SEATWORK # 1

1. Initialize an empty list to store student data: students = [].

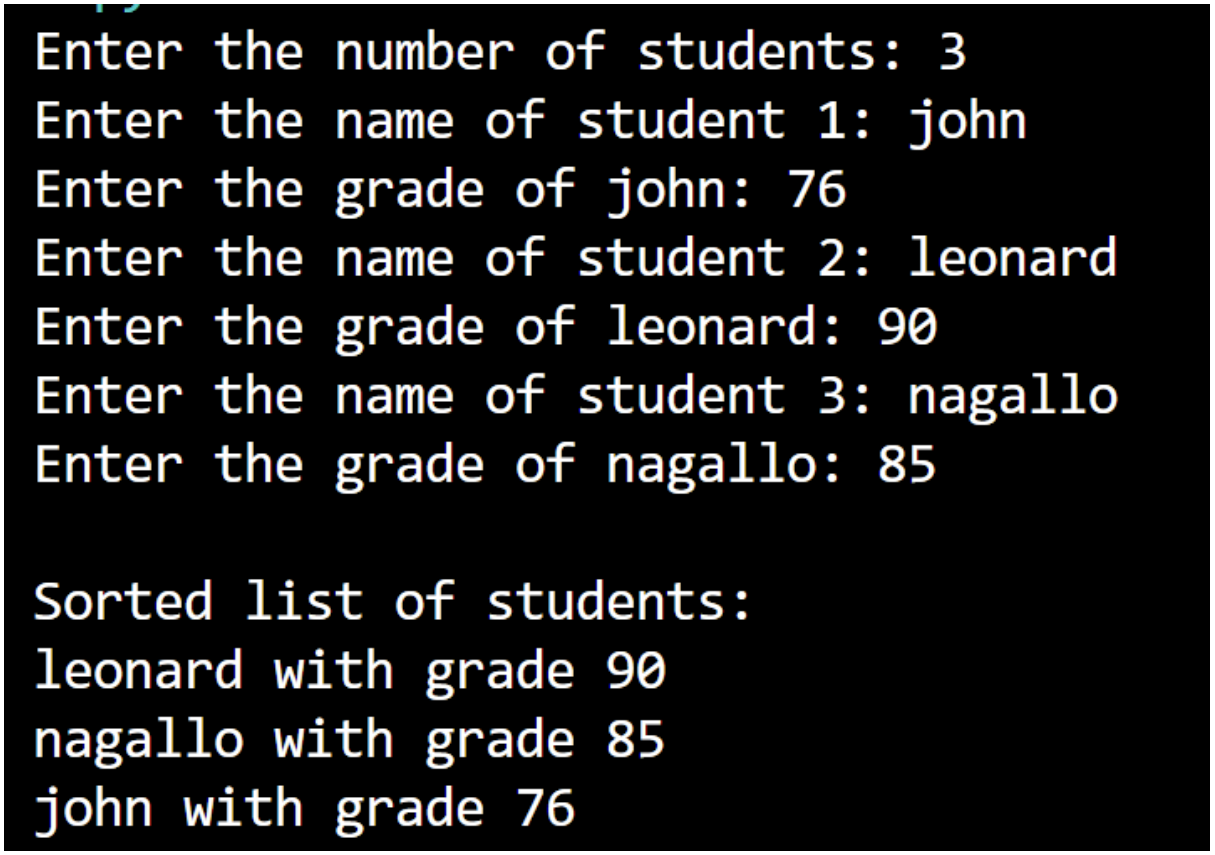
2. Get the number of students by prompting the user to input a value, and store it in num_students.

3. Loop through each student using a for loop that runs num_students times:
   - For each iteration, ask for the student's name and grade.
   - Create a dictionary for each student with "name" and "grade" keys.
   - Add each dictionary to the students list using append().

4. Sort the students list:
   - Use the sorted() function with a lambda function as the key.
   - Sort primarily by student["grade"] to order by grade in descending order.
   - Sort secondarily by student["name"] to order alphabetically if grades are the same.

5. Display the sorted list:
   - Print each student's name and grade in the sorted order.

Output:



SEATWORK# 2

Instruction:

Begin by creating a list called people that contains tuples, with each tuple representing a person's name (as a string) and age (as an integer). Next, use the sorted() function with a lambda function as the key argument to sort the people list by age, which is the second element in each tuple. Store this sorted result in a new variable called people_sorted_by_age. Then, define another list called words that includes words (as strings) of different lengths. To sort this list by the length of each word, use the sorted() function again, but this time with the key=len argument, and store the sorted result in a variable named sorted_words.

Following this, generate a dictionary called cubes using dictionary comprehension. In this dictionary, the keys should be numbers from 1 to 6, and the values should be their corresponding cubes. Finally, print the results to display your outputs: first, print the header Seatwork#: to label your output. Then, print the sorted list of people by age by displaying people_sorted_by_age, the sorted list of words by length by displaying sorted_words, and the dictionary of cubes by displaying cubes. Follow the input, process output structure in the code.

Expected output:

```
Seatwork #2
People sorted by age: [('Bob', 20), ('Alice', 25), ('Charlie', 30)]
Words sorted by length: ['apple', 'banana', 'cherry', 'blueberry']
Cubes: {1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
```

LAB
Directions:

File name: th3qu1ckbr0wnf0x.py

- Create a main function named "*main*" that displays a menu to the user with four options:
  - **Option 1**: Filter odd numbers from a set of 5 numbers.
  - **Option 2**: Union the odd numbers set with another set of 5 numbers.
  - **Option 3**: Sort a list of three names and their ages.
  - **Option 0**: Exit the program.
- For each option:
  - **Option 1**: Ask the user to enter 5 numbers, store them in a set named *numbers_set* , and call your function named *"filter_odd_numbers"* that will filter all the even numbers, leaving only the odd numbers. Save the odd numbers in a new set called *"odd_numbers_set"* and print them inside the *"filter_odd_numbers"* function
  - **Option 2**: Ensure that Option 1 has been run first (to populate the odd numbers set). Then, ask the user to enter another 5 numbers, save it in a set named *"new_numbers_set"* and call your function named *"union"* that will combine the *"odd_numbers_set"* and *"new_numbers_set"* , and print the *"new_numbers_set"*
  - **Option 3**: Prompt the user to enter three names and ages. Store them in a list of tuples named *"people"* and call your function named *"sortAge"*. The function should sort the ages from youngest to eldest in list called *"sorted_people"*, and print the outcome.
  - **Option 0**: Exit the program.
- Use input validation to ensure users enter correct options and handle any invalid input gracefully. Print *"Invalid choice. Please try again."* if the user enters numbers outside 0-3

Final Step: Run the Program

- Once the main function is complete, call it to start the program. The menu should guide users through each functionality.

```
Enter your choice: 1
Please enter 5 numbers:
Enter number 1: 1
Enter number 2: 2
Enter number 3: 3
Enter number 4: 4
Enter number 5: 5
Filtered set with only odd numbers: {1, 3, 5}
```

```
Enter your choice: 2
Please enter another 5 numbers for union operation:
Enter number 1: 6
Enter number 2: 7
Enter number 3: 8
Enter number 4: 9
Enter number 5: 10
Union of filtered odd numbers set and new set: {1, 3, 5, 6, 7, 8, 9, 10}
```

```
Enter your choice: 3
Please enter three names and their corresponding ages:
Enter name 1: yamada
Enter age for yamada: 12
Enter name 2: haru
Enter age for haru: 26
Enter name 3: tanaka
Enter age for tanaka: 8
Sorted list by age: [('tanaka', 8), ('yamada', 12), ('haru', 26)]
```

```
Enter your choice: 4
Invalid choice. Please try again.
```

```
Laboratory Activity
Choose an option:
Option 1: Enter 5 numbers and filter odd numbers only
Option 2: Enter another 5 numbers and union with filtered odd numbers set
Option 3: Enter names and ages, and sort by age
Option 0: Exit
Enter your choice: 0
Leaving the program.
```

```
Laboratory Activity
Choose an option:
Option 1: Enter 5 numbers and filter odd numbers only
Option 2: Enter another 5 numbers and union with filtered odd numbers set
Option 3: Enter names and ages, and sort by age
Option 0: Exit
Enter your choice:
```

Source Code:

SW1:

```python
students = []

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

for i in range(num_students):
    name = input(f"Enter the name of student {i + 1}: ")
    grade = int(input(f"Enter the grade of {name}: "))
    students.append({"name": name, "grade": grade})

sorted_students = sorted(
    students,
    key=lambda student: (-student["grade"], student["name"])
)

print("\nSorted list of students:")
for student in sorted_students:
    print(f"{student['name']} with grade {student['grade']}")
```

Output:

```
Enter the number of students: 3
Enter the name of student 1: john
Enter the grade of john: 76
Enter the name of student 2: leonard
Enter the grade of leonard: 90
Enter the name of student 3: nagallo
Enter the grade of nagallo: 85

Sorted list of students:
leonard with grade 90
nagallo with grade 85
john with grade 76
```

SW2:

```
people = [("Alice", 25), ("Bob", 20), ("Charlie", 30)]

words = ["banana", "apple", "cherry", "blueberry"]

cubes = {x: x**3 for x in range(1, 6)}




people_sorted_by_age = sorted(people, key=lambda person: person[1])

sorted_words = sorted(words, key=len)



print("\nSeatwork #2")

print("People sorted by age:", people_sorted_by_age)

print("Words sorted by length:", sorted_words)

print("Cubes:", cubes)
```

Output:

```
Seatwork #2
People sorted by age: [('Bob', 20), ('Alice', 25), ('Charlie', 30)]
Words sorted by length: ['apple', 'banana', 'cherry', 'blueberry']
Cubes: {1: 1, 2: 8, 3: 27, 4: 64, 5: 125}
```

Lab:

```python
odd_numbers_set = set()

def filter_odd_numbers(numbers_set):
    global odd_numbers_set
    odd_numbers_set = {num for num in numbers_set if num % 2 != 0}
    print("Filtered set with only odd numbers:", odd_numbers_set)

def union(new_numbers_set):
    union_set = odd_numbers_set.union(new_numbers_set)
    print("Union of filtered odd numbers set and new set:", union_set)

#Sorting of tuples
def sortAge(people):
    sorted_people = sorted(people, key=lambda person: person[1])
    print("Sorted list by age:", sorted_people)

def main():
    while True:
        print("\nLaboratory Activity")
        print("Choose an option:")
        print("Option 1: Enter 5 numbers and filter odd numbers only")
        print("Option 2: Enter another 5 numbers and union with filtered odd numbers set")
        print("Option 3: Enter names and ages, and sort by age")
        print("Option 0: Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            numbers_set = set()
            print("Please enter 5 numbers:")
            for i in range(5):
                number = int(input(f"Enter number {i+1}: "))
                numbers_set.add(number)

            filter_odd_numbers(numbers_set)
        elif choice == '2':
            if not odd_numbers_set:
                print("You need to run Option 1 first to generate a filtered odd numbers set.")
                return

            new_numbers_set = set()

            print("Please enter another 5 numbers for union operation:")
            for i in range(5):
                number = int(input(f"Enter number {i+1}: "))
                new_numbers_set.add(number)
            union(new_numbers_set)
        elif choice == '3':
            people = []
            print("Please enter three names and their corresponding ages:")
            for i in range(3):
                name = input(f"Enter name {i+1}: ")
                age = int(input(f"Enter age for {name}: "))
                people.append((name, age))
            sortAge(people)
        elif choice == '0':
            print("Leaving the program.")
            break
        else:
            print("Invalid choice. Please try again.")

# Run the main function to start the program
main()
```

Output:

```
Enter your choice: 1
Please enter 5 numbers:
Enter number 1: 1
Enter number 2: 2
Enter number 3: 3
Enter number 4: 4
Enter number 5: 5
Filtered set with only odd numbers: {1, 3, 5}
```

```
Enter your choice: 2
Please enter another 5 numbers for union operation:
Enter number 1: 6
Enter number 2: 7
Enter number 3: 8
Enter number 4: 9
Enter number 5: 10
Union of filtered odd numbers set and new set: {1, 3, 5, 6, 7, 8, 9, 10}
```

```
Enter your choice: 3
Please enter three names and their corresponding ages:
Enter name 1: yamada
Enter age for yamada: 12
Enter name 2: haru
Enter age for haru: 26
Enter name 3: tanaka
Enter age for tanaka: 8
Sorted list by age: [('tanaka', 8), ('yamada', 12), ('haru', 26)]
```

```
Enter your choice: 4
Invalid choice. Please try again.
```

```
Laboratory Activity
Choose an option:
Option 1: Enter 5 numbers and filter odd numbers only
Option 2: Enter another 5 numbers and union with filtered odd numbers set
Option 3: Enter names and ages, and sort by age
Option 0: Exit
Enter your choice: 0
Leaving the program.
```