two decimal places (using :.2f).
i += 1 increments the counter i by 1. This prepares the loop for the next iteration, where it will handle grades for the next student (i + 1).



```
    average_grade = total_grade / num_subjects
    print(f"Average grade for student {i}: {average_grade:.2f}")
    i += 1
```

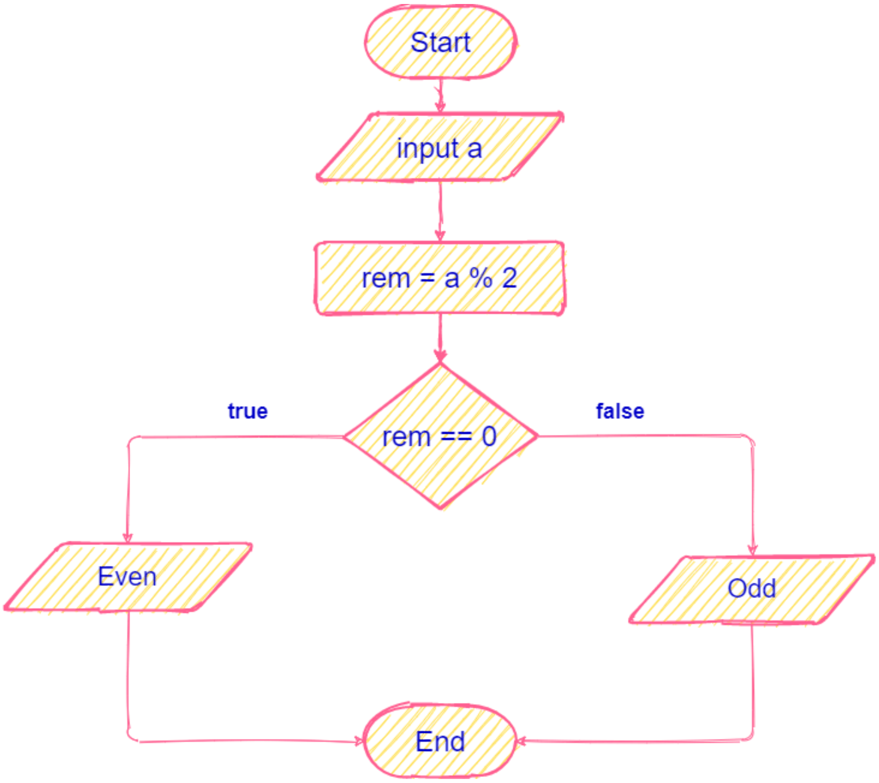
### Flow chart

A flowchart is a diagrammatic representation of an algorithm. It is read from top to bottom and uses shapes and arrows to show what happens in a program. A flowchart can be helpful for both writing programs and explaining the program to others.



| Symbol | Name | Function |
|---|---|---|
| ⬭ | Start/end | An oval represents a start or end point |
| → | Arrows | A line is a connector that shows relationships between the representative shapes |
| ▱ | Input/Output | A parallelogram represents input or output |
| ▭ | Process | A rectagle represents a process |
| ◆ | Decision | A diamond indicates a decision |

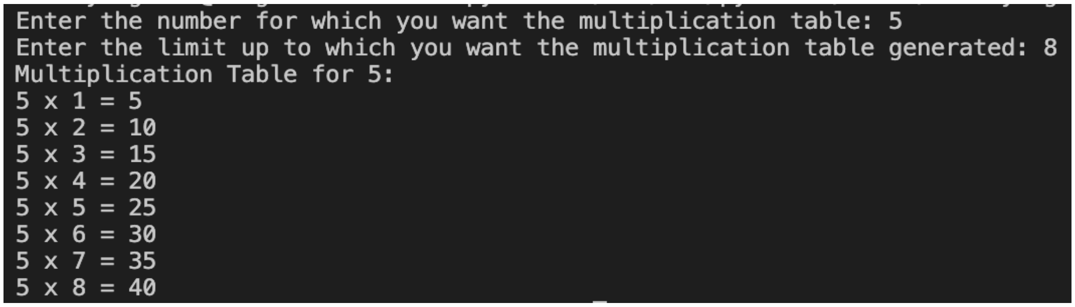
**Example:** To find whether an entered integer is even or odd, the flow chart is as follows:



**Exercises:**

1. Write Python code to generate a multiplication table for a specified number up to a specified limit entered by the user. The code should prompt the user to enter the number for which they want the multiplication table and the limit up to which they want the table generated. The code should then use loops to generate the multiplication table for the specified number up to the specified limit.
   Also design a flowchart that represents the logic for the program.

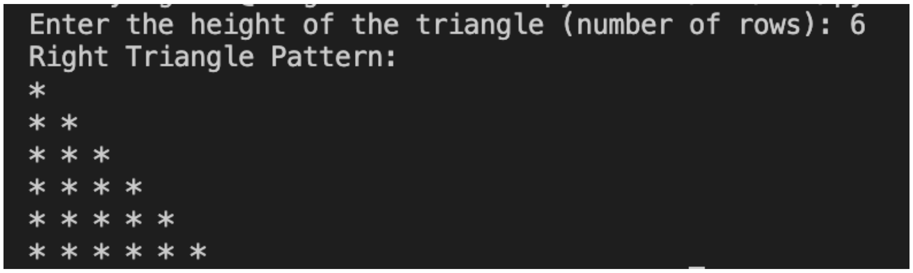
   Input/Output sample:



```
Enter the number for which you want the multiplication table: 5
Enter the limit up to which you want the multiplication table generated: 8
Multiplication Table for 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
```

2. Write Python code to generate a pattern of stars (*) in the shape of a right triangle. The code should prompt the user to enter the height of the triangle (number of rows). Then, using nested loops, the code should print a pattern of stars representing the right triangle.

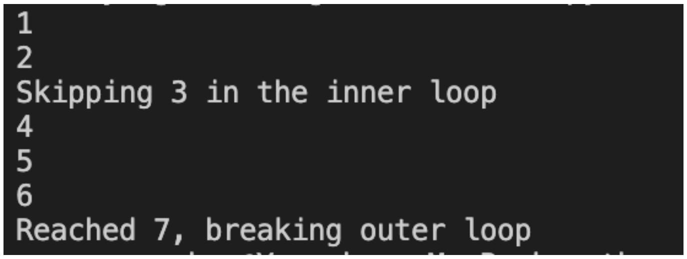
   Input/Output sample:



```
Enter the height of the triangle (number of rows): 6
Right Triangle Pattern:
*
* *
* * *
* * * *
* * * * *
* * * * * *
```

3. Write Python code to print numbers from 1 to 9, skipping the number 3 in the inner loop, and breaking out of the outer loop when reaching the number 7. Use nested loops to achieve this task.

   Input/Output sample:



```
1
2
Skipping 3 in the inner loop
4
5
6
Reached 7, breaking outer loop
```